

A Time Efficient Approach for Distributed Feature Selection Partitioning by Features

L. Morán-Fernández^(✉), V. Bolón-Canedo, and A. Alonso-Betanzos

Laboratory for Research and Development in Artificial Intelligence (LIDIA),
Computer Science Department, University of A Coruña, 15071 A Coruña, Spain
`laura.moranf@udc.es`

Abstract. With the advent of high dimensionality, feature selection has become indispensable in real-world scenarios. However, most of the traditional methods only work in a centralized manner, which —ironically— increase the running time requirements when they are applied to this type of data. For this reason, we propose a distributed filter approach for vertically partitioned data. The idea is to split the data by features and apply a filter at each partition performing several rounds to obtain a final subset of features. Different than existing procedures to combine the partial outputs of the different partitions of data, we propose a merging process according to the theoretical complexity of these feature subsets instead of classification error. Experimental results tested in five datasets show that the running time decreases considerably. Moreover, regarding the classification accuracy, our approach was able to match, and in some cases even improve, the standard algorithms applied to the non-partitioned datasets.

1 Introduction

In the last few years, there has been an increase in the size of datasets in all fields of application. The advent of this type of high dimensional datasets has posed a big challenge for machine learning researchers, since it is difficult to deal with a high number of input features due to the curse of dimensionality [14]. The scaling up problem appears in any algorithm when the data size increases beyond the capacity of the traditional data mining algorithms, damaging their classification performance and efficiency. This problem can affect negatively in some other aspects such as excessive storage requirements, increase of time complexity and, finally, it affects generalization accuracy, introducing noise and overfitting. To confront the problem of the high number of features it is natural —and perhaps essential— to investigate the effects of the application of feature selection. Feature selection methods have received an important amount of attention in the classification literature, where three kinds of algorithms have generally been studied: *filter*, *wrapper* and *embedded* methods. The main difference between the first two is that a wrapper makes use of the algorithm that will be employed to build the final classifier, while a filter method does not. The embedded methods generally use classification learning models, and then an optimal subset of

features is built by the classification algorithm. So, the interaction with the classifier required by wrapper and embedded methods comes with an important computational burden.

The use of an adequate feature selection method can lead to an improvement of the inductive learner, either in terms of learning speed, generalization capacity or simplicity of the induced model. However, we will have to deal with a scalability problem if we apply these techniques to large datasets. The advantages of feature selection come at a certain price, as the search for a subset of relevant features introduces an extra layer of complexity to the modeling task. This new layer increases the memory and running time requirements, making these algorithms very inefficient when applied to problems that involve very large datasets. Ironically, standard feature selection becomes impracticable on large datasets, which are the ones that would benefit most from its application.

Trying to overcome the drawbacks mentioned above, over the last years many distributed methods have been developed instead of the traditional centralized approaches. The first reason is that, with the advent of network technologies, data is being increasingly available already distributed in multiple locations, and it is not economic or legal to gather it in a single location. And, second, when dealing with large amounts of data, most existing feature selection 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 feature selection method on each subset of data and then combine the results. There are two main techniques for partitioning and distributing data: by samples (horizontally), and by features (vertically). Most of the distributed feature selection methods have been used to scale up datasets that are too large for batch learning in terms of samples [2,6,7,18]. While not so common, there are some other developments which distribute the data by features [16,17].

In our previous work [5], we presented a methodology for distributing the data vertically which combined the partial feature subsets based on improvements in the classification accuracy. Although the experiments showed that the execution time was considerably shortened whereas the performance was maintained or even improved compared to the standard algorithms applied to the non-partitioned datasets, the drawback of this methodology was that it was dependent on the classifier. In order to overcome this issue, we proposed in [4] a new framework for distributing the feature selection process by samples which performed a merging procedure to update the final feature subset according to the theoretical complexity of these features, by using data complexity measures [13] instead of the classification error. In this way, we provided a framework for distributed feature selection which not only was independent of the classifier, but also reduced drastically the computational time needed by the algorithm, thus paving the way for its application in high dimensional datasets.

