This is a postprint version of the following published document:

Martín, R., Aler, R. & Galván, I.M. A filter attribute selection method based on local reliable information. *Applied Intelligence* 48, 35–45 (2018).

DOI: [10.1007/s10489-017-0959-3](https://doi.org/10.1007/s10489-017-0959-3)

© 2017, Springer Science+Business Media, LLC, part of Springer Nature

A Filter Attribute Selection Method based on Local Reliable Information

Ricardo Martín , Ricardo Aler *, and Inés M. Galván

gricardo.martin@gmail.com, {aler,igalvan}@inf.uc3m.es

Computer Science Department. Carlos III University. Spain

January 12, 2017

Abstract

In this article, a filter feature weighting technique for attribute selection in classification problems is proposed (**LIA**). It has two main characteristics. First, unlike feature weighting methods, it is able to consider attribute interactions in the weighting process, rather than only evaluating single features. Attribute subsets are evaluated by projecting instances into a grid defined by attributes in the subset. Then, the joint relevance of the subset is computed by measuring the information present in the cells of the grid. The final weight for each attribute is computed by taking into account its performance in each of the grids it participates. Second, many real problems contain low signal-to-noise ratios, due for instance to high noise levels, class overlap, class imbalance, or small training samples. **LIA** computes reliable local information for each of the cells by estimating the number of target class instances not due to chance, given a confidence value. In order to study its properties, **LIA** has been evaluated with a collection of 18 real datasets and compared to two feature weighting methods (**Chi-Squared** and **ReliefF**) and a subset feature selection algorithm (**CFS**). Results show that the method is significantly better in many cases, and never significantly worse. **LIA** has also been tested with different grid dimensions (1, 2, and 3). The method works best when evaluating attribute subsets larger than 1, hence showing the usefulness of considering attribute interactions.

Keywords: Feature selection, Filter methods, Feature weighting methods, Noisy domains

*Corresponding author: aler@inf.uc3m.es

1 Introduction

Feature selection is one of the important field of study within the Machine Learning field [8, 7, 19, 24] because of the many benefits it can bring to supervised learning. Among others, reducing overfitting, improving prediction accuracy, or removing irrelevant or redundant features, which can contribute to better performance for classification algorithms [20]. Attribute selection methods may be grouped into two main categories: filter and wrapper [8]. Filter approaches use general characteristics of data (such as correlation with the class) to evaluate and select attributes, while wrapper approaches select attributes according to the evaluation provided by a specific machine learning algorithm [18]. Generally, wrapper methods achieve better accuracy than filter methods because the search process is guided by the performance of the classifier [25], but when the number of features is very large, the wrapper approach can become computationally very costly (see for instance [12]). Additionally, wrappers may result in overfitting if the search process is too exhaustive and the training set is small [21, 23]. On the other hand, filter methods are computationally very fast, yet they usually achieve very competitive results [15]. There are also hybrid systems that combine the strengths of both approaches by using an independent measure (i.e. filter) to quickly narrow the search space and a final stage where evaluation by a learning algorithm (i.e. wrapper) is used to decide on the final subset. Thus, filter approaches can also be seen as a preprocessing stage for wrapper methods [5, 27].

This paper focuses on the filter approach to feature selection. Filter methods can be divided into two groups of algorithms, namely feature weighting algorithms and subset search algorithms [28, 29, 26]. The first group assigns weights individually to each attribute depending on their relevance to the class. Then, attributes are ranked using these weights. There are many ways to measure relevance. For instance, Information Gain weights attributes by using Shannon entropy to measure the prediction uncertainty reduction [3]. Another common measure is Mutual Information [4], that measures dependency between attributes and the class. Chi-square is an statistical method that can be used with the same purpose. The second group of filter algorithms evaluates subsets, instead of individual attributes. This is commonly used to penalize subsets that contain redundant attributes, rather than to evaluate their joint predictive capabilities [13, 10, 29].

In this work, we propose a filter weighting selection method - Attribute Selection based on Local Information Analysis (LIA) - but with the ability to consider the joint relevance of subsets of attributes during the ranking process. In order to evaluate subsets of attributes, the algorithm projects training instances into a space whose dimensions are the attributes to be evaluated. This space is then divided into a grid of rectangular regions (cells). The subset of attributes is evaluated by computing the reliable information provided for the different cells in the grid. Finally, weights are computed for each attribute by aggregating the relevance of the subsets where the attribute participates.

