Tackling the supervised label ranking problem by bagging weak learners

Juan A. Aledo[*], José A. Gámez[†], David Molina[‡]

Abstract

Preference learning is the branch of machine learning in charge of inducing preference models from data. In this paper we focus on the task known as *label ranking problem*, whose goal is to predict a ranking among the different labels the class variable can take. Our contribution is twofold: (i) taking as basis the tree-based algorithm LRT described in [1], we design weaker tree-based models which can be learnt more efficiently; and (ii) we show that bagging these weak learners improves not only the LRT algorithm, but also the state-of-the-art one (IBLR [1]). Furthermore, the bagging algorithm which takes the weak LRT-based models as base classifiers is competitive in time with respect to LRT and IBLR methods. To check the goodness of our proposal, we conduct a broad experimental study over the standard benchmark used in the label ranking problem literature.

Keywords: Supervised classification, label ranking problem, bagging, ensemble, Mallows model, decision tree.

[*]Department of Mathematics, University of Castilla-La Mancha
[†]Department of Computer Systems, University of Castilla-La Mancha
[‡]Department of Computer Systems, University of Castilla-La Mancha

1 Introduction

Preferences are comparative judgments about a set of alternatives, choices or options. The goal of preference (choice) modeling is to study individual or collective decision processes and procedures from a set of previously stated preferences. *Preference Learning* [2] has arisen as a new branch of machine learning, with the goal of inducing preference models from data which contain information on the past preferences of some individuals. Once the model is learnt, it can be used to predict preferences in future scenarios.

Although a big deal of the research on preference learning has been related to *recommender systems* [3] or to the *learning to rank problem* [4], in the last years there has been a growing interest in studying rank data from a data mining perspective [5]. In this paper we follow this direction. In particular, we focus on a task known as *label ranking problem* [1], whose goal is to predict a ranking among a set of labels given the value of the predictive attributes. As an example, suppose that we want to recommend to a forthcoming student a ranked list of the degrees which can be studied in our University. For instance, we could recommend maths $\succ$ computer science $\succ$ biology $\succ$ medicine to one student with good skills in mathematics and programming, and biology $\succ$ medicine $\succ$ maths $\succ$ computer science to other student with good marks in chemistry and natural sciences but who does not like computers. The task has resemblance to supervised classification, in the sense that we have several predictive attributes (e.g. high school marks on maths, physics, chemistry, etc., IQ score, age, etc.) and a distinguished target variable taking values in a set of disjoints labels ({maths, computer science, biology, medicine}). However there are two important differences:

- The goal is not to predict the best class label for an unseen student, but to provide a *ranking* of the class labels, by ordering first the degree we think best fits to the student, then the second one, etc.
- We use how previous students have ranked the degrees according to their abilities and preferences. Thus, our training instances will be labelled with (possibly incomplete) rankings of the available degrees, which will be used to train the label ranker.

Two problems somewhat related to label ranking, although quite different from the point of view of the machine learning task they carry out, are *ordinal classification* [6] and *learning to rank* [4].

In ordinal classification a ranking is defined among the class labels. However, the instances are labelled with a single label and the machine learning task consists in the induction of a *standard* classifier, but exploiting the inner structure of the class variable during the learning process. Learning to rank is a classical problem in *information retrieval*, although it also has been applied to other fields as machine translation, computational biology and recommender systems. In its basic form, it outputs a ranked collection of documents given an input query, although it also refers to more complex settings. In the *listwise* approach to learning to rank [7], the information retrieval task is helped by using machine learning. In this framework, the input instances contain a query, a list of relevant documents for the query and a rating for each document. The ranking of the documents is then obtained from the ratings. However, for the machine learning process different feature vectors are used by transforming each instance into a set of triplets (query, document, rating), which are used to learn a model $f(\text{query},\text{document}) \rightarrow \text{rating}$. Thus, once a new query is received, the information retrieval model gets the relevant documents, which are then ranked by applying the learned model.

In this paper we follow the approach to label ranking introduced in [1]. The goal is to induce a model able to predict complete label rankings by taking advantage in the learning process of all the available information, that is, the (possibly partial) rankings of the instances in the training set.

Methods based on the transformation of the whole problem into a set of single-class classifiers (e.g. label-wise [8, 9], pair-wise approaches [10, 11] or chain classifiers [12]) are not considered, as we aim to deal with all the dependences simultaneously. To do this, we rely on the work of Cheng et al. [1], where they manage the problem in a non-standard classification setting, by designing instance-based (IBLR) and decision/regression tree-based (LRT) classifiers tailored to cope with training instances labelled with a (partial) ranking. In order to do that, rankings are managed properly by using the Mallows probability distribution [13] to model a sample of rankings. Moreover, a proper distance for rankings comparison is used to obtain the *consensus* ranking for the sample [14] (details are provided in Section 2.1). These two algorithms obtain a good performance in comparison with competing approaches [15, 16, 17], IBLR being better than LRT [1] (details in Section 5.2). However, from the computational point of view, IBLR shows two main drawbacks: (i) it does not scale well to datasets having a large number of variables and/or instances, and (ii) it needs far more time at inference/query time than LRT.

Our goal is to improve the performance (accuracy) of the algorithms in [1] by developing new methods based on the LRT algorithm. In particular our main contributions are:

- We design two weak learners based on the LRT algorithm by using unsupervised discretization to select the splitting point. From the complexity study (see Sections 4.1 and 4.2) it follows that the time needed to learn the weak classifiers is reduced proportionally to N (the number of instances in the dataset) with respect to LRT. In practice, when the number of variables grows, they need about 1% of the time needed by the original LRT (see Section 5.4.3).

- We consider the use of ensembles by using bagging [18]. The results show that bagging the weak learners is competitive with the ensemble of LRT in terms of accuracy, but much more efficient in terms of time. In fact, the approach based on applying bagging to the original LRT algorithm is not practical under conditions of restricted CPU time.

  The approach based on bagging the weak learners is competitive (in accuracy) not only with respect to the LRT-based ensemble, but also with respect to the state-of-the-art IBLR algorithm.

- We study the problem of dealing with partial information, that is, the case when the training instances are labelled with an incomplete ranking. In this scenario, our proposals based on bagging significantly outperform the IBLR algorithm, the difference being bigger as the number of missing labels grows.

The paper is structured as follows. In Section 2 we review some basic notions needed to deal with rank data and introduce the *label ranking prediction* problem. In Section 3 we describe the decision tree-based algorithm (LRT) introduced in [1] to deal with the label ranking problem. Section 4 is devoted to detail our proposal. We pay special attention to analyze the complexity of the method described in [1]. In Section 5 we set forth the empirical study carried out to test the methods designed in this paper, analyzing the results in detail. Finally, in Section 6 we provide some conclusions.

2 Preliminaries

In this section we review some notions needed to deal with rank data. Then we properly define the *label ranking prediction* task.

2.1 Dealing with rankings

Rankings are a natural way to express preferences. Specifically, given a set of items $\mathcal{I} = \{1, 2, \ldots, k\}$, a ranking π is an order of preference over (some of) these items. Rankings can be *complete* (the k items are ranked) or *incomplete* (only p items are ranked, $2 \leq p < k$). A ranking is denoted as a vector of items, from most to least preferred, separated by commas.

