A Distributed Feature Selection Approach Based on a Complexity Measure

Verónica Bolón-Canedo(✉), Noelia Sánchez-Maroño, and Amparo Alonso-Betanzos

Laboratory for Research and Development in Artificial Intelligence (LIDIA), Computer Science Department, University of A Coruña, 15071 A Coruña, Spain
{vbolon,nsanchez,ciamparo}@udc.es

Abstract. Feature selection is often required as a preliminary step for many machine learning problems. However, most of the existing methods only work in a centralized fashion, i.e. using the whole dataset at once. In this paper we propose a new methodology for distributing the feature selection process by samples which maintains the class distribution. Subsequently, it performs a merging procedure which updates the final feature subset according to the theoretical complexity of these features, by using data complexity measures. In this way, we provide a framework for distributed feature selection independent of the classifier and that can be used with any feature selection algorithm. The effectiveness of our proposal is tested on six representative datasets. The experimental results show that the execution time is considerably shortened whereas the performance is maintained compared to a previous distributed approach and the standard algorithms applied to the non-partitioned datasets.

1 Introduction

The technological per-capita capacity to store information has almost doubled every 40 months since the 1980s, and for example in 2012, every day 2.5 exabytes of data were created. In this scenario, where data is not only big in volume, but also in complexity and variety, machine learning techniques have become indispensable when extracting useful information from huge amounts of meaningless data. If one analyzes the size (samples $\times$ features) of the datasets posted in the UCI Machine Learning Repository [1], it is easy to see that it has increased dramatically [2]. In the 1980s, the maximal size of the data was about 100; then in the 1990s, this number increased to more than 1500; and finally in the 2000s, it further increased to about 3 million. The proliferation of this type of datasets had brought unprecedented challenges to machine learning researchers. Learning algorithms can degenerate their performance due to overfitting, learned models decrease their interpretability as they are more complex, and finally speed and efficiency of the algorithms decline in accordance with size.

Machine learning can take advantage of feature selection to be able to reduce the dimensionality of a given problem. *Feature selection* (FS) is the process of detecting the relevant features and discarding the irrelevant and redundant ones,

I. Rojas et al. (Eds.): IWANN 2015, Part II, LNCS 9095, pp. 15–28, 2015.
DOI: 10.1007/978-3-319-19222-2_2

with the goal of obtaining a small subset of features that describes properly the given problem with a minimum degradation or even improvement in performance [3]. Feature selection, as it is an important activity in data preprocessing, has been an active research area in the last decade, finding success in many different real world applications, especially those related with classification problems.

Feature selection methods are divided into three categories: filters, wrappers and embedded methods. While wrappers involve optimizing a predictor as part of the selection process, filters rely on the general characteristics of the training data to select features with independence of any predictor. The embedded methods use machine learning models for classification, and then an optimal subset of features is built by the classifier algorithm. As stated in [4], even when the subset of features might be not so accurate as with embedded and wrapper methods, filters are preferable due to their computational and statistical scalability, so they will be the focus of this research. Traditionally, FS methods have been applied in a centralized manner, i.e. a single learning model to solve a given problem. However, there might be several reasons to use a FS method in a distributed way. First, nowadays the data are sometimes distributed in multiple locations. Second, and more common, when dealing with large amounts of data, most existing FS methods are not expected to scale well and their efficiency may significantly deteriorate or even become inapplicable. Therefore, a possible solution might be to distribute the data, run a FS method on each partition and then combine the results. There are two main techniques for partitioning and distributing data: vertically, i.e. by features, and horizontally, i.e. by samples. Distributed learning has been used to scale up datasets that are too large for batch learning in terms of samples [5–8]. While not common, there are some other developments that distribute the data by features [9–11]. Even less common, there are also approaches that address the distribution both vertically and horizontally [12].

This paper will be focused on the most common approach: distribution by samples. We will present a methodology in which several rounds of FS will be performed on different partitions of the data. Then, the partial outputs need to be combined into a single subset of relevant features. In our previous work [13], the distribution of the samples across the different nodes was performed randomly, which could lead to situations where some classes are not represented in all the nodes. Moreover, our previous approach used of a method to combine the partial outputs that involves classification algorithms, which introduced an important overhead. Different than our previous study, and trying to overcome these two disadvantages, in this work (i) we partition the data taking into account the class distribution, (ii) we propose a new method to combine the partial outputs which makes use of data complexity measures, leading to an impressive reduction in the time necessary for this task, and (iii) we present a case study trying to determine if, since we reduced the running time impressively, it is worth considering more computations trying to improve the accuracy. The experimental results from six different databases demonstrate that our new proposal can maintain the performance of both the original FS methods and our previous approach in [13], as well as showing important savings in running times.