It is common in classification domains the presence of low signal-to-noise

ratios. Some examples are domains where one of the classes contains fewer instances, compared to the rest of classes (imbalanced domains), or domains with small training samples, high levels of noise, or class overlap. As mentioned in previous paragraphs, **LIA** needs to compute the information contained in each of the cells of a grid in order to evaluate the joint performance of the subset of attributes that generate the grid. In order to deal with noise issues, **LIA** uses a measure that estimates 'reliable information' (for a given confidence value), where effects due to noise are removed before determining how much information is contained in the cell about the target class (typically, the minority class).

The algorithm has been evaluated with binary classifications domains including problems with different degrees of class imbalance, sample size, and number of features. **LIA** has been compared with three other filter selection algorithms. Two of them are feature weighting algorithms, **Chi-Squared** [17] and **ReliefF** [14, 16], and the third one is **CFS** [11, 10], a filter method that evaluates subsets of attributes. **Chi-Squared** is a well-performing and widely available method in most machine learning platforms. It evaluates individual attributes by computing the chi-squared statistic with respect to the class. **ReliefF** estimates the quality of attributes according to how well their values discriminate different-class instances that are near to each other. It has been selected because, although it is a feature weighting method, there is evidence that is robust to feature interactions and noise [16]. Finally, **CFS** has been chosen because it is a subset feature algorithm selection that in addition to evaluating the individual predictive ability of each feature, it also evaluates the subset by penalizing the redundancy between pairs of attributes. In order to evaluate the performance of the feature selection algorithms, a classifier must be used. Different classification algorithms may deal differently with the same subset of selected attributes, because for instance, they may have different degrees of robustness against irrelevant or redundant attributes. Therefore, in this work, three classifier algorithms have been used: J48, Multilayer Perceptron, and K-NN.

The rest of the paper is organized as follows. The attribute selection algorithm is described in Section 2. In Section 3 the experimental validation of the algorithm is presented. Finally, Section 4 ends the paper by drawing some conclusions and describing future lines of research.

2 Attribute Selection Method

The proposed feature selection algorithm ranks attributes according to the information they supply about a target class. The method takes into account possible interactions between attributes by considering all possible combinations of 1, 2, and 3 attributes. The feature space defined by a subset of attributes is partitioned into a grid of cells by dividing attribute ranges $[0, 1]$ into P parts (all attributes have been previously discretized and normalized). If $[0, 1]$ is divided into two parts ($P = 2$), $[0, 0.5]$ and $[0.5, 1]$, and subsets have 3 attributes, the grid has $2 \times 2 \times 2 = 8$ cells. Then, the number of instances for each class within

each cell can be counted and a function is used to compute how much reliable information is provided to the target class in every cell in the grid. Every attribute participates in many attribute combinations, and therefore, in many different grids. Thus, attributes can be ranked by adding the information they provide in the grids they participate. The process at the core of attribute ranking is the computation of reliable information in a grid cell. This is described below.

2.1 Function F to evaluate the local reliable information

LIA local reliable information is computed by function F for every cell. It is 'local' because it considers one cell at a time. It is 'reliable' because it discounts the effects of randomness on the actual observed number of C_1 instances in the cell. In order to describe F , the following notation will be used:

- C_1 and C_0 : C_1 is the target class (typically the minority class). C_0 is the non-target class.
- $Total.C_1$ and $Total.C_0$: Number of C_1 and C_0 patterns in the dataset.
 $Total = Total.C_1 + Total.C_0$
- $cell$: A cell in the grid
 - $cell.C_1$ and $cell.C_0$: Number of C_1 and C_0 patterns in the cell.
 - $cell.Total = cell.C_0 + cell.C_1$
- $confidence$: it is the required confidence on the estimate computed by F . The larger this number, the more conservative will be the estimate returned by function F
- $density = \frac{Total.C_1}{Total}$. It is the density of C_1 instances for the whole dataset.

The computation of F is based on the concept of number of 'reliable patterns' of the target class within a cell ($reliable.C_1$). Basically, $reliable.C_1$ is a conservative estimation of how many of the observed $cell.C_1$ patterns in the cell can be trusted, given a confidence level. Or equivalently, how many of the observed $cell.C_1$ patterns are not due to randomness. If the required confidence level is very high, that means that the estimate returned by F must be very conservative ($reliable.C_1 \ll cell.C_1$) (or in other words, that the level of noise is assumed to be very high). The number of reliable patterns is estimated by calculating the success probability (or density d_{C_1}) of a binomial distribution $B(n, d_{C_1})$, with number of trials $n = cell.Total$, that could generate the observed $cell.C_1$ patterns with a probability equal to $confidence$ ¹, and then considering that only $d_{C_1} \times cell.Total$ patterns are reliable. $reliable.C_1$ will be always smaller or equal than $cell.C_1$. And the larger the confidence required, the smaller or more conservative $reliable.C_1$ will be.

