Fast wrapper feature subset selection in high-dimensional datasets by means of filter re-ranking

Pablo Bermejo, Luis de la Ossa, José A. Gámez, José M. Puerta

Department of Computing Systems. Intelligent Systems and Data Mining Laboratory (I3A). University of Castilla-La Mancha. Albacete, 02071, Spain.

Abstract

This paper deals with the problem of supervised wrapper-based feature subset selection in datasets with a very large number of attributes. Recently the literature has contained numerous references to the use of hybrid selection algorithms: based on a filter ranking, they perform an incremental wrapper selection over that ranking. Though working fine, these methods still have their problems: (1) depending on the complexity of the wrapper search method, the number of wrapper evaluations can still be too large; and (2) they rely on a univariate ranking that does not take into account interaction between the variables already included in the selected subset and the remaining ones.

Here we propose a new approach whose main goal is to drastically reduce the number of wrapper evaluations while maintaining good performance (e.g. accuracy and size of the obtained subset). To do this we propose an algorithm that iteratively alternates between filter ranking construction and wrapper feature subset selection (FSS). Thus, the FSS only uses the first block of ranked attributes and the ranking method uses the current selected subset in order to build a new ranking where this knowledge is considered. The algorithm terminates when no new attribute is selected in the last call to the FSS algorithm. The main advantage of this approach is that only a few blocks of variables are analyzed, and so the number of wrapper evaluations decreases drastically.

The proposed method is tested over eleven high-dimensional datasets (2400-46000 variables) using different classifiers. The results show an impressive reduction in the number of wrapper evaluations without degrading the quality of the obtained subset.
Keywords: Feature subset selection, High-dimensional datasets, Wrapper algorithms, Filter measures, Complexity, Rank-based algorithms, Re-ranking

1. Introduction

Feature (or variable, or attribute) Subset Selection (FSS) is the process of identifying the input variables which are relevant to a particular learning (or data mining) problem [1, 2], and is a key process in supervised classification. FSS helps to improve classification performance (accuracy, AUC, etc.) and also to obtain more interpretable classifiers or to detect outliers [3]. In the case of high-dimensional datasets, e.g., datasets with thousands of variables, FSS is even more important because otherwise the number of instances needed to obtain reliable models will be enormous (impracticable for many real applications such as microarray domains).

Most algorithms for supervised FSS can be classified as filter or wrapper approaches. In the filter approach an attribute (or attribute subset) is evaluated by only using intrinsic properties of the data (e.g. statistical or information-based measures). Filter techniques have the advantage of being fast and general, in the sense that the subset obtained is not biased in favor of a specific classifier. On the other hand wrapper algorithms are those that use a classifier (usually the one to be used later) in order to assess the quality of a given attribute subset [4]. Wrapper algorithms have the advantage of achieving greater accuracy than filters but with the disadvantage of being (far) more time-consuming and obtaining an attribute subset that is biased towards the classifier used. Over the last decade wrapper-based FSS has been an active area of research. Different search algorithms (greedy sequential [5], floating [6], best-first search, branch and bound [7], evolutionary algorithms [8, 9, 10], etc.) have been used to guide the search process while some classifier (e.g. Naive Bayes, KNN, etc.) is used as a surrogate in order to evaluate the goodness of the subset proposed by the search algorithm. There is no doubt that the results provided by wrapper methods are better than those obtained by using filter algorithms, but the main problem is that they do not scale well. Thus, while datasets of up to 100 or 500 variables were the norm in the last decade of the 20th century, at the start of the 21st century new datasets which involve thousands of variables appeared (e.g. genetics or information-retrieval-based datasets), and the result is that the use of
pure wrapper algorithms is intractable in many cases [11]. Because of this, hybrid filter-wrapper algorithms have become the focus of attention in the last few years. The idea is to use a filter algorithm whose output guides the wrapper algorithm. In this way the advantages of the wrapper approach are retained whilst the number of wrapper evaluations is (considerably) reduced. Examples of these algorithms are [11, 12], which incrementally explore the attributes by following the ranking obtained by a filter measure; [13], which applies a wrapper sequential forward search but only over the first \( k \) (e.g. 100) attributes in the filter ranking; and [14, 15], which use the filter-based ranking for a better organization of the search process.

Our idea in this paper is to improve the efficiency of these so-called hybrid filter-wrapper FSS algorithms. To do this, our aim is to drastically reduce the number of wrapper evaluations by increasing the number of the filter evaluations carried out. Our proposal is based on working incrementally not only at the attribute level, but also at the block or set of attributes level, taking into account the selected subset (\( S \)) in the previous blocks. Thus, we start by using a filter measure to rank the attributes, then an incremental filter-wrapper algorithm \( A \) is applied but only over the first block, that is, over the first \( B \) ranked attributes. Let \( S \) be the subset of attributes selected from this first block. Then, a new ranking is computed over the remaining attributes but taking into account the already selected ones (\( S \)). Then, algorithm \( A \) is run again over the first block in this new ranking but initializing the selected subset to \( S \) instead of \( \emptyset \). This process is iterated until no modification in the selected subset is obtained. As we show, in our experiments, the number of re-ranks carried out is very small, and so only a small percentage of attributes is explored, which leads to a great reduction in wrapper evaluations (and so in CPU time) but without decreasing the accuracy of the output obtained. Even the size of the selected subset is reduced.

