Exploring the consequences of distributed feature selection in DNA microarray data

Verónica Bolón-Canedo*†, Konstantinos Sechidis†, Noelia Sánchez-Maroño*, Amparo Alonso-Betanzos*, Gavin Brown†
*Department of Computer Science, University of A Coruña, Campus de Elviña s/n, A Coruña 15071, Spain
{vbolon, nsanchez, ciamparo}@udc.es
†School of Computer Science, University of Manchester, M13 9PL Manchester, UK
{konstantinos.sechidis, gavin.brown}@manchester.ac.uk

***Abstract*—Microarray data classification has been typically seen as a difficult challenge for machine learning researchers mainly due to its high dimension in features while sample size is small. Because of this particularity, feature selection is usually applied trying to reduce its high dimensionality. However, existing algorithms may not scale well when dealing with this amount of features, and a possible solution is to distribute the features into several nodes. In this work we explore the process of distribution on microarray data —which has recently gained attention— and we evaluate to what extent it is possible to obtain similar results as those obtained with the whole dataset. We performed experiments with different aggregation methods, feature rankers and also evaluated the effect of distributing the feature ranking process in the subsequent classification performance.**

I. Introduction

In the last years, microarray data classification has been established as a serious challenge for machine learning researchers. This type of data is used to collect information from tissue and cell samples regarding gene expression differences that are likely to be useful for diagnosing diseases or even for distinguishing among specific types of tumor. The particularities that make microarray data interesting for machine learning researchers are related with its high dimensionality (from thousands to tens of thousands of genes) in contrast with a small sample size (often less than 100 patients). In this scenario, the typical classification task is to separate healthy patients from cancer patients, based on their gene expression "profile" [1].

Research studies have shown that the great majority of the genes measures in a DNA microarray experiment are not relevant for the classification task [2]. Thus, to avoid the "curse of dimensionality" [3], feature selection is almost mandatory in this domain. Feature selection is defined as the process of identifying and removing irrelevant features from the data, so that the learning algorithm focuses only on those aspects of the data useful for analysis and future prediction [4].

Feature selection methods can be broadly divided into approaches that are classifier-independent (*filters*), and classifier-dependent (*wrapper* and *embedded* methods) [4]. Wrappers involve optimizing a predictor as part of the selection process, employing the accuracy of a particular classifier as the measure of utility for a candidate subset. This process produces subsets of features that are highly specific to the classifier chosen, so any change in the learning model may render the subset of features suboptimal. Embedded methods simply select features as a fundamental part of building a classifier, like decision trees, for example. These methods are less computationally expensive than wrappers, and also less prone to overfitting, but still use quite strict model structure assumptions [5]. Finally, filter methods are independent of any particular classifier model, and rank features based on statistical dependencies present between the features and the target class in the data. Therefore, they extract features that are generic, having incorporated few assumptions. Because of its speed of computation, filters are faster than embedded and wrapper methods, so they are usually preferred to deal with microarray data and hence they will be the focus of this work.

Usually, feature selection is applied in a centralized manner, i.e. a single learning model is used to solve a given problem. However, with the recent advent of Big Data, the need of distributing the feature selection process has arisen. Most of the existing algorithms were developed when dataset sizes were smaller and now they do not scale well and their efficiency significantly deteriorates or even become inapplicable. To solve this problem, the feature selection process can be parallelized by distributing the subsets of data to multiple processors, learning in parallel and then combining them, reducing the training time considerably.

Although this approach can be applied to any feature-abundant classification problem, it is especially suitable for application to microarray data, due to the large number of input features/genes that can be distributed across the available nodes. In fact, some works have already used a distributed feature selection approach to deal with microarray data [6], [7], [8]. However, the reduction in complexity that is achieved by distributing the data, usually comes at the cost of losing information, since the partial outputs need to be eventually combined. Since the data is divided by features, it is necessary to work with partial rankings, and so information about interaction or redundancy between features might be omitted.

The aim of this work is to explore if the particularities of DNA microarray data make them suitable for applying a distributed feature process. Specifically, we examine the effects of including some overlap in the subsets of data, and analyze different possibilities for combining the partial

results. Finally, we provide some recommendations about the most appropriate aggregation method to combine the partial rankings or which feature selection methods are more robust to dealing with incomplete rankings.

II. Microarray data

DNA is a molecule that encodes the "program" for future organisms. DNA has coding and non-coding segments. The coding segments, also known as genes, specify the structure of proteins, which do the essential work in every organism. Genes make proteins in two steps: DNA is transcribed into mRNA and then mRNA is translated into proteins. Advances in molecular genetics technologies, such as DNA microarrays, allow us to obtain a global view of the cell, with which it is possible to measure the simultaneous expression of tens of thousands of genes [9]. The gene expressions acquired from a DNA microarray can be used as inputs to large-scale data analysis to increase our understanding of normal and diseased states.

Apart from the obvious problem of having an extremely high dimensionality (in the order of tens of thousands of genes), microarray data present other experimental complications that may hinder the process of classification [1]. The small sample size might impact on the error estimation, leading to unsound applications of classification methods. Moreover, microarray datasets are usually unbalanced (i.e. a dataset is dominated by a major class with significantly more samples than the minority one) so the minority class instances are sometimes ignored by the learning algorithms. These characteristics render the analysis of microarray data an interesting and challenging domain for machine learning researchers.