¹Specifically, $prob_B(X \leq cell.C_1) = confidence$. This will be explained in more detail in the next paragraphs.

Algorithm 1 shows the steps followed to compute F for a *cell* and a *confidence* level. In step 1, the density of C_1 patterns is computed for the complete dataset (this is the background density of the dataset). Then, in step 2, $\delta density$ is computed by means of the *DeltaDensity* function. This function will be described later in Algorithm 2), but basically, $density + \delta density$ is the smallest density (probability success rate) of a binomial, that for the given confidence level, the C_1 patterns within the cell could have been generated by that distribution. The latter could not be claimed if $\delta density$ was just slightly smaller. Steps 4 and 5 display the calculation of the number of reliable patterns, related to the density computed in step 2. From this point on, only the C_1 reliable patterns are considered to actually belong to C_1 , the rest are included in C_0 . Taking the latter into account, steps 5 and 6 recompute the number of C_1 and C_0 patterns for the rest of the dataset (i.e. outside the cell, represented by *difference* in Algorithm 2). Finally, the entropy-based information gain (IG) [22] is used to measure how much entropy is reduced by partitioning the dataset into the within / without cell partitions (with the correction that only the C_1 reliable patterns are considered to belong to C_1 within the cell). r and $1 - r$ are the proportions of these two partitions. IG is considered to be the amount of information contained in the cell.

Algorithm 1 FUNCTION $F(cell, confidence, total)$

Input:

cell: a cell in the grid.

confidence: required confidence on the estimate computed by F

total: the complete dataset.

Output: F returns the Information Gain for the cell.

- 1: $density \leftarrow \frac{total.C_1}{total.C_1 + total.C_0}$
 - 2: $\delta density \leftarrow \text{DeltaDensidad}(cell.C_1, cell.C_0, confidence, total.C_1, total.C_0)$
 - 3: $reliable.C_1 \leftarrow cell.Total * (density + \delta density)$
 - 4: $reliable.C_0 \leftarrow cell.Total - reliable.C_1$
 - 5: $difference.C_0 \leftarrow Total.C_0 - reliable.C_0$
 - 6: $difference.C_1 \leftarrow Total.C_1 - reliable.C_1$
 - 7: $r \leftarrow reliable.Total / Total$
 - 8: **return** $F_{Ent}(Total) - (r \times F_{Ent}(reliable) + (1 - r) \times F_{Ent}(difference))$
-

Algorithm 2 describes function *DeltaDensity*, which computes the increase in the background density ($density + \delta density$), so that a binomial distribution can generate the C_1 patterns observed in the cell, with a given *confidence*. In other words, $B(cell.Total, density + \delta density)$ is a binomial such that Eq. 1 holds.

$$prob_B(X \leq cell.C_1) = confidence \quad (1)$$

As explained in previous paragraphs, this is used in function F to determine the number of reliable C_1 patterns in a cell, as $cell.Total \times (density + \delta density)$. Algorithm 2 solves Eq. 1 for $\delta density$ by means of a binary search.

$CDFBINOM(k, p, n) = prob_{B(n,p)}(X \leq k)$ is the cumulative distribution function of a binomial distribution $B(n, p)$, with n the number of trials, and p the trial success probability.² Binary search is implemented in Algorithm 2 by updating variables d_1 and d_2 , which are the lower and upper bounds of $d = density + \delta density$. p_1 and p_2 are the cumulative probabilities at bounds d_1 and d_2 . The updating process is iterated until the midpoint between the bounds d has a cumulative probability p as close to the target (*confidence*) as required ($confidence - p < \epsilon$).

Algorithm 2 FUNCTION *DeltaDensity*($NC_1, NC_0, confidence, totalC_1, totalC_0$)

Input:

NC_1, NC_0 : number of C_1 and C_0 patterns within the cell.