Besides this introduction, this paper is organized as follows: the following section presents a set of well-known incremental selection algorithms; in Section 3 we introduce the motivation for reconstructing the ranking in search-time; in Section 4 we introduce our contribution/improvement based on re-ranking; then, Section 5 contains the experimental evaluation carried out, and finally in Section 6 we provide a summary of the main conclusions and future work.
2. Previous work on wrapper FSS for high-dimensional data

In this section we briefly review previous works in the literature for speeding-up wrapper subset selection when working with high-dimensional datasets. We focus on methods based on the use of a filter ranking, so we start with the construction of the ranking and then we briefly describe some algorithms that make use of it.

2.1. Filter step: creating the ranking

As we are in a supervised problem, in order to create the ranking a measure \( m(A_i; C) \) is computed for each predictive attribute \( A_i \) with respect to the class feature \( C \). Therefore, this stage requires \( O(n) \) filter evaluations\(^1\). It is very common to use correlation and information-based metrics as filter measure \( m(A_i; C) \). In our case, we follow [11, 12, 14, 16] and Symmetrical Uncertainty (SU) [17] is used to evaluate the individual merit with respect to the class for each attribute. SU is a nonlinear information-theory-based measure that can be interpreted as a sort of Mutual Information normalized to interval \([0,1]\):

\[
SU(A_i, C) = 2 \left( \frac{H(C) - H(C|A_i)}{H(C) + H(A_i)} \right),
\]

\( C \) being the class and \( H() \) being the Shannon entropy. Attributes are ranked in decreasing SU order; that is, more informative attributes are placed first.

2.2. Rank-based FSS algorithms

In the following algorithms we assume that the ranking \( r \) has already been computed.

**Rank Search**

The Rank Search algorithm [18] evaluates exactly \( n \) subsets, containing the first ranked variable, the first two ranked variables, the first three ranked variables, . . . Therefore, it is linear in the number of wrapper evaluations, that is, \( O(n) \), however its main drawback is that it usually chooses relatively large subsets.

\(^1\)Notice that this ranking can also be created by using the wrapper approach, but in this case it needs more CPU time which is biased to the classifier used
Incremental Selection
This approach starts with $S = \emptyset$ and runs over the ranking by iteratively testing $S \cup A_r$ in a wrapper way. Then, if the wrapper evaluation obtained is better than the current one (corresponding to $S$), $A_r$ is added to $S$, otherwise it is discarded. Obviously, this approach is also linear in the number of variables with the extra advantage over Rank Search that the evaluated models in practice have (far) fewer variables, $|S|$ vs $\frac{n+1}{2}$ on average. Note that Rank Search evaluates exactly $n$ subsets with cardinality 1, 2, 3, …, $n$, and therefore the average cardinality is $\frac{1}{n} \cdot \frac{n(n+1)}{2} = \frac{n+1}{2}$. However, the cardinality of the subsets evaluated by IWSS is bounded by $|S|$. Thus, in the worst case, if all the $n$ variables are selected, IWSS evaluates exactly the same subsets as Rank Search. However, from our experiments (see Table 2) $|S| << n$, and so the complexity of the wrapper evaluations is clearly favorable for IWSS. Different proposals follow this idea with some modifications. In [12, 19] a look-ahead parameter $l$ is used to allow the search to finish when $l$ consecutive attributes have been explored and discarded. BIRS [11] uses a relevance criterion based on the use of t-tests over the output of an inner 5-folds cross validation. Later, in IWSS [20] alternative relevance criteria to the use of a t-test are studied. In the experiments described in this paper we follow the suggestion of [20] and a variable is considered to be relevant (and so added to $S$) if besides having a better mean in the 5-fold cross validation, it is also better in at least 2 out of the 5 folds.

Incremental Selection with Replacement
In [14] a more sophisticated incremental wrapper algorithm is presented: IWSSr. Now, when an attribute ranked in position $i$ is analyzed, then not only its inclusion is studied but also its interchange with any of the variables already included in $S$. Thus, the algorithm can retract from some of its previous decisions, that is, a previously selected variable can become useless after adding some others. As shown in [14] this new algorithm behaves in a similar way to the simpler (IWSS) incremental approach with respect to accuracy but obtains more compact subsets. Of course, this search is more complex and has worst-case complexity $O(n^2)$, although its actual (experimental) complexity reduces to $O(n^{1.3})$ [14].

Best Agglomerative Ranked Subset (BARS)
BARS [15] alternates between the construction of a ranking of the available subsets (initially single variables) and a growing heuristic process that
obtains all the combinations (by merging) of the first three subsets in the ranking with each one of the remaining ones. After the growing phase, all the subsets with worse accuracy than the current best are pruned. A new ranking is created and so on. Worst case complexity of BARS is exponential, but in practice, when few attributes are selected it evaluates less candidate subsets than IWSSr.

**Linear Forward Selection**

Linear Forward subset selection (LFS) [13] tries to speed-up the wrapper search by limiting the number of available attributes to those ranked in the top \( k \) positions. Then, any wrapper algorithm can be used. The authors propose the use of Sequential Forward Selection (SFS) [5], whose complexity in this case reduces from \( O(n^2) \) to \( O(k^2) \).

### 3. Motivating re-ranking

As we can observe in the previous section, the methods discussed therein mainly deal with the high-dimensionality problem in two different manners.

The first one is by using a ranking filter as a guide for the wrapper approach [11, 12, 19, 20, 14, 15], decreasing in this way the number of wrapper evaluations. However, in the case of dealing with thousands of variables, the number of wrapper evaluations is still very high even for the simpler (linear) algorithm, that is, if we have \( n=100000 \) variables, then 1000 000 wrapper evaluations are required. If we use a better but also more complex algorithm, then more evaluations are needed.