III. Feature ranking

As mentioned in the Introduction, this work will be focused on filter methods for feature selection. Filter methods can be classified according to the output they produce. Some of them —so-called subset methods— return a subset of optimal features, while others —known as ranker methods— produce an ordered ranking of all the features obtained from assigning them weights according to their degrees of relevance [10].

Furthermore, feature ranking methods can be divided into univariate methods—those that take into account only the individual relevance of each feature—and multivariate methods—those which take into account feature dependencies. Theoretically, univariate methods are expected to be more tolerant to incomplete datasets (i.e. datasets in which not the whole set of features is available) than multivariate methods, although at the cost of missing feature dependencies. Feature ranking methods are very common in the literature (see [5], [11]) and in this work we chose two popular representative methods[1]: one univariate (MIM, Mutual Information Maximization [12]) and one multivariate (mRMR, minimum Redundancy Maximum Relevance [13]). In fact, the latter was specifically proposed for dealing with DNA microarray data, since it can take into account the redundancy usually present in this type of data.

[1]We used MIM and mRMR implementations from FEAST Toolbox for Matlab http://www.cs.man.ac.uk/~gbrown/fstoolbox/

IV. The rationale of the approach

In the last few years, there has been an important amount of attention devoted to DNA microarray classification. Because of the high dimensionality of this type of data (together with the redundancy of the genes), feature selection is usually applied as a preprocessing step [1].

Since the end of the 1990s, when microarray datasets began to be dealt with, a large number of feature selection methods have been applied —because of their high dimensionality together with the redundancy of the genes. In the literature, one can find both classical methods and methods developed especially for this kind of data. Due to the high computational resources that these datasets demand, wrapper and embedded methods have mostly been avoided, in favor of less expensive approaches such as filters.

In fact, the problem is that, when dealing with a high number of input features, the majority of algorithms were designed under the assumption that the dataset can be represented as a single memory-resident table. So, if the entire dataset does not fit in main memory, these algorithms are not applicable.

A possible solution to this problem is to distribute the dataset into several nodes (or processors). In this manner, a feature selection algorithm may take advantage of preprocessing multiple subsets in sequence of concurrently. Nevertheless, a common problem with this approach is the unavoidable information loss, since the partial outputs (the results obtained from each node) need to be eventually combined.

There are two main techniques for partitioning and distributing data: vertically, i.e. by features, and horizontally, i.e. by samples. In the case of microarray data, the distribution has to be by features, as can be seen in Figure 1. Notice that, under this scenario, it is necessary to work with partial rankings of features (obtained from the features in each node), and so information about interaction or redundancy between features might be omitted.

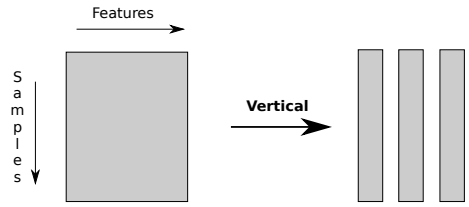


Fig. 1. Vertical distribution of the data

A. An example

We can characterize the problem of ranking a set of features $\mathbf{X} = \{X_1, \ldots, X_F\}$ as the problem of obtaining an ordered ranking of them according to some metric (related to the degree of importance) usually given by a feature selection method. As we mentioned before, and because of distributing the data across several nodes, we need to combine the partial ranks and try to not incur in an important loss of information.

In this example, let us suppose that we have a number of features (genes) F that we distribute across the different nodes and that need to be ranked. Depending on the system in which this approach is tested on, the bottleneck might be either the number of nodes available or the number of features. If the number of nodes is too high, each will have a small amount of data so it will be more difficult to recover the *ideal* ranking. On the contrary, if the number of nodes is low, each will have to take more of the computational burden, hence there will be less advantages to the distributed strategy.

To illustrate this problem, let N be the number of nodes available, while the relevance of each feature is randomly generated as a number between 0 and 1. Once the partial rankings (obtained from each node) are combined, we need a measure to determine to what extent the *combined* ranking is close to the *ideal* ranking (obtained when data is all together). For this task, we use the Normalized Discounted Cumulative Gain (NDCG) [14], which is often used to measure effectiveness of web search engine algorithms or related applications. This method returns a value between 0 and 1, where 1 means that the rankings are identical. The pseudocode for this example can be found in Algorithm 1.