Complete rankings are permutations of the items in $\mathcal{I}$, i.e. the set of complete rankings on the items of $\mathcal{I}$ is the symmetric group $\mathbb{S}_k$. We use $\widetilde{\mathbb{S}}_k$ to denote the set of (complete or incomplete) rankings on the items of $\mathcal{I}$. Given $\pi \in \widetilde{\mathbb{S}}_k$, we will denote by $\pi(i)$ the i-th ranked element in π. Given $a, b \in \mathcal{I}$, we use $a \succ_\pi b$ to indicate that a precedes b in the ranking π.

2.1.1 Consensus permutation

Given a dataset or sample with N rankings $\mathbf{D} = \{\pi_1, \pi_2, \ldots, \pi_N\}$, $\pi_i \in \widetilde{\mathbb{S}}_k$, the *rank aggregation problem* [14] consists in obtaining the permutation $\pi_0 \in \mathbb{S}_k$ which better represents the rankings contained in the sample. Such a permutation π_0 is known as the *consensus ranking*.

Formally, in the rank aggregation problem we look for the permutation π_0 such that:

$$\pi_0 = argmin_{\pi \in \mathbb{S}_k} \frac{1}{N} \sum_{i=1}^{N} D\left(\pi_i, \pi\right) \tag{1}$$

where $D(\pi, \tau)$, $\pi, \tau \in \widetilde{\mathbb{S}}_k$, is a distance measure which counts the number of item pairs (a, b), $a, b \in \mathcal{I}$, $a < b$, over which π and τ disagree, ignoring those pairs non ranked in both rankings π and τ. There is disagreement over a pair (a, b) ($a, b \in \mathcal{I}$, $a < b$) of items ranked in both π and τ, if the relative order of a and b is different in π and τ. This distance is a generalized version of the Kendall distance, which takes as input two permutations (see for instance [19]). When $\mathbf{D}$ only contains permutations and the Kendall distance is used in (1), this problem is known as the *Kemeny ranking problem* [20].

Computing the consensus permutation is an NP-hard problem. However, good approximate algorithms can be used. In particular, Borda (or Borda count) algorithm [21] deserves to be highlighted because of its good trade-off between efficiency and accuracy [22].

When dealing with complete rankings in $\mathbb{S}_k$, *Borda count* method proceeds as follows: first, for each permutation π in the dataset it assigns $k-i+1$ points to the i-th item of π; then, after processing the whole dataset, it returns the permutation that orders the items from the most valued item to the least valued one. On the other hand, incomplete rankings are managed by using *generalized* Borda count methods [23, 14, 30]. In particular, given an incomplete ranking π we use a method that manages the uncertainty about the non-ranked items by taking into account all the permutations compatible with π (see [1] for the details). Thus, for an incomplete ranking π which ranks p elements, $2 \leq p < k$, the scoring scheme proposed in [1, Proposition 2] assigns $(p-i+1)(k+1)/(p+1)$ points to items in positions $i = 1, .., p$ and $(k+1)/2$ points to non ranked items.

2.1.2 Mallows probability distribution

The Mallows model [13] is a distance-based probability distribution over permutations which belongs to the exponential family. It is defined by two parameters: the central permutation $\pi_0 \in \mathbb{S}_k$ and a spread parameter $\theta \in [0, +\infty)$. It is defined in terms of a distance over permutations, e.g. the Kendall distance (see [24] for details).

Given a permutation π and a Mallows model parameterized by π_0 and θ, the probability assigned to π is

$$P(\pi; \pi_0, \theta) = \frac{e^{-\theta \cdot D(\pi, \pi_0)}}{\psi(\theta)},$$

where $\psi(\theta)$ is a normalization constant.

The Mallows distribution is strongly unimodal, being π_0 the permutation with the highest probability. On the other hand, θ quantifies the concentration of the distribution around its peak π_0. Thus, if $\theta = 0$ we get the uniform distribution, while when $\theta > 0$, the probability of each permutation $\pi \in \mathbb{S}_k$ decreases according to $D(\pi, \pi_0)$ and θ.

2.2 Label ranking prediction problem

In this section we formally define the problem addressed in this paper. As our approach falls in the supervised classification task, we start by introducing this setting. *Standard* supervised classification involves learning a model from a set of labelled data, in order to assign one *class* label or category to every new example. Formally, $\mathbf{D} = \{\mathbf{x}^j, c^j\}_{j=1}^N$ is a dataset containing N labelled instances; $\mathbf{x}^j$ is a configuration of values defined over the n predictive variables or *attributes*, $X_1, X_2, \ldots, X_n$, i.e. $\mathbf{x}^j = (x_1^j, \ldots, x_n^j)$, where $x_i^j \in dom(X_i)$; and $c^j \in dom(C) = \{c^1, \ldots, c^k\}$ is the class label. The goal is to learn a *classifier* model from $\mathbf{D}$

$$\mathcal{C} : dom(X_1) \times dom(X_2) \times \cdots \times dom(X_n) \longrightarrow dom(C),$$

which generalizes well on unseen data.

In the last years, several classification-based tasks have appeared which do not follow the standard definition of the supervised classification setting. This is, for example, the case of weakly supervised problems, where the information on the class is not fully known for the training instances, or the case of non-standard classification problems, where the task is no longer to predict *only* the most probable class label (see [25] for a recent review on this topic). The label-ranking problem can be viewed as a non-standard supervised classification problem, as we know the (partial) rank for all the training instances, but our goal is to predict a complete ranking, not a single label.

We define the model to be induced as follows:

Definition 1 (LR-Classifier)**.** *Given a set of n discrete or numerical predictive variables or* attributes, $X_1, X_2, \ldots, X_n$ and a target class variable C, with domain $dom(C) = \{1, \ldots, k\}$, a *Label Ranking Classifier* (LR-classifier) $\mathcal{C}$ is a function (model):

$$\begin{array}{rrcl} \mathcal{C}: & dom(X_1) \times dom(X_2) \times \cdots \times dom(X_n) & \longrightarrow & \mathbb{S}_k, \\ & (x_1, x_2, \ldots, x_n) & \longmapsto & \pi \end{array}$$

that is, a function that assigns a permutation of the values in $dom(C)$ to any configuration $(x_1, x_2, \ldots, x_n) \in dom(X_1) \times \cdots \times dom(X_n)$.

The goal of the label ranking prediction problem is to learn an LR-classifier from the data. Notice that a LR-classifier always returns a complete ranking. However, as will be immediately detailed, we allow the presence of incomplete rankings in the training instances.

Definition 2 (LR prediction problem)**.** *Given a dataset* $\mathbf{D} = \{(x_1^j, \ldots, x_n^j, \pi^j)\}_{j=1}^N$, *where* $x_i^j \in dom(X_i)$ *and* $\pi^j \in \widetilde{\mathbb{S}}_k$*, the Label Ranking Prediction problem (LR prediction problem) consists in learning an LR-classifier from* $\mathbf{D}$ *which generalizes well on unseen data.*

3 Decision tree algorithm for label ranking

Decision tree induction is one of the most used machine learning methods, both for classification [26] and regression tasks [27]. In this section we describe the algorithm proposed in [1] to adapt the decision tree induction algorithm to the label ranking problem.