The second one consists in a different usage of the filter ranking. The idea is that better variables are ranked in the first positions and so variables having a worse ranking can be directly discarded. This idea is used for example in BARS [15], where the authors recommend that only the first half of the ranking should be explored. This is also used in IWSS and IWSSr [21], where an option termed as early stopping limits the exploration of the search to the first 20%-50% of the ranking. However, perhaps the most representative algorithm exploiting this idea is LFS [13], where only a small portion of the rank is explored, e.g. the first 100 variables (default value suggested by authors).

Although this second proposal drastically reduces the number of wrapper evaluations, it usually obtains worse results with respect to the case of running over the complete ranking, that is, allowing the algorithms to use all
the variables. Below, we give two critical examples regarding this proposal.

*Example 1.* First, we use a worst-case theoretical example to illustrate the drawbacks of only using the first variables in the ranking. Let us consider Figure 1, where we use a graphical model (specifically a Bayesian network) [22] to show the dependence relations between the variables. Thus, from this figure and using the criterion of *d-separation* [22], we can observe:

- Variables $K_1, \ldots, K_n$ are irrelevant (marginally and conditionally) to *Class*.
- Variable $Y$ is relevant (dependent) to *Class*.
- Variable $Z$ is marginally irrelevant (independent) to *Class* but conditionally relevant (dependent) to *Class* given the value of $Y$.

![Figure 1: A Bayesian network illustrating a FSS problem.](image)

Therefore, in this case the minimum set of variables that makes the rest independent (irrelevant) to *Class* is $\{Y, Z\}$. However, if we create a rank using a measure $I(X;\text{Class})$, we should obtain:

$$Y, X_1, \ldots, X_n, Z, K_1, \ldots, K_n$$

where boxes mean that the variables inside them can be placed in any order. Therefore, if we only select the first $k$ variables of the rank, then we will never obtain the best selected subset because $Z$ will always be discarded (at least for reasonable values for $k$).
Example 2. Figure 2 shows the position in the filter rank of the last variable included in the selected subset by the SFS algorithm when using the Naive Bayes classifier for the 11 datasets used in our experiments (see Section 5 for a description). As we can observe, only in one out of 11 cases does the algorithm not select variables after position 100 (the default for LFS). Furthermore, if we set the threshold to be the first 10% of the variables, which can result in a high number of variables given the cardinality of the datasets used, the previous conclusion extends to only 3 out of the 11 datasets.

4. Re-Ranking-based Feature Subset Selection

In the above sections we have seen that reducing the number of variables prior to running the FSS process is the approach that most dramatically reduces the number of wrapper evaluations, and so the one achieving a better speed-up in the case of high-dimensional datasets. However, from the two
examples discussed in Section 3 we have also realized that choosing only
the first variables of the filter (univariate) ranking can represent a serious
limitation for the posterior FSS process, because that ranking does not take
into account possible interactions between the attributes.

In this paper we propose to deal with the number of variables to be used in
a dynamic way, but we also propose to manage the ranking dynamically. That
is, instead of setting the number of variables to be used at the beginning, we
allow the algorithm to modify this number during the FSS selection process.
This behavior is not new, for example in [21] the number of variables to be
still explored is increased each time a new variable is added to the selected
subset and in [19] a look-ahead parameter is used to define the number of
consecutive variables still to be explored without including a new one in the
selected subset. However, if we only manage the number of variables to be
analyzed in the FSS process in a dynamic way, we still have the problem
of using a static ranking, where no interactions (conditional dependences)
between variables are considered. Thus, the main novelty and contribution
of this paper is to deal with the ranking in a dynamic way.

Our approach, termed Re-Ranking, tries to overcome the aforementioned
problems by iteratively using blocks (subsets) of variables taken from the
beginning of the ranking, but instead of always using the initial (univariate)
ranking we propose to re-rank the remaining attributes by taking into account
the current selected subset $S$. In this way, (1) attributes correlated with $S$
will be placed at the end of the new ranking because they add nothing to
the class once we know the value of variables in $S$; and (2) variables that are
conditionally correlated with the class will be placed early in the ranking if
the conditional relevance is due to variables already included in $S$.

The algorithm for re-ranking-based incremental selection is shown in Fig-
ure 3. The following points should be made:

- **Selection algorithm.** As a selection algorithm, any one suitable for being
  seeded with a starting selected subset can be used. Of course, for those
  algorithms requiring a ranking such as IWSS or IWSS$_r$ there is no need
  to (re)compute it because they will receive it as a parameter ($B$).

- **Stop criterion.** As can be observed the algorithm stops when analyzing
  a new block does not produce a modification in the selected subset,
  that is, it returns the same subset received as seed. Thus, the number
  of attributes to explore is decided dynamically and its specific value
  will depend on the evolution of the selection process.
In: $\mathbf{T}$ training set, $M$ filter measure, $\mathcal{C}$ classifier, $B$ block size

Out: $\mathbf{S}$ // The selected subset

1. list $\mathbf{R} = \{\}$ // The ranking, best attributes first
2. for each predictive attribute $A_i$ in $\mathbf{T}$
   
   3. $\text{Score} = M_{\mathbf{T}}(A_i, \text{class})$
   
   4. insert $A_i$ in $\mathbf{R}$ according to $\text{Score}$
   
   5. $\text{sol.}\mathbf{S} = \emptyset$ // selected variables
   
   6. $\text{sol.eval} = \text{null}$ // data about the wrapper evaluation of $\text{sol.}\mathbf{S}$
   
   7. $\mathbf{B} =$ first block of size $B$ in $\mathbf{R}$ // $\mathbf{B}$ is ordered
   
