Insights into distributed feature ranking

Verónica Bolón-Canedo[a,b,*], Konstantinos Sechidis[b], Noelia Sánchez-Maroño[a], Amparo Alonso-Betanzos[a], Gavin Brown[b]

[a]*Department of Computer Science, Universidade da Coruña, 15071 A Coruña, Spain*
[b]*School of Computer Science, University of Manchester, M139PL Manchester, UK*

Abstract

In an era in which the volume and complexity of datasets is continuously growing, feature selection techniques have become indispensable to extract useful information from huge amounts of data. However, existing algorithms may not scale well when dealing with huge datasets, and a possible solution is to distribute the data in several nodes. In this work we explore the different ways of distributing the data (by features and by samples) and we evaluate to what extent it is possible to obtain similar results as those obtained with the whole dataset. Trying to deal with the challenge of distributing the feature ranking process, we have performed experiments with different aggregation methods and feature rankers, and also evaluated the effect of distributing the feature ranking process in the subsequent classification performance.

Keywords: feature selection, feature ranking, distributed learning

1. Introduction

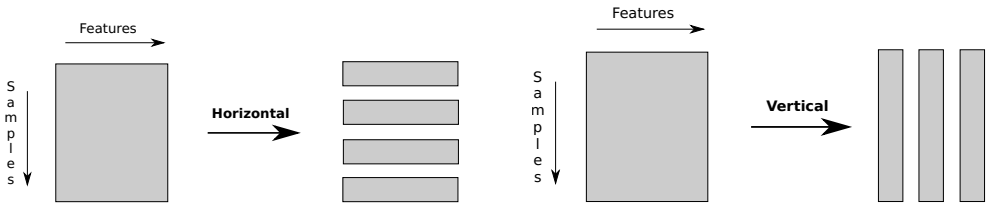
Feature selection has been one of the high activity research areas during the last few years, due to the appearance of datasets containing hundreds of thousands of features. Thus, as feature selection allows to reduce the dimensionality of the problems, it can be used for maintaining and –most of the time– improving the machine learning algorithms' performance, while re-

*Corresponding author.
Email addresses: vbolon@udc.es (Verónica Bolón-Canedo), konstantinos.sechidis@manchester.ac.uk (Konstantinos Sechidis), nsanchez@udc.es (Noelia Sánchez-Maroño), ciamparo@udc.es (Amparo Alonso-Betanzos), gavin.brown@cs.manchester.ac.uk (Gavin Brown)

ducing computational costs. For these reasons, feature selection has become an indispensable preprocessing step in many applications, showing outstanding results, such as in the fields of bioinformatics [38, 30], text analysis and classification [4, 39, 3], network analysis and classification [16, 43], or dimensionality reduction for the sake of visualization [37], just to name a few.

However, when dealing with an extremely high number of input features, learning algorithms' performance can degenerate due to over fitting, learned models decrease their interpretability as they become more complex and, finally, speed and efficiency of the algorithms decline in accordance with size [18]. Moreover, the problem is that the majority of existing feature selection algorithms were designed under the assumption that the dataset would be represented as a single memory-resident table. So if the entire data set does not fit in main memory, these algorithms are challenging to apply.

A practical solution to the aforementioned problems might be to distribute the dataset into several nodes or processors in a computer cluster. Then, if the data is distributed, feature selection methods may take advantage of processing multiple subsets in sequence or concurrently. There are two simple strategies for distributing the data, that can be consulted in Figure 1. On the one hand, it is possible to partition the data by samples (known as horizontal distribution), give a subset of samples to each node, in such a way that each node estimates the score for *all* the features and sort them. At the end, the rankings obtained by each node need to be combined. On the other hand, another possibility is to partition the data by features (vertically), give a *subset* of features to each node to estimate their scores, and then combine the partial rankings.



(a) Horizontal (b) Vertical

Figure 1: Types of partition of the data

In this work we explore the two strategies depicted above. We also examine the effects of including some overlap in the subsets of data, and analyze different possibilities for combining the partial results, proposing a corollary

that can help in deciding how to distribute the data across the available nodes. Finally, we provide some recommendations about the most appropriate aggregation method to combine the partial rankings and which feature selection methods are more robust for dealing with incomplete rankings. A small part of this article has been published in a previous conference paper [13], in particular some experiments concerning the vertical distribution on microarray datasets.

2. An illustrative example

The problem of feature ranking can be characterized as the problem of having a set of features and obtaining an ordered ranking of them according to some degree of importance, that usually is given by a feature selection criterion. Although feature ranking may well be one of the better known preprocessing techniques, extensively studied for years, the present Big Data scenario has come with new challenges for the field [11]. One available alternative to deal with this scenario is the use of distributed approaches (that will be described in the next section), and thus it is important not to overlook the implications in a distributed computing environment. If we distribute the data across different nodes, the problem becomes that of having partial rankings that need to be combined, without incurring in an important loss of information.

Suppose that we have a number of features that need to be ranked, but distributed across different computational nodes. Depending on the scenario we are working with, the bottleneck might be the number of nodes available or the number of features per node. To illustrate this, we show an illustrative toy example for which pseudocode can be consulted in Algorithm 1. For determining the degree of closeness between the ideal ranking (obtained when data is all together) and the ranking achieved after combining the partial rankings, we use the Normalized Discounted Cumulative Gain (NDCG) [24], 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.

Figure 2(a) shows an example in which the number of features to rank is $M = 100$ and the maximum number of nodes available is $K = 100$. The color in the figure represents the NDCG value, as depicted in the colorbar in the right of the figure. As expected, when all the features are present on each node, the NDCG value is 1 since the rankings are identical. However,

Algorithm 1: Pseudo-code for generating the toy example

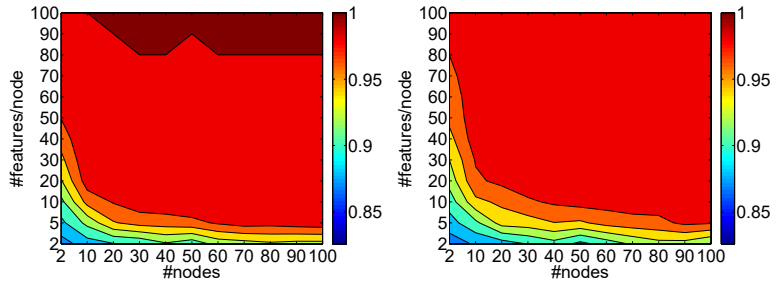
Data: $D_{(M)} \leftarrow$ training dataset with M input features
$K \leftarrow$ number of nodes
$M \leftarrow$ number of features
$\mathbf{X} \leftarrow$set of features, $\mathbf{X} = \{X_1, \ldots, X_M\}$
$m_k \leftarrow$ number of features to go to each node
Result: $NDCG \leftarrow$ similarity between the true ranking and the combined ranking

```
1 Generate a true ranking Rank_t by generating a random value between 0 and 1, Score(X_i), for
    each feature X_i ∈ X
2 for k ← 1 to K do
3 |   D_(m_k) ← subset of data with m_k random features
4 |   Rank the features in this node according to their Score, obtaining a partial ranking
  |     Rank_p(k)
  end
  /* Combining partial rankings into a final ranking                                  */
5 for each feature m in X do
6 |   Avg(m) ← calculate the average of its position in all the partial rankings Rank_p(k), ∀k ∈ K
  end
7 Obtain a combined ranking Rank_c by ordering Avg
8 NDCG ← compare (Rank_t, Rank_c)
```

even in the case in which we have 10 nodes and 90 features on each node, the rankings are not exactly the same, which gives us an idea about the complexity of the ranking combination task. As can be seen, for obtaining good results it is necessary that each node has a large number of features, even when the number of nodes available is also large. The problem is that, by doing so, the time complexity is not sufficiently reduced. Notice that in a real scenario, each node has only partial information to rank the features (because of not having the whole dataset), so the importance given to each feature cannot be computed in an accurate way, as opposite as what happens in this example, in which each node uses the true importance of each feature.

To simulate a real scenario, in which a feature ranking method has to determine the importance of a feature according only to the data it has, we include some random noise to the scores that represent the true importance of each feature, since it is likely that, in some nodes, two features with close relevance can be misranked. Figure 2(b) shows the same scenario as before, but with some noise (in this case, added with a normal distribution with mean 0 and variance 0.2). In this scenario, as can be seen, the distributed ranking problem becomes even more complex.

As this simple example reveals, the problem of distributing the feature ranking process is not easy to solve, even when we can use the true importance of the features, obtained from the whole set of examples. In fact, Ar-



(a) No noise (b) Noise

Figure 2: Example to illustrate the problem of distributing feature rankings, with and without noise, with $M = 100$ features.

row's impossibility theorem [5] 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 (more details can be found in [5]). So, this theorem also acknowledges how challenging it is to distribute the feature ranking process.

In the next section a review of distributed approaches to ranking feature selection algorithms is given.

3. Background

Traditionally, feature selection is applied in a centralized manner, i.e., a single learning model is used to solve a given problem. However, since nowadays data may be distributed, feature selection can take advantage of processing multiple subsets in sequence or concurrently. There are several ways to distribute a feature selection task [20]. In this work we have considered that the data set is together in one very large dataset or that either the data may be in different datasets in different locations (e.g., in different parts of a company). In the former case data can be distributed on several processors, whereas the latter can use the original distribution. Next, an identical feature selection algorithm can be run on each and the results combined. Other ways of distributing the feature selection task are a) *streaming*, where large volumes of data may be arriving in a continuous infinite stream in real time, and b) *ensembles*, where the dataset is not particularly large but different feature selection methods need to be applied to learn unseen

instances and combine results. For further details of both, the interested reader can consult the paper by Bolón-Canedo et col. [11].