As its name indicates, a decision tree is a tree-shaped decision model which is used to make predictions given an input. In contrast to classification [26] or regression/model trees [27, 28, 29], in the label ranking problem each leaf contains a complete label ranking [1] (Figure 1).

As in [1], we assume that there is no missing values in the training set regarding the predictive attributes, but that some instances can be labelled with incomplete rankings. We describe the *complete* case and comment on the particularities of dealing with *incomplete* rankings.

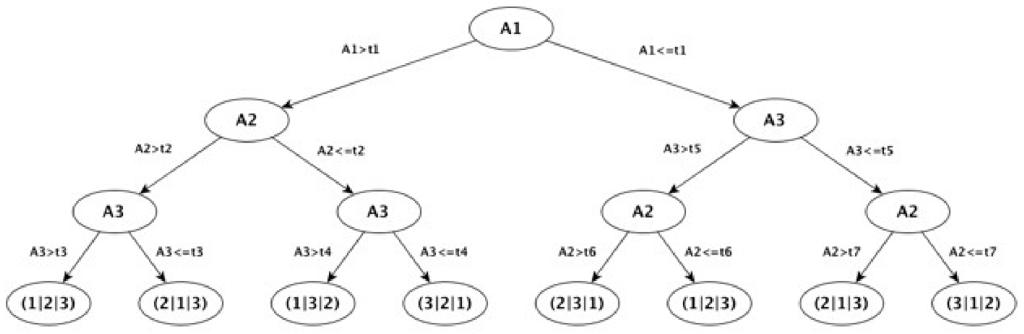

Figure 1: Decision tree example for label ranking.

3.1 Learning the decision tree

Most of the decision tree induction algorithms [26, 27] work recursively. Thus, at each call the method receives a set of instances $\mathbf{R} = \{(x_1^j, \ldots, x_k^j, \pi^j)\}_{j=1}^s$,

$k \leq n, s \leq N$ and must decide either stopping the process by creating a leaf (target) node or using a predictive attribute X_i to split $\mathbf{R}$ into several subsets according to the value taken by X_i. Let $\mathbf{R}_\Pi = \{\pi^j\}_{j=1}^s$ be the set of label rankings in $\mathbf{R}$.

Let us start with the stopping criteria. In [1] a leaf node is created if any of the two following conditions is true:

- $D(\pi^i, \pi^j) = 0$ for all i, j with $1 \leq i, j \leq s$, i.e. there is no disagreement among the rankings included in $\mathbf{R}_\Pi$.
  - In the complete case this means that all the rankings are the same, and so this is the value taken as output for the leaf node.
  - When $\mathbf{R}_\Pi$ contains incomplete rankings this process is more complex. Even if no disagreement exists (e.g. $\mathcal{I} = \{a, b, c\}$ and $\mathbf{R}_\Pi = \{a \succ_{\pi_1} b, a \succ_{\pi_2} c, b \succ_{\pi_3} c\}$) the consensus permutation for $\mathbf{R}_\Pi$ must be computed. In [1] a two-steps procedure is used: first, a consensus permutation for $\mathbf{R}_\Pi$ is computed by using a generalized Borda algorithm based on the notion of extension sets [1, 23, 30]; second, an iterative process is run which consists in (i) the *completion* of the incomplete rankings in $\mathbf{R}_\Pi$ by using the consensus previously obtained, and (ii) in the estimation of a new consensus from the completed rankings. This iterative process finishes when the new consensus is equal to the previous one.

    Remark 1. *In our experiments we realized that the use of completion does not significantly increase the accuracy of the obtained models, but it significantly increases the required CPU time. Because of this and since our goal is to apply bagging over LRT-based algorithms, we decided to avoid the* completion *phase, setting as consensus permutation the one obtained by directly using the extension-sets-based generalized Borda count algorithm.*

- $s \leq 2n$. That is, the number of instances arriving to the current node is less than or equal to twice the number of class labels. In this case, the output is set to the consensus permutation for $\mathbf{R}_\Pi$ obtained by using the (generalized) Borda count algorithm.

  This stopping criterion has a side effect, as it also works as a sort of pre-pruning. In fact, no post-pruning step is used in [1].

If none of the two previous conditions occurs, then the algorithm selects an attribute X_i to split $\mathbf{R}$. Greedy inductive algorithms select the attribute that most reduces the uncertainty on the target variable for the obtained splitting. For example, the one that most reduces the conditional entropy $H(C|X_i)$ is selected for classification trees [26] and the one that most reduces the variance on the target variable Y is selected for regression trees [27]. The authors in [1] propose to exploit the resemblance between the spread parameter θ (from the Mallows distribution) when dealing with rankings and the variance when dealing with numerical variables, to select as decision variable the one reducing the uncertainty of the resulting partition. Formally, given an attribute X_i taking values in $dom(X_i) = \{x_i^1, \ldots, x_i^{r_i}\}$, the *uncertainty* associated to a partition $\{\mathbf{R}_1, \ldots, \mathbf{R}_{r_i}\}$ is inversely proportional to

$$f(X_i) = f(\{\mathbf{R}_1, \ldots, \mathbf{R}_{r_i}\}) = \sum_{j=1}^{r_i} \frac{|\mathbf{R}_j| \cdot \theta_j}{|\mathbf{R}|}, \tag{2}$$

where θ_j is the spread parameter corresponding to the Mallows distribution estimated for the set of label rankings in $\mathbf{R_j}$, the highest θ_j the more concentrated around the peak is the sample. Then, the algorithm selects as the splitting variable the one maximizing $f(X_i)$, $1 \leq i \leq n$.

For the estimation of the Mallows model parameters given in $\mathbf{R}_\Pi$, a two-steps maximum likelihood estimation (MLE) approach is followed. First, the consensus permutation π_0 is obtained by using the (generalized) Borda count method. Then, once π_0 is known, the computation of θ can be carried out by means of standard numerical optimization methods (see [1, 22, 24] for the details).

In the case of numerical variables, the algorithm proceeds in a standard way, that is, selecting a threshold t and dealing with the resulting two-states discrete variable X_i^t: $dom(X_i^t) = \{X_i \leq t, X_i > t\}$. To select the threshold t, all[1] the (different) values for X_i in $\mathbf{R}$ are analyzed as possible thresholds and the one maximizing $f(X_i^t)$ is selected.

3.2 Inference

Given an instance, inference is, in general, quite simple: just start at the root node and traverse the tree by following the appropriate path according to

[1]In fact, we skip the first/last values which lead to partitions with less than n instances in one of the branches. This *rule* is aligned with the pre-pruning strategy described above.

the values for the predictive attributes in the instance being classified. Once a leaf node is achieved, it returns its associated permutation.

However, note that in the incomplete case some leaves could not contain a (complete) ranking. In this case, the returned ranking is completed by using the information gathered in the predecessor nodes of such leaves [1].