Algorithm 1: Pseudo-code for generating the toy example

Data: $D_{(S\times F)} \leftarrow$ training dataset with S samples and F input features
$N \leftarrow$ number of nodes
$S \leftarrow$ number of samples
$F \leftarrow$ number of features
$\mathbf{X} \leftarrow$set of features, $\mathbf{X} = \{X_1, \ldots, X_F\}$
$f_n \leftarrow$ number of features to go to each node

Result: $NDCG \leftarrow$ similarity between the true ranking and the combined ranking

Generate a random value between 0 and 1, $Score(X_i)$, for each feature $X_i \in \mathbf{X}$, obtaining a true ranking $Rank_t$
for $n \leftarrow$ *1 to N* **do**
  $D_{(S\times f_n)} \leftarrow$ subset of data with f_n random features
  Rank the features in this node according to their $Score$, obtaining a partial ranking $Rank_p(n)$
end
for *each feature* f *in* $\mathbf{X}$ **do**
  $Avg(f) \leftarrow$ calculate the average of its position in all the partial rankings $Rank_p(n)$, $\forall n \in N$
end
Obtain a combined ranking $Rank_c$ by ordering Avg
$NDCG \leftarrow$ compare ($Rank_t$, $Rank_c$)

Figure 2 shows an example in which the number of features to rank is $F = 1000$ and the maximum number of nodes available is $N = 1000$. The NDCG value is represented by the color, as the colorbar in the right side of the figure depicts. As expected, when all the features are present on each node, the NDCG value is 1 since the rankings are identical. Nevertheless, even when we have 100 nodes and 800 features on each node, the rankings are not exactly the same, which gives us an idea about the complexity of the ranking combination task. From the figure, we can see that, for ensuring good results to be obtained, it is necessary to send a large number of features to each node, even when the number of nodes available is also large. And it has to be noted that, by doing so, the time complexity is barely reduced.

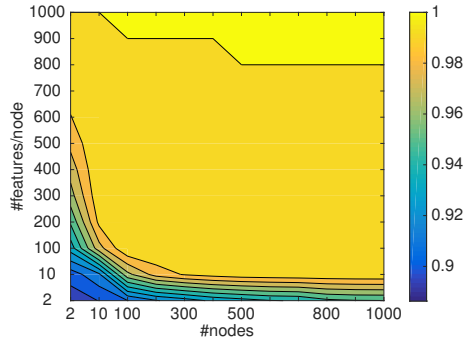


Fig. 2. Example to illustrate the problem of distributing feature rankings, with $F = 1000$ features.

As can be seen through this simple example, the problem of distributing the features and obtaining accurate rankings of them is not trivial, even when we can use the true importance of the features, obtained from the whole set of examples. Arrow's impossibility theorem [15] states that, when having *at least* two rankers (in this case, nodes) and *at least* three options to rank (in this case features), it is impossible to design an aggregation function that satisfies in a strong way a set of desirable conditions at once (see more details in [15]). So, this theorem also acknowledges how challenging it is to distribute the feature ranking process.

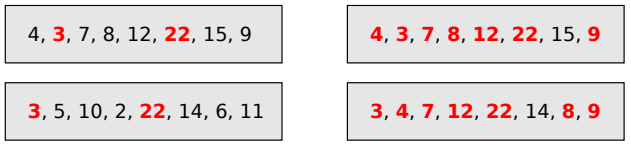
V. The distributing methodology

In this section we will present some strategies that are possible when distributing features across several nodes or processors. When building the subsets of features, it is possible to define a level of overlap among them or leave it random. Moreover, there are several strategies to combine the rankings obtained from the different nodes into a final ranking of features. In the following subsections, we will comment on these strategies in detail.

A. Distribution by features

Although the most common approach in the literature when dealing with distributed learning is to distribute the data in subsets of samples, there are situations —such as DNA microarray classification— in which the number of features is the bottleneck of the algorithms and it is more practical to divide the data by features. Notice that in this case, and contrary to the situation in which data is divided by samples, the rankings are partial (i.e. they do not contain all possible features) so their combination poses a big challenge to researchers.

To be able to combine the partial rankings, a certain level of overlap among the different subsets of features on the available nodes is not only desirable but also mandatory. The overlap between two sets of features can be defined as the subset of features that are common among them. In Figure 3 we can see an example of overlap between two subsets of features. This level of overlap needs to be carefully chosen. If the overlap is too low (see Figure 3(a)), it is difficult to connect the partial rankings and the performance is expected to be poor. On the contrary, when the overlap is too high (see Figure 3(b)), the performance is expected to be good, but at the cost of barely reducing the complexity of the original problem.



(a) Low overlap (b) High overlap

Fig. 3. Example of two cases of overlap between rankings, where the numbers represent the indices of the features; in red the overlapping features.

A possible strategy to distribute the features is to randomly select the features for each node, allowing that each feature can be picked for more than a node. In other words, for each node we randomly pick f_n features from the whole set of features, and this is repeated for each node. In this way we suppose that some overlap might occur, although we cannot ensure to what extent. Because of the random sampling of the features, we need to make sure that all the features were selected at least once to appear on each node, at the end of the process. This can be solved, for example, by adding the features which are not present in any node (if any) to the last node. In the following, we will refer to this strategy as *randomOverlap*.

However, if we want to force a certain level of overlap (between the features in a node and the remaining features), a possible strategy might be the following: let $\mathbf{X_i} = \{X_1, \ldots, X_j\}$ be the set of features assigned to node i and $\mathbf{X_r} = \{X_{j+1}, \ldots, X_F\}$ the remaining features that will be assigned to other nodes. If we are considering 20% of overlap, we will add to $\mathbf{X_i}$ the 20% of features in $\mathbf{X_r}$, randomly picked. The drawback within this approach (named *predefOverlap*) is that it is impossible to ensure a certain overlap between pairs of nodes. In Section VI-A we will explore how these proposed strategies behave.