The two main approaches to partitioned data distribution are by features (vertically) or by samples (horizontally). Distributed learning has been extensively studied during the last years, mostly focusing on scale up datasets that are too large for batch learning by samples [8, 17, 45]. Due to its importance, there are some frameworks available such as MLlib, Spark's open-source distributed machine learning library [32] or MADlib that provides an evolving suite of SQL-based algorithms for machine learning, data mining and statistics that run at scale within a database engine [21]. However, when dealing with big dimensionality datasets, researchers, of necessity, have to partition by features. For instance, in the case of DNA microarray data, the small sample size combined with big dimensionality prevent the use of horizontal partitioning. Some methods that take into account some of the particularities of these datasets, such as the high redundancy among features, were presented by Sharma et al. [42] and Bolón-Canedo et al. [10], the latter at a much lower computational cost. Including other type of datasets, Prasad et al. proposed a distributed parallel feature selection technique that employs a vertical distribution strategy for a dataset to exploit parallel computation [36]. It uses Information Gain filter-based ranking method which evaluates multiple disjoint feature subsets of dataset in parallel. Zadeh et al. [47] presented a vertically distributed filter method which handles the redundancy with consistently comparable results with centralized methods. They formalized the feature selection problem as a diversity maximization problem by introducing a mutual-information-based metric distance on the feature.

Finally, there are some approximations that are able to deal with horizontal as well as vertical data partitioning. Banerjee & Chakravarty proposed a secure distributed protocol that will allow feature selection for multiple parties without revealing their own data [7]. Another is a distributed parallel feature selection method that can read data in distributed form and perform parallel feature selection in symmetric multiprocessing mode via multithreading and massively parallel processing [49]. Recently, Hogde et al. present a distributed processing framework for feature selection using the AURA neural network [6] and Apache Hadoop [22]. They have used five feature selectors (mutual information, Correlated Feature Selection, Gain Ratio, Chi Square and Odds Ratio) which may be used independently or coupled with the AURA k-NN for classification or prediction. Moreover, there are

some approximations adapted to specific issues, such as the framework presented by Zhao et col. that combines distributed feature selection methods and econometric models for efficient economic analysis [48]. They developed a subtractive clustering based feature selection algorithm and an attribute coordination based clustering algorithm to select and identify the important features of data in both horizontally and vertically distributions.

Scalability of feature selection is still an open challenge in the machine learning community and how to design efficient and distributed algorithms to speed up the computation is still a fertile area and needs deeper investigation [29]. The work presented herein tries to study the problem of the distribution of feature rankings in both the two typical scenarios exposed above (i.e. when the data is divided by features and by samples), aiming at finding a way to reduce the computational burden of the feature ranking algorithms in cases in which we deal with large datasets.

Also, a particularity of the works mentioned in this section is that they were focused on distributing the feature selection process without degrading the classification accuracy. Different from them, in this work we study the distributed problem from a perspective *independent* of the dataset, trying to give some recommendations about the behavior of different feature ranking and feature combination methods, in this scenario. Our idea is to provide guidelines to the interested reader, according to the resources available, considering the number of nodes usable and the number of features to deal with. The remaining of this article aims at answering questions such as how important the selection of a good ranking aggregation method is, or the tolerance of feature selection methods for working on partial data.

4. Distributing feature rankings

In this section we will present some strategies that are possible when distributing feature rankings. The first choice to make is if dividing the data by samples or by features, usually depending on if the dataset is too large in terms of samples or in terms of features. Then, the distribution of the data can be done in disjoint subsets or allowing for some overlap. And, finally, we need another strategy to combine the rankings obtained over the different partitions into a final ranking of features. In Table 1, we can see an overview of some of the possible strategies that we have to deal with when distributing the feature ranking problem. In the following subsections, we will comment on them in more detail.

Table 1: Some strategies for distributing feature rankings.

Distrib. by samples	**Distrib. by features**	**Aggregation methods**
- Disjoint subsets of samples	- Random subsets of features	- Best rank
- Bootstrap subsets of samples	- Subsets with predefined overlap	- Arith. mean
		- Geom. mean
		- Median

4.1. Distribution by samples

In distributed learning, the most common approach is to partition data by samples. In the scenario at hand, this means that we have different rankings, obtained from different examples, but all the rankings are complete.

How to divide the available data between different nodes is not an easy-to-solve question. The first option that might cross our minds is to divide the total number of samples into the number of nodes available, into *disjoint subsets*. So, if we have K nodes and N samples, the task is to put N/K samples on each node, so that all the nodes have the same number of samples, but each node contains different samples. Although it might seem quite simple, this approach can produce acceptable results as long as the number of samples on each node is not too small (some feature ranking methods might suffer from overfitting when having more features than samples).

Then, another approach is the popular *bootstrap sampling*, in which we select subsets of N/K samples for each node, randomly taken, with replacement. The reason to select subsets of N/K samples (instead of N, as in bagging) is to maintain the computational complexity reduction to the same ratio as in the disjoint subsets approach. Notice that it is well-known that bootstrap has some desirable properties, for example for reducing bias in estimates [15], so this is the approach that will be used in the experiments in Section 5.

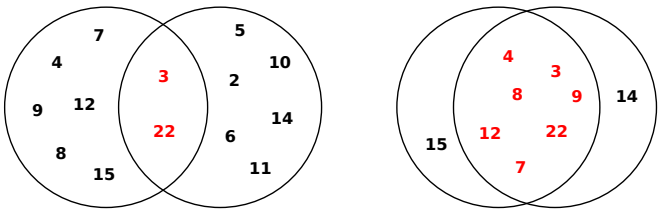
At this point, it is necessary to bear in mind that some feature ranking methods return both a ranking and a score for each feature, such as Mutual Information Maximization (MIM) [28]; while others only return a ranking, such as minimum Redundancy Maximum Relevance (mRMR) [35] or Recursive Feature Elimination for Support Vector Machines (RFE-SVM) [19]. In the case of having both a ranking and a score, after running the feature selection algorithm on each node, for each feature X_m we will have a set of K different scores $\{Score(m,1),\ldots,Score(m,K)\}$. So, instead of

using aggregation methods, we can estimate the score of a given feature $Score(X_m)$ computed as the average of the scores for each feature across the different nodes, such that $Score(X_m) = avg(Score(m, 1), \ldots, Score(m, K))$. However, if we only have a ranking, we will need to use other aggregation methods, which are commented on Section 4.3.

4.2. Distribution by features

Although the most common approach when dealing with distributed learning is to divide the data by samples, it might be also interesting to divide the data by features, especially in some situations in which the number of features is, indeed, the bottleneck of the algorithm —as happens with DNA microarray data [12]. In this case and, unlike the situation in which the distribution is made by samples, the rankings are partial, so their combination is much more challenging.

In this case, it is not possible to distribute the data in *disjoint subsets* of features, since some common features across the different nodes are necessary to connect the partial rankings. Therefore, a certain level of overlap is strictly required when deciding how to distribute the features over the available nodes. The overlap is defined as the subset of features that are common across different nodes. Based on this definition, in the case of horizontal partition (section 4.1), the overlap is complete. In Figure 3 we can see an example of overlap between two nodes. Notice that the level of overlap is something that needs to be carefully examined. If the overlap is too low, as in Figure 3(a), the performance is expected to be very poor, since it would be very difficult to connect the different partial ranks. And, on the contrary, if the overlap is too high, as in Figure 3(b) the performance is expected to be good, but we will not be reducing the complexity of the original problem.



(a) Low overlap (b) High overlap

Figure 3: Example of two cases of overlap between sets of features where the numbers represent the features' ID. Overlapping features are highlighted in red.

A possible strategy is to randomly select the features for each node, in a similar way as bootstrap but with replacement. In this scenario, a natural question that arises is "Which is the minimum number of nodes that we need in order to ensure (in expectation) that a tuple of features will be ranked in a fixed number of nodes?" Because of the random selection, we can answer this question with the following corollary.

Corollary 1. *In order to ensure that, in expectation, each tuple-t (i.e. $t = 2$ for a pair, $t = 3$ for a triplet etc.) of features will be ranked in v nodes, the total number of nodes that we need is given by the following expression:*

$$K = \frac{M!}{(M-t)!}\frac{(m_k - t)!}{m_k!}v, \tag{1}$$

where M is the total number of features and m_k is the number of features per node.

Proof. The total number of features can generate $\binom{M}{t}$ different tuples. On the other hand, in each node we have m_k features, and thus $\binom{m_k}{t}$ different tuples. As a result, in expectation, we need $\binom{M}{t}/\binom{m_k}{t}$ nodes to make sure that every tuple-t is ranked in *one* node. Moreover, if we want to make sure that each tuple-t will be ranked in v nodes, the number of nodes that we need is given by the following expression:

$$K = \frac{\binom{M}{t}}{\binom{m_k}{t}}v = \frac{M!}{(M-t)!}\frac{(m_k - t)!}{m_k!}v$$

□

While this corollary seems counterintuitive, we can further motivate it by exploring the limit cases. If we assume that the number of features per node is equal to the total number of features, i.e. $m_k = M$, then, in order to make sure that each tuple t of features will be ranked in v nodes, we just need v nodes. While, if we assume that the number of features per node is equal to t, then, in order to make sure that each tuple t of features will be ranked in v nodes, we just need $\binom{M}{t} \times v$ nodes in expectation.