2 Methods

2.1 Distributed Feature Selection Algorithm

The idea of distributing the data horizontally builds on the assumption that combining the output of multiple experts is better than the output of any single expert. The proposed methodology consists of performing several fast feature selectors on several partitions of the data, combining then the partial outputs into a single subset of features. The feature selection distributed algorithm is applied to the training dataset in several iterations or rounds. This repetition ensures capturing enough information for the combination stage. For this sake, at each round we start by dividing each dataset D into several small disjoint subsets D_i. Since the partition is being made by samples, it is necessary to bear in mind that random distributions of the data might imply that some of the classes are not represented exhaustively in all nodes. To solve this inconvenience, we divide the data maintaining the original class proportions in the training dataset, i.e. if a dataset has 70% of instances from the positive class and 30% from the negative class, in each partition of the dataset this distribution will be maintained. In this manner, we ensure that all the feature selectors in the different partitions are able to learn all classes.

After having the dataset partitioned into smaller subsets of data D_i, each feature selector is run on each of them, generating a corresponding selection in which the features selected to be removed receive a vote. At this point, a new round starts so a new partition of the dataset is performed and another round of voting is accomplished until reaching the predefined number of rounds. After all the small datasets D_i in each round have been used (which could be done in parallel, as all of them are independent of each other), the combination method builds the final selection S as the result of the filtering process that we will explain in detail in the next paragraph. This set S will be used to train a classifier C and to test its performance over a new set of samples (test dataset).

Combining the partial outputs on the different partitions of data is not an easy-to-solve question. Since at each round and partition, each feature to be removed receives a vote, at the end of the process we will have a number of votes for each feature which may range from 0 to v_{max}, being v_{max} the number of features or attributes in each dataset n times the number of rounds r. To decide which features to remove, it is necessary to estimate a threshold of votes. In our previous work [13], the method to calculate this threshold estimated the best value for the number of votes from its effect on the training set. Following the recommendations exposed in [14], we selected the number of votes taking into account two different criteria: the training error and the percentage of features retained. Both values must be minimized to the extent possible, by minimizing the fitness criterion $e[v]$:

$$e[v] \leftarrow \alpha \times error + (1 - \alpha) \times featPercentage \tag{1}$$

To calculate this criterion, a term α was introduced to measure the relative relevance of both values and was set to $\alpha = 0.75$ as suggested in [14], giving more

influence to the classification error. Since the maximum number of votes v_{max} in some cases might be in the order of thousands, instead of evaluating all the possible values for the number of votes we opted for delimiting it into an interval $[minVote, maxVote]$ computed by using the mean and standard deviation such that $minVote = avg - 1/2std$ and $maxVote = avg + 1/2std$.

The drawback of our previous approach is that, by involving a classifier in the process of selecting the optimal threshold, in some cases the time necessary for this task was higher than the time required by the feature selection process, even without distributing the data, which introduced an important overhead in the running time. Furthermore, this fact made our methodology dependent on the classifier chosen, in a similar way than the wrapper approach does.

Trying to overcome the aforementioned problems, in this paper we propose to modify the function for calculating the threshold of votes by making use of data complexity measures, which have been proposed in the last few years in order to characterize the complexity of datasets beyond estimates of error rates [15]. Thus, instead of evaluating the merit of a candidate subset of features by its classification error rate, we propose to calculate the complexity of the dataset with the candidate features. The rationale behind this decision is that we assume that good candidate features would contribute to decrease the complexity and must be maintained, whilst bad candidate features would contribute to increase the complexity and must be discarded. Since our intention is to propose a framework that could be applicable to both binary and multiclass datasets, among the existing complexity measures, the Fisher discriminant ratio [15] was chosen, which is applicable to problems with any number of classes. Fisher's multiple discriminant ratio for C classes is defined as:

$$f = \frac{\sum_{i=1,j=1,i\neq j}^{C} p_i p_j (\mu_i - \mu_j)^2}{\sum_{i=1}^{C} p_i \sigma_i^2}, \tag{2}$$

where μ_i, σ_i^2, and p_i are the mean, variance, and proportion of the ith class, respectively. In this work we will use the inverse of the Fisher ratio, $1/f$, such that a small complexity value represents an easier problem. Therefore, we propose to replace the formula for calculating $e[v]$ defined in Eq. (1) with the one that we present in Eq. (3), expecting to achieve two important goals (1) a reduction in the time necessary to calculate the threshold and (2) a method independent on the classifier.

$$e[v] \leftarrow \alpha \times 1/f + (1 - \alpha) \times featPercentage \tag{3}$$

The algorithm for the whole methodology is detailed in Algorithm 1. At the end, the final selection S returned by the algorithm is applied to the training and test sets in order to obtain the ultimate classification accuracies. It must be noted that this algorithm can be used with any feature selection method, although the use of filters is recommended since they are faster than other techniques. Note that the method can be also applied on ranker methods, however it is required to establish another threshold to determine the number of features to be removed in each subset of data.