B. Aggregation

Once the features are divided across the different nodes and different partial rankings are obtained, it is necessary to combine them in a single final ranking. Several strategies can be used to aggregate the rankings, and in this paper we have chosen the following four methods [16]:

- **best.rank**: assigning to each element to be ranked the best position that it has achieved among all rankings.
- **median**: assigning to each element to be ranked the median of all the positions that it has achieved among all rankings.
- **arith.mean**: assigning to each element to be ranked the mean of all the positions that it has achieved among all rankings.
- **geom.mean**: assigning to each element to be ranked the geometric mean of all the positions that it has achieved among all rankings.

We illustrate the behavior of these methods with a simple example. When the data is distributed by features, we have to deal with partial rankings. Imagine that we have 6 different features to be ranked $\{a, b, c, d, e, f\}$, and 3 nodes, such that 2 features go to each node. Suppose that we are considering 50% of overlap, so an extra feature from the remaining nodes goes to each node. Then, for example, we will have elements $\{a, b, c\}$ in the first node, elements $\{c, d, f\}$ in the second node, and elements $\{e, f, a\}$ in the third node. The three partial rankings can be seen in Table I, in which elements not present on a given node are being assigned the last position in the ranking, according to the implementation provided in [16] (6, in this example). In this case, the *best.rank* method will return $\{a, d, c, f, b, e\}$, whereas the remaining methods will return $\{a, f, c, d, b, e\}$. Notice that, for the sake of this example, we are choosing the alphabetical order in case of ties.

TABLE I
EXAMPLE OF HOW THE AGGREGATION METHODS WORK WITH PARTIAL RANKS.

Element	R_1	R_2	R_3	best.rank	median	arith.mean	geom.mean
a	1	6	1	1	1	2.7	1.8
b	3	6	6	3	6	5.0	4.8
c	2	3	6	2	3	3.7	3.3
d	6	1	6	1	6	4.3	3.3
e	6	6	3	3	6	5.0	4.8
f	6	2	2	2	2	3.3	2.9

It has to be noted that certain aggregation methods are more prone to deal with ties. For example, the *best.rank* method, since the set of possible values obtained when applying this strategy is much more reduced than the possible values obtained by *arith.mean*. Experiments using the different aggregation methods can be found in Section VI-C.

VI. EXPERIMENTS

In this section we empirically evaluate the strategies commented on the previous section about how to deal with the distribution of feature rankings. We have considered nine widely-used binary class microarray datasets [1], which are available for download in [17], [18], [19]. The reason for choosing binary datasets is that they are much more common in the literature than the multiclass ones. As a matter of fact, a typical microarray dataset consists of distinguishing between having a given cancer or not, therefore the great majority of the datasets are binary. Table II summarizes the properties of the selected datasets: for each dataset, the number of features (# Feats.), number of samples (# Samp.) and the percentage of examples of each class is shown. The imbalance ratio [20] (IR) is defined as the number of negative class samples divided by the number of positive class samples,

in which a high level indicates that the dataset is highly imbalanced. Finally, F1 (*maximum Fisher's discriminant ratio*) [21] checks for overlapping among the classes in which the higher the F1, the more separable the data is. In order to estimate mutual information of continuous features, they were discretized, using an equal-width strategy into 5 bins.

TABLE II
SUMMARY DESCRIPTION OF THE DATASETS USED IN THE EXPERIMENTAL STUDY

Dataset	# Feats.	# Samp.	(%min,%maj)	IR	F1
Brain	12625	21	(33.33, 66.67)	2.00	0.89
CNS	7129	60	(35.00, 65.00)	1.86	0.45
Colon	2000	62	(35.48, 64.52)	1.82	1.08
DLBCL	4026	47	(48.94, 51.06)	1.04	2.91
GLI	22283	85	(30.59, 69.41)	2.27	2.35
Ovarian	15154	253	(35.97, 64.03)	1.78	6.94
SMK	19993	187	(48.13, 51.87)	1.08	0.41