   8. remove first $B$ variables from $\mathbf{R}$
   
   9. $\text{sol} =$ IncrementalSelection($\mathbf{T}, \mathbf{B}, \mathcal{C}, \mathbf{S}$)
   
   10. $\text{continue} =$ true
   
   while $\text{continue}$ do
      
      11. $\mathbf{R}' = \{\}$
      
      12. for each predictive attribute $A_i$ in $\mathbf{R}$
      
      13. $\text{Score} = M_{\mathbf{T}}(A_i, \text{class}|\mathbf{S})$
      
      14. insert $A_i$ in $\mathbf{R}'$ according to $\text{Score}$
      
      15. $\mathbf{R} = \mathbf{R}'$
      
      16. $\mathbf{B} =$ first block of size $B$ in $\mathbf{R}$ // $\mathbf{B}$ is ordered
      
      17. remove first $B$ variables from $\mathbf{R}$
      
      18. $\text{sol'} =$ IncrementalSelection($\mathbf{T}, \mathbf{B}, \mathcal{C}, \mathbf{S}$)
      
      19. if($\text{sol.}\mathbf{S} == \text{sol'.}\mathbf{S}$) //no new feature selected
      
      20. then $\text{continue} =$ false
      
      21. else $\text{sol} =$ sol'
      
   22. return ($\text{sol.}\mathbf{S}$)

Figure 3: Re-Ranking Canonical Algorithm.

- **Block size.** The block size is a key parameter in this approach. This value must be large enough to give some freedom to the wrapper algorithm, but not so large as to explore a large number of useless attributes, which would cancel out the advantages of using re-ranking. As we will see in the experiments, only a few re-ranks are carried out in most datasets, so we think that values of 30-50 are adequate to get the aforementioned balance.

- **Re-ranking algorithm.** In order to build the ranking of the remaining attributes, $\{A_1, \ldots, A_r\}$, but considering the current selected subset, we need to score $M(A_i, \text{class} | \mathbf{S})$ for each $i = 1, \ldots, r$. As we know,
exact computation of this term is not feasible even for moderate sizes of \( S \) because we will need very large (#instances) training sets and too much time and space. Of course, if the size of \( S \) grows, then this expression is directly computationally intractable.

In the literature we can find different ways to approximate this score:

1. **Conditional Mutual Information Maximization (CMIM).**
   Based on using conditional mutual information [23], CMIM tries to balance the amount of information present for each candidate attribute \( A_i \) and class \( C \), and the fact that this information might have been already caught by some feature \( A_j \in S \). Thus, this method selects features maximizing their mutual information with the class but minimizing their pair-to-pair dependency. In this case, given that we have a selected subset \( S \) and a set of attributes to rank \( \{A_1, \ldots, A_r\} \), the merit \( M(A_i, C|S), i = 1, \ldots, k \) is computed as:
   \[
   M(A_i, C|S) = \min_{A_j \in S} I(A_i; C|A_j)
   \]

2. **Mutual Information-Based Feature Selection (MIFS).**
   Similar to the main idea in CMIM, Battiti presented MIFS in [24]. Thus, Battiti suggests approximating the merit \( M(A_i, C|S), i = 1, \ldots, k \) by computing it as:
   \[
   M(A_i, C|S) = I(A_i; C) - \beta \sum_{A_j \in S} I(A_i; A_j)
   \]
   Where \( \beta \in [0, 1] \) and its commonly suggested value is 0.5.

3. **Max-Relevance and Min-Redundancy (MRMR).**
   Peng et. al [25] present an approximation similar to MIFS; in this case, the merit \( M(A_i, C|S), i = 1, \ldots, k \) is computed as:
   \[
   M(A_i, C|S) = I(A_i; C) - \frac{1}{|S|} \sum_{A_j \in S} I(A_i; A_j)
   \]

In our proposal we use Symmetrical Uncertainty \( SU() \) instead of mutual information \( I() \). As we can see, the number of calls to the filter measure each time we have to re-rank attributes is \( r \cdot |S| \). However these computations are more than compensated for the dramatic reduction in the number of calls to the wrapper evaluator.
5. Experiments

5.1. Test suite

As we stated in the introduction to this paper, our research in this study is motivated by high-dimensional datasets, thus we consider 11 publicly obtained datasets from the ASU Feature Selection Repository\(^2\): the first 4 of them (warpAR10P, warpAR10P, pixraw10P and orlraws10P) come from the digital image field, the next 5 (TOX-171, SMK-CAN-187, GLI-85, GLA-BRA-180, CLL-SUB-111 and CLL-SUB-111) are bioinformatics datasets, specifically microarrays, and the last 2 (pcmac and basehock) come from the text mining field (textual databases). Table 1 shows the number of features, records/instances and the class cardinality for all datasets. As we can observe from this table, the number of variables is very high, ranging from 2400 to 46151, while the number of instances is very small except for the two text databases. All of them are balanced, that is, the number of instances belonging to a certain class value is the same (or almost the same) as for the other class values.

| Dataset       | #Feats. | #Inst. | | | Dataset       | #Feats. | #Inst. | | |
|---------------|--------|--------| | |---------------|--------|--------| | |
| warpPIE10P    | 2421   | 210    | 10 | | warpAR10P     | 2400   | 130    | 10 |
| pixraw10P     | 10000  | 100    | 10 | | orlraws10P    | 10304  | 100    | 10 |
| TOX-171       | 5749   | 171    | 4  | | SMK-CAN-187   | 46151  | 180    | 4  |
| GLI-85        | 22283  | 85     | 2  | | GLA-BRA-180   | 19993  | 187    | 2  |
| CLL-SUB-111   | 11340  | 111    | 3  | | pcamc         | 3289   | 1943   | 2  |
| basehock      | 4862   | 1993   | 2  | |                |        |        |    |

Table 1: Number of attributes, instances and class cardinality in the used datasets.