Algorithm 1. Pseudo-code for the proposed distributed methodology

Data: $\mathbf{d}_{(m \times n+1)}$ ← labeled training dataset with m samples and n input features

X ←set of features, X = $\{x_1, \ldots, x_n\}$
s ← number of submatrices of **d** with p samples
V ← vector of votes
r ← number of rounds
$\alpha \leftarrow 0.75$

Result: S ← subset of features \S ⊂ X

`/* Obtaining a vector of votes for discarding features */`

initialize the vector of votes V to 0, $|V|$=n
for *each round* **do**
  Split **d** into s disjoint submatrices maintaining the class distribution
  for *each submatrix* **do**
    apply a feature selection algorithm
    F ← features selected by the algorithm
    E ← features eliminated by the algorithm \E ∪ F = X
    increment one vote in vector V for each feature in E
  end
end

`/* Obtain threshold of votes, Th, to remove a feature */`

$minVote \leftarrow$ minimum threshold considered
$maxVote \leftarrow$ maximum threshold considered
for $v \leftarrow$ *mixVote to maxVote with increment 5* **do**
  F_{th} ← subset of selected features (number of votes $< v$)
  $1/f \leftarrow$ inverse of Fisher ratio computed on training dataset **d** using only features in F_{th}
  $featPercentage \leftarrow$ percentage of features retained $\left(\frac{|F_{th}|}{|X|} \times 100\right)$
  $e[v] \leftarrow \alpha \times 1/f + (1 - \alpha) \times featPercentage$
end
$Th \leftarrow min(e)$, Th is the value which minimizes the function e
S ← subset of features after removing from X all features with a number of votes $\geq Th$

2.2 Experimental Setup

In order to test our distributed framework for feature selection, we have chosen the same six benchmark datasets as in our previous work [13], which are described in Table 1 depicting their properties (number of features, number of training and test samples and number of classes). These datasets can be considered representative of problems from medium to large size, since the horizontally distribution is not suitable for small-sample datasets. All of them can be free downloaded from the UCI Machine Learning Repository [1]. Those datasets originally divided into training and test sets were maintained, whereas, for the sake

of comparison, datasets with only training set were randomly divided using the common rule 2/3 for training and 1/3 for testing. The number of packets (s) to divide the dataset in each round is also displayed in the last column of Table 1. This number was calculated trying to maintain a proportion between the number of samples and the number of features with the constraint of having, at least, three packets per dataset.

Table 1. Dataset description

Dataset	Features	Training	Test	Classes	Packets
Connect4	42	45038	22519	3	45
Isolet	617	6238	1236	26	5
Madelon	500	1600	800	2	3
Ozone	72	1691	845	2	11
Spambase	57	3067	1534	2	5
Mnist	717	40000	20000	2	5

The distributed approach proposed herein can be used with any feature selection method, although a subset of features is mandatory and so, a threshold is required for ranker methods. In this work, five well-known filters, based on different metrics, were chosen. While three of them return a feature subset (CFS, Consistency-based and INTERACT), the other two (ReliefF and Information Gain) are ranker methods so, as aforementioned, a threshold is necessary in order to obtain a subset of features. In this research we have opted for retaining the c top features, being c the number of features selected by CFS, since it is a widely-used method and, among the three subset methods chosen, it is the one which usually selects the larger number of features. It is also worth noting that although most of the filters work only over nominal features, the discretization step is done by default by Weka [16], working as a black box for the user.

- **Correlation-based Feature Selection** (CFS) is a simple filter algorithm that ranks feature subsets according to a correlation based heuristic evaluation function [17]. Theoretically, irrelevant features should be ignored and redundant features should be screened out.
- The **Consistency-based Filter** [18] evaluates the worth of a subset of features by the level of consistency in the class values when the training instances are projected onto the subset of attributes.
- The **INTERACT** algorithm [19] is based on symmetrical uncertainty (SU). The authors stated that this method can handle feature interaction, and efficiently selects relevant features. The first part of the algorithm requires a threshold, but since the second part searches for the best subset of features, it is considered a subset filter.
- **Information Gain** [20] is one of the most common attribute evaluation methods. This filter provides an ordered ranking of all the features and then a threshold is required.

– **ReliefF** [21] is an extension of the original Relief algorithm that adds the ability of dealing with multiclass problems and is also more robust and capable of dealing with incomplete and noisy data. This method may be applied in all situations, has low bias, includes interaction among features and may capture local dependencies which other methods miss.

3 Experimental Results

In this section we present and discuss the experimental results in terms of (a) the number of selected features; (b) the classification accuracy; and (c) the feature selection runtime. Three different approaches will be compared in the tables of this section: the centralized standard approach (C), the distributed approach presented in our previous work, which distributes the data randomly and merge the partial feature selection results by taking classification accuracy into account (D-Clas) and the distributed approach proposed herein, which distributes the data maintaining the class distribution and merges the partial outputs using data complexity measures (D-Comp). The name of the specific filter used will be added at the beginning. For example, the centralized approach employing CFS will be represented as CFS-C. Finally, the last subsection presents a case study to determine the most suitable interval for the threshold of votes.