4 Proposal: An ensemble of simpler learners

The use of an ensemble of classifiers instead of a single one leads, in general, to an improvement in accuracy [31]. One of the simplest techniques to build such an ensemble is *Bagging* [18], which in its canonical form basically consists in:

- Using bootstrap sampling to generate b different bags $\{\mathbf{D}_1, \ldots, \mathbf{D}_b\}$ from the original dataset $\mathbf{D}$.
- Using a machine learning algorithm to learn a model M_i for each dataset $\mathbf{D}_i$.
- Given a new instance, $\mathbf{x} = (x_1, \ldots, x_n)$ to be classified, it returns $g(M_1(\mathbf{x}), \ldots, M_b(\mathbf{x}))$, where $M_i(\mathbf{x})$ is the value returned by model M_i and g is an aggregation function, e.g., majority voting.

In this paper we aim to study the effect of bagging decision trees in the problem of label ranking. In particular, the idea is to learn b decision trees from the set of bags and then to use the consensus permutation obtained from $\{M_1(\mathbf{x}), \ldots, M_b(\mathbf{x})\}$ as aggregation function. However, as well as we expect to increase the accuracy by using bagging, we also expect an increase in the computational complexity. Because of this, *our first goal* is to reduce the complexity of the base classifier to be bagged. Thus, we propose two simpler algorithms to learn the trees. Basically, we reduce the learning complexity by strongly decreasing the number of computations needed to decide the splitting point. As a consequence, *weak* models are obtained by using these faster learners, but when combined into an ensemble, it gets accurate results.

In this section we first study the (time) complexity of learning decision trees for label ranking and bagging them. Next we introduce our modifications to get less expensive learning.

4.1 Complexity of tree induction for label ranking

We focus our study on the case of numerical predictors as in [1]. This is the most computationally demanding case and the one we consider to simplify the LRT algorithm. Furthermore, we only deal with the case of complete rankings, as the proposed modifications do not take into account the completeness of the rankings.

Firstly, we consider some standard assumptions: $k \leq n$, $k \leq N$ and $n \leq N$. Secondly, let us assume that the depth of the tree is $log(N)$, which is the standard assumption for a balanced tree [32]. Finally, as usual, we consider that at each level of the tree, each attribute is processed once over the whole dataset, i.e. the N instances[2]. Therefore, the time complexity of learning is $log(N)$ times the complexity of the operations performed at each node/level:

- *Stopping* condition. In essence, it reduces to check wether all the rankings are the same, which requires $O(kN)$.

- *Leaf* node. Borda count method is used to obtain the consensus permutation. Its computational complexity is $O(kN + klog(k))$, where the first part is due to counting and the second one to the sorting process.

- *Decision* node. All the n attributes must be checked in order to decide the most informative one. For each attribute X_i the threshold leading to the best binary split must be identified. This implies to check all the N values as potential thresholds. Therefore, the first step is to sort[3] the values of X_i which requires $O(Nlog(N))$. For each threshold, the variable is considered as a discrete one X_i^t with two states $\{X_i \leq t, X_i > t\}$ and (2) is used to evaluate its goodness. This requires to compute π_0 and θ twice, one for each side of the threshold, which computationally is equivalent to compute the two parameters only once for the N values. As mentioned above, computing π_0 by using Borda count is $O(kN + klog(k))$. On the other hand, the complexity of computing

[2]At each level, the number of instances processed per node is different, but the union of them is N.

[3]In fact, if the algorithm is implemented carefully, this sorting only is needed the first time an attribute is analyzed. Therefore, this requires $O(nNlog(N))$ to sort all the variables before starting the decision tree construction process.

θ is bounded by[4] $O(k^2N + k^2)$ [33, pg. 51]. Thus, the complexity of computing the two parameters is $kN + klog(k) + k^2N + k^2$, which leads to $O(k^2N)$. Finally, as these parameters are computed for each one of the n attributes and taking every value (N) as threshold, we get $O(nk^2N^2)$.

Then, the complexity order for the whole learning process is given by

$$O(nNlog(N) + log(N)(kN + max\{kN + k\,log(N), nk^2N^2\}))$$

that is,

$$O(nk^2N^2log(N)). \tag{3}$$

Regarding inference, classifying a new instance requires $O(log(N))$, as this is the depth of the tree.

With respect to bagging, as learning dominates the sampling process, the complexity of the learning stage is b times the complexity of learning a single tree. On the other hand, the complexity of inference is $O(b\,log(N))$ for classifying the given instance using the b trees, plus $O(kb + k\,log(k))$ for the aggregation phase (Borda count).

4.2 W-LRT and F-LRT

In this section our goal is to reduce the complexity of computing the splitting point. The splitting criterion described above can be viewed as a supervised discretization criterion which produces *optimal* splits for the label ranking problem. As studied above, this is the most time demanding step in the construction of the tree. Therefore, in this section, we propose two modified versions of the LRT algorithm that differ from the original one only in the way they compute the splitting point for each attribute. In particular, we propose to select the splitting point by using two well-known unsupervised[5] discretization criteria: equal-width and equal-frequency.

Our aim is to avoid the analysis of all the N possible splitting points. Thus, instead of checking the N possible thresholds, we propose to set the

[4]In fact, the complexity reported in [33, pg. 51] is bounded by $O(k^2)$. However a precedence matrix is required which needs $O(k^2N)$.

[5]A supervised discretization for the label ranking problem has been proposed in [34]. We did not use it because (i) applying it at each node will not decrease the complexity of the process, since it tries all the possible thresholds, and (ii) applying it as a pre-processing procedure leads to trees with a large branching degree, which traduces into a low accuracy.

mid point without carrying out any test. On the other hand, in this way we use far less knowledge than when doing the supervised selection, and so we may expect a weaker *model*. However, the combination of these weaker models with bagging will allow to obtain better results, as bagging predictors have shown to be effective especially when using weak base classifiers.

As said above, we propose the use of two simple and well known unsupervised discretization techniques:

- Equal width (W-LRT). In this algorithm we choose as threshold for each variable X_i the value which splits its domain into two intervals of equal width. Running over all the N values of X_i in order to identify the smallest and biggest ones requires $O(N)$.

  Example.
  X_i: (4, 16, $\underline{0}$, 12, $\underline{28}$, 24, 26, 16, 18)
  $t = mean(0, 28) = 14 \rightsquigarrow \{X_i \leq 14, X_i > 14\}$

  Then, the complexity order for the whole learning process is given by

  $$O(log(N)(kN + max\{kN + k\,log(N), (k^2N + N)n\})$$

  that is,

  $$O(nk^2Nlog(N)). \tag{4}$$

- Equal frequency (F-LRT). In this algorithm we choose as threshold for each variable X_i the value which creates two bins having the same number of values. This requires $O(Nlog(N))$ to sort the N values of X_i, and then take the value in the middle position as threshold. As already commented, this sorting process only needs to be performed once for each variable. Hence, addressing the middle point of such a sorted array requires $O(1)$.