5.2. Methodology

The goal of our experiments is to test the goodness of using Re-ranking as a framework in which incremental feature subset selection algorithms can be plugged in. To do this, we perform several experiments in order to demonstrate that the use of Re-ranking drastically reduces the number of wrapper evaluations (which is proportional to CPU time) while not degrading the performance (accuracy and number of selected attributes) of the FSS algorithms used. We also carry out additional experiments in order to study

\(^2\)http://featureselection.asu.edu
the block size and the measure used to approximate \(M(A_i; C|S)\). Thus, our experiments follow this methodology:

1. First we focus on studying the use of Re-ranking in combination with the following four FSS algorithms frequently used with high-dimensional datasets: SFS, IWSS (mf=2), IWSSr (mf=2) and BARS (\(\epsilon=100\) and \(k=3\)). The goal is to check that the performance of original FSS algorithms (in terms of accuracy and cardinality of selected subset) does not decrease when the number of wrapper evaluations is drastically reduced. For this study, block size is set to two different values, \(B=10\) and \(B=30\), and the re-ranking criterion is CMIM. Four well-known classifiers coming from different paradigms are used: Naive Bayes [26], c4.5 [27, 28], AODE [29] and ibK [30]. In all the cases the implementation provided in the WEKA toolkit [31] with default parameters has been used.

2. Our second experiment is devoted to studying the impact of the block size \(B\). In addition, we also study the effect of working incrementally at the block-level but without using re-ranking.

3. Finally, from the knowledge acquired in the previous two experiments we set a third one in order to compare between the three different approximations for computing the measure used to re-rank the attributes described in Sec. 4.

In all the cases we use a 10-fold cross validation and the average over the ten test folds is reported as the output statistic.

5.3. Experimental Results

5.3.1. Experiment 1.- Studying the impact of introducing Re-ranking in the FSS process

Tables 2 and 3 show the results obtained when running the feature selection algorithms SFS, IWSS, IWSSr and BARS with and without using our Re-ranking proposal. Two different block sizes \((B=10\) and \(B=30\)) and the CMIM score used to create the conditioned ranking are considered. We denote the Re-ranking-based algorithms by using the superscript \(R\). For each pair (FSS method, classifier) we report the accuracy and cardinality of the selected subset averaged over the ten folds of the cross-validation carried out. We also report the average over the 11 datasets. Furthermore, the row under the mean shows the result of carrying out the statistical analysis. Thus, we compare each pair (with and without using re-ranking) by using
the non-parametrical Wilcoxon Matched-Pairs Signed-Ranks Test [32, 33], applying a confidence level of $\alpha = 0.05$. The result of a test is indicated in the corresponding cell as: = if there is no statistical difference, $\uparrow$ if the re-rank algorithm returns a significantly greater value, and $\downarrow$ if the re-rank algorithm returns a significantly smaller value.

Regarding accuracy, we can observe that the conclusions are almost the same for both block sizes (10 and 30). Thus, when $B = 10$ and according to the statistical tests output, from the 16 compared pairs we get 15 ties and only one case in which the re-ranked version significantly improves upon the version without re-ranking. In the case of $B = 30$ we obtain 13 ties and 3 cases in which the re-ranked version is significantly better. With respect to the number of selected attributes, we can observe that the use of re-ranking is significantly better in 8 (7) cases, worse in 3 (5) and there are no statistically significant differences in 5 (4) cases for $B = 10$ ($B = 30$). Therefore, for the classifiers used, and for datasets similar to those considered, we find that the use of our re-ranking proposal does not degrade the performance of the FSS algorithms tested.

On the other hand, when we study the number of wrapper evaluations carried out, then the difference is dramatic. Tables 4 and 5 show the number of wrapper evaluations carried out when using the Naive Bayes classifier (the ratios are also representative for the other three classifiers).

An additional remark/advantage is that the use of Re-ranking makes it possible to finish execution (in a pre-fixed large amount of time) of datasets pcmac and basehoek using the classifier ibK and the SFS algorithm (empty cells in Tables 2 and 3; in this case statistical tests are done using the results for the other 9 datasets).

In Figure 4 the number of evaluations is shown for a better comparison (please note the logarithmic scale). Evaluations of the FSS algorithm with and without Re-ranking are shown for block sizes $B = 10$ and $B = 30$. In all the algorithms the Naive Bayes classifier has been used. We also included the number of evaluations carried out by the LFS algorithm (Section 2.2) in this graph. Although not shown here for the sake of clarity, a comparison of accuracy between LFS (e.g. using SFS and Naive Bayes) and SFS$^R$ (also using Naive Bayes) outputs a statistically significant difference in favour of SFS$^R$ for block sizes $\geq 30$ and no difference for the smallest block sizes, although in these cases the Re-ranking algorithm carries out (far) less wrapper evaluations.

Since the time spent in a feature selection process does not depend just on
<table>
<thead>
<tr>
<th>Dataset</th>
<th>SFS</th>
<th>SFS</th>
<th>IWSS</th>
<th>IWSS</th>
<th>IWSSr</th>
<th>IWSSr</th>
<th>BARS</th>
<th>BARS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DataList</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 2: Comparison of FSS algorithms Vs. the same algorithm + (CMIM) Re-ranking (B=10).