3.1 Number of Selected Features

Table 2 shows the number of features selected by each approach. In the first block of the table we can see the features selected by the centralized and distributed "D-Comp" approaches, which are not dependent on the classifier. Then, the table visualizes the number of features selected by the distributed approach "D-Clas" for each classifier, since the stage devoted to finding the threshold of votes makes use of a given learning algorithm. As can be seen, the number of features selected by centralized and distributed approaches is similar, in some cases being even larger in the centralized approach (see Connect4 with INT or Cons). Therefore, we can affirm that applying a distributed approach does not imply a larger selection of features.

3.2 Classification Accuracy Results

This section presents the classification accuracy obtained by C4.5 [22], naive Bayes [23], k-NN [24] and SVM [25] classifiers both with the centralized and distributed approaches (Table 3). The best result for each dataset and classifier is highlighted in bold face, whilst the best result for each dataset (regardless of the classifier employed) is also shadowed. As expected, the results are very variable depending on the dataset and the classifier. For some datasets (Connect4 and Isolet) the highest accuracies are achieved by centralized approaches, although the results obtained by distributed approaches are only inferior in 1 or 2%. For other datasets (Madelon, Spambase and Mnist) the best results are reported

Table 2. Number of features selected by the different approaches tested

		Connect4	Isolet	Madelon	Ozone	Spambase	Mnist
	Full set	42	617	500	72	57	717
	CFS-C	7	186	18	20	19	61
	CFS-D-Comp	8	105	9	8	18	77
	INT-C	36	56	23	16	26	40
	INT-D-Comp	7	62	11	6	15	62
	Cons-C	39	11	22	16	20	18
	Cons-D-Comp	7	31	11	5	12	48
	IG-C	7	186	18	20	19	61
	IG-D-Comp	7	131	10	9	18	67
	ReliefF-C	7	186	18	20	19	61
	ReliefF-D-Comp	9	138	13	17	15	67
C4.5	CFS-D-Clas	9	132	14	14	19	90
C4.5	IG-D-Clas	9	142	15	13	19	79
C4.5	ReliefF-D-Clas	9	146	18	12	17	74
C4.5	INT-D-Clas	9	75	14	12	19	72
C4.5	Cons-D-Clas	9	32	15	6	18	50
NB	CFS-D-Clas	9	132	14	14	20	90
NB	IG-D-Clas	9	142	15	13	19	79
NB	ReliefF-D-Clas	9	146	18	13	19	74
NB	INT-D-Clas	9	75	14	12	19	72
NB	Cons-D-Clas	10	32	15	6	20	50
k-NN	CFS-D-Clas	9	131	14	14	19	90
k-NN	IG-D-Clas	10	142	15	13	19	79
k-NN	ReliefF-D-Clas	11	145	18	12	17	74
k-NN	INT-D-Clas	10	78	14	12	19	72
k-NN	Cons-D-Clas	10	34	15	6	18	50
SVM	CFS-D-Clas	9	137	14	14	19	90
SVM	IG-D-Clas	9	142	15	13	20	79
SVM	ReliefF-D-Clas	9	149	18	12	17	74
SVM	INT-D-Clas	9	70	14	12	19	72
SVM	Cons-D-Clas	9	28	15	6	18	50

by a distributed approach, improving in up to 6% the centralized approach (see Mnist). Finally, for Ozone dataset the three approaches tested obtain the highest accuracy. The important conclusion, however, is that by distributing the data there is not a significant degradation in classification accuracy. In fact, in some cases the accuracy is improved. It is worth mentioning, for example, the case of Isolet, in which Cons-D-Clas and Cons-D-Comp combined with SVM classifier report 68.12% and 60.49% accuracy, respectively, whilst the same filter method in the standard centralized approach degrades its performance until 31.17%, probably due to the small number of features selected by the centralized filter (see Table 2).

3.3 Runtime

Table 4 reports the runtime of the feature selection algorithms, both for centralized and distributed approaches. Notice that in both distributed approaches (D-Clas and D-Comp), the feature selection stage at each subset of data is the same, so the time required will be referred as "D" for both of them. Also, in the distributed approach, considering that all the subsets can be processed at the same time, the time displayed in the table is the maximum of the times required by the filter in each subset generated in the partitioning stage. In these

Table 3. Test classification accuracy. Best results are highlighted.