Because of the random sampling of the features, we need to make sure that, at the end of the process, all the features were selected at least once to appear on each node. This can be solved, for example, by adding the features not selected (if any) to the last node.

Another way to ensure that all the features are ranked at least once is to force the subsets to have a predefined level of overlap between the features in one node and the remaining features. 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 10% of overlap, we will add to $\mathbf{X}_i$ the 10% of features in $\mathbf{X}_r$, randomly picked. The drawback within this approach is that it is impossible to ensure a certain overlap between *tuples* of nodes, as we did with Corollary 1 for the random selection. In Section 5.1 we will explore how these different proposed strategies behave.

4.3. Aggregation

After partitioning the data into several nodes and applying a feature ranker to each node, we will have different rankings that need to be combined. Several strategies can be used to perform this task, but in this paper we have chosen the following four methods [26]:

- **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 will illustrate the behavior of these methods with a simple example. First, suppose that we are working with complete rankings of features (those obtained when data is partitioned by samples). Imagine that we have 5 different features to be ranked $\{a, b, c, d, e\}$, and we have 5 different rankings of them $R_1, R_2, ..., R_5$, as can be seen in Table 2. The last four columns of the table illustrate the calculations made by each method. For example, the method *best rank* computes the best value achieved by each feature along the different rankings, where best means the highest position. Notice that with this method, there was a tie. When this happens, this method returns the

Table 2: Example of how the aggregation methods work with complete rankings.

Element	R_1	R_2	R_3	R_4	R_5	best rank	median	arith mean	geom mean
a	1	2	3	1	1	1	1	1.6	1.4
b	2	1	1	2	3	1	2	1.8	1.6
c	3	3	2	5	2	2	3	3.0	2.8
d	4	4	5	3	4	3	4	4.0	3.9
e	5	5	4	4	5	4	5	4.6	4.6

elements which are tied in their original position. Therefore, in this example, all the methods will return the ranking $\{a, b, c, d, e\}$.

When the data is distributed by features, each node cannot rank all the features, so the rankings are called partial. Now, imagine that we have 6 different features to be ranked $\{a, b, c, d, e, f\}$, and 3 nodes, such that 2 elements 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 3, in which elements not present in a given node are being assigned the last position in the ranking, according to the implementation provided in [26] (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 $\{a, f, c, d, b, e\}$.

Table 3: 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

Notice that some of these methods are more likely to have to deal with ties than others (e.g. the *best rank* method is prone to have ties, since the set of possible values obtained by *best rank* is much more reduced than the

possible values received by *arith mean*). A further discussion about this issue can be found in Section 5.2.

5. Experiments

In this section we empirically evaluate some of the strategies commented on the previous section. We use six datasets from the UCI repository which are detailed in the first six rows of Table 4. These are chosen to have a variety of sample-feature ratios, and a range of multi-class problems. Moreover, we have included seven widely-used binary microarray datasets [12], which are available for download in [1, 44, 2]. These data sets are known to have many redundant features. This fact complicates the problem of vertically distributed feature ranking because if two redundant (and relevant) features are in the same node, a good feature ranking method can deal with this. But, if these two redundant features are in different nodes, the partial rankings built on each node cannot take care of this redundancy among them. However, we will show good performance results. 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. In order to estimate mutual information of continuous features, the datasets were discretized, using an equal-width strategy into 5 bins.

In the following sections, we investigate the questions: "how can we deal with overlap between features?", "which is the best aggregation method?", "which is the behavior of the different feature ranking methods when distributing the data?", and, finally, "which is the relationship of these results with the classification accuracy?". To address these questions, we use the datasets detailed in Table 4.

Notice that, in the experiments carried out in this section, we do not take into account the computational time, except the fact that the training time is assumed to be reduced when dividing the data into the available nodes. 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.

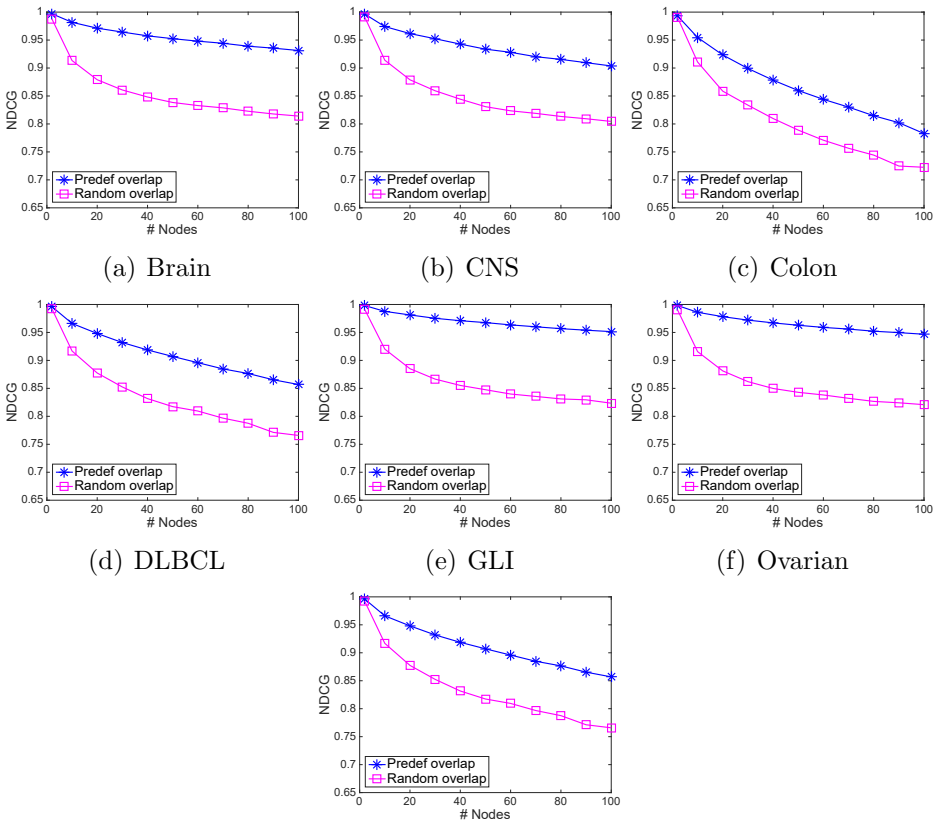
Table 4: Summary description of the datasets used in experiments.

Dataset	Features	Samples	Classes
Ionosphere	34	351	2
Landsat	36	6435	6
Lungcancer	56	32	3
Semeion	256	1593	10
Soybeansmall	35	47	4
Waveform	40	5000	3
Brain	12625	21	2
CNS	7129	60	2
Colon	2000	62	2
DLBCL	4026	47	2
GLI	22283	85	2
Ovarian	15154	253	2
SMK	19993	187	2

5.1. Which options are indicated for dealing with overlap when distributing by features?

In Section 4.2 we commented on the different options that we had to ensure some overlap between the features on each node, when distributing the data by features, which is essential for a correct integration of the partial rankings. In Figures 4 and 5 we can see a comparison between forcing a predefined level of overlap (in this case, 50%) and selecting randomly the features to go to each node.

For these experiments, we have chosen the microarray datasets and three of the UCI datasets (Ionosphere, Landsat and Waveform). In the case of UCI datasets, we divided the data into up to 10 nodes, while in the case of microarray datasets, and because they have a larger number of features, we distributed the data into up to 100 nodes. For simplicity, the MIM feature ranking method and the *best rank* aggregation method were chosen; the experiments were repeated 100 times and we are showing the average NDCG values (obtained by comparing the combined ranking with the ranking obtained with the whole data). As the NDCG measure tends to obtain higher results if the number of elements to rank is high (as happens with microarray data), it is common to compare only the top ranked elements. For instance, when evaluating the performance of web search engines, it is common to com-



(a) Brain (b) CNS (c) Colon

(d) DLBCL (e) GLI (f) Ovarian

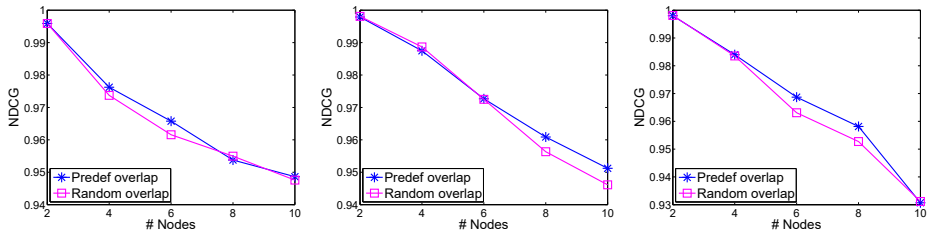
(g) SMK

Figure 4: Experiments with different numbers of features per node and different techniques for dealing with overlap (50% predefined overlap and random sampling) on microarray datasets. NDCG measures the similarity with the ideal ranking.

pare only the top 10 entries. In DNA microarray analysis, it is also common to focus only on the top ranked features, so in the experiments concerning microarray datasets 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 from the experimental results shown in Figure 4, 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.



(a) Ionosphere (b) Landsat (c) Waveform

Figure 5: Results of the experiments for measuring similarity (by using NDCG) with different numbers of features per node and different techniques for dealing with overlap (50% predefined overlap and random sampling), on Waveform, Landsat and Ionosphere dataset

However, focusing on the results from Figure 5 that test datasets with a much smaller number of features, we can see that using a predefined level of overlap very slightly outperforms the results achieved when the overlap is random, although in light of the small differences, the authors cannot recommend one method over the other in this setting.