The number of evaluations but the expensive ness of such evaluations (cardinal ity of candidate subset) we also show, in Table 6, the mean selection time (in minutes) of SFS, IWSS, IWSSr and BARS with and without re-ranking.
<table>
<thead>
<tr>
<th>Dataset</th>
<th>SFS</th>
<th>SFS^a</th>
<th>IWS</th>
<th>IWS^a</th>
<th>IWSSr</th>
<th>IWSSr^a</th>
<th>BARS</th>
<th>BARS^a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cox</td>
<td>0.15</td>
<td>0.15</td>
<td>0.15</td>
<td>0.15</td>
<td>0.15</td>
<td>0.15</td>
<td>0.15</td>
<td>0.15</td>
</tr>
<tr>
<td>AML</td>
<td>0.25</td>
<td>0.25</td>
<td>0.25</td>
<td>0.25</td>
<td>0.25</td>
<td>0.25</td>
<td>0.25</td>
<td>0.25</td>
</tr>
<tr>
<td>CLL</td>
<td>0.35</td>
<td>0.35</td>
<td>0.35</td>
<td>0.35</td>
<td>0.35</td>
<td>0.35</td>
<td>0.35</td>
<td>0.35</td>
</tr>
</tbody>
</table>

**Table 3:** Comparison of FSS algorithms Vs. the same algorithm + (CMIM)Re-ranking (B=30).

(CMIM and B = 30). The Ratio row shows the number of times faster than an algorithm is when re-ranking is applied. Although not shown here, the common behaviour is that for lower values for B the greater the ratio
Table 4: Number of wrapper evaluations. $B = 10$ and CMIM are used for versions including re-ranking.

<table>
<thead>
<tr>
<th>DataSet</th>
<th>SFS</th>
<th>SFS$^b$</th>
<th>IWSS</th>
<th>IWSS$^b$</th>
<th>IWSSr</th>
<th>IWSSr$^b$</th>
<th>BARS</th>
<th>BARS$^b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>warpPIE</td>
<td>29778.3</td>
<td>269.0</td>
<td>2420.0</td>
<td>56.0</td>
<td>34801.0</td>
<td>399.3</td>
<td>19307.0</td>
<td>897.0</td>
</tr>
<tr>
<td>warpAR</td>
<td>34320.0</td>
<td>225.0</td>
<td>2400.0</td>
<td>41.0</td>
<td>29622.1</td>
<td>225.8</td>
<td>8406.9</td>
<td>742.0</td>
</tr>
<tr>
<td>pixraw</td>
<td>35000.0</td>
<td>104.0</td>
<td>10000.0</td>
<td>33.0</td>
<td>71849.9</td>
<td>171.6</td>
<td>37468.8</td>
<td>291.0</td>
</tr>
<tr>
<td>orlraws</td>
<td>75219.2</td>
<td>157.0</td>
<td>10304.0</td>
<td>39.0</td>
<td>83600.6</td>
<td>151.5</td>
<td>41369.7</td>
<td>462.0</td>
</tr>
<tr>
<td>TOX</td>
<td>77611.5</td>
<td>293.0</td>
<td>5748.0</td>
<td>51.0</td>
<td>86706.7</td>
<td>299.2</td>
<td>17067.4</td>
<td>754.0</td>
</tr>
<tr>
<td>SMK</td>
<td>175938.4</td>
<td>118.0</td>
<td>19993.0</td>
<td>30.0</td>
<td>188950.7</td>
<td>141.5</td>
<td>44232.1</td>
<td>270.0</td>
</tr>
<tr>
<td>GLI</td>
<td>111415.0</td>
<td>79.0</td>
<td>22283.0</td>
<td>25.0</td>
<td>121454.6</td>
<td>102.4</td>
<td>59671.3</td>
<td>264.0</td>
</tr>
<tr>
<td>GLA</td>
<td>470740.2</td>
<td>98.0</td>
<td>49151.0</td>
<td>40.0</td>
<td>522086.2</td>
<td>118.2</td>
<td>133257.1</td>
<td>275.0</td>
</tr>
<tr>
<td>CLL</td>
<td>80814.6</td>
<td>165.0</td>
<td>11340.0</td>
<td>41.0</td>
<td>116075.3</td>
<td>170.3</td>
<td>21856.6</td>
<td>376.0</td>
</tr>
<tr>
<td>pcmac</td>
<td>379221.7</td>
<td>141.0</td>
<td>3289.0</td>
<td>60.0</td>
<td>164961.4</td>
<td>247.7</td>
<td>22954.7</td>
<td>916.0</td>
</tr>
<tr>
<td>basehock</td>
<td>369025.8</td>
<td>805.0</td>
<td>4862.0</td>
<td>64.0</td>
<td>192413.9</td>
<td>1427.7</td>
<td>16660.2</td>
<td>2047.0</td>
</tr>
<tr>
<td>Mean</td>
<td>107162.19</td>
<td>274.9</td>
<td>12890.00</td>
<td>46.5</td>
<td>141138.40</td>
<td>386.8</td>
<td>38304.71</td>
<td>606.7</td>
</tr>
<tr>
<td>Ratio</td>
<td>608.1</td>
<td>285.0</td>
<td>382.8</td>
<td>63.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Number of wrapper evaluations. $B = 30$ and CMIM are used for versions including re-ranking.