		Connect4	Isolet	Madelon	Ozone	Spambase	Mnist
C4.5	CFS-C	61.22	81.59	80.50	97.63	81.16	86.99
	CFS-D-Clas	61.25	**82.23**	76.88	95.86	79.27	88.65
	CFS-D-Comp	61.25	81.53	80.62	96.09	79.73	88.55
	INT-C	60.48	78.96	80.63	96.92	78.16	87.24
	INT-D-Clas	61.66	79.03	82.38	94.79	80.83	88.62
	INT-D-Comp	61.25	79.35	80.62	96.92	80.44	88.45
	Cons-C	60.49	56.00	80.63	**98.70**	84.62	87.00
	Cons-D-Clas	61.66	77.10	82.63	96.33	79.34	**90.46**
	Cons-D-Comp	61.25	72.87	80.62	97.75	85.27	89.46
	IG-C	**63.90**	81.40	72.75	98.22	83.83	87.83
	IG-D-Clas	62.34	81.08	79.63	97.87	**85.33**	87.88
	IG-D-Comp	62.27	79.41	80.62	97.87	81.68	87.77
	ReliefF-C	63.49	79.54	73.88	98.11	78.81	87.34
	ReliefF-D-Clas	63.00	80.56	**87.50**	98.46	84.75	87.95
	ReliefF-D-Comp	63.00	81.53	84.12	95.98	84.88	88.06
NB	CFS-C	60.28	75.05	71.75	78.22	57.69	71.88
	CFS-D-Clas	58.83	73.89	70.13	76.69	57.24	73.34
	CFS-D-Comp	58.83	**75.30**	70.50	77.63	58.87	73.49
	INT-C	53.85	71.26	70.00	78.22	57.95	70.94
	INT-D-Clas	59.16	70.75	70.13	75.03	74.77	71.06
	INT-D-Comp	58.83	69.60	70.00	76.21	78.42	71.30
	Cons-C	54.12	42.78	70.00	**98.70**	91.00	72.78
	Cons-D-Clas	59.16	69.92	70.38	73.25	**92.89**	**75.74**
	Cons-D-Comp	58.83	64.91	70.00	98.46	92.05	74.61
	IG-C	60.42	69.34	70.38	74.08	76.53	70.74
	IG-D-Clas	60.28	67.54	70.63	77.63	89.70	68.09
	IG-D-Comp	60.20	66.77	70.50	78.46	66.95	68.07
	ReliefF-C	60.42	62.67	68.63	71.36	41.85	69.82
	ReliefF-D-Clas	**60.50**	56.51	71.50	60.95	91.79	70.93
	ReliefF-D-Comp	**60.50**	53.69	**72.25**	66.86	92.05	70.89
kNN	CFS-C	53.90	56.00	85.63	96.45	79.14	87.93
	CFS-D-Clas	57.61	54.78	65.63	96.57	77.31	91.65
	CFS-D-Comp	53.68	54.65	88.50	94.56	79.92	91.57
	INT-C	58.27	52.92	88.75	94.44	79.73	86.87
	INT-D-Clas	57.61	49.84	71.75	95.27	76.86	91.79
	INT-D-Comp	53.68	50.42	88.75	94.79	76.92	91.72
	Cons-C	58.06	49.90	88.75	**98.70**	**80.83**	87.36
	Cons-D-Clas	57.61	58.31	71.63	95.27	77.38	**96.31**
	Cons-D-Comp	53.68	57.41	88.75	95.50	76.79	95.14
	IG-C	51.29	54.78	74.25	95.98	78.62	89.63
	IG-D-Clas	57.01	59.72	86.13	95.50	78.42	90.77
	IG-D-Comp	54.52	**61.83**	88.50	94.79	77.71	90.97
	ReliefF-C	**61.81**	59.14	75.25	95.98	76.99	89.97
	ReliefF-D-Clas	57.01	57.09	**90.88**	96.80	80.70	91.35
	ReliefF-D-Comp	57.01	55.93	88.38	96.33	80.18	91.77
SVM	CFS-C	**60.42**	83.45	66.50	**98.70**	**85.85**	79.58
	CFS-D-Clas	**60.42**	82.42	67.13	**98.70**	82.27	**81.52**
	CFS-D-Comp	**60.42**	82.30	66.75	**98.70**	85.46	81.49
	INT-C	**60.42**	73.83	66.38	**98.70**	80.31	78.54
	INT-D-Clas	**60.42**	78.00	**68.50**	**98.70**	81.49	80.84
	INT-D-Comp	**60.42**	75.18	66.38	**98.70**	81.10	80.87
	Cons-C	**60.42**	31.17	66.38	**98.70**	81.88	75.14
	Cons-D-Clas	**60.42**	68.12	66.50	**98.70**	81.94	80.85
	Cons-D-Comp	**60.42**	60.49	66.38	**98.70**	81.16	80.52
	IG-C	**60.42**	82.94	67.13	**98.70**	83.83	78.28
	IG-D-Clas	**60.42**	79.67	67.13	**98.70**	83.38	79.30
	IG-D-Comp	**60.42**	80.12	66.75	**98.70**	83.38	79.15
	ReliefF-C	**60.42**	**84.61**	67.50	**98.70**	81.94	75.43
	ReliefF-D-Clas	**60.42**	82.36	67.50	**98.70**	83.57	75.72
	ReliefF-D-Comp	**60.42**	81.98	67.25	**98.70**	83.77	75.86