$totalC_1, totalC_0$: Total number of C_1 and C_0 patterns in the dataset.

confidence: This function performs a binary search for the success probability d of a binomial distribution $B(NC_1 + NC_0, d)$, so that $prob_B(X \leq NC_1) \sim confidence$

Output: The increase over the background density, so that the associated binomial distribution can generate the observed NC_1 patterns in the cell with a probability equal to *confidence*.

```

density  $\leftarrow (totalC_1 / (totalC_1 + totalC_0))$ 
 $d_1 \leftarrow density$ 
 $p_1 \leftarrow CDFBINOM(NC_1, d_1, NC_1 + NC_0)$ 
if  $p_1 < confidence$  then
    return 0
end if
 $d_2 \leftarrow NC_1 / (NC_1 + NC_0)$ 
 $p_2 \leftarrow CDFBINOM(NC_1, d_2, NC_1 + NC_0)$ 
 $\epsilon \leftarrow \frac{1 - confidence}{10}$ 
 $p \leftarrow -1$ 
while  $|confidence - p| \geq \epsilon$  do
     $d \leftarrow \frac{d_1 + d_2}{2}$ 
     $p \leftarrow CDFBINOM(NC_1, d, NC_1 + NC_0)$ 
    if  $confidence < p$  then
         $d_1 \leftarrow d$ 
         $p_1 \leftarrow p$ 
    else
         $d_2 \leftarrow d$ 
         $p_2 \leftarrow p$ 
    end if
end while
return  $\delta density = (d - density)$ 

```

²The GSL library has been used for computing $CDFBINOM$:
<https://www.gnu.org/software/gsl>

2.2 LIA Algorithm for attribute selection

LIA procedure is displayed in Algorithm 3. Basically, LIA uses function F (defined in Algorithm 1) to evaluate all possible subsets of $maxDepth$ attributes. For each grid, the cell with maximum information is determined. This information is accumulated for every attribute and every confidence level. This is then used to rank the attributes.

In Algorithm 3, steps 1 to 5 initialize vector $VConf$ with values $1 - 10^{-cp}$, ranging cp from CP_{min} to CP_{max} by CP_{step} . For instance, in this article, $CP_{min} = 0.30103$, $CP_{max} = 4.30103$, and $CP_{step} = 1$ have been chosen, resulting in $VConf = (0.50000, 0.95000, 0.99500, 0.99950, 0.99995)$. At the end of this loop, C contains the number of different confidence values that are going to be considered. Step 6 creates the $MI[a, c]$ array, where the amount of information for every attribute a and every confidence c is contained. The loop that starts at step 7 goes through all possible grids defined by $depth$ attributes ($depth$ is the size of attribute subsets considered, which in turn define the dimensionality of the grids of cells). Steps 8 and 9 create the grid for the subset of attributes $(a_1, \dots, a_n)$ and instantiate it with the patterns in the dataset (basically, instances in the dataset are distributed among the cells of the grid defined by attributes $(a_1, \dots, a_n)$). $Depth$ determines the dimension of the grid (i.e. the size of the subset of attributes used to create the grid). As mentioned in the introduction, this article considers triplets of attributes ($Depth=3$), but in the experimental section we will study the effects of having smaller subsets, with $Depth=2$ and $Depth=1$. Step 10 determines the best cell in the grid, such that the summation of its information F for all confidence levels is maximum.³ Once $bestCell$ has been determined, its information broken down per confidence level is stored in vector $VInfoMax$ (steps 11, 12, and 13). $VInfoMax$ contains an element for each one of the C confidence levels considered. Then, in steps 14 to 18, MI is updated by adding the information for the current grid, for attribute a_j and confidence level i . In short, at the end of the loop from steps 7 to 19, matrix $MI[a, c]$ contains, for every attribute a and every confidence level c , the summation of information for all the different grids where attribute a participates. Finally, steps 20, 21, and 22 compute vector MR , which summarizes the amount of information for each attribute a_j . This is done by adding the information MI of attribute a_j for all possible confidence levels. However, for each confidence level, MI is normalized by dividing by the maximum MI for that confidence level, so each term in the summation of step 21 ranges from 0 to 1. Attributes can now be ranked by their MR : the higher $MR[a]$, the more relevant is the a -th attribute.

The time complexity of LIA is $O(G * N)$, where G is the number of grids (subsets of attributes of size $Depth$) evaluated. For triplets, as in this article, the time complexity is $O(Depth^3 * N)$. In order to improve this time complexity, in this article the following optimization has been applied: 3-dimensional grids

³Technically, this is done for all cells, but also, all cell groupings, or hiper-cells, with a paralelepide (rectangular-like) shape. The reason is that in some cases, the region that contains the information is larger than a single cell.

Algorithm 3 Algorithm LIA

Input:

Values: a $nAtrs \times nPatterns$ matrix containing the dataset.
depth: the size of an attribute subset / the dimensionality of the grid .
 CP_{min} , CP_{max} : minimum and maximum confidence parameters.
 CP_{step} : confidence parameter step.