<table>
<thead>
<tr>
<th>DataSet</th>
<th>SFS</th>
<th>SFS$^b$</th>
<th>IWSS</th>
<th>IWSS$^b$</th>
<th>IWSSr</th>
<th>IWSSr$^b$</th>
<th>BARS</th>
<th>BARS$^b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>warpPIE</td>
<td>29778.3</td>
<td>780.0</td>
<td>2420.0</td>
<td>142.0</td>
<td>13681.0</td>
<td>1268.2</td>
<td>19307.0</td>
<td>2591.0</td>
</tr>
<tr>
<td>warpAR</td>
<td>34320.0</td>
<td>720.0</td>
<td>2400.0</td>
<td>124.0</td>
<td>13681.0</td>
<td>1268.2</td>
<td>8406.9</td>
<td>1857.0</td>
</tr>
<tr>
<td>pixraw</td>
<td>35000.0</td>
<td>363.0</td>
<td>10000.0</td>
<td>103.0</td>
<td>71849.9</td>
<td>171.6</td>
<td>37468.8</td>
<td>945.0</td>
</tr>
<tr>
<td>orlraws</td>
<td>75219.2</td>
<td>474.0</td>
<td>10304.0</td>
<td>112.0</td>
<td>83600.6</td>
<td>151.5</td>
<td>41369.7</td>
<td>1384.0</td>
</tr>
<tr>
<td>TOX</td>
<td>77611.5</td>
<td>726.0</td>
<td>5748.0</td>
<td>110.0</td>
<td>86706.7</td>
<td>1298.0</td>
<td>17067.4</td>
<td>1682.0</td>
</tr>
<tr>
<td>SMK</td>
<td>175938.4</td>
<td>360.0</td>
<td>19993.0</td>
<td>110.0</td>
<td>188950.7</td>
<td>496.6</td>
<td>44232.1</td>
<td>908.0</td>
</tr>
<tr>
<td>GLI</td>
<td>111415.0</td>
<td>249.0</td>
<td>22283.0</td>
<td>110.0</td>
<td>121454.6</td>
<td>277.8</td>
<td>59671.3</td>
<td>724.0</td>
</tr>
<tr>
<td>GLA</td>
<td>470740.2</td>
<td>510.0</td>
<td>49151.0</td>
<td>110.0</td>
<td>522086.2</td>
<td>481.1</td>
<td>133257.1</td>
<td>1168.0</td>
</tr>
<tr>
<td>CLL</td>
<td>80814.6</td>
<td>2355.0</td>
<td>11340.0</td>
<td>110.0</td>
<td>116075.3</td>
<td>618.6</td>
<td>21856.6</td>
<td>1211.0</td>
</tr>
<tr>
<td>pcmac</td>
<td>379221.7</td>
<td>2355.0</td>
<td>3289.0</td>
<td>110.0</td>
<td>164961.4</td>
<td>380.3</td>
<td>22954.7</td>
<td>3118.0</td>
</tr>
<tr>
<td>basehock</td>
<td>369025.8</td>
<td>2700.0</td>
<td>4862.0</td>
<td>110.0</td>
<td>192413.9</td>
<td>16660.2</td>
<td>16660.2</td>
<td>2047.0</td>
</tr>
<tr>
<td>Mean</td>
<td>107162.19</td>
<td>274.9</td>
<td>12890.00</td>
<td>46.5</td>
<td>141138.40</td>
<td>386.8</td>
<td>38304.71</td>
<td>606.7</td>
</tr>
<tr>
<td>Ratio</td>
<td>608.1</td>
<td>285.0</td>
<td>382.8</td>
<td>63.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.3.2. Experiment 2.- Analysis of the block size ($B$)

In the above experiments we have seen that similar accuracies are obtained for the same (FSS algorithm, Classifier), so in this experiment we only show the results for the pair (IWSS, Naive Bayes) to avoid overwhelming the reader with too many tables. Thus, the upper part of Table 7 shows the
results obtained when setting the block size $B$ to the following values: 5, 10, 20, 30, 40 and 50.

In order to obtain sound conclusions in this comparison we perform a Friedman test [34] followed by a post-hoc Holm test [35], as suggested in [33] and using the code provided in [36]; the confidence level is set to $\alpha = 0.05$. The control algorithm is automatically selected by the code used [34], commonly being the algorithm with the highest accuracy when comparing by accuracy, or the lowest cardinality when comparing by number of attributes. We first perform the test comparing by accuracy, and algorithm with $B = 50$ is selected as control. The result is that block sizes $B = 5$ and $B = 10$ perform significantly worse (we cross out their means using $\cdot$). Then, if we repeat the study over the survival block sizes but using the cardinality of the selected subset as criterion, then $B = 40$ and $B = 50$ are discarded (crossed out). Therefore, from the study we can conclude that values around 20 and 30 are a reasonable choice for $B$.

In this experiment we also study the possibility of incrementally working
Table 7: Studying the impact of block-size: with re-ranking (top) and without re-ranking (bottom).

at the block level but without performing re-ranking, that is, all blocks are defined over the original (marginal) ranking, and used to decide when to stop. This behaviour is obtained by removing lines 12-16 from the algorithm in Fig. 3. From the results shown in the lower half of Table 7 we can observe how the use of re-ranking improves on the accuracy of its counterpart algorithm using first-ranking-based blocks, although it is also observable (and expected) that the gain is reduced when increasing $B$.