To examine the research problem in detail, this paper will be focused on the vertically distributed approach making use of data complexity measures. The experimental results from five different datasets demonstrate that our new proposal shows important savings in running times, as well as matching—and in some cases even improving—the classification accuracy.

2 Distributed Feature Selection

The methodology that we propose in this work consists in a distributed framework for feature selection by partitioning the data vertically, i.e., by features. Basically, the distributed algorithm adopts the following 3-step experimental framework: (1) partition of the training datasets, (2) application of the distributed algorithm to the subsets of features in several “rounds” and (3) combination of the results into a single subset of features. The repetition of the process in several rounds ensures that we have gathered enough information for the combination step to be useful, since each feature in the vector of votes can only take values 0 or 1 for each round, so we would not have enough data to decide which features form the final subset. The algorithm for the whole methodology is detailed in Algorithm 1.

For each round, we start by dividing the data D without replacement — as usually happens in real world when different features are collected on different locations — by randomly assigning groups of t features to each partition. Then, the chosen filter is applied to each partition separately and the features selected to be removed receive a vote. At this point, a new round is performed leading to a new partition of the dataset and another iteration of voting is accomplished until reaching the predefined number of rounds. Finally, the features that have received a number of votes above a certain threshold are removed. Notice that the maximum number of votes is the number of rounds r .

In order to calculate the threshold of votes to employ, an automatic method was used. In [9], the authors recommend that the selection of the number of votes must take into account two different aspects: the training classification error and the percentage of features retained. Both values must be minimized to the extent possible, by minimizing the fitness criterion $e[v]$:

$$e[v] \leftarrow \alpha \times error + (1 - \alpha) \times featPercentage, \quad (1)$$

where α is a parameter with value in the interval $[0, 1]$. It measures the relative relevance of *error* and *featPercentage* values. Different values can be used if the researcher is more interested in reducing the storage requirements or the error. For this work, it was set to $\alpha = 0.75$, giving more influence to the classification error. At the end, the features with the number of votes above the obtained threshold are removed from the final subset of features S . This subset will be used in the training and test sets in order to obtain the final classification accuracy.

The problem with this approach is that, by involving a classifier in the process of selecting the optimal threshold, it makes our methodology dependent on the classifier chosen. Moreover, 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. Trying to overcome these issues, we propose to modify the function for calculating the threshold of votes by making use of data complexity measures. These measures are a recent proposal to represent characteristics of the data

which are considered difficult in classification tasks beyond estimates of error rates [13]. The rationale for this decision is that we assume that good candidate features would contribute to decrease the complexity and must be maintained. Since our intention is to propose a framework that could be independent of the classifier and applicable to both binary and multiclass datasets, among the existing complexity measures, the Fisher discriminant ratio was chosen. Fisher’s multiple discriminant ratio for C classes is defined as:

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

where μ_i, σ_i^2 , and p_i are the mean, variance, and proportion of the i th class, respectively. In this work we will use the inverse of the Fisher ratio, $1/f$, where a small complexity value represents an easier problem. Therefore, the new formula for calculating $e[v]$ is defined as:

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

Thus, the fundamental goals that we expect to achieve with the new formulation of Eq. (1) are a reduction in time to calculate the threshold and, also, a classifier-independent method.

3 Materials and Methods

In order to examine the effect of the proposed distributed framework for feature selection by partitioning the data vertically, we use five benchmark datasets, which are described in Table 1 depicting their main characteristics (number of features, number of training and test samples and number of classes). In this work the common partition 2/3 for training and 1/3 for testing was used.

Table 1. Summary of datasets’ main characteristics

Dataset	Features	Samples		Classes	Download
		Training	Test		
Isolet	617	6238	1236	26	[3]
Madelon	500	1600	800	2	[3]
Mnist	717	40000	20000	2	[3]
Breast	24481	78	19	2	[1]
Lung	12600	136	68	5	[19]

It must be noted that the proposed framework explained above can be used with any feature selection method, although the use of filters is recommended since they are faster than other methods. In this work, five different filters were involved, all of them are implemented in the Weka [10] environment:

Algorithm 1. Pseudo-code for the proposed distributed methodology

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

$X \leftarrow$ set of features, $X = \{x_1, \dots, x_n\}$
 $s \leftarrow$ number of submatrices of \mathbf{D} with m samples and t features
 $V \leftarrow$ vector of votes
 $r \leftarrow$ number of rounds (5 in this experimentation)
 $\alpha \leftarrow 0.75$

Result: $S \leftarrow$ subset of features $\setminus S \subset X$

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

1. initialize the vector of votes V to 0, $|V|=n$
 2. **for** each round **do**
 3. Split \mathbf{D} into s disjoint submatrices with m samples and t features
 4. **for** each submatrix **do**
 5. apply a feature selection algorithm
 6. $F \leftarrow$ features selected by the algorithm
 7. $E \leftarrow$ features eliminated by the algorithm $\setminus E \cup F = X$
 8. increment one vote in vector V for each feature in E
 9. **end**
 10. **end**
- /** Obtaining a threshold of votes, Th , to remove a feature */
11. $minVote \leftarrow$ minimum threshold considered (1)
 12. $maxVote \leftarrow$ maximum threshold considered (number of rounds, r)
 13. **for** $v \leftarrow minVote$ to $maxVote$ with increment 1 **do**
 14. $F_{th} \leftarrow$ subset of selected features (number of votes $< v$)
 15. $1/f \leftarrow$ inverse of Fisher ratio computed on training dataset \mathbf{D} using only features in F_{th}
 16. $featPercentage \leftarrow$ percentage of features retained $\left(\frac{|F_{th}|}{|X|} \times 100 \right)$
 17. $e[v] \leftarrow \alpha \times 1/f + (1-\alpha) \times featPercentage$
 18. **end**
 19. $Th \leftarrow min(e)$, Th is the value which minimizes the function e
 20. $S \leftarrow$ subset of features after removing from X all features with a number of votes $\geq Th$
-

- **Correlation-based Feature Selection** (CFS) is a simple multivariate filter algorithm which measures the goodness of feature subsets according to the usefulness of individual features for predicting the class label along with the level of intercorrelation among them [11].
- The **Consistency-based Filter** (CONS) evaluates the worth of a subset of features by the level of consistency in the class values when the training samples are projected onto the subset of features [8].
- The **INTERACT** algorithm [20] is based on *symmetrical uncertainty* (SU). It consists of two major parts. In the first part, the features are ranked in

descending order based on their SU values. In the second part, features are evaluated one by one starting from the end of the ranked feature list. The authors stated that this method can effectively reduce the number of features, and maintain or improve predictive accuracy in dealing with interacting features.

- **Information Gain** [12] is one of the most common attribute evaluation methods. The univariate filter provides an ordered ranking of all the features which requires a threshold.
- **ReliefF** [15] 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.

While the first three methods return a feature subset, the other two are ranker methods, so we need to establish the number of selected features. In this paper we have opted for retaining the p top features, being p 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 largest number of features. To test the adequacy of our proposal, four different classifiers, belonging to different families, were selected: C4.5, naive Bayes, k NN and SVM. Notice that two of them are linear (naive Bayes and SVM using a linear kernel) whilst the other two are non-linear (C4.5 and k NN).

4 Experiments and Analysis

This section presents the results over the datasets described in Table 1, comparing three different approaches: the centralized standard approach (C), the distributed approach which merges the partial subsets of features by taking classification error into account (“D-Clas”) and, last, the distributed approach proposed in this work, which combines the partial outputs using data complexity measures (“D-Comp”). We will discuss the experimental results in terms of number of selected features, classification accuracy and runtime. In the case of the first three datasets (Isolet, Madelon and Mnist), we have opted for dividing the datasets in 5 packets, so that each packet will contain 20% of features, without replacement. For the microarray datasets (Breast and Lung), the data is split by assigning groups of t features to each subset, where the number of features t in each subset is half the number of the samples, to avoid overfitting. In this manner, the considered datasets will have enough features to lead to a correct learning.