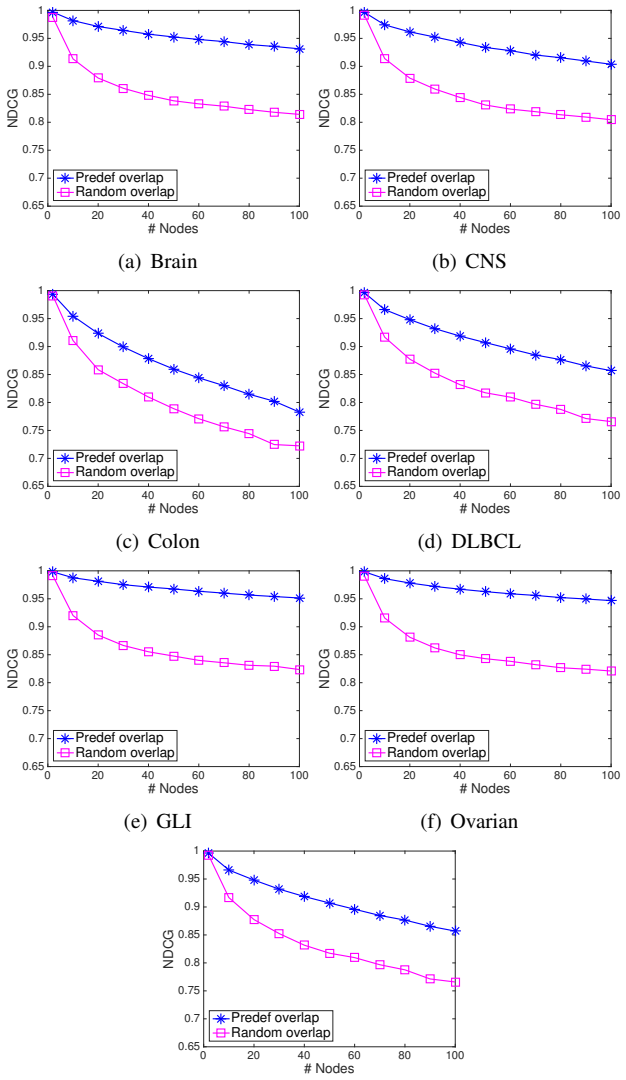
In the following sections, we ask the questions: "how can we deal with overlap between features?", "how different feature ranking methods behave when distributing the data?", "which is the best aggregation method in this scenario?", and, finally, "what is the relationship of these results with the classification accuracy?". To address these questions, we use some of the datasets detailed in Table II.

Notice that, in the experiments carried out in this section, we do not take into account the computational time. This is because the focus of this paper is to find how much information about the features is lost when distributing the feature ranking process. As for the training time, it is assumed that the time is reduced when dividing the data into the available nodes.

A. Overlap between nodes

As mentioned in Section V-A, there are different options to induce some overlap between the features on each node, which is essential for a correct integration of the partial rankings.

In Figure 4 we can see a comparison between both strategies, randomOverlap and predefOverlap, using a 50% of overlap in both cases; however, randomOverlap only guarantees that some overlap may occur because we randomly added 50% extra features to each node. For simplicity, the Mutual Information Maximization (MIM) [12] feature ranking method and the best.rank aggregation method were chosen. Notice that, by using MIM we could have used the mutual information values computed for each feature and then build the final ranking based upon these values, and the ranking recovering problem would be trivial. However, we are using MIM as representative of any feature ranking method and omitting the mutual information values, but only working with the partial rankings. The experiments were repeated 100 times and we are showing the average NDCG values (computed by comparing the combined ranking with the *ideal* ranking obtained with the whole data). As the NDCG measure tends to obtain high results if the number of elements to rank is high (as happens with microarray data), it is common to compare only



(a) Brain (b) CNS

(c) Colon (d) DLBCL

(e) GLI (f) Ovarian

(g) SMK

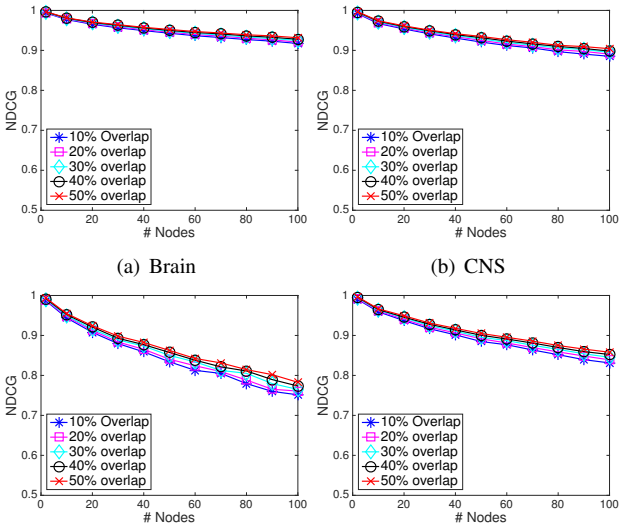
Fig. 4. Experiments with different numbers of features per node and two different strategies for dealing with overlap (50% in this case)

the top ranked elements. For instance, when evaluating the performance of web search engines, it is common to compare only the top 10 entries. In DNA microarray analysis, it is also common to focus only on the top ranked features, so in these experiments we will compute NDCG comparing only the top X features, being X the 10% of the total number of features (so, for instance, if we are dealing with Colon dataset, we compare the top 200 features).

As can be seen, the experimental results show that using a predefined level of overlap clearly outperforms the results achieved when the overlap is random, and this improvement is more pronounced as the number of nodes increases —which implies a higher complexity of the problem. Notice that, for the sake of fairness, the number of features to go to each node is the same both when we are using a predefined level of overlap and when we are randomly selecting the features belonging to each node. It is necessary to bear in mind that, when picking

random subsets of features for each node, it might be even possible that some features are never selected, and hence they are never ranked. For all these reasons, the authors recommend the use of predefined overlap when distributing the features across the available nodes.