Table 4. Maximum untime (hh:mm:ss) for the feature selection methods tested

	Connect4	Isolet	Madelon	Ozone	Spambase	Mnist
CFS-C	00:01:40	00:04:10	00:00:36	00:00:10	00:00:12	00:29:47
CFS-D	00:00:10	00:01:17	00:00:25	00:00:08	00:00:06	00:04:17
IG-C	00:01:37	00:02:51	00:00:41	00:00:09	00:00:11	00:24:11
IG-D	00:00:04	00:00:54	00:00:29	00:00:09	00:00:05	00:03:55
ReliefF-C	00:28:00	00:09:13	00:01:02	00:00:14	00:00:21	08:26:53
ReliefF-D	00:00:11	00:01:43	00:00:40	00:00:08	00:00:04	00:22:26
INT-C	00:01:52	00:03:16	00:00:40	00:00:09	00:00:13	00:52:25
INT-D	00:00:11	00:01:10	00:00:31	00:00:08	00:00:04	00:03:19
Cons-C	00:06:08	00:04:05	00:00:52	00:00:11	00:00:14	01:42:43
Cons-D	00:00:10	00:01:20	00:00:25	00:00:06	00:00:02	00:03:17

Table 5. Average runtime (hh:mm:ss) for obtaining the threshold of votes

Method	D-Clas-C4.5	D-Clas-NB	D-Clas-kNN	D-Clas-SVM	D-Comp
CFS	00:00:36	00:00:26	00:00:48	00:01:36	00:00:01
INT	00:00:31	00:00:24	00:00:50	00:01:23	00:00:01
Cons	00:00:29	00:00:23	00:00:46	00:01:41	00:00:01
IG	00:00:38	00:00:28	00:00:46	00:01:43	00:00:01
ReliefF	00:00:33	00:00:26	00:00:41	00:02:02	00:00:01

experiments, all the subsets were processed in the same machine, but the proposed algorithm could be executed in multiple processors. Please note that this filtering time is independent of the classifier chosen. The lowest time for each dataset is shadowed.

As expected, the advantage of the distributed approaches in terms of execution time over the standard method is significant. The time is reduced for all datasets and filters, except for Ozone with the IG filter, in which it is maintained. It is worth mentioning the important reductions when the dimensionality of the dataset grows. For Mnist dataset, which has 717 features and 40000 training samples, the reduction is more than notable. In fact, for ReliefF filter, the processing time is reduced from more than 8 hours to 22 minutes, proving the adequacy of the distributed approach when dealing with large datasets.

For the distributed approaches, it is necessary to take into account the time required to calculate the threshold to build the final subset of features. Since the distributed approach "D-Clas" makes uses of a classifier to establish the threshold, the time required by this approach depends highly on the classifier, whilst with the distributed approach "D-Comp" this time is independent of the classifier. Therefore, in Table 5, we can see the average runtime on all datasets for each filter and distributed approach. It is easy to note that the time required to find the threshold in the proposed distributed approach "D-Comp" is notable lower than the one in our previous approach "D-Clas", especially with the SVM

classifier, in which in some cases the reduction goes from 2 minutes to 1 second. In light of these results, we can conclude that the distributed approaches performed successfully, since the running time was considerably reduced and the accuracy did not drop to inadmissible values. In fact, our approach is able to match and in some cases even improve the standard algorithms applied to the non-partitioned datasets. Moreover, it has been demonstrated that the distributed approach proposed in this paper, "D-Comp", outperforms our previous proposal "D-Clas", since the time for obtaining the threshold of votes was also reduced and the performance results are similar.

3.4 Case Study: Determining the Optimal Interval of Votes

Bearing in mind that with our proposed approach based on complexity measures the time required to find the threshold of votes has been significantly shortened,

Table 6. Test classification accuracy with different intervals of votes. Best results are highlighted.