Figure 1(a) displays the average number of features selected by the five filters for the three different approaches. The full number of features for each dataset is shown in parenthesis. As we can see, the number of features selected by distributed methods is larger than those selected by the centralized approaches. This is caused by the fact that, with the vertical partition, the features are distributed

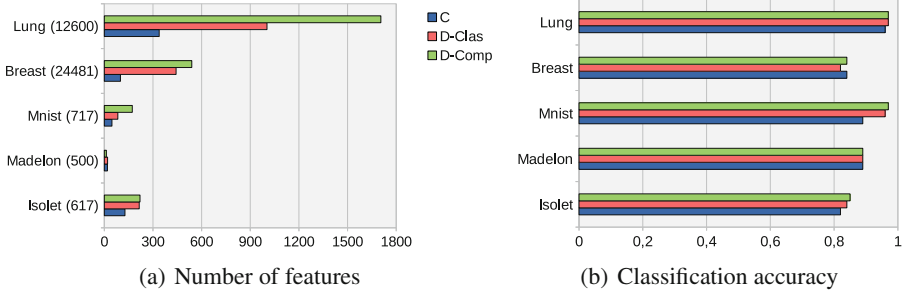


Fig. 1. Number of features and classification accuracy for the different approaches

across the packets and it is more difficult to detect redundancy between features if they are in different partitions.

In terms of classification accuracy, Table 2 shows the best result for each dataset and classifier in bold face, whilst the best result for each dataset and approach is presented in Fig. 1(b). As we can see, the best results were obtained by our distributed approach (“D-Comp”) for Isolet and Mnist, whilst for Madelon, Breast and Lung it matches the best classification accuracy achieved by the other approaches. However, the important conclusion is that by distributing the data there is not a significant degradation in classification accuracy. It is worth mentioning the case of Mnist dataset, in which the test classification accuracy improves from 0.89 (centralized approach combined with *k*NN classifier and IG filter) to 0.97 (“D-Comp” approach with *k*NN classifier for both INT and CONS filters). This is probably due to the higher number of features selected by the filters in the “D-Comp” approach, which turn out to be more appropriate (see Fig. 1(a)).

Finally, Table 3 shows the runtime required by the feature selection methods for both centralized and distributed approaches. Notice that in the two distributed approaches (“D-Clas” and “D-Comp”), the feature selection stage at each packet 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 experiments, all the subsets were processed in the same machine, but the proposed algorithm could be executed in multiple processors. The lowest time for each dataset is highlighted in bold face.

As expected, the time required by the distributed methods is drastically reduced for all datasets and filters if compared with that of the centralized approach, in some cases from 9434s to 0.29 (Lung dataset with CFS filter). Notice that, the larger the dataset, the larger the reduction in time.

Moreover, for the distributed approaches, it is necessary to take into account the time required to calculate the threshold to combine the partial outputs of features. As was mentioned in Sect. 2, the distributed approach “D-Clas” makes