Output:

MR: a vector with $nAtrs$ components. $MR[i]$ contains the information of the i -th attribute.

```
1:  $C \leftarrow 0$ 
2: for  $cp$  in  $[CP_{min}..CP_{max}, step\ CP_{step}]$  do
3:    $VConf[C] \leftarrow 1 - 10^{-cp}$ 
4:    $C \leftarrow C + 1$ 
5: end for
6:  $MI \leftarrow new\ Array(nAtrs, C)$ 
7: for  $(a_1, \dots, a_n)$  in  $[combination(nAtrs, Depth)]$  do
8:    $grid \leftarrow new\ Grid(P, depth)$ 
9:    $grid.fill(Values, a_1, \dots, a_n)$ 
10:   $bestCell \leftarrow \arg\max_{cell} \sum_{i=1}^{t=C} F(cell, VConf[i], total)$ 
11:  for  $i$  in  $[1..C]$  do
12:     $VInfoMax[i] \leftarrow F(bestCell, VConf[i], total)$ 
13:  end for
14:  for  $i$  in  $[1..C]$  do
15:    for  $j$  in  $[1..n]$  do
16:       $MI[a_j, i] \leftarrow MI[a_j, i] + VInfoMax[i]$ 
17:    end for
18:  end for
19: end for
20: for  $j$  in  $[1..n]$  do
21:   $MR[a_j] = \sum_{i=1}^{t=C} (MI[a_j, i]) / \max_a(MI[a, i])$ 
22: end for
23: return MR
```

are processed only for those combinations of attributes in which there is an attribute that has provided information for grids of dimension 2. In that case, the time complexity tends to $O(\text{Depth}^2 * N)$.

LIA is able to control the accuracy vs. time complexity tradeoff via the *Depth* parameter. *Depth*=3 can be used for best accuracy. *Depth*=1 will obtain answers very quickly, while *Depth*=2 can be used as a convenient tradeoff. Section 3 provides results for *Depth*=3 and compares them with those with *Depth*=2 and *Depth*=1. In order to offer a guide to select the *Depth* parameter, depending on the size of the domain and the available computing time, we provide the number of attributes that LIA is able to handle within a time limit of 30 seconds, in an Intel I5 computer with 4Gb RAM, and a domain with 50000 training patterns. Under those conditions, LIA is able to process domains with $4,500 \times 10^3$, 3×10^3 , and 2×10^3 attributes, for *Depth*= 1, 2, and 3, respectively. Given that LIA is linear in the number of instances, computing times can be calculated for larger domains.

2.3 A note on implementation

LIA has been implemented carefully in order to be able to apply it to problems with thousands of attributes. In particular, LIA is able to use parallelism. Given that there are no temporal dependencies, all grids can be processed in parallel, and results combined at the end of the process. Also, the following design principles have been followed in order to make execution times fast:

1. Low use of conditional sentences: grouping instances into cell grids can be achieved with simple mathematical operations, integer arithmetic, and no conditional sentences.
2. Locality of access: the sequential implementation of LIA allows to use the cache hierarchy optimally.
3. SIMD (Single Instruction, Multiple Data): given that LIA can be mostly implemented with integer arithmetic, and access to data stored in grid cells can be done sequentially, the algorithm can be implemented with SSE3 instructions.

The current implementation of LIA is able to process, approximately, 400000 grids (subsets of attributes with *Depth* = 3) per second, in an Intel I5 with 4GB RAM.

3 Experimental Validation on real domains

We have carried out experiments using 18 binary real classification domains, 15 of them have been obtained from the KEEL-dataset repository [1]⁴ and three (Musk-2, Splice, and Kr-vs-kp) from the UCI Machine learning repository [2].

⁴<http://keel.es/>

Table 1: Description of Real Domains

Name	Attributes	Instances	IR
Musk-2	166	5279	5,49
Sonar	60	166	1,16
Splice	60	2540	1,08
Spambase	57	3677	1,54
Kr-vs-kp	37	2556	1,09
Dermatology	34	358	16.9
Segment0	19	2308	6.02
Vehicle0	18	846	3.25
Vehicle1	18	846	2.9
Vehicle2	18	846	2.88
Vehicle3	18	846	2.9
Page-blocks0	10	5472	8.79
Wisconsin	9	683	1.86
Pima	8	768	1.87
Yeast1	8	1484	2.46
Yeast3	8	1484	8.1
Ecoli-0_vs_1	7	220	1.86
New-thyroid1	5	215	5.14

⁵ Table 1 summarizes the characteristics of the domains: number of attributes, number of instances, and imbalance ratio (*IR*).⁶

For comparison purposes, experiments have been carried out using Weka’s implementation [9] of the **Chi-Squared**, **ReliefF**, and **CFS** attribute selection algorithms. In this implementation the attribute evaluators are CfsSubsetEval, ChiSquaredAttributeEval and ReliefFAttributeEval, respectively. As search method, **Chi-Squared** and **ReliefF** use the Ranker method (which ranks the attributes according to their weights) and Best First for the **CFS** algorithm.