It is necessary to bear in mind that, when picking random subsets of features for each node, the level of overlap cannot be determined nor chosen a priori (in fact, it might be even possible that some features are never selected, and hence they are never ranked). However, if we want to control this, we can use the formula presented in Eq.(1), which allows us to estimate how many nodes we need if we want to ensure a certain number of features present in tuples of nodes.

Table 5 shows, according to Eq. (1), the number of nodes necessary for datasets Waveform, Landsat and Ionosphere (check characteristics in Table 4) when we require different numbers of features per node (10, 20 and 30), subject to the restriction that each combination of features (single feature, pairs of features, triplets of features) goes to at least one node ($v = 1$). Notice that, even with these datasets that do not have a very large number of features, the number of nodes necessary increases exponentially when increasing the requirements, unless we put a large number of features per node.

Taking into account this estimation about the desirable number of nodes, we have performed several experiments on these datasets with MIM filter and

Table 5: Estimated number of nodes necessary for different conditions computed from Eq. (1)

	Waveform			Landsat			Ionosphere		
	Feats per node			Feats per node			Feats per node		
Combinations	10	20	30	10	20	30	10	20	30
Single	4	2	2	4	2	2	4	2	2
Pairs	18	5	2	14	4	2	13	3	2
Triplets	83	9	3	60	7	2	50	6	2

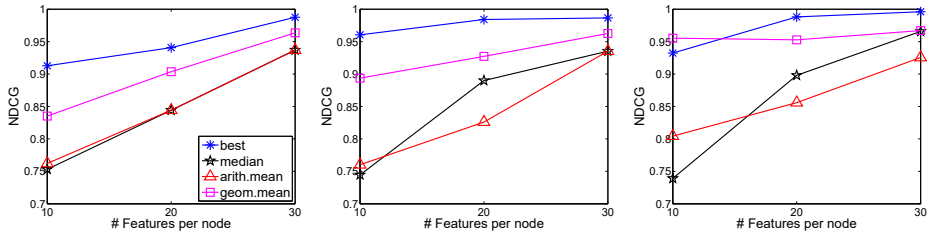
four different aggregation methods. The results can be seen in Figure 6, in which the experiments were run 100 times and we are showing the average.

As can be seen, even when using the number of nodes suggested by Eq. (1) (which can be consulted in Table 5), and in theory ensuring that each single feature, pair or triplet of features is present in at least one node, in some cases the obtained rankings are far from the ideal (i.e. the ranking obtained with the whole dataset, NDCG=1). Again, there results reinforce the idea on how complex dealing with partial ranking is. Also, for the three datasets considered, it is interesting to see that the improvement when using triplets instead of pairs of features is not very notable, so this is an indication that perhaps it is better to use simpler models.

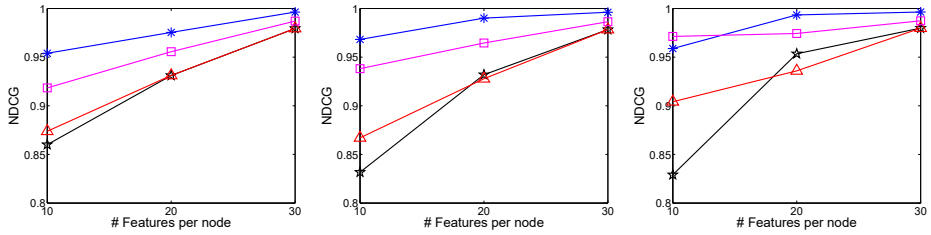
Finally, it is worth noticing the different performance achieved when using the four different aggregation methods considered (see Figure 6), being *best rank* the one obtaining the best results in this case. The next subsection will try to shed light on the issue of which of those is the best aggregation method on the datasets we choose.

5.2. Which is the best aggregation method?

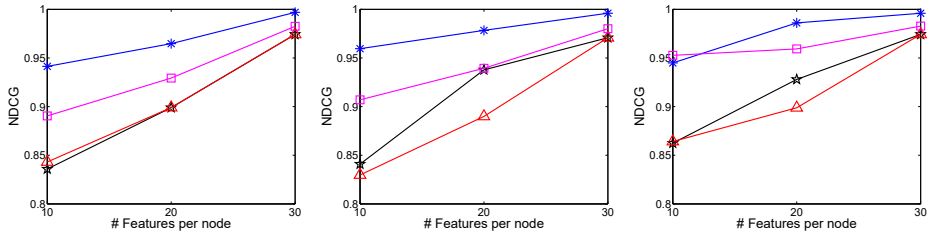
In Section 4, we mentioned that there are several strategies that can be used to combine the partial rankings into a final ranking of features. Specifically, we chose four different methods, named *best rank*, *arith mean*, *geom mean* and *median*. As could be seen in the previous subsection, these four methods lead to very different NDCG values, so in this subsection we will perform some experiments to see if some of the aggregation methods are superior to the others in both cases considered, i.e. partitioning by samples and by features.



(a) Waveform, Single (b) Waveform, Pairs (c) Waveform, Triplets



(d) Landsat, Single (e) Landsat, Pairs (f) Landsat, Triplets



(g) Ionosphere, Single (h) Ionosphere, Pairs (i) Ionosphere, Triplets

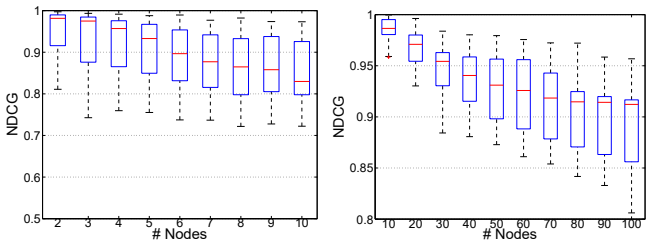
Figure 6: NDCG results on the experiments with different numbers of features per node and different numbers of combinations (tuples), on Waveform, Landsat and Ionosphere dataset

First, we study the behavior of the aggregation methods when the partition of the dataset is made by samples. Since the number of samples of the UCI datasets presented in Table 4 ranges from 32 to 6435, the number of nodes has to vary accordingly. Thus, we perform separate experiments. For the UCI datasets with a small number of samples (Ionosphere, Lungcancer and Soybeansmall), we tested up to 10 nodes. Figure 7 (left column) shows NDCG values on average over these three datasets, with 100 repetitions, for the four aggregation methods considered. In both cases we are using MIM as feature ranking method. As can be seen, *best rank* is more affected by the increase in the number of nodes and shows a poorer performance than

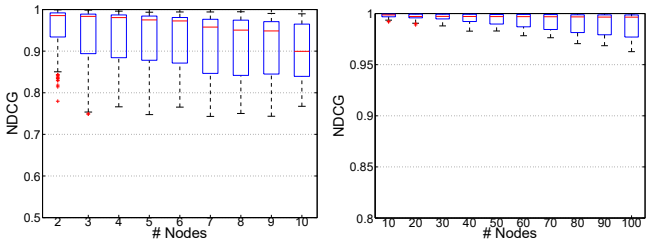
their counterparts. Figure 7 (right column) shows also the averaged NDCG values, but in this case we selected the UCI datasets with the highest number of samples (Landsat, Semeion and Waveform) and so we tested a number of nodes between 10 and 100. Again, the *best rank* method is clearly inferior to the other approaches.

Trying to understand the poor performance of *best rank* in this scenario, let us recall how the aggregation methods behave when dealing with complete rankings, checking the example shown in Table 2. Notice that, having an important number of elements with the same score (as it is likely to happen with *best rank*, and to a lesser extent, with *median*), might produce poor results. In fact, the larger the number of nodes, the smaller the number of samples that go to each node, when dividing the data by samples. And, when losing such an amount of information, it is likely that the rankings are inaccurate, leading to ties and, eventually, to a degradation in the performance of the *best rank* method. In fact, this phenomenon can be seen in Figure 7. This brings up an interesting line of future research, that might be to develop an aggregation method that can deal more efficiently with draws.

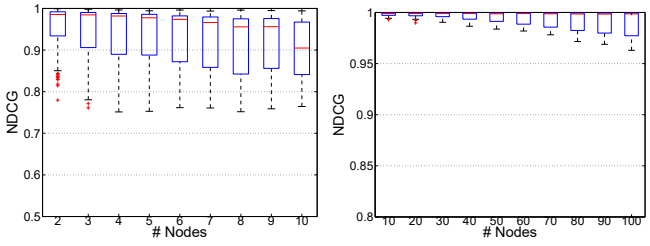
Regarding vertical partitioning, left column of Figure 8 shows the averaged NDCG values over the six UCI datasets in Table 4 (six first rows), when the features are divided into up to 10 nodes, with 50% of overlap, and for 100 repetitions. In contrast to what happened when dividing the data by samples, in this case the *best rank* method seems to be superior to the others, although of course its performance degrades as the number of nodes increases. Finally, right column of Figure 8 shows the averaged NDCG values over the seven microarray datasets, when the features are divided into up to 100 nodes, with 50% of overlap, and for 100 repetitions for the four aggregation methods considered. The results confirm that the *best rank* method clearly outperforms the remaining aggregation methods when distributing by features. Again, trying to understand the reasons behind the behavior of the *best rank* method, let's remember the example depicted in Table 3, 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 8(b). The superiority of the *best rank* method, in this vertical scenario, 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. Thus, another possible line of future work can be to improve the treatment of partial rankings by these



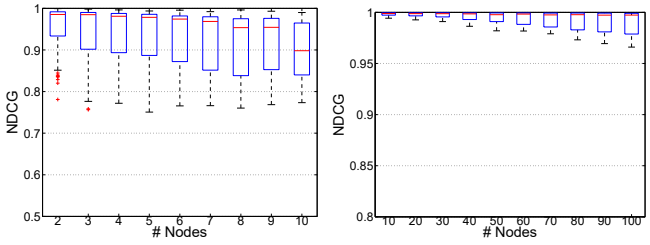
(a) Best rank



(b) Median



(c) Arith mean



(d) Geom mean

Figure 7: Averaged NDCG across 3 datasets for each column and 100 repetitions; when dividing by samples and using MIM feature ranking method. First column represents the UCI datasets with the fewest samples (Ionosphere, Lungcancer and Soybeansmall) and second column represents the UCI datasets with the largest number of samples (Landsat, Semeion and Waveform). 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.