  Example.
  $sort(X_i)$: (0, 4, 12, 16, $\underline{16}$, 18, 24, 26, 28)
  $t = 16 \rightsquigarrow \{X_i \leq 16, X_i > 16\}$

  Then, the complexity order for the whole learning process is given by

  $$\begin{aligned}&O(nNlog(N)+\\&\quad log(N)(kN + max\{kN + k\,log(N), (k^2N + 1)n\})\end{aligned}$$

  that is,

  $$O(nk^2Nlog(N)). \tag{5}$$

All in all, the complexity regarding LRT has been reduced by $O(N)$.

5 Experimental evaluation

In this section we carry out an empirical study of the proposed methods. Next, we describe the used datasets, the involved algorithms and the experimental methodology.

5.1 Datasets

We use as benchmark a total of 21 datasets. The first 16 were proposed in [1] and used since then as a standard benchmark for the label ranking problem. They can be considered semi-synthetic, as they were obtained by transforming a multi-class ($_{(c)}$) or regression ($_{(r)}$) problem to a label ranking one (see [1] for the transformation details). The last 5 datasets correspond to real-world biological data problems and were used in [10]. In this case the expression profile of the output gene (e.g. (1.7, 2.9, 0.3, -2.4)) was directly converted into a rank (e.g. (2, 1, 3, 4)). All the datasets, conveniently formatted for the label-ranking problem, can be downloaded from `https://www.uni-marburg.de/fb12/kebi/research/repository/labelrankingdata`. Table 1 shows the main features of each dataset. The columns max #rankings and #rankings stand for the maximum number of label rankings ($k!$) and the actual number of different label rankings in the dataset, respectively.

5.2 Algorithms

In the study we consider the following algorithms:

- The LRT algorithm [1] (see Section 3).
- The two proposed algorithms based on unsupervised discretization: W-LRT and F-LRT (see Section 4.2).
- The ensemble approaches constructed by applying bagging to the three LRT based algorithms: LRTb, W-LRTb and F-LRTb. Here b denotes the number of models considered in bagging.
- As reference we also consider the IBLR algorithm proposed in [1], an instance-based method which returns as outcome the consensus permutation for the k nearest neighbours, computed by combining the generalized Borda count with an iterative improving method. Euclidean dis-

Table 1: Datasets.

Dataset	#instances	#attributes	max #rankings	#rankings
authorship$_{(c)}$	841	70	4!	17
bodyfat$_{(r)}$	252	7	7!	236
calhousing$_{(r)}$	20640	4	4!	24
cpu-small$_{(r)}$	8192	6	5!	119
elevators$_{(r)}$	16599	9	9!	131
fried$_{(r)}$	40769	9	5!	120
glass$_{(c)}$	214	9	6!	30
housing$_{(r)}$	506	6	6!	112
iris$_{(c)}$	150	4	3!	5
pendigits$_{(c)}$	10992	16	10!	2081
segment$_{(c)}$	2310	18	7!	135
stock$_{(r)}$	950	5	5!	51
vehicle$_{(c)}$	846	18	4!	18
vowel$_{(c)}$	528	10	11!	294
wine$_{(c)}$	178	13	3!	5
wisconsin$_{(r)}$	194	16	16!	194
spo	2465	24	11!	2361
heat	2465	24	6!	622
dtt	2465	24	4!	24
cold	2465	24	4!	24
diau	2465	24	7!	967

tance is taken to identify the nearest neighbours. The number of neighbours considered is set by cross validation, taking values in $[1, \sqrt{N}]$.

To justify the selection of the IBLR as the baseline algorithm we have collected the results of several published studies that propose LR algorithms based on different machine learning paradigms and that use the benchmark of datasets proposed originally in [1]. Thus, Table 2 shows the results (averaged Kendall coefficient for a 5×10 cross validation) reported in the source references for the following algorithms in the complete case (see next subsection):

- IBLR and LRT [1]. In this case we report the results obtained by our own implementation.
- IB-PL [17]. An instance based method similar to IBLR but using the Plackett-Luce model [35, 36] instead of the Mallows one to model the probability distribution over permutations.
- Lin-PL [17]. An algorithm based on the estimation of generalized linear models by using the Plackett-Luce model to model the distribution over permutations.
- MLP [16]. A label ranking algorithm based on a multilayer perceptron. Several algorithms result from the way in which the weights are corrected during the back propagation process. Here we report the results of the best one, a combination between global and local correction processes.
- Apriori [15]. A label ranking algorithm based on adapting the Apriori algorithm to discover association rules whose consequents are rankings. The authors apply the algorithm using two different discretizations. We have taken the best result for each dataset.

Although the results reported above may consider different partitions for the repeated cross validation, we think that the average over the 50 runs (5×10 cv) is representative enough to provide a fair estimation. In Table 2, between parenthesis and close to the algorithm name, we show the average rank of each algorithm. In the light of the ranking obtained, IBLR is taken as the baseline algorithm for comparison in this study.

Table 2: Mean accuracy for complete rankings comparing different approaches.

method	aut	bod	cal	cpu	ele	fri	gla	hou	iri	pen	seg	sto	veh	vow	win	wis
IBLR (1.63)	0.933	0.200	0.367	0.484	0.726	0.949	**0.883**	**0.764**	**0.966**	**0.946**	**0.958**	**0.928**	**0.860**	**0.919**	0.932	0.487
IB-PL (2.41)	**0.936**	0.230	0.326	**0.495**	0.721	0.894	0.841	0.711	0.960	0.939	0.950	0.922	0.859	0.851	0.947	0.479
LRT (3.25)	0.885	0.193	**0.398**	0.475	**0.773**	0.886	0.845	0.740	0.928	0.924	0.944	0.892	0.825	0.692	0.883	0.330
Lin-PL (3.56)	0.930	**0.272**	0.220	0.426	0.712	**0.996**	0.825	0.659	0.832	0.909	0.902	0.710	0.838	0.586	**0.954**	**0.635**
Apriori (4.34)	0.570	0.160	0.290	0.440	0.640	0.770	0.850	0.760	0.960	0.690	0.900	0.890	0.750	0.720	0.910	0.280
MLP (5.63)	0.829	0.074	0.106	0.357	0.684	0.660	0.757	0.574	0.800	0.752	0.842	0.745	0.800	0.545	0.874	0.235

5.3 Methodology

The following design decisions have been adopted:

- In all the cases the algorithms are assessed by using five repetitions of a ten-folds cross-validation (5×10cv).
- As in [37, 1], the Kendall coefficient is used as goodness score. Specifically, given two permutations π_1 and π_2 in $\mathbb{S}_k$ this coefficient is given by

$$\tau_K(\pi_1, \pi_2) = \frac{C(\pi_1, \pi_2) - D(\pi_1, \pi_2)}{\frac{k(k-1)}{2}}, \tag{6}$$

  where $C(\pi_1, \pi_2)$ (resp. $D(\pi_1, \pi_2)$) is the number of concordant (resp. discordant) pairs between π_1 and π_2.

  The Kendall coefficient ranges between -1 and 1. The closer to 1 the better the correlation. Values close to -1 can also be useful as they show a good correlation between a ranking and its inverse. Values close to 0 mean poor correlation between the two rankings.

- We consider three different scenarios: (i) complete case; (ii) incomplete case with 30% of missing labels in the rankings; and (iii) incomplete case with 60% of missing labels in the rankings.