The performance of the different attribute selection methods has been evaluated by their effects on classification algorithms. Given that all selection methods belong to the filter category, their attribute ranking is independent of specific classification algorithms. However, different classification algorithms react differently to the same subset of attributes, because, for instance, they deal differently with irrelevant or redundant attributes. Therefore, in this work, we have used three classification algorithms: J48 (the Weka implementation of C4.5), Nearest neighbor, and MLP (the Weka implementation of the Multilayer Perceptron). All classifiers have been run with Weka default parameters.

As we have mentioned, **LIA** has been designed for selecting attributes that provide more information on a target class, in this case the minority class. It is known that global accuracy is not a good performance measure for imbalanced

⁵<http://archive.ics.uci.edu/ml/>

⁶*IR* is the ratio between the number of instances of the majority class vs. the minority class.

data sets because classifiers tend to focus on the majority class at the expense of the minority class. Therefore, in this work, *G-mean*, a metric for imbalanced problems, has been used to evaluate and compare attribute selection methods. *G-mean* is the geometric mean of sensitivity and specificity of the classifiers. Let's *TP*, *FP*, and *FN* the number of true/false positives, and false negatives, respectively, *G-mean* is defined as $G-mean = \sqrt{Sensitivity \times Specificity}$ where $Sensitivity = TP / (TP + FN)$ and $Specificity = TN / (TN + FP)$.

3.1 Experimental results

The method has been tested on the real domains described in Table 1 using 5-fold cross validation. *G-mean* has been computed for every fold and averaged over the five folds.

The Wilcoxon test has been used in order to determine if the number of domains where **LIA** outperforms the other selection methods is statistically significant [6]. This test has been carried out for every proportion of selected features, from 10% to 90% by 10% increase. This means that after the attributes have been ranked, different subset sizes are tested, from 10% (of the ranked attributes) to 90%. In order to study the influence of the dimensionality of the grid used by **LIA** to evaluate attributes, experiments with *Depth* = 1, *Depth* = 2 and, *Depth* = 3 have also been carried out.

Results of the Wilcoxon test for *Depth* = 1, *Depth* = 2 and, *Depth* = 3 can be seen in Figure 1. The Figure contains three main vertical sections, one for each classification algorithm (J48, KNN, and MLP), as can be seen at the top of the figure. Within each vertical section, there are three columns, one for each of the attribute selection algorithms (CFS, CHI, and ReliefF) that have been compared to **LIA**, as can be seen in the x-axis labels (at the bottom). Each column displays the significance results of the test for each of the percentage of attributes (from 10% to 90%, labels in the y-axis). Within each column, significance for *Depth* = 1, *Depth* = 2, and *Depth* = 3 are shown from left to right. A black-filled circle means that **LIA** is better than the attribute selection algorithm indicated in the x-axis, white-filled meaning the opposite. Differences are statistically significant if the circle is large, non-significant otherwise.

In most cases, it can be observed that performance of **LIA** is improved as *Depth* is increased from 1 to 3. In other words, white circles become black circles, or small black circles become large black circles (i.e. the improvement is statistically significant). Next, the results in Figure 2 will be analyzed comparing **LIA** with *Depth* = 3 and the other three attribute selection algorithms. In order to ease visualization, **LIA** results for *Depth* = 3 only, have been replicated in Figure 2:

- **LIA** and CFS: It can be seen that **LIA** performs better than CFS in general: all circles are black for all the classification algorithms (J48, KNN and MLP). Also, the improvement is significant for all percentages (all black circles at CFS and *depth* = 3 are large).

- **LIA and CHI:** LIA is almost always better (all circles but one are black) and differences are significant in many cases. The only case where **Chi** is better is when used with KNN and 10% of attributes are selected, but even in that case, LIA is not significantly worse.
- **LIA and ReliefF:** LIA is again usually better than **ReliefF**, and significant improvements can be observed in many cases (only two white circles when using J48, but LIA is not significantly worse even in those cases).

In summary, LIA is never significantly worse than the other methods (i.e. no big white-filled circles can be observed in Figure 2). Also, very few small white-filled circles can be observed, which means that even if we consider non-significant differences, it is seldom the case that LIA is worse than the other selection algorithms.