		Connect4	Isolet	Madelon	Ozone	Spambase	Mnist
C4.5	CFS-D-Comp	61.25	**81.53**	80.62	96.09	79.73	88.55
	CFS-D-Comp+	61.20	79.35	79.62	97.63	78.68	88.15
	INT-D-Comp	61.25	79.35	80.62	96.92	80.44	88.45
	INT-D-Comp+	61.16	77.42	80.62	97.75	79.47	88.52
	Cons-D-Comp	61.25	72.87	80.62	97.75	85.27	**89.46**
	Cons-D-Comp+	61.16	73.06	80.62	97.75	83.90	89.06
	IG-D-Comp	62.27	79.41	80.62	**97.87**	81.68	87.77
	IG-D-Comp+	61.16	77.55	80.62	97.40	**85.40**	87.61
	ReliefF-D-Comp	63.00	**81.53**	**84.12**	95.98	84.88	88.06
	ReliefF-D-Comp+	**63.70**	80.37	77.50	97.28	81.75	87.39
NB	CFS-D-Comp	58.83	**75.30**	70.50	77.63	58.87	73.49
	CFS-D-Comp+	60.28	75.05	70.62	80.83	52.74	72.77
	INT-D-Comp	58.83	69.60	70.00	76.21	78.42	71.30
	INT-D-Comp+	60.28	68.63	70.00	84.26	88.98	71.19
	Cons-D-Comp	58.83	64.91	70.00	**98.46**	**92.05**	**74.61**
	Cons-D-Comp+	60.28	61.96	70.00	**98.46**	79.79	73.33
	IG-D-Comp	60.20	66.77	70.50	78.46	66.95	68.07
	IG-D-Comp+	60.28	65.55	70.50	79.17	90.35	69.20
	ReliefF-D-Comp	**60.50**	53.69	**72.25**	66.86	**92.05**	70.89
	ReliefF-D-Comp+	60.44	50.99	70.62	60.71	87.94	70.60
kNN	CFS-D-Comp	53.68	54.65	88.50	94.56	79.92	91.57
	CFS-D-Comp+	51.48	52.02	86.12	95.03	78.10	89.77
	INT-D-Comp	53.68	50.42	**88.75**	94.79	76.92	91.72
	INT-D-Comp+	50.38	54.14	**88.75**	95.62	77.25	91.26
	Cons-D-Comp	53.68	57.41	**88.75**	95.50	76.79	**95.14**
	Cons-D2	50.38	61.90	**88.75**	95.50	76.27	93.73
	IG-D-Comp	54.52	61.83	88.50	94.79	77.71	90.97
	IG-D2	50.38	**61.96**	88.50	95.27	78.03	90.23
	ReliefF-D-Comp	**57.01**	55.93	88.38	**96.33**	**80.18**	91.77
	ReliefF-D-Comp+	56.91	54.91	84.25	96.09	77.38	90.42
SVM	CFS-D-Comp	**60.42**	**82.30**	66.75	**98.70**	85.46	**81.49**
	CFS-D-Comp+	**60.42**	79.09	67.12	**98.70**	**85.85**	80.91
	INT-D-Comp	**60.42**	75.18	66.38	**98.70**	81.10	80.87
	INT-D-Comp+	**60.42**	74.15	66.38	**98.70**	82.46	80.30
	Cons-D-Comp	**60.42**	60.49	66.38	**98.70**	81.16	80.52
	Cons-D-Comp+	**60.42**	54.71	66.38	**98.70**	76.99	79.49
	IG-D-Comp	**60.42**	80.12	66.75	**98.70**	83.38	79.15
	IG-D-Comp+	**60.42**	78.64	66.75	**98.70**	83.51	78.78
	ReliefF-D-Comp	**60.42**	81.98	67.25	**98.70**	83.77	75.86
	ReliefF-D-Comp+	**60.42**	79.73	**67.88**	**98.70**	80.70	75.42

in this case study we would like to analyze if it is possible to increase the interval of possible number of votes. In Section 2.1 we have explained that this interval was set to $[avg \pm 1/2std]$, trying to avoid a high number of calculations which could lead to unaffordable computing times. However, since we have seen that this time is not prohibitive anymore, we performed some experiments setting this interval to $[avg \pm std]$. In Table 6 we can see the accuracy results comparing our distributed approach "D-Comp" and this new proposal that we have called "D-Comp+". If we compare both approaches for each combination of dataset, filter and classifier, it turns out that "D-Comp" outperforms or matches "D-Comp+" in 89 out of 120 cases. Regarding the number of selected features, "D-Comp" selects in almost all cases a slightly larger number of features than "D-Comp+", which apparently leads to better performances. In terms of running time, the computation of the optimal threshold does not take more than 1 second in any case, regardless of the interval chosen. Therefore, we can conclude that, although the low computational times required for finding the threshold would allow us to try a larger number of possible votes, it is better to maintain our original proposed approach in which the interval was set to $[avg \pm 1/2std]$. Moreover, if in the future we need to deal with datasets with millions of data, it is better to reduce the computation as much as possible.

4 Conclusions

Feature selection is usually applied in a centralized manner. However, if the data are distributed, feature selection may take advantage of processing multiple subsets in sequence or concurrently. The need to use distributed feature selection can be two-fold. On the one hand, with the advent of network technologies, the data are sometimes distributed in multiple locations and often with multiple parties. On the other hand, most existing feature selection algorithms do no scale well and their efficiency significantly deteriorates when dealing with large-scale data.

In this paper we propose a methodology for distributing the process of feature selection by tackling the most common distribution in the literature: the horizontal partition. A previous proposal was able to confront this problem, but presented certain drawbacks, mainly (1) the partitioning of the data did not take into account the class distribution, a fact which might lead to partitions in which a certain class is not represented and (2) the method has shown to be dependent on the classifier and time-consuming in the process of merging the partial results from the different partitions.

In this new proposal, we aimed at achieving a method able to overcome these drawbacks, especially in terms of running time, since in high dimensional datasets this will be a core issue. For this sake, we modify our previous methodology so that the partitions of the dataset maintain the original class distribution. Then, we propose a new methodology for combining the partial results from different partitions which makes use of data complexity measures. The rationale behind this was that features that contribute to decrease the complexity

of a dataset must be maintained, whereas those that contribute to increase the complexity must be discarded. By using this new methodology, we were able to reduce significantly the running time while maintaining the classification performance. Moreover, our new approach is independent on the subsequent classifier. Finally, it is worth mentioning that the proposed method can be used with any feature selection algorithm without any modifications, so it could be seen as a general framework for distributed feature selection.