  To deal with the incomplete cases, we remove labels from the rankings in the corresponding training sets with probability 0.3 and 0.6, respectively. In any case, incomplete rankings cannot have less than 2 labels[6].

- In the case of the ensembles we set b in $\{5, 10, 15, 20, 25, 30, 50, 100\}$.

[6]This is also the procedure followed in [1]

- All the algorithms have been written in Java. The experiments have been carried out in computers running Linux operating system and with a maximum of 20 GB of RAM memory.

5.4 Results

In this section we present the results obtained as well as their analysis. We divide the study into accuracy, tree-size and time.

5.4.1 Accuracy

Tables 3, 4 and 5 show the results for the three scenarios considered. The values in the cells correspond to the mean Kendall coefficient τ_K between the current and the predicted permutations, averaged over the test sets of the $5{\times}10$cv. In order to make easier the interpretation of the results, the algorithm(s) obtaining the best result for each dataset has(have) been boldfaced.

Table 3: Mean accuracy for each algorithm with complete rankings applying bagging.

# Bagging		aut	bod	cal	cpu	ele	fri	gla	hou	iri	pen	seg	sto	veh	vow	win	wis	spo	hea	dtt	col	dia
	LRT	0.885	0.193	0.398	0.475	0.773	0.886	0.845	0.740	0.928	0.924	0.944	0.892	0.825	0.692	0.883	0.330	0.120	0.045	0.093	0.058	0.169
	W-LRT	0.829	0.146	0.359	0.457	0.767	0.851	0.806	0.696	0.942	0.907	0.928	0.879	0.799	0.668	0.861	0.262	0.125	0.048	0.084	0.053	0.188
	F-LRT	0.857	0.154	0.356	0.474	0.772	0.854	0.809	0.686	0.896	0.904	0.928	0.872	0.814	0.654	0.856	0.325	0.126	0.045	0.086	0.057	0.175
-	IBLR	**0.933**	0.200	0.367	0.484	0.726	0.949	**0.883**	**0.764**	**0.966**	**0.946**	**0.958**	**0.928**	0.860	**0.919**	**0.932**	**0.487**	0.089	**0.216**	0.127	0.073	0.139
	LRT	0.902	0.196	0.426	0.498	0.778	0.923	0.848	0.749	0.944	0.938	0.951	0.896	0.849	0.731	0.899	0.374	0.132	0.051	0.101	0.069	0.195
	W-LRT	0.886	0.191	0.397	0.479	0.774	0.872	0.801	0.705	0.945	0.919	0.938	0.888	0.834	0.696	0.904	0.318	0.136	0.052	0.105	0.065	0.204
5	F-LRT	0.894	0.181	0.382	0.493	0.775	0.895	0.828	0.725	0.932	0.920	0.939	0.885	0.833	0.695	0.887	0.368	0.137	0.048	0.099	0.068	0.198
	LRT	0.910	0.212	0.450	0.511	0.784	0.936	0.853	0.756	0.946	0.942	0.955	0.902	0.858	0.744	0.910	0.383	0.140	0.060	0.117	0.082	0.213
	W-LRT	0.899	0.207	0.421	0.489	0.779	0.880	0.808	0.709	0.951	0.922	0.940	0.892	0.846	0.704	0.915	0.331	0.142	0.061	0.119	0.078	0.218
10	F-LRT	0.906	0.192	0.406	0.504	0.780	0.909	0.835	0.731	0.941	0.924	0.942	0.893	0.841	0.706	0.901	0.377	0.144	0.058	0.115	0.080	0.215
	LRT	0.912	0.220	0.459	0.516	0.786	0.943	0.855	0.758	0.951	0.943	0.956	0.904	0.862	0.749	0.917	0.386	0.144	0.066	0.125	0.090	0.221
	W-LRT	0.905	0.212	0.430	0.492	0.781	0.884	0.810	0.711	0.956	0.923	0.942	0.893	0.850	0.707	0.920	0.334	0.144	0.065	0.126	0.086	0.224
15	F-LRT	0.909	0.199	0.414	0.508	0.782	0.916	0.837	0.734	0.945	0.926	0.944	0.896	0.844	0.711	0.905	0.379	0.147	0.064	0.122	0.086	0.223
	LRT	0.914	0.224	0.464	0.519	0.787	0.946	0.855	0.760	0.950	0.944	0.956	0.905	0.864	0.752	0.919	0.386	0.145	0.069	0.131	0.093	0.226
	W-LRT	0.908	0.217	0.434	0.494	0.782	0.886	0.811	0.713	0.958	0.923	0.942	0.893	0.853	0.708	0.924	0.336	0.146	0.068	0.130	0.090	0.227
20	F-LRT	0.911	0.204	0.418	0.510	0.783	0.919	0.838	0.735	0.948	0.927	0.944	0.898	0.845	0.713	0.908	0.381	0.149	0.068	0.126	0.091	0.227
	LRT	0.915	0.226	0.467	0.520	0.788	0.948	0.856	0.761	0.952	0.944	0.957	0.906	0.865	0.753	0.920	0.387	0.146	0.071	0.135	0.097	0.229
	W-LRT	0.909	0.218	0.437	0.495	0.783	0.887	0.811	0.714	0.959	0.924	0.943	0.894	0.854	0.709	0.926	0.337	0.146	0.071	0.132	0.094	0.230
25	F-LRT	0.912	0.206	0.421	0.511	0.783	0.922	0.839	0.736	0.950	0.927	0.945	0.898	0.846	0.714	0.909	0.382	0.149	0.069	0.129	0.094	0.229
	LRT	0.915	0.227	0.469	0.521	0.788	0.949	0.856	0.761	0.952	0.944	0.957	0.906	0.865	0.754	0.920	0.387	0.147	0.074	0.136	0.099	0.230
	W-LRT	0.910	0.222	0.439	0.496	0.783	0.888	0.813	0.714	0.958	0.924	0.943	0.894	0.855	0.710	0.928	0.338	0.147	0.072	0.136	0.096	0.231
30	F-LRT	0.913	0.207	0.423	0.512	0.784	0.923	0.839	0.736	0.950	0.928	0.945	0.899	0.847	0.715	0.910	0.382	0.150	0.071	0.133	0.097	0.231
	LRT	0.916	0.230	0.473	0.523	0.789	0.952	0.857	0.761	0.955	0.945	0.957	0.907	**0.867**	0.756	0.923	0.388	0.149	0.076	0.144	0.104	0.235
	W-LRT	0.912	0.223	0.443	0.497	0.784	0.889	0.813	0.714	0.961	0.924	0.943	0.895	0.857	0.711	0.929	0.340	0.147	0.076	0.141	0.102	0.234
50	F-LRT	0.915	0.210	0.427	0.514	0.784	0.926	0.840	0.737	0.953	0.928	0.946	0.900	0.849	0.717	0.913	0.383	0.152	0.076	0.136	0.102	0.235
	LRT	0.916	**0.233**	**0.476**	**0.524**	**0.790**	**0.955**	0.856	0.761	0.956	0.945	**0.958**	0.907	**0.867**	0.758	0.924	0.389	0.149	0.080	**0.148**	**0.107**	**0.239**
	W-LRT	0.914	0.225	0.447	0.498	0.784	0.891	0.813	0.715	0.961	0.925	0.944	0.894	0.858	0.711	0.928	0.340	0.148	0.081	0.145	**0.107**	0.236
100	F-LRT	0.916	0.210	0.429	0.515	0.785	0.929	0.839	0.738	0.956	0.929	0.946	0.901	0.850	0.719	0.916	0.384	**0.153**	0.079	0.140	0.106	0.237

Before going into a deeper (statistical) analysis, let us draw some preliminary conclusions from the obtained results:

- Although the original *supervised* LRT algorithm obtains, in general, better results than W-LRT and F-LRT, these two algorithms based on the use of unsupervised discretization show a good performance.