Table 2. Test classification accuracy

		Isolet	Madelon	Mnist	Breast	Lung			Isolet	Madelon	Mnist	Breast	Lung		
C4.5	CFS	C	0.81	0.75	0.85	0.68	0.78	kNN	C	0.56	0.69	0.86	0.63	0.96	
		D-Clas	0.81	0.75	0.86	0.59	0.81		D-Clas	0.58	0.72	0.91	0.57	0.96	
		D-Comp	0.83	0.80	0.89	0.74	0.84		D-Comp	0.57	0.76	0.94	0.79	0.96	
	INT	C	0.77	0.78	0.85	0.79	0.78		C	0.47	0.73	0.85	0.53	0.94	
		D-Clas	0.78	0.78	0.87	0.64	0.86		INT	D-Clas	0.53	0.73	0.85	0.57	0.96
		D-Comp	0.80	0.79	0.91	0.74	0.75		D-Comp	0.45	0.78	0.97	0.86	0.96	
	Cons	C	0.54	0.78	0.85	0.68	0.84		C	0.40	0.70	0.72	0.47	0.84	
		D-Clas	0.75	0.77	0.85	0.65	0.81		Cons	D-Clas	0.57	0.74	0.96	0.59	0.93
		D-Comp	0.76	0.81	0.92	0.79	0.76		D-Comp	0.41	0.89	0.97	0.74	0.94	
	IG	C	0.79	0.80	0.87	0.53	0.87		C	0.54	0.78	0.89	0.68	0.96	
		D-Clas	0.80	0.78	0.89	0.75	0.82		IG	D-Clas	0.56	0.79	0.94	0.65	0.95
		D-Comp	0.82	0.72	0.88	0.74	0.82		D-Comp	0.53	0.66	0.94	0.79	0.97	
Relieff	C	0.79	0.83	0.86	0.74	0.88	C	0.57	0.89	0.89	0.74	0.96			
	D-Clas	0.80	0.81	0.89	0.64	0.88	Relieff	D-Clas	0.59	0.89	0.96	0.74	0.94		
	D-Comp	0.81	0.76	0.90	0.54	0.82	D-Comp	0.58	0.74	0.95	0.74	0.94			
NB	CFS	C	0.72	0.70	0.72	0.37	0.93	SVM	C	0.82	0.65	0.80	0.68	0.94	
		D-Clas	0.74	0.70	0.73	0.49	0.95		D-Clas	0.84	0.65	0.82	0.67	0.96	
		D-Comp	0.74	0.72	0.74	0.37	0.96		D-Comp	0.85	0.66	0.83	0.79	0.99	
	INT	C	0.67	0.70	0.70	0.37	0.94		C	0.73	0.65	0.78	0.74	0.94	
		D-Clas	0.65	0.69	0.76	0.52	0.94		INT	D-Clas	0.80	0.65	0.78	0.67	0.94
		D-Comp	0.74	0.70	0.75	0.37	0.94		D-Comp	0.79	0.66	0.85	0.74	0.94	
	Cons	C	0.40	0.70	0.72	0.37	0.85		C	0.30	0.65	0.74	0.32	0.79	
		D-Clas	0.60	0.70	0.74	0.46	0.88		Cons	D-Clas	0.76	0.66	0.80	0.61	0.92
		D-Comp	0.64	0.70	0.73	0.37	0.91		D-Comp	0.69	0.66	0.86	0.79	0.97	
	IG	C	0.70	0.70	0.69	0.37	0.88		C	0.81	0.66	0.79	0.79	0.96	
		D-Clas	0.71	0.70	0.71	0.39	0.88		IG	D-Clas	0.83	0.65	0.80	0.73	0.96
		D-Comp	0.70	0.70	0.73	0.37	0.91		D-Comp	0.83	0.68	0.80	0.79	0.97	
Relieff	C	0.65	0.70	0.70	0.84	0.88	C	0.82	0.66	0.75	0.63	0.96			
	D-Clas	0.70	0.70	0.70	0.82	0.92	Relieff	D-Clas	0.84	0.66	0.77	0.70	0.97		
	D-Comp	0.69	0.67	0.60	0.84	0.91	D-Comp	0.85	0.68	0.80	0.68	0.97			

Table 3. Runtime (seconds) for the feature selection methods tested

		Isolet	Madelon	Mnist	Breast	Lung
CFS	C	250	36	1787	7969	9434
	D	40	18	287	0.47	0.29
INT	C	196	40	3145	179	135
	D	46	16	225	0.68	1.12
Cons	C	368	52	6163	14.35	8.11
	D	58	16	225	0.33	0.45
IG	C	171	41	1451	1.88	1.99
	D	42	15	273	0.61	0.24
Relieff	C	533	62	30413	3.95	5.05
	D	190	26	5522	0.70	0.26

use of a classifier to establish this threshold. Therefore, the time required by this method depends highly on the classifier, whilst with the distributed approach “D-Comp” the time is independent of the classifier. Table 4 depicts the average runtime on all datasets for each filter and distributed approach, as well as the speed up values, which indicate the performance improvement of the approach

Table 4. Runtime (seconds) for obtaining the threshold of votes

Method	D-Clas					D-Comp	SpeedUp
	C4.5	NB	kNN	SVM	Average		
CFS	149.67	85.2	171.3	641.41	261.90	2.33	112.41
INT	92.33	67.66	153.47	607.52	230.25	1.51	152.48
Cons	82.25	59.10	96.77	625.45	215.89	0.97	222.57
IG	101.10	76.23	132.54	576.87	222.44	2.19	101.59
ReliefF	114.20	101.45	149.76	989.75	338.79	2.27	149.25

“D-Comp” with respect to the average time of the four approaches “D-Clas”. It is easy to notice that the time required to find the threshold in the proposed distributed approach “D-Comp” is notable lower than the approach “D-Clas”, specially with the ReliefF filter, in which in some cases the time was reduced from 989.75 s to only 2.27.