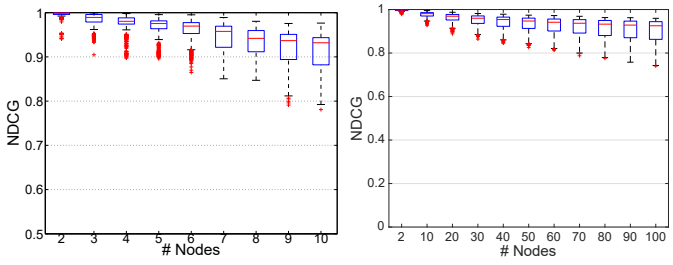
aggregation methods.

In light of these results, the authors suggest the use of the *arith mean* method when dividing by samples, and the *best rank* method when dividing by features.

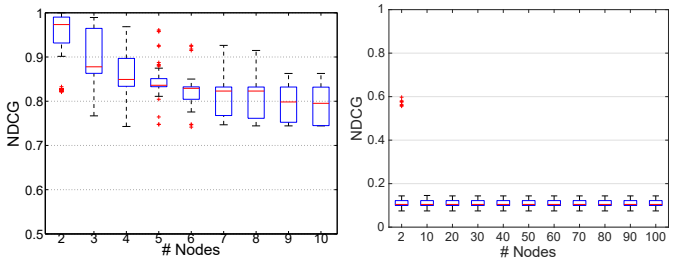
5.3. Which is the behavior of the different feature ranking methods when distributing the data?

In this subsection, we will study the tolerance of some popular feature ranking methods when working with distributed data, and thus forced to deal with incomplete datasets (either because the data was partitioned by samples or by features). 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. In theory, it is expected that univariate methods are more tolerant to incomplete data than multivariate methods, although at the cost of missing feature dependencies. Figure 9 shows the results of a small experiment with Waveform dataset and both MIM (univariate) and Relief [25] (multivariate) feature rankers, when dividing the data by samples into up to 100 nodes and with 100 repetitions. Notice that, since these two methods return both a ranking and a score for each feature, we do not need an aggregation method since the total score is calculated by adding partial scores, as commented in Section 4.1. As can be seen, the results achieved by MIM are better than those obtained by Relief, showing NDCG values close to 1 for any number of nodes tested.

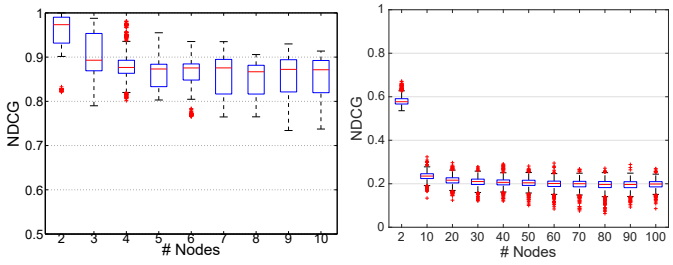
In Figure 10, we can see another small experiment with microarray datasets Colon and DLBCL, comparing univariate method MIM with multivariate method mRMR. This last method was chosen as it is very popular for being applied to microarray datasets. When dealing with microarray data, it is common that researchers employ multivariate methods, since it is well-known that most genes in a microarray experiment are redundant. 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 $\mathcal{O}(NM)$ (where N is the number of samples and M is the number of features) whilst that of mRMR is $\mathcal{O}(NM^2)$. When the number of features is in the order of thousands (as it is the case with microarrays), this increase in complexity becomes, in some cases, unbearable.



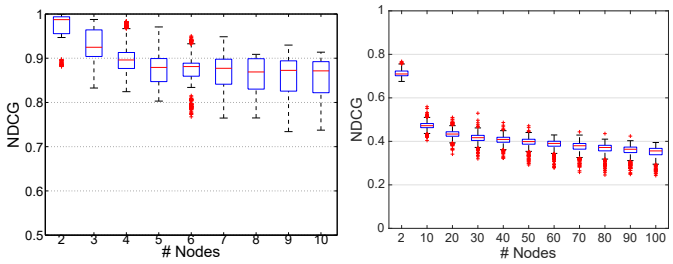
(a) Best rank



(b) Median



(c) Arith mean



(d) geom mean

Figure 8: Averaged NDCG across sevaral datasets and 100 repetitions when dividing by features and using MIM feature ranking method, with 50% overlap. First column uses the 6 first datasets on Table 4 and second column uses 7 microarray datasets on the same table. 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.

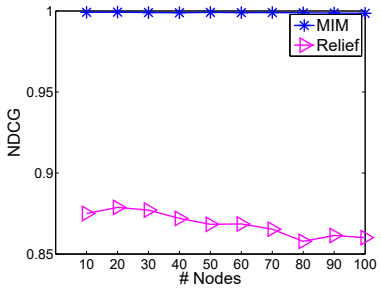
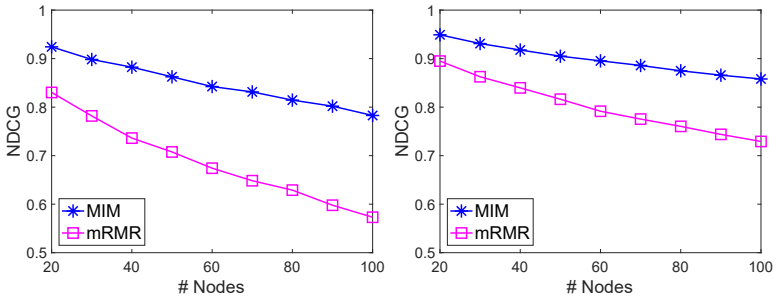


Figure 9: NDCG results when dividing data by samples and estimating the combined score, for a univariate (MIM) and a multivariate (ReliefF) method.

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 Figure 10, the results when using mRMR are worse than when using MIM. For DLBCL dataset, the minimum NDCG obtained with MIM was around 0.87, whereas with mRMR it around 0.73. Even more drastic is the deterioration in the case of Colon, in which the worst NDCG value with mRMR drops below 0.6. In light of these results, when using multivariate methods (and although these are the ones which can benefit more from the reduction in complexity), the information loss when aggregating the rankings affects the results much more than in the case of the less complex univariate methods.



(a) Colon (b) DLBCL

Figure 10: NDCG results for Colon and DLBCL datasets when dividing by features, comparing univariate (MIM) and multivariate (mRMR) methods on microarray data.

From these small experiments we can infer that, in fact, not all the fea-

ture ranking methods are equally tolerant for dealing with incomplete data. Therefore, in order to study this issue in more detail, we have chosen a suite of 7 information theoretic feature selection methods, that also have been analyzed in a previous work [14] (see Table 6 for the details of the methods). Notice that, since not all of these methods return both a ranking and a score for each feature, we need to use aggregation methods to combine the rankings. This new set of experiments will be performed on UCI datasets, due to the computational burden of multivariate methods when dealing with microarray data.

Table 6: Feature ranking methods used in the experiments.

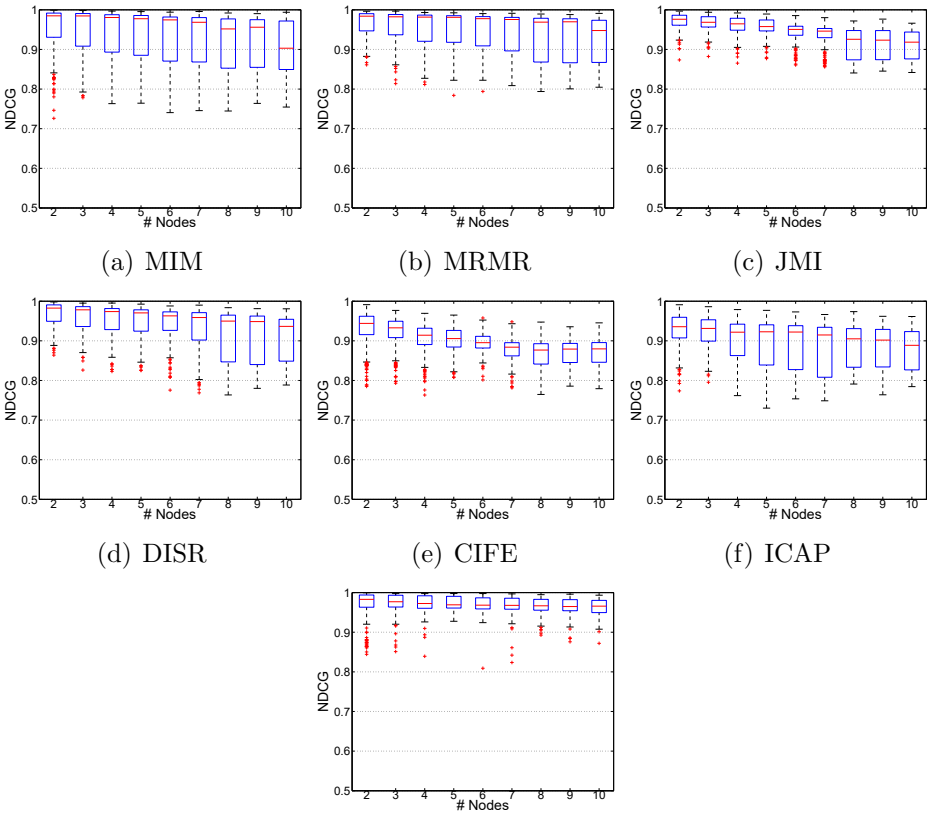
Method	Uni/Multivariate	Authors
Mutual Information Maximization (MIM)	Univariate	Lewis [28]
minimum Redundancy Maximum Relevance (mRMR)	Multivariate	Peng et al. [35]
Joint Mutual Information (JMI)	Multivariate	Yang & Moody [46]
Double Input Symmetrical Relevance (DISR)	Multivariate	Meyer & Bontempi [33]
Conditional Infomax Feature Extraction (CIFE)	Multivariate	Lin & Tang [31]
Interaction Capping (ICAP)	Multivariate	Jakulin [23]
Conditional Redundancy (CONDRED)	Multivariate	Brown et al. [14]