Table 4: Mean accuracy for each algorithm using rankings with 30% missing labels applying bagging.

# Bagging		aut	bod	cal	cpu	ele	fri	gla	hou	iri	pen	seg	sto	veh	vow	win	wis	spo	hea	dtt	col	dia
	LRT	0.854	0.152	0.363	0.448	0.760	0.863	0.826	0.712	0.918	0.916	0.935	0.873	0.815	0.673	0.848	0.331	0.106	0.030	0.069	0.047	0.156
	W-LRT	0.817	0.134	0.330	0.432	0.744	0.827	0.790	0.678	0.907	0.894	0.917	0.863	0.784	0.651	0.856	0.255	0.116	0.037	0.076	0.041	0.170
	F-LRT	0.846	0.141	0.313	0.448	0.755	0.834	0.796	0.669	0.885	0.893	0.918	0.854	0.801	0.645	0.836	0.317	0.115	0.034	0.065	0.035	0.155
-	IBLR	**0.923**	0.163	0.342	0.475	0.712	0.926	0.809	0.717	0.909	0.924	0.931	**0.901**	0.836	**0.823**	0.885	**0.445**	0.078	**0.199**	0.112	0.058	0.127
	LRT	0.890	0.174	0.395	0.478	0.768	0.905	0.831	0.726	0.931	0.932	0.944	0.881	0.839	0.715	0.884	0.365	0.125	0.043	0.088	0.057	0.180
	W-LRT	0.871	0.168	0.370	0.460	0.758	0.848	0.785	0.686	0.922	0.908	0.929	0.872	0.821	0.685	0.887	0.305	0.131	0.044	0.094	0.057	0.187
5	F-LRT	0.883	0.157	0.342	0.474	0.762	0.871	0.814	0.706	0.914	0.910	0.930	0.866	0.821	0.684	0.877	0.355	0.129	0.043	0.085	0.054	0.181
	LRT	0.901	0.194	0.425	0.497	0.779	0.924	0.839	0.736	0.936	0.939	0.949	0.890	0.852	0.730	0.902	0.376	0.137	0.051	0.103	0.070	0.200
	W-LRT	0.890	0.185	0.400	0.475	0.769	0.861	0.793	0.694	0.932	0.914	0.935	0.880	0.836	0.698	0.905	0.318	0.139	0.053	0.110	0.067	0.206
10	F-LRT	0.900	0.176	0.372	0.491	0.773	0.890	0.823	0.717	0.930	0.917	0.937	0.877	0.834	0.700	0.898	0.365	0.139	0.051	0.102	0.067	0.201
	LRT	0.904	0.201	0.436	0.504	0.783	0.933	0.841	0.740	0.938	0.941	0.950	0.893	0.857	0.735	0.910	0.380	0.141	0.056	0.112	0.076	0.211
	W-LRT	0.896	0.193	0.412	0.481	0.773	0.867	0.796	0.697	0.938	0.916	0.937	0.883	0.843	0.702	0.909	0.323	0.142	0.057	0.117	0.074	0.214
15	F-LRT	0.905	0.183	0.384	0.496	0.776	0.898	0.826	0.720	0.936	0.920	0.939	0.881	0.838	0.704	0.904	0.369	0.143	0.056	0.106	0.073	0.211
	LRT	0.906	0.205	0.442	0.508	0.784	0.937	0.843	0.741	0.938	0.942	0.951	0.894	0.859	0.739	0.913	0.382	0.143	0.060	0.116	0.079	0.216
	W-LRT	0.899	0.197	0.418	0.483	0.775	0.869	0.798	0.698	0.939	0.917	0.938	0.885	0.845	0.704	0.912	0.325	0.143	0.061	0.123	0.078	0.218
20	F-LRT	0.907	0.186	0.391	0.499	0.778	0.902	0.827	0.723	0.938	0.921	0.940	0.883	0.841	0.707	0.906	0.370	0.145	0.060	0.113	0.076	0.216
	LRT	0.907	0.209	0.446	0.510	0.785	0.940	0.843	0.743	0.940	0.943	0.952	0.895	0.861	0.740	0.916	0.384	0.145	0.062	0.121	0.083	0.219
	W-LRT	0.901	0.202	0.422	0.485	0.777	0.871	0.799	0.699	0.943	0.918	0.939	0.885	0.847	0.705	0.913	0.326	0.144	0.063	0.126	0.081	0.220
25	F-LRT	0.909	0.189	0.394	0.501	0.779	0.905	0.827	0.725	0.941	0.922	0.941	0.885	0.842	0.709	0.910	0.372	0.146	0.062	0.116	0.080	0.219
	LRT	0.908	0.211	0.448	0.511	0.786	0.942	0.844	0.743	0.941	0.943	0.952	0.896	0.861	0.742	0.917	0.385	0.146	0.064	0.124	0.087	0.221
	W-LRT	0.902	0.204	0.425	0.486	0.777	0.872	0.799	0.700	0.942	0.918	0.939	0.886	0.848	0.706	0.916	0.327	0.144	0.064	0.129	0.083	0.222
30	F-LRT	0.911	0.189	0.397	0.502	0.779	0.907	0.827	0.726	0.943	0.922	0.941	0.886	0.844	0.710	0.912	0.372	0.147	0.064	0.119	0.081	0.221
	LRT	0.910	0.217	0.453	0.514	0.788	0.946	0.843	**0.744**	0.940	0.944	0.952	0.897	0.863	0.744	0.921	0.387	0.148	0.068	0.130	0.090	0.227
	W-LRT	0.905	0.206	0.430	0.488	0.779	0.875	0.800	0.702	0.944	0.919	0.940	0.887	0.850	0.708	0.919	0.329	0.146	0.067	0.134	0.088	0.226
50	F-LRT	0.913	0.195	0.402	0.504	0.781	0.912	0.828	0.727	0.942	0.923	0.942	0.887	0.846	0.712	0.915	0.373	0.149	0.068	0.124	0.086	0.225
	LRT	0.912	**0.218**	**0.457**	**0.517**	**0.789**	**0.949**	**0.845**	**0.744**	0.941	**0.945**	**0.953**	0.897	**0.866**	0.746	**0.923**	0.388	0.150	0.075	**0.140**	**0.094**	**0.231**
	W-LRT	0.907	0.209	0.435	0.490	0.780	0.877	0.801	0.702	0.943	0.920	0.940	0.888	0.852	0.708	0.915	0.331	0.146	0.070	0.139	0.093	0.229
100	F-LRT	0.916	0.200	0.407	0.505	0.782	0.915	0.828	0.729	**0.945**	0.924	0.942	0.889	0.847	0.714	0.918	0.374	**0.151**	0.072	0.129	0.091	0.230

Table 5: Mean accuracy for each algorithm using rankings with 60% missing labels applying bagging.