The previous paragraphs analyzed the influence of depth on LIA compared to the other three attribute selection algorithms. For completeness sake, Figure 3 studies the statistical significance of $Depth = 3$ vs. $Depth = 1$, and $Depth = 3$ vs. $Depth = 2$. It can be seen that in most cases, $Depth = 3$ performs better than $Depth = 1$, and in many cases the improvement is significant. With respect to $Depth = 3$ vs. $Depth = 2$, $Depth = 3$ tends to perform better than $Depth = 1$, but the number of times the improvement is significant decreases. Therefore, larger $Depth$ tends to improve results, but with diminishing returns.

The previous results have analyzed whether differences between the attribute selection methods are significant or not, but it is also important to analyze the magnitude of the improvements, to determine whether differences are large or not, and in which cases. Figure 4 displays the average normalized *G-mean* for every percentage of attributes. There is one curve for each of the attribute selection algorithms. *G-mean* for each domain and each percentage of attributes has been normalized by dividing each result by the best *G-mean* obtained for that domain. This normalization factor measures the difficulty for that domain and avoids that results obtained in easy domains outweigh the rest of the domains.

In Figure 4 it can be seen that LIA outperforms the rest of the methods. For J48, LIA is better for all percentages except for 70%, 80%, and 90%, where it is similar to **Chi-Squared** and **ReliefF**. For KNN, LIA performs also better than the other techniques. Only for 10% of attributes, **Chi-Squared** is slightly better than LIA. For MLP, LIA is always better than the rest of techniques in terms of the average of *G-mean*.

Finally, Figure 5 displays the average normalized *Gmean* for $Depth = 1$, $Depth = 2$, and $Depth = 3$. It can be seen that $Depth = 3$ performs better than lower depths, although the improvement depends on the classification algorithm (larger for KNN).

Figure 1: Statistical Wilcoxon significance tests for LIA with $Depth = 1, 2, \text{ and } 3$ versus CFS, Chi-Squared, and ReliefF for different classification methods. Black circle means LIA wins. Large black circle means that the improvement is statistically significant. Y-axis displays the percentage of best-performing attributes used after being ranked.

4 Conclusions

In this article, we have proposed a filter attribute selection algorithm named LIA for binary classification problems. Although the algorithm is a feature weighting method (i.e. it ranks attributes), attribute interactions are considered by measuring their joint performance in multidimensional grids. The joint performance of a subset of attributes that defines a grid is measured by estimating the reliable information about the target class, in each of the grid cells. Reliable information is determined by first removing the effects of noise, given a

Figure 2: Statistical Wilcoxon significance tests for $LIA-Depth = 3$ versus CFS, Chi-Squared, and ReliefF for different classification methods. Black circle means LIA wins. Large black circle means that the improvement is statistically significant. Y-axis displays the percentage of best-performing attributes used after being ranked.

confidence value.

LIA has been validated in 18 real domains and compared to three other attribute selection methods (CFS, CHI, and ReliefF). The performance of the different attribute selection methods has been evaluated via three different classification algorithms (J48, KNN, and MLP). A Wilcoxon test has been used to measure statistical significance of results.

First, we have studied the influence of the dimension of the grid ($Depth$) considered by LIA (or equivalently, the size of the attribute subset that defines

Figure 3: Statistical Wilcoxon significance tests comparing LIA at different depths: $Depth = 3$ vs. $Depth = 1$ (left) and $Depth = 3$ vs. $Depth = 2$.

the grid). LIA performs better as $Depth$ increases from 1 to 3, although the largest improvement occurs when comparing $Depths$ 1 and 3. With regard to the comparison of LIA with the other feature selection methods, experiments show that LIA is never significantly worse than any of them, and in many cases it is significantly better. The latter assertion is true for all the classifiers used. Therefore, we can conclude that LIA has a good performance overall.

Taking into account that wrapper attribute selection methods can in principle achieve better accuracy than filter methods, a future research line could be to develop hybrid selection methods, where a classification algorithm uses as input the attributes initially selected by LIA. Then, depending on the accuracy in classification, attributes could be removed or added to the list previously se-

Figure 4: Averaged Normalized $Gmean$ for the different selection methods.

lected. Also, although LIA has been developed for classification problems, there could be several strategies for extending it to regression problems, like using a new evaluation function or transforming regression problems into classification ones via discretization.

Acknowledgements

The authors acknowledge financial support granted by the Spanish Ministry of Science under contract ENE2014-56126-C2-2-R.

Figure 5: Averaged Normalized $Gmean$ for $Depth = 1$, $Depth = 2$, and $Depth = 3$.