Analogously to Section 5.2, we start by analyzing the case of dividing by samples, separating the datasets according to the number of samples. So, Figure 11 shows the averaged NDCG values over the three datasets with the smallest number of samples (Ionosphere, Lungcancer and Soybeansmall), dividing them in up to 10 nodes, for 100 repetitions. Since in Section 5.2 we have seen that the aggregation method *arith mean* was the most appropriate for a horizontal distribution, we restricted these experiments to this method.

As expected, the performance of the different feature selection methods worsens as the number of nodes increases, since the loss of information is higher. In light of these results, CONDRED seems to be the best option in this scenario. Notice that in these experiments we are not evaluating the efficiency of feature selection methods in selecting useful features[1], but the tolerance of these methods to deal with incomplete data.

Figure 12 shows also the average NDCG values for 100 repetitions, in this case for datasets Landsat, Semeion and Waveform (with the largest sample

[1]For an exhaustive review of these feature selection methods, please check Brown et al.[14]



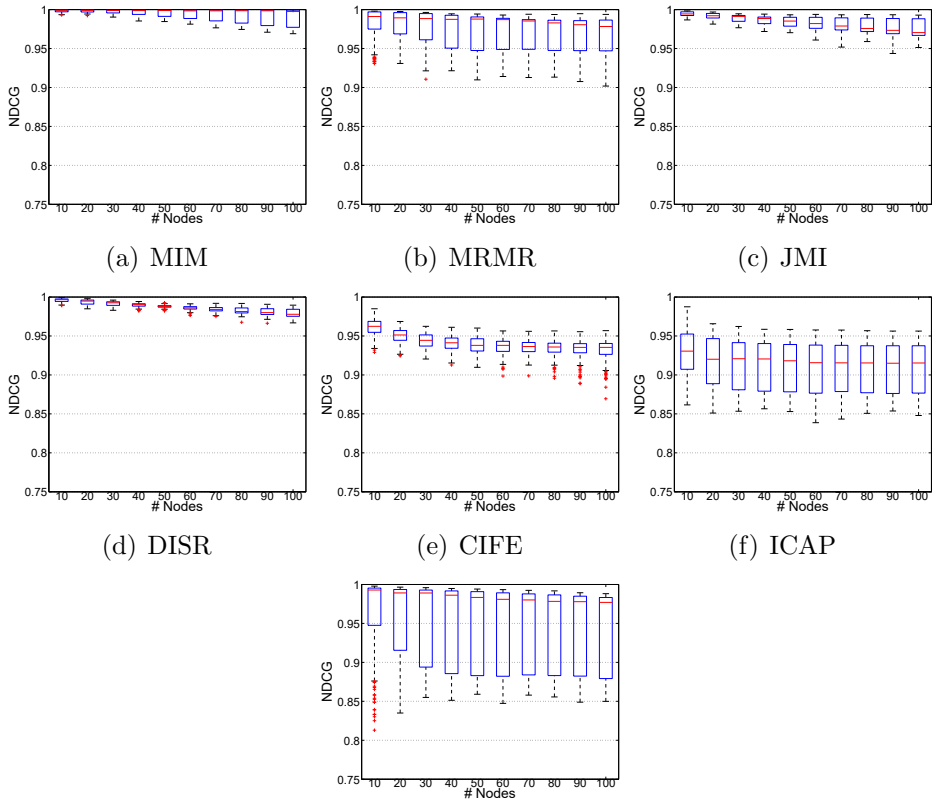
(a) MIM (b) MRMR (c) JMI

(d) DISR (e) CIFE (f) ICAP

(g) CONDRED

Figure 11: Averaged NDCG across the 3 datasets with the smallest number of samples (Ionosphere, Lungcancer and Soybeansmall) and 100 repetitions when dividing by samples and using *arith mean* as aggregation method. 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.

size), dividing the samples in a number of nodes between 10 and 100. In this case, the methods that are more robust to incomplete data are MIM, JMI and DISR, with NDCG values over 0.95, even when dividing the samples in up to 100 nodes.



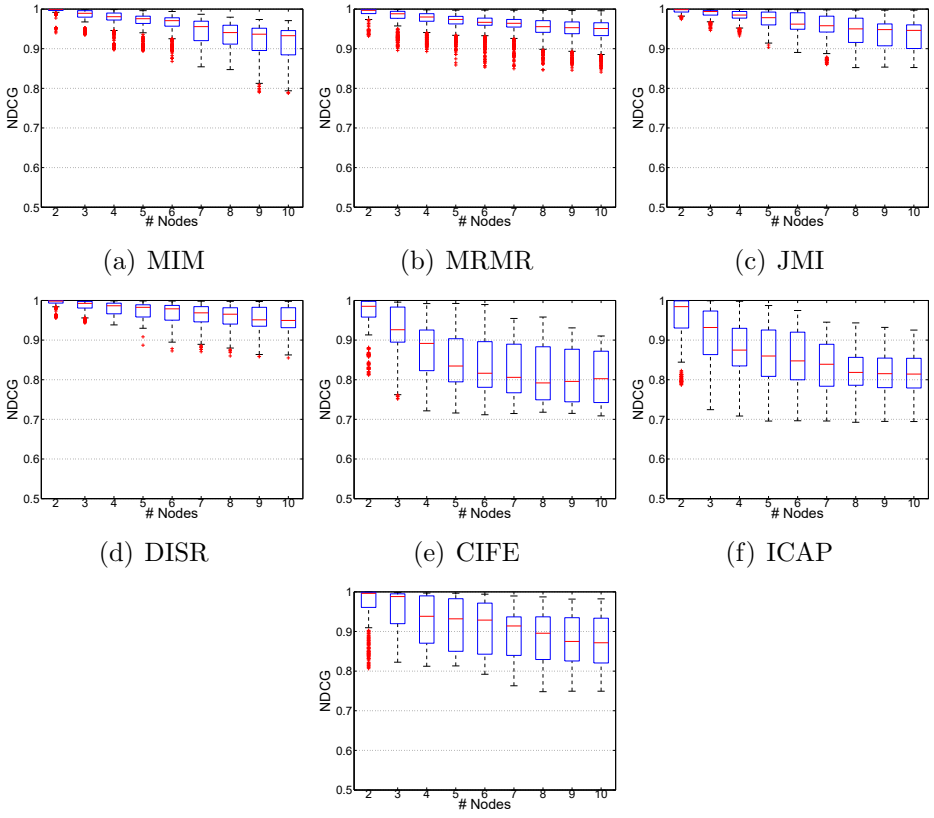
(a) MIM (b) MRMR (c) JMI

(d) DISR (e) CIFE (f) ICAP

(g) CONDRED

Figure 12: Averaged NDCG across the 3 datasets with the largest number of samples (Landsat, Semeion and Waveform) and 100 repetitions, when dividing by samples and using *arith mean* as aggregation method. 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.

Figure 13 shows the averaged NDCG values over the six first datasets described in Table 4, when dividing the features in up to 10 nodes, with 50% of overlap and 100 repetitions. In this case, it is necessary to highlight the good behavior of the methods MIM, mRMR, JMI and DISR, with NDCG median values over 0.9 for any number of nodes tested.



(a) MIM (b) MRMR (c) JMI

(d) DISR (e) CIFE (f) ICAP

(g) CONDRED

Figure 13: Averaged NDCG across the first 6 datasets (Ionosphere, Lungcancer, Soybeans-mall, Landsat, Semeion and Waveform) in Table 4 and 100 repetitions, when dividing by features and using *best rank* as aggregation method, with 50% overlap. 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.

To sum up, it seems that all the methods tested can deal quite satisfactorily with incomplete data, even when losing information. It is necessary to highlight the good performance shown by the MIM and JMI methods, for any scenario tested. In fact, JMI was identified in [14] as having the three desirable characteristics of an information-based selection criteria: it takes into account conditional redundancy, it keeps a balance between relevance and redundancy and it is usable with small sample sizes. Moreover, it presented the best trade-off (in the Pareto-optimal sense) of accuracy and stability. JMI has also shown to have superior performance than other feature selection methods based on mutual information in semi-supervised [40] and multi-label data scenarios [41].