However, the required level of overlap comes at a computational price. In fact, suppose that we are dealing with Colon dataset (2000 features) and 10 nodes. Forcing a 50% overlap implies that we would have 300 features per node, instead of 200 features per node —although with this situation there would be no overlap and it would not be possible to combine the partial rankings. Trying to shed light on this issue, we performed some experiments varying the level of overlap. In Figure 5 we can see some experiments in which the level of overlap ranges from 10% to 50% (again with MIM feature ranker and best.rank aggregation method). As can be seen, there are no big differences among the different levels of overlap, although the variance is more pronounced as the number of nodes increases —as expected. For the sake of brevity, we are showing the results only for four of the datasets.



(a) Brain (b) CNS

(c) Colon (d) DLBCL

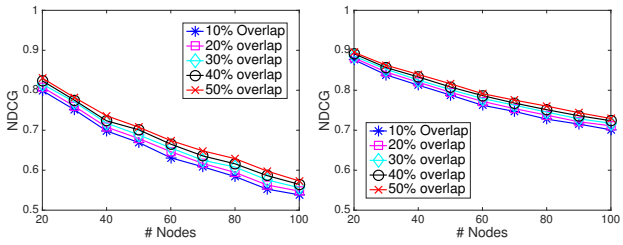
Fig. 5. Experiments with different numbers of features per node and different levels of predefined overlap when using MIM

B. Univariate vs. multivariate methods

The experiments in the previous subsection (testing different levels of overlap) were executed with MIM feature ranker, which is a univariate method so it is expected that it is more robust with respect to incomplete data —since it does not have to take into account interactions between features, such as redundancy. However, when dealing with microarray data, it is common that researchers employ multivariate data, since it is well-known that most genes in a microarray experiment are redundant between each other. The problem is that dealing with redundancy implies a higher computational cost, which makes more necessary to distribute the feature ranking process. For example, the theoretical complexity of MIM is $\mathcal{O}(SF)$ (where S is the number of samples and F is the number of features) whilst that of mRMR is $\mathcal{O}(SF^2)$. When the number of features is in the order of thousands (as it is the case in the scenario at hand), this increase in complexity becomes, in some cases, unbearable.

Figure 6 shows the same experiment we perform with MIM univariate ranker method (testing different nodes and different levels of overlap) but, in this case, we used the mRMR multivariate method. Because of the time complexity restrictions of mRMR (it takes in the order of days to compute the whole ranking for some datasets), we run the experiment only with Colon and DLBCL datasets, and we started the number of nodes in 20.

As can be seen from the figures, the results when using mRMR are worse than when using MIM (Figure 5). For DLBCL dataset, the minimum NDCG obtained with MIM was around 0.83, whereas with mRMR it was 0.7. Even more drastic is the deterioration in the case of Colon, in which the worst NDCG value with mRMR drops until 0.5. In light of these results we can see that, the more we need to distribute the data —i.e. because of the computational complexity or multivariate methods—, the more information we lose when recovering the ranking.



(a) Colon (b) DLBCL

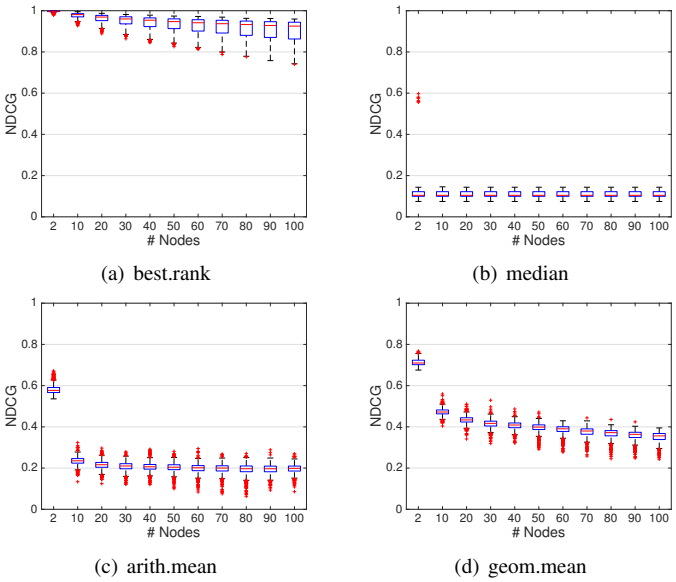
Fig. 6. Experiments with different numbers of features per node and different levels of predefined overlap when using mRMR

C. Best aggregation method

In Section V-B, we mentioned that there are several strategies to combine the partial rankings into a final ranking of features. Specifically, we chose four different methods, named *best.rank*, *arith.mean*, *geom.mean* and *median*. So far, in the experiments we used the *best.rank* method, but in this section we will perform some experiments to compare it with the other aggregation methods.

Figure 7 shows the NDCG values on average over the seven datasets presented in Table II, when the features are divided from 2 to 100 nodes, with 50% of overlap, and for 100 repetitions; for the four aggretation methods considered. As feature ranking method, we chose MIM. The results show that the *best.rank* method clearly outperforms the remaining aggregation methods, although of course its performance degrades as the number of nodes increases.