As future work, we plan to test the scalability properties of the proposed method with datasets larger than 100 000 samples. Moreover, it would be interesting to perform a sensitivity analysis on the value chosen for α. Finally, another line of future research would be trying other complexity measures.

Acknowledgments. This research has been economically supported in part by the Ministerio de Economía y Competitividad of the Spanish Government through the research project TIN 2012-37954, partially funded by FEDER funds of the European Union; and by the Consellería de Industria of the Xunta de Galicia through the research project GRC2014/035.

References

1. Bache, K., Lichman, M.: UCI machine learning repository (2013). http://archive.ics.uci.edu/ml (accessed January 2015)
2. Zhao, Z.A., Liu, H.: Spectral feature selection for data mining. Chapman & Hall/CRC (2011)
3. Guyon, I.: Feature extraction: foundations and applications, vol. 207. Springer, Heidelberg (2006)
4. Saeys, Y., Inza, I., Larrañaga, P.: A review of feature selection techniques in bioinformatics. Bioinformatics **23**(19), 2507–2517 (2007)
5. Chan, P.K., Stolfo, S.J.: Toward parallel and distributed learning by meta-learning. In: AAAI workshop in Knowledge Discovery in Databases, pp. 227–240 (1993)
6. Ananthanarayana, V.S., Subramanian, D.K., Murty, M.N.: Scalable, distributed and dynamic mining of association rules. In: Prasanna, V.K., Vajapeyam, S., Valero, M. (eds.) HiPC 2000. LNCS, vol. 1970, pp. 559–566. Springer, Heidelberg (2000)
7. Tsoumakas, G., Vlahavas, I.: Distributed data mining of large classifier ensembles. In: Proceedings Companion Volume of the Second Hellenic Conference on Artificial Intelligence, pp. 249–256 (2002)
8. Das, K., Bhaduri, K., Kargupta, H.: A local asynchronous distributed privacy preserving feature selection algorithm for large peer-to-peer networks. Knowledge and information systems **24**(3), 341–367 (2010)
9. McConnell, S., Skillicorn, D.B.: Building predictors from vertically distributed data. In: Proceedings of the 2004 Conference of the Centre for Advanced Studies on Collaborative Research, pp. 150–162. IBM Press (2004)
10. Skillicorn, D.B., McConnell, S.M.: Distributed prediction from vertically partitioned data. Journal of Parallel and Distributed computing **68**(1), 16–36 (2008)
11. Rokach, L.: Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography. Computational Statistics & Data Analysis **53**(12), 4046–4072 (2009)

12. Banerjee, M., Chakravarty, S.: Privacy preserving feature selection for distributed data using virtual dimension. In: Proceedings of the 20th ACM international conference on Information and knowledge management, pp. 2281–2284. ACM (2011)
13. Bolón-Canedo, V., Sánchez-Maroño, N., Cerviño-Rabuñal, J.: Scaling up feature selection: a distributed filter approach. In: Bielza, C., Salmerón, A., Alonso-Betanzos, A., Hidalgo, J.I., Martínez, L., Troncoso, A., Corchado, E., Corchado, J.M. (eds.) CAEPIA 2013. LNCS, vol. 8109, pp. 121–130. Springer, Heidelberg (2013)
14. de Haro García, A.: Scaling data mining algorithms. Application to instance and feature selection. Ph.D. thesis, Universidad de Granada (2011)
15. Basu, M., Ho, T.K.: Data complexity in pattern recognition. Springer (2006)
16. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The Weka data mining software: an update. ACM SIGKDD Explorations Newsletter **11**(1), 10–18 (2009)
17. Hall, M.A.: Correlation-based feature selection for machine learning. Ph.D. thesis, The University of Waikato (1999)
18. Dash, M., Liu, H.: Consistency-based search in feature selection. Artificial intelligence **151**(1), 155–176 (2003)
19. Zhao, Z., Liu, H.: Searching for interacting features. In: IJCAI, vol. 7, pp. 1156–1161 (2007)
20. Hall, M.A., Smith, L.A.: Practical feature subset selection for machine learning. Computer Science **98**, 181–191 (1998)
21. Kononenko, I.: Estimating attributes: analysis and extensions of relief. In: Bergadano, F., De Raedt, L. (eds.) ECML 1994. LNCS, vol. 784, pp. 171–182. Springer, Heidelberg (1994)
22. Quinlan, J.R.: C4. 5: programs for machine learning. Morgan kaufmann (1993)
23. Rish, I.: An empirical study of the naive bayes classifier. In: IJCAI 2001 workshop on empirical methods in artificial intelligence, vol. 3, pp. 41–46 (2001)
24. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Machine learning **6**(1), 37–66 (1991)
25. Vapnik, V.N.: Statistical learning theory. Wiley (1998)