Nevertheless, the results presented in this subsection only evaluate if these feature selection methods are able to build a final ranking from incomplete rankings as similar as possible to the ranking obtained with the whole dataset. To measure the classification accuracy that is lost in the process is a different question, which will be discussed in the next subsection.

5.4. 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. It was demonstrated in a previous work [9] that certain classifiers can obtain outstanding accuracy levels even when the subset of features is not optimal and, on the contrary, other classifiers perform their own embedded feature selection. For those reasons, in these experiments we use a simple nearest neighbor classifier ($k = 3$), as it makes few (if any) assumptions about the data, and avoids also the need for parameter tuning.

Figure 14 shows the classification accuracy between 0 and 1 obtained by a 3-NN classifier on the six UCI datasets presented in the first six rows of Table 4 using the four feature selection methods that reported good results both in the previous subsection and in a previous work [14]: MIM, mRMR, JMI and DISR. 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 ("ideal" case), with the rankings achieved when distributing the data by samples and by features. In the former case, we used the aggregation method *arith mean*, whereas in the latter we used *best rank*, as suggested in Section 5.2. In both cases we divided the data in 10 nodes and, in the case of distribution by features, we added a 50% of overlap

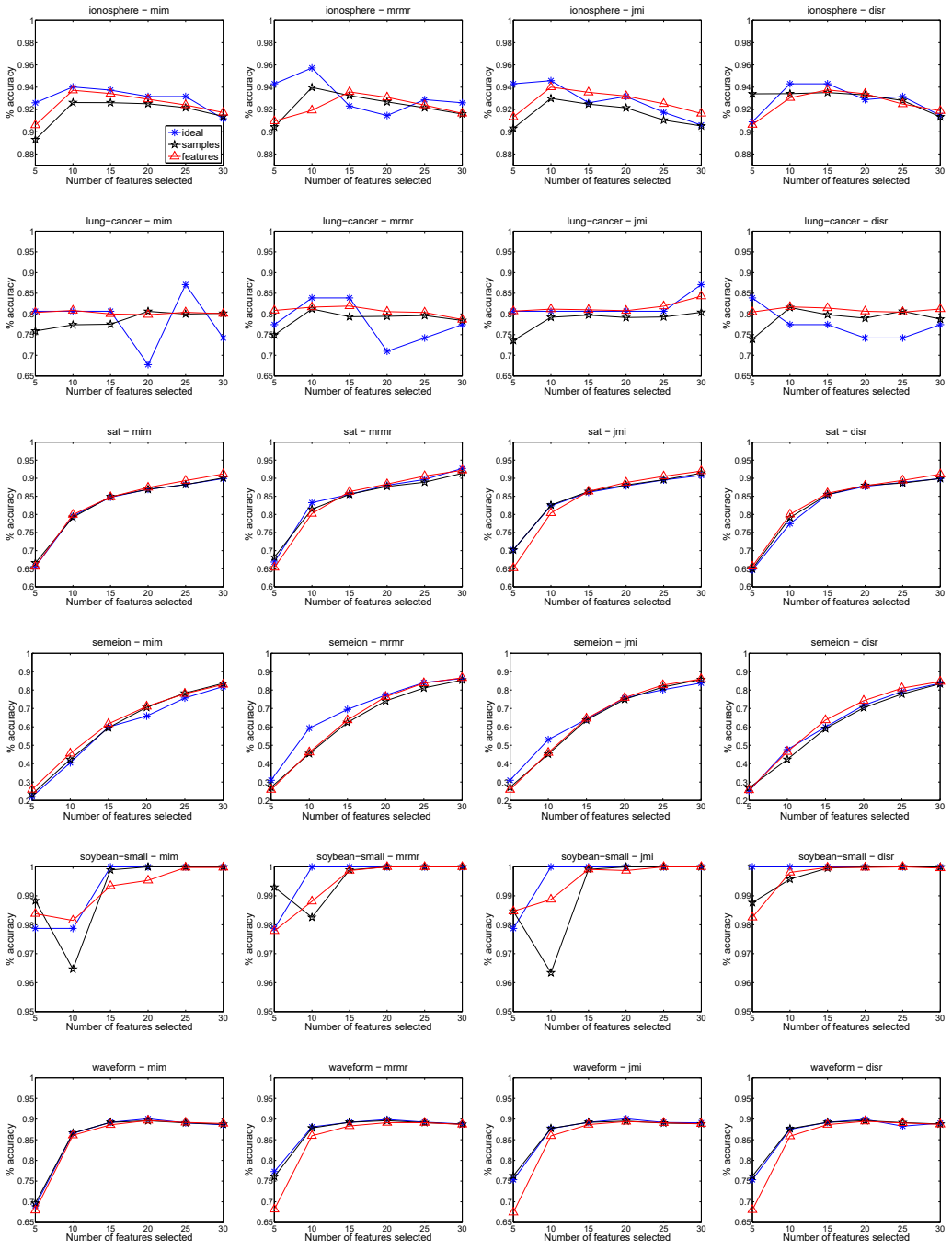


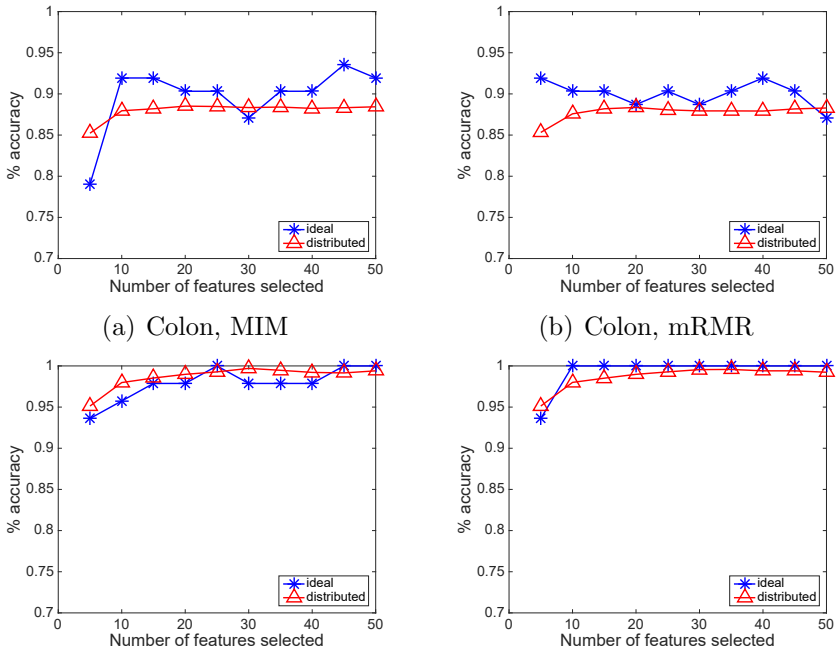
Figure 14: Classification accuracy obtained with a 3-NN classifier on the first 6 datasets on Table 4 and reported on average for 100 repetitions. The three curves in each graph correspond to the ideal situation (all data in a unique node), and two distributed situations, "samples" correspond to an horizontal distribution (by samples), and "features" to a vertical distribution (by features). When data is divided by samples, the *arith mean* aggregation method was used while when data is divided by features, the *best rank* aggregation method was employed.

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 30, with increments of 5. For calculating the classification accuracy of both distributed approaches, the whole process was repeated and averaged 100 times.

The experimental results show two clear tendencies according to the number of samples of the datasets tested. In the case of datasets with the highest number of samples (such as Landsat, Semeion and Waveform), the loss of accuracy when applying a distributed approach is not relevant. For Waveform dataset, the worst option seems to be the distribution by features, which is reasonable since distributing 40 features into 10 nodes intuitively implies an important loss of information. On the other hand, the datasets with a small number of samples (Ionosphere, Lungcancer and Soybeansmall) show more pronounced differences between distributed and centralized approaches. It is also interesting to see that in some datasets having more features contributes to better classification performances whereas in other datasets the contrary happens. However, analyzing the importance of specific features in these datasets and their relation with classification accuracy is out of the scope of this paper.

Finally, Figure 15 shows the classification accuracy obtained by a 3-NN classifier on Colon and DLBCL microarray datasets using MIM (univariate) and mRMR (multivariate) feature selection methods. As aggregation method, we used *best.rank*, since we are distributing by features, and in this case we divided the data into up 50 nodes and we added a 50% of overlap between nodes. As 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.

These 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 [27]. Again, analyzing specific aspects related to classification is not the goal of this paper.



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

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

Figure 15: 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.

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

6. Conclusions and future work

This work has presented an overview of the different strategies than can be performed when distributing the feature ranking process. Whether the partition is being made by features or by samples, 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:

- **How to distribute by features?** In this case it is essential to introduce some level of overlap between the features appearing in each node, to ensure a correct combination of the partial rankings. We suggest two different ways to ensure this overlap—establishing a desired level of overlap or using a random level of overlap. The experimental results showed that they have similar performance, thus both techniques can be employed in practical applications. The aggregation method that works better in this scenario is the *best rank*.

- **How to distribute by samples?** In this case to divide the available data in the different nodes, the authors recommend using bootstrap samples and *arith mean* as an aggregation method.

- **Which feature selection algorithm performs better?** Regarding the tolerance of feature ranking methods to incomplete data, we found that, among a suite of information theoretic feature selection methods, MIM and JMI show a good performance, which is interesting since these methods also demonstrated a good trade-off accuracy/ stability in a previous work [14].