5 Final Remarks

This paper presents a new methodology for distributed feature selection over vertically partitioned data. Typical procedures make use of classification algorithms to combine the partial outputs obtained from each partition. However, these approaches present two important drawbacks: (1) the method is dependent on the classifier and (2) it is necessary to add an extra running time in the process of merging the partial results from the different partitions. Trying to overcome these issues, we have designed a new approach based on a complexity data measure which is independent of the classifier and does not consume so much time in the merging stage.

In light of the results obtained using five datasets, we can conclude that the distributed approach proposed in this work (“D-Comp”) performs successfully, since the classification accuracy matches and, in some cases, is even better than the use of the original feature selection algorithms over the whole datasets. In terms of execution time, we are able to reduce it significantly with respect to the distributed approach “D-Clas”, being this fact (together with the independence from any classifier) the most important advantage of our method.

As future research, we plan to test the scalability properties of the proposed method by increasing the size of the datasets and modifying the number of packets. Moreover, another interesting line of research might be to deal with disjoint partitions, as well as the use of other complexity measures.

Acknowledgements. 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. V. Bolón-Canedo acknowledges support of the Xunta de Galicia under postdoctoral Grant code ED481B 2014/164-0.

References

1. Technology agency for science and research. Kent ridge bio-medical dataset repository. <http://datam.i2r.a-star.edu.sg/datasets/krbd/>
2. 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, p. 559. Springer, Heidelberg (2000)
3. Bache, K., Lichman, M.: UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences (2013). <http://archive.ics.uci.edu/ml/>. Accessed January 2015
4. Bolón-Canedo, V., Sánchez-Maróño, N., Alonso-Betanzos, A.: A distributed feature selection approach based on a complexity measure. In: International Work Conference on Artificial Neural Networks (2015, in press)
5. Bolón-Canedo, V., Sánchez-Maróño, N., CerViño-Rabuñal, J.: Toward parallel feature selection from vertically partitioned data. In: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (2014)
6. 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)
7. Das, K., Bhaduri, K., Kargupta, H.: A local asynchronous distributed privacy preserving feature selection algorithm for large peer-to-peer networks. *Knowl. Inf. Syst.* **24**(3), 341–367 (2010)
8. Dash, M., Liu, H.: Consistency-based search in feature selection. *Artif. Intel.* **151**(1), 155–176 (2003)
9. de Haro García, A.: Scaling data mining algorithms. Application to instance and feature selection. Ph.D. thesis, Universidad de Granada (2011)
10. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.: The weka data mining software: an update. *ACM SIGKDD Explor. Newsl.* **11**(1), 10–18 (2009)
11. Hall, M.A.: Correlation-based feature selection for machine learning. Ph.D. thesis, The University of Waikato (1999)
12. Hall, M.A., Smith, L.A.: Practical feature subset selection for machine learning. *Comput. Sci.* **98**, 181–191 (1998)
13. Ho, T.K., Basu, M.: *Data Complexity in Pattern Recognition*. Springer, London (2006)
14. Jain, A., Zongker, D.: Feature selection: evaluation, application, and small sample performance. *IEEE Trans. Patter Anal. Mach. Intel.* **19**(2), 153–158 (1997)
15. 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)
16. McConnell, S., Skillicorn, D.B.: Building predictors from vertically partitioned data. In: *Proceedings of the 2004 Conference of the Centre for Advanced Studies on Collaborative Research*, pp. 150–162. IBM Press (2004)
17. Skillicorn, D.B., McConell, S.M.: Distributed prediction from vertically partitioned data. *J. Parallel Distrib. Comput.* **68**(1), 16–36 (2008)
18. 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)
19. Vanderbilt University. Gene expression model selector. <http://www.gems-system.org/>
20. Zhao, Z., Liu, H.: Searching for interacting features. *IJCAI* **7**, 1156–1161 (2007)