Trying to understand the reasons behind the superiority of the *best.rank* method, let's remember the example depicted in



(a) best.rank (b) median

(c) arith.mean (d) geom.mean

Fig. 7. NDCG across 7 datasets and 100 repetitions; when using MIM feature ranking method, with 50% overlap and predefOverlap strategy. The box indicates the upper/lower quartiles, the horizontal line within each shows the median value, while the dotted crossbars indicate the maximum/minimum values.

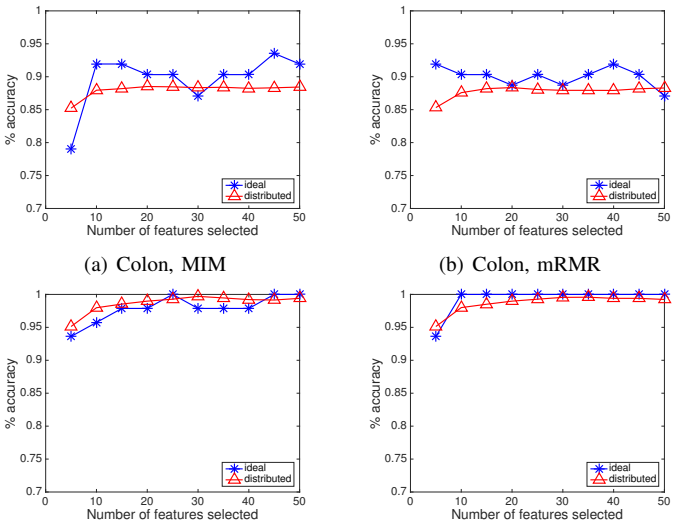
Table I, which showed how these methods work with partial rankings. In this case, it is very likely that ties happen with the *median* method, leading to a poor performance as the number of nodes increases. In fact, this behavior can be seen in Figure 7(b). The superiority of the *best.rank* method is explained because it is the only method that can overlook the presence of many "last" positions for each element, which greatly affects the performance of the other methods. A line of future work can be to improve the treatment of partial rankings by these aggregation methods.

In light of these results, the authors suggest the use of the *best.rank* method for dealing with distributed DNA microarray data.

D. Relationship with classification accuracy

At this point, it is necessary to clarify that including classifiers in our experiments is likely to obscure the experimental observations related to feature selection performance, since they include their own assumptions and particularities, and some classifiers even perform their embedded feature selection—such as decision trees. Therefore, in these experiments we use a simple nearest neighbor classifier ($k = 3$), this is chosen as it makes few (if any) probabilistic assumptions about the data, and we avoid the need for parameter tuning.

Figure 8 shows the % classification accuracy obtained by a 3-NN classifier on Colon and DLBCL datasets using MIM and mRMR feature selection methods. For evaluating the loss in classification accuracy when distributing the data, we are comparing the results obtained when using the ranking built with the whole dataset, with the rankings achieved when distributing the data by features. As aggregation method, we used *best.rank*, as suggested in Section VI-C. We divided the data in 50 nodes and we added a 50% of overlap between nodes. Since for classification we need to establish a threshold in the ranking returned by the feature selection method, we opted for considering the top features from 5 to 50, with increments of 5. For calculating the classification accuracy of both distributed approaches, the whole process was repeated 100 times and averaged.



(a) Colon, MIM (b) Colon, mRMR

(c) DLBCL, MIM (d) DLBCL, mRMR

Fig. 8. Classification accuracy obtained with a 3-NN classifier on Colon and DLBCL datasets and reported on average for 100 repetitions. The *best.rank* aggregation method was employed.

The experimental results show that, overall, there is a negligible loss in accuracy when applying a distributed approach. Moreover, we can see that the results obtained by distributing the data are, in some cases, more stable than those achieved with the whole ranking. This is because, in some way, distributing the features across nodes and then combining the partial results into a final one is the same idea also known as ensemble learning or mixture of experts, which states that combining the outputs of several experts yields better and more robust results than a single expert [22]. However, analyzing specific aspects related to classification is out of the scope of this paper.

In summary, these experiments demonstrate that it is possible to distribute the data without significantly compromising the classification accuracy, since this measure is the ultimate form of evaluation of the goodness of a feature ranking method.

VII. Conclusion

This work has presented an analysis of how distributed strategies affect to DNA microarray data. These data are characterized by having a much larger number of features than of samples, so distributing the features across different processing nodes might alleviate the computational work of feature ranking methods and therefore the training time will be reduced. However, when distributing data, the crucial point is the combination of the incomplete rankings generated at

each node, aiming at losing the smallest amount of information possible, compared to the *ideal* ranking. From the experiments carried out in this paper, we can draw some conclusions and recommendations to the users:

- When the bottleneck of the problem is the large number of features, and so we distribute the data by features, it is essential to introduce some level of overlap between the features appearing in the different nodes, to ensure a correct combination of the partial rankings. We have explored several techniques of introducing overlap, and the experimental results showed that it is better to ensure a determined level of overlap. The level of overlap has to be carefully chosen, having in mind that a high degree of overlap implies a higher computational cost.
- When it comes to aggregate the partial ranks, in this scenario the best option according to the experiments was the so-called the *best.rank* aggregation method.
- Regarding the choice of univariate or multivariate feature ranking methods, we found that when using multivariate methods, the loss of information given by distributing the data is higher. However, depending on the degree of redundancy present in the data, it might be necessary to employ a multivariate approach. In this case, the number of nodes needs to be reduced to the extent possible, trying to find a trade-off between computational complexity reduction and loss of information.
- In terms of classification performance, experiments with a simple nearest neighbor classifier allowed us to check that distributing the data—to an acceptable extent—does not seem to compromise the classification accuracy.