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. Also, the problem of obtaining imbalanced subsets when distributing datasets or the data set is already imbalanced in its original distribution, can be an interesting line of research. Furthermore, the distribution in both samples and features simultaneously might also be explored. It is also interesting to note that the methodology presented in this work is also directly applied to graph learning tasks, and more particularly structure learning of Bayesian networks. All of the information based feature selection methods we used can be seen as approximate iterative maximizers of the conditional mutual information (CMI) [14]. Markov Blanket (MB) discovery algorithms, such as IAMB, can be also seen as a special case of the above framework. As a result this methodology can be naturally used with MB discovery algorithms. Markov Blanket is an important concept that links the feature selection with the structure learning. Using Pellet & Elisseef's [34] wording "Feature selection and causal structure learning are related by a common concept: the Markov blanket." Pellet & Elisseef [34] presented how a generic feature selection algorithm returning strongly relevant variables can

be turned into a causal structure-learning algorithm. Our work can thus be used to derive distributed versions of the above structure-learning procedure.

Acknowledgments

This research has been economically supported in part by the Spanish Ministerio de Economía y Competitividad and FEDER funds of the European Union through the research project TIN2015-65069-C2-1-R; and by the Consellería de Industria of the Xunta de Galicia through the research project GRC2014/035. Financial support from the Xunta de Galicia (Centro singular de investigación de Galicia accreditation 2016-2019) and the European Union (European Regional Development Fund - ERDF), is gratefully acknowledged (research project ED431G/01). V. Bolón-Canedo acknowledges support of the Xunta de Galicia under postdoctoral Grant code ED481B 2014/164-0.

References

[1] Feature Selection Datasets at Arizona State University. `http://featureselection.asu.edu/datasets.php`. [Online; accessed September-2018].

[2] Kent Ridge Bio-Medical Dataset. `http://datam.i2r.a-star.edu.sg/datasets/krbd`. [Online; accessed September-2018].

[3] Amancio, D. R. (2015). A complex network approach to stylometry. *PLoS One*, 10(8):e0136076.

[4] Amancio, D. R., Oliveira Jr, O. N., and Costa, L. d. F. (2012). Structure–semantics interplay in complex networks and its effects on the predictability of similarity in texts. *Physica A: Statistical Mechanics and its Applications*, 391(18):4406–4419.

[5] Arrow, K. J. (1951). *Social choice and individual values*. Wiley.

[6] Austin, J. (1996). AURA, a distributed associative memory for high speed symbolic reasoning. *Connectionist Symbolic Integration. Kluwer*.

[7] Banerjee, M. and Chakravarty, S. (2011). Privacy preserving feature selection for distributed data using virtual dimension. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 2281–2284. ACM.

[8] Bekkerman, R., Bilenko, M., and Langford, J. (2011). *Scaling up machine learning: Parallel and distributed approaches*. Cambridge University Press.

[9] Bolón-Canedo, V., Sánchez-Maroño, N., and Alonso-Betanzos, A. (2013). A review of feature selection methods on synthetic data. *Knowledge and Information Systems*, 34(3):483–519.

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

[11] Bolón-Canedo, V., Sánchez-Maroño, N., and Alonso-Betanzos, A. (2015b). Recent advances and emerging challenges of feature selection in the context of big data. *Knowledge-Based Systems*, 86:33–45.

[12] Bolón-Canedo, V., Sánchez-Marono, N., Alonso-Betanzos, A., Benítez, J. M., and Herrera, F. (2014). A review of microarray datasets and applied feature selection methods. *Information Sciences*, 282:111–135.

[13] Bolón-Canedo, V., Sechidis, K., Sánchez-Marono, N., Alonso-Betanzos, A., and Brown, G. (2017). Exploring the consequences of distributed feature selection in dna microarray data. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN*, pages 1665–1672. INNS.

[14] Brown, G., Pocock, A., Zhao, M.-J., and Luján, M. (2012). Conditional likelihood maximisation: a unifying framework for information theoretic feature selection. *The Journal of Machine Learning Research*, 13(1):27–66.

[15] Chernick, M. R. (2011). *Bootstrap methods: A guide for practitioners and researchers*, volume 619. John Wiley & Sons.

[16] Costa, L. d. F., Rodrigues, F. A., Travieso, G., and Villas Boas, P. R. (2007). Characterization of complex networks: A survey of measurements. *Advances in Physics*, 56(1):167–242.

[17] Gupta, P., Sharma, A., and Jindal, R. (2016). Scalable machine-learning algorithms for big data analytics: a comprehensive review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 6(6):194–214.

[18] Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3(Mar):1157–1182.

[19] Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422.

[20] Hand, D. J., Mannila, H., and Smyth, P. (2001). *Principles of data mining*. MIT press.

[21] Hellerstein, J. M., Ré, C., Schoppmann, F., Wang, D. Z., Fratkin, E., Gorajek, A., Ng, K. S., Welton, C., Feng, X., Li, K., et al. (2012). The madlib analytics library: or mad skills, the sql. *Proceedings of the VLDB Endowment*, 5(12):1700–1711.

[22] Hodge, V. J., OKeefe, S., and Austin, J. (2016). Hadoop neural network for parallel and distributed feature selection. *Neural Networks*, 78:24–35.

[23] Jakulin, A. (2005). *Machine learning based on attribute interactions*. PhD thesis, Univerza v Ljubljani.

[24] Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446.

[25] Kira, K. and Rendell, L. A. (1992). A practical approach to feature selection. In *Proceedings of the ninth International Workshop on Machine Learning*, pages 249–256. Morgan Kaufmann Publishers Inc.

[26] Kolde, R., Laur, S., Adler, P., and Vilo, J. (2012). Robust rank aggregation for gene list integration and meta-analysis. *Bioinformatics*, 28(4):573–580.

[27] Kuncheva, L. I. (2014). Ensemble methods. *Combining Pattern Classifiers: Methods and Algorithms, Second Edition*, pages 186–229.

[28] Lewis, D. D. (1992). Feature selection and feature extraction for text categorization. In *Proceedings of the Workshop on Speech and Natural Language*, pages 212–217. Association for Computational Linguistics.

[29] Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., and Liu, H. (2017). Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, 50(6):94.

[30] Li, T., Zhang, C., and Ogihara, M. (2004). A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. *Bioinformatics*, 20(15):2429–2437.

[31] Lin, D. and Tang, X. (2006). Conditional infomax learning: an integrated framework for feature extraction and fusion. In *Computer Vision–ECCV 2006*, pages 68–82. Springer.

[32] Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., et al. (2016). Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research*, 17(1):1235–1241.

[33] Meyer, P. E. and Bontempi, G. (2006). On the use of variable complementarity for feature selection in cancer classification. In *Applications of Evolutionary Computing*, pages 91–102. Springer.

[34] Pellet, J.-P. and Elisseeff, A. (2008). Using markov blankets for causal structure learning. *Journal of Machine Learning Research*, 9(Jul):1295–1342.

[35] Peng, H., Long, F., and Ding, C. (2005). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8):1226–1238.

[36] Prasad, B. R., Bendale, U. K., and Agarwal, S. (2016). Distributed feature selection using vertical partitioning for high dimensional data. In *Advances in Computing, Communications and Informatics (ICACCI), 2016 International Conference on*, pages 807–813. IEEE.

[37] Raymer, M. L., Punch, W. F., Goodman, E. D., Kuhn, L. A., and Jain, A. K. (2000). Dimensionality reduction using genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 4(2):164–171.

[38] Saeys, Y., Inza, I., and Larrañaga, P. (2007). A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517.

[39] Sarkar, S. D., Goswami, S., Agarwal, A., and Aktar, J. (2014). A novel feature selection technique for text classification using naive bayes. *International Scholarly Research Notices*, 2014.

[40] Sechidis, K. and Brown, G. (2018). Simple strategies for semi-supervised feature selection. *Machine Learning*, 107(2):357–395.

[41] Sechidis, K., Nikolaou, N., and Brown, G. (2014). Information theoretic feature selection in multi-label data through composite likelihood. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 143–152. Springer Berlin Heidelberg.

[42] Sharma, A., Imoto, S., and Miyano, S. (2012). A top-r feature selection algorithm for microarray gene expression data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 9(3):754–764.

[43] Silva, T. C. and Amancio, D. R. (2012). Word sense disambiguation via high order of learning in complex networks. *EPL (Europhysics Letters)*, 98(5):58001.

[44] Statnikov, A., Aliferis, C., and Tsamardinos, I. Gems: Gene expression model selector. `http://www.gems-system.org`. [Online; accessed September-2018].

[45] Wu, X., Zhu, X., Wu, G.-Q., and Ding, W. (2014). Data mining with big data. *IEEE Transactions on Knowledge and Data Engineering*, 26(1):97–107.

[46] Yang, H. H. and Moody, J. E. (1999). Data visualization and feature selection: New algorithms for nongaussian data. In *NIPS*, pages 687–702. Citeseer.

[47] Zadeh, S. A., Ghadiri, M., Mirrokni, V. S., and Zadimoghaddam, M. (2017). Scalable feature selection via distributed diversity maximization. In *AAAI*, pages 2876–2883.

[48] Zhao, L., Chen, Z., Hu, Y., Min, G., and Jiang, Z. (2016). Distributed feature selection for efficient economic big data analysis. *IEEE Transactions on Big Data*.

[49] Zhao, Z., Zhang, R., Cox, J., Duling, D., and Sarle, W. (2013). Massively parallel feature selection: an approach based on variance preservation. *Machine Learning*, 92(1):195–220.