References

- [1] Jesús Alcalá-Fdez, Alberto Fernández, Julián Luengo, Joaquín Derrac, Salvador García, L Sánchez, and F Herrera. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing*, 17, 2011.
- [2] Arthur Asuncion and David Newman. Uci machine learning repository, 2007.
- [3] M Ben-Bassat. Pattern recognition and reduction of dimensionality. *Handbook of Statistics*, 2:773–910, 1982.
- [4] Mohamed Bennasar, Yulia Hicks, and Rossitza Setchi. Feature selection using joint mutual information maximisation. *Expert Systems with Applications*, 42(22):8520–8532, 2015.
- [5] Sanmay Das. Filters, wrappers and a boosting-based hybrid for feature selection. In *ICML*, volume 1, pages 74–81. Citeseer, 2001.
- [6] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.
- [7] George Forman. An extensive empirical study of feature selection metrics for text classification. *The Journal of machine learning research*, 3:1289–1305, 2003.
- [8] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.
- [9] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.
- [10] Mark A Hall. *Correlation-based feature selection for machine learning*. PhD thesis, The University of Waikato, 1999.
- [11] Mark A Hall and Lloyd A Smith. Feature subset selection: a correlation based filter approach. In *International Conference on Neural Information Processing and Intelligent Information Systems*, pages 855–858, 1997.
- [12] Iñaki Inza, Pedro Larrañaga, Rosa Blanco, and Antonio J Cerrolaza. Filter versus wrapper gene selection approaches in dna microarray domains. *Artificial intelligence in medicine*, 31(2):91–103, 2004.
- [13] Asha Gowda Karegowda, AS Manjunath, and MA Jayaram. Comparative study of attribute selection using gain ratio and correlation based feature selection. *International Journal of Information Technology and Knowledge Management*, 2(2):271–277, 2010.

- [14] Kenji Kira and Larry A Rendell. The feature selection problem: Traditional methods and a new algorithm. In *AAAI*, pages 129–134, 1992.
- [15] Ron Kohavi and George H John. The wrapper approach. In *Feature extraction, construction and selection*, pages 33–50. Springer, 1998.
- [16] Igor Kononenko, Edvard Šimec, and Marko Robnik-Šikonja. Overcoming the myopia of inductive learning algorithms with relief. *Applied Intelligence*, 7(1):39–55, 1997.
- [17] Huan Liu and Rudy Setiono. Chi2: Feature selection and discretization of numeric attributes. In *In Proceedings of the Seventh IEEE International Conference on Tools with Artificial Intelligence*, page 388. IEEE, 1995.
- [18] Huan Liu and Lei Yu. Toward integrating feature selection algorithms for classification and clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 17(4):491–502, 2005.
- [19] Huawen Liu, Jigui Sun, Lei Liu, and Huijie Zhang. Feature selection with dynamic mutual information. *Pattern Recognition*, 42(7):1330–1339, 2009.
- [20] Huawen Liu, Xindong Wu, and Shichao Zhang. A new supervised feature selection method for pattern classification. *Computational Intelligence*, 30(2):342–361, 2014.
- [21] John Loughrey and Pádraig Cunningham. Overfitting in wrapper-based feature subset selection: The harder you try the worse it gets. In *Research and Development in Intelligent Systems XXI*, pages 33–43. Springer, 2005.
- [22] Ryszard S Michalski, Jaime G Carbonell, and Tom M Mitchell. *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.
- [23] Juha Reunanen. Overfitting in making comparisons between variable selection methods. *The Journal of Machine Learning Research*, 3:1371–1382, 2003.
- [24] Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. A review of feature selection techniques in bioinformatics. *bioinformatics*, 23(19):2507–2517, 2007.
- [25] Petr Somol, Bart Baesens, Pavel Pudil, and Jan Vanthienen. Filter-versus wrapper-based feature selection for credit scoring. *International Journal of Intelligent Systems*, 20(10):985–999, 2005.
- [26] Alper Kursat Uysal. An improved global feature selection scheme for text classification. *Expert Systems with Applications*, 43:82–92, 2016.
- [27] Eric P Xing, Michael I Jordan, Richard M Karp, et al. Feature selection for high-dimensional genomic microarray data. In *ICML*, volume 1, pages 601–608, 2001.

- [28] Lei Yu and Huan Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *ICML*, volume 3, pages 856–863, 2003.
- [29] Lei Yu and Huan Liu. Efficient feature selection via analysis of relevance and redundancy. *The Journal of Machine Learning Research*, 5:1205–1224, 2004.