In summary, DNA microarray data can benefit from a distribution process, although some choices have to be carefully made. For example, the method for aggregating rankings must be *best.rank*, and a predefined level of overlap is necessary among nodes. Having said that, the authors think that DNA microarray data is a suitable focus for research into distributed feature selection strategies.

As future work, we have identified an important need for developing new aggregation methods that can deal more efficiently with partial rankings and with ties along different rankings. Furthermore, new scenarios in which the distribution has to be done by samples must be explored. It would be also interesting to add some comparisons of computational cost in terms of computational time and memory usage and the use of more feature rankers in the experimental study.

ACKNOWLEDGMENT

This research has been economically supported in part by the Ministerio de Economía y Competitividad of the Spanish Government through the research project TIN2015-65069-C2-1-R, 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 Xunta de Galicia postdoctoral funding (grant ED481B 2014/164-0).

REFERENCES

[1] V. Bolón-Canedo, N. Sánchez-Maroño, A. Alonso-Betanzos, J. M. Benítez, and F. Herrera, "A review of microarray datasets and applied feature selection methods," *Information Sciences*, vol. 282, pp. 111–135, 2014.

[2] T. Golub, D. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. Mesirov, H. Coller, M. Loh, J. Downing, M. Caligiuri *et al.*, "Molecular classification of cancer: class discovery and class prediction by gene expression monitoring," *Science*, vol. 286, no. 5439, pp. 531–537, 1999.

[3] A. Jain and D. Zongker, "Feature selection: Evaluation, application, and small sample performance," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 2, pp. 153–158, 1997.

[4] I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, *Feature extraction: foundations and applications*. Springer, 2006.

[5] G. Brown, A. Pocock, M.-J. Zhao, and M. Luján, "Conditional likelihood maximisation: a unifying framework for information theoretic feature selection," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 27–66, 2012.

[6] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, "Distributed feature selection: An application to microarray data classification," *Applied Soft Computing*, vol. 30, pp. 136–150, 2015.

[7] R. B. Ray, M. Kumar, and S. K. Rath, "Fast computing of microarray data using resilient distributed dataset of apache spark," in *Recent Advances in Information and Communication Technology 2016*. Springer, 2016, pp. 171–182.

[8] L. Morán-Fernández, V. Bolón-Canedo, and A. Alonso-Betanzos, "A time efficient approach for distributed feature selection partitioning by features," in *Conference of the Spanish Association for Artificial Intelligence*. Springer International Publishing, 2015, pp. 245–254.

[9] G. Piatetsky-Shapiro and P. Tamayo, "Microarray data mining: facing the challenges," *ACM SIGKDD Explorations Newsletter*, vol. 5, no. 2, pp. 1–5, 2003.

[10] L. Yu and H. Liu, "Efficient feature selection via analysis of relevance and redundancy," *The Journal of Machine Learning Research*, vol. 5, pp. 1205–1224, 2004.

[11] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.

[12] D. D. Lewis, "Feature selection and feature extraction for text categorization," in *Proceedings of the workshop on Speech and Natural Language*. Association for Computational Linguistics, 1992, pp. 212–217.

[13] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 8, pp. 1226–1238, 2005.

[14] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 4, pp. 422–446, 2002.

[15] K. J. Arrow, *Social choice and individual values*. Wiley, 1951.

[16] R. Kolde, S. Laur, P. Adler, and J. Vilo, "Robust rank aggregation for gene list integration and meta-analysis," *Bioinformatics*, vol. 28, no. 4, pp. 573–580, 2012.

[17] "Feature Selection Datasets at Arizona State University," http://featureselection.asu.edu/datasets.php, [Online; accessed November-2016].

[18] A. Statnikov, C. Aliferis, and I. Tsamardinos, "Gems: Gene expression model selector," http://www.gems-system.org, [Online; accessed November-2016].

[19] "Kent Ridge Bio-Medical Dataset," http://datam.i2r.a-star.edu.sg/datasets/krbd, [Online; accessed November-2016].

[20] A. Orriols-Puig and E. Bernadó-Mansilla, "Evolutionary rule-based systems for imbalanced data sets," *Soft Computing*, vol. 13, no. 3, pp. 213–225, 2009.

[21] T. K. Ho and M. Basu, "Complexity measures of supervised classification problems," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 3, pp. 289–300, 2002.

[22] L. I. Kuncheva, "Ensemble methods," *Combining Pattern Classifiers: Methods and Algorithms, Second Edition*, pp. 186–229, 2014.