5.3.3. Experiment 3.- Studying the score used to re-rank attributes.

In the above experiments we have used CMIM as the score to approximate the conditional mutual information needed to construct the ranking conditioned on the current selected subset, $S$. We chose CMIM because, from the scores reviewed in Section 4, it is the only one including (order-one) conditional mutual information in its computation. In this experiment we carry out a comparison between CMIM and the other two scores (MIFS and MRMR) discussed in Section 4 to approximate $M(A_i; C|S)$. As in experiment 2 and for the sake of simplicity we show the results (Table 8) for a single configuration (IWSSr and Naive Bayes), though two block sizes ($B = 10$ and

<table>
<thead>
<tr>
<th>Dataset</th>
<th>IWSS</th>
<th>B=5</th>
<th>B=10</th>
<th>B=20</th>
<th>B=30</th>
<th>B=40</th>
<th>B=50</th>
</tr>
</thead>
<tbody>
<tr>
<td>warpPIE</td>
<td>74.62</td>
<td>61.74</td>
<td>54.18</td>
<td>48.40</td>
<td>42.01</td>
<td>35.05</td>
<td></td>
</tr>
<tr>
<td>pixelr</td>
<td>90.00</td>
<td>84.76</td>
<td>79.33</td>
<td>74.00</td>
<td>68.67</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GLraw</td>
<td>66.67</td>
<td>60.00</td>
<td>53.33</td>
<td>48.33</td>
<td>43.33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOX</td>
<td>91.43</td>
<td>86.67</td>
<td>81.67</td>
<td>76.67</td>
<td>71.67</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMK</td>
<td>93.00</td>
<td>87.00</td>
<td>81.00</td>
<td>75.00</td>
<td>69.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GLA</td>
<td>85.00</td>
<td>78.33</td>
<td>71.67</td>
<td>65.00</td>
<td>58.33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLL</td>
<td>79.33</td>
<td>72.67</td>
<td>66.67</td>
<td>60.00</td>
<td>53.33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>picmac</td>
<td>78.49</td>
<td>72.00</td>
<td>65.00</td>
<td>58.00</td>
<td>51.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>basehock</td>
<td>89.51</td>
<td>84.00</td>
<td>78.89</td>
<td>73.00</td>
<td>67.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>79.44</td>
<td>73.81</td>
<td>68.18</td>
<td>62.63</td>
<td>57.00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Stop B=5</th>
<th>Stop B=10</th>
<th>Stop B=20</th>
<th>Stop B=30</th>
<th>Stop B=40</th>
<th>Stop B=50</th>
</tr>
</thead>
<tbody>
<tr>
<td>warpPIE</td>
<td>74.62</td>
<td>61.74</td>
<td>54.18</td>
<td>48.40</td>
<td>42.01</td>
<td>35.05</td>
</tr>
<tr>
<td>pixelr</td>
<td>90.00</td>
<td>84.76</td>
<td>79.33</td>
<td>74.00</td>
<td>68.67</td>
<td></td>
</tr>
<tr>
<td>GLraw</td>
<td>66.67</td>
<td>60.00</td>
<td>53.33</td>
<td>48.33</td>
<td>43.33</td>
<td></td>
</tr>
<tr>
<td>TOX</td>
<td>91.43</td>
<td>86.67</td>
<td>81.67</td>
<td>76.67</td>
<td>71.67</td>
<td></td>
</tr>
<tr>
<td>SMK</td>
<td>93.00</td>
<td>87.00</td>
<td>81.00</td>
<td>75.00</td>
<td>69.00</td>
<td></td>
</tr>
<tr>
<td>GLA</td>
<td>85.00</td>
<td>78.33</td>
<td>71.67</td>
<td>65.00</td>
<td>58.33</td>
<td></td>
</tr>
<tr>
<td>CLL</td>
<td>79.33</td>
<td>72.67</td>
<td>66.67</td>
<td>60.00</td>
<td>53.33</td>
<td></td>
</tr>
<tr>
<td>picmac</td>
<td>78.49</td>
<td>72.00</td>
<td>65.00</td>
<td>58.00</td>
<td>51.00</td>
<td></td>
</tr>
<tr>
<td>basehock</td>
<td>89.51</td>
<td>84.00</td>
<td>78.89</td>
<td>73.00</td>
<td>67.00</td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>79.44</td>
<td>73.81</td>
<td>68.18</td>
<td>62.63</td>
<td>57.00</td>
<td></td>
</tr>
</tbody>
</table>
Table 8: Comparing the use of CMIM, MIFS and MRMR to approximate conditional mutual information when performing re-ranking. IWSSr is used as the FSS algorithm, Naive Bayes as classifier and two block sizes are considered: $B = 10$ and $B = 30$.

6. Conclusions and Future Work

In this paper we have presented a significant improvement for incremental feature subset selection algorithms. The idea is based on working on blocks/sets of attributes. The novelty is that the ranking changes each time we analyze a block. Thus, after analyzing the current block we perform a re-rank based on conditional mutual information that takes into account the variables already included in the selected subset. The advantage of this approach is that attributes that become redundant because of the current content of the selected subset will go to the end of the new ranking, while attributes that become relevant because of interactions with the content of the current selected subset will be placed at the beginning of the new ranking.

The results of our experiments show that this way of proceeding leads to a reduced number of re-ranks, which means that only a few blocks (and
so attributes) are analyzed, and so the method is (far) more efficient. Furthermore, the results also show that the quality of the selected subset is not degraded. On the contrary, the same (classification) accuracy, and sometimes the number of selected attributes is reduced. Regarding parameters, we recommend the use of CMIM-based Re-ranking and a block size of around 20-30. These conclusions are supported by an extensive set of experiments: 4 classifiers, 4 different FSS algorithms and 11 high-dimensional datasets, statistically proving the improvements in performance.

7. Acknowledgements

This work has been supported by the JCCM under project PCI08-0048-8577, and MEC under project TIN2007-67418-C03-01.

References