# Bagging		aut	bod	cal	cpu	ele	fri	gla	hou	iri	pen	seg	sto	veh	vow	win	wis	spo	hea	dtt	col	dia
	LRT	0.791	0.087	0.283	0.368	0.710	0.786	0.768	0.634	0.772	0.882	0.906	0.799	0.743	0.638	0.721	0.303	0.086	0.024	0.049	0.026	0.121
	W-LRT	0.718	0.099	0.244	0.369	0.680	0.746	0.716	0.627	0.667	0.851	0.872	0.786	0.699	0.608	0.683	0.238	0.096	0.029	0.055	0.034	0.132
	F-LRT	0.770	0.098	0.225	0.379	0.698	0.765	0.741	0.606	0.698	0.855	0.872	0.771	0.706	0.608	0.644	0.288	0.092	0.021	0.058	0.023	0.120
-	IBLR	0.779	0.126	0.279	0.450	0.678	0.885	0.681	0.637	0.642	0.896	0.877	0.831	0.713	0.704	0.639	**0.375**	0.057	**0.162**	0.088	0.040	0.103
	LRT	0.868	0.137	0.325	0.429	0.729	0.860	0.783	0.663	0.816	0.902	0.919	0.830	0.798	0.673	0.811	0.333	0.108	0.030	0.074	0.039	0.146
	W-LRT	0.822	0.129	0.284	0.411	0.709	0.792	0.738	0.637	0.735	0.874	0.891	0.811	0.764	0.651	0.748	0.280	0.114	0.029	0.073	0.037	0.153
5	F-LRT	0.844	0.119	0.258	0.425	0.718	0.815	0.758	0.644	0.768	0.879	0.896	0.805	0.767	0.648	0.739	0.326	0.115	0.029	0.069	0.037	0.145
	LRT	0.888	0.154	0.357	0.462	0.755	0.893	0.804	0.684	0.842	0.919	0.930	0.851	0.820	0.697	0.835	0.352	0.122	0.037	0.087	0.047	0.169
	W-LRT	0.858	0.147	0.315	0.439	0.737	0.818	0.759	0.655	0.761	0.891	0.908	0.834	0.793	0.672	0.774	0.293	0.127	0.035	0.087	0.045	0.177
10	F-LRT	0.875	0.137	0.289	0.455	0.744	0.846	0.783	0.666	0.793	0.896	0.915	0.831	0.794	0.673	0.763	0.341	0.128	0.036	0.083	0.045	0.170
	LRT	0.895	0.160	0.371	0.474	0.765	0.906	0.812	0.690	0.843	0.925	0.934	0.858	0.827	0.706	0.852	0.359	0.129	0.040	0.094	0.053	0.182
	W-LRT	0.872	0.155	0.328	0.450	0.748	0.828	0.769	0.662	0.761	0.896	0.914	0.841	0.803	0.680	0.788	0.299	0.132	0.038	0.096	0.052	0.187
15	F-LRT	0.887	0.144	0.302	0.467	0.754	0.858	0.791	0.674	0.802	0.902	0.922	0.841	0.804	0.681	0.777	0.347	0.133	0.040	0.089	0.051	0.182
	LRT	0.897	0.167	0.379	0.481	0.770	0.913	0.815	0.693	0.849	0.928	0.936	0.862	0.830	0.711	0.857	0.363	0.132	0.041	0.100	0.057	0.189
	W-LRT	0.878	0.161	0.336	0.456	0.753	0.833	0.771	0.664	0.768	0.899	0.916	0.846	0.807	0.684	0.796	0.303	0.134	0.039	0.099	0.055	0.193
20	F-LRT	0.893	0.147	0.309	0.473	0.759	0.865	0.793	0.679	0.809	0.905	0.925	0.846	0.809	0.686	0.788	0.350	0.135	0.043	0.092	0.056	0.189
	LRT	0.899	0.171	0.384	0.485	0.773	0.917	0.817	0.695	0.849	0.930	0.937	0.864	0.832	0.714	0.860	0.366	0.134	0.044	0.102	0.060	0.193
	W-LRT	0.883	0.164	0.341	0.459	0.757	0.836	0.773	0.666	0.767	0.901	0.918	0.848	0.810	0.686	0.801	0.304	0.136	0.041	0.102	0.056	0.198
25	F-LRT	0.896	0.151	0.314	0.476	0.762	0.869	0.796	0.681	0.812	0.907	0.927	0.849	0.812	0.689	0.794	0.351	0.137	0.044	0.096	0.058	0.194
	LRT	0.901	0.173	0.387	0.488	0.775	0.920	0.818	0.697	0.851	0.931	0.937	0.865	0.834	0.715	0.865	0.367	0.135	0.045	0.105	0.061	0.197
	W-LRT	0.885	0.167	0.344	0.462	0.759	0.837	0.775	0.668	0.770	0.902	0.919	0.850	0.812	0.688	0.801	0.306	0.137	0.043	0.103	0.057	0.201
30	F-LRT	0.898	0.152	0.317	0.479	0.764	0.872	0.798	0.683	0.815	0.908	0.928	0.851	0.814	0.690	0.797	0.353	0.139	0.046	0.097	0.059	0.197
	LRT	0.903	0.177	0.394	0.494	0.779	0.927	0.823	**0.699**	**0.856**	0.933	0.939	0.868	0.838	0.719	**0.868**	0.370	0.138	0.047	0.109	0.065	0.206
	W-LRT	0.891	0.170	0.350	0.467	0.764	0.842	0.778	0.670	0.767	0.905	0.921	0.854	0.816	0.691	0.806	0.308	0.138	0.045	0.109	0.061	0.207
50	F-LRT	0.902	0.156	0.324	0.484	0.768	0.878	0.800	0.687	0.822	0.911	0.930	0.855	0.818	0.693	0.800	0.355	0.141	0.049	0.103	0.063	0.205
	LRT	0.905	**0.180**	**0.400**	**0.499**	**0.782**	**0.932**	**0.825**	**0.699**	0.852	**0.935**	**0.940**	**0.870**	**0.841**	**0.722**	0.867	0.372	0.141	0.050	**0.114**	**0.070**	**0.213**
	W-LRT	0.895	0.175	0.356	0.471	0.767	0.845	0.781	0.672	0.765	0.907	0.923	0.855	0.818	0.694	0.809	0.310	0.140	0.048	**0.114**	0.065	0.212
100	F-LRT	**0.907**	0.156	0.330	0.488	0.771	0.883	0.801	0.690	0.828	0.912	0.932	0.857	0.820	0.697	0.802	0.356	**0.143**	0.054	0.110	0.069	**0.213**

- The algorithms using bagging have better accuracy when compared to the based models alone. This fact can be clearly observed from Figures 2, 3 and 4, where we show the accuracy averaged over the 21 datasets (y axis) as a function of the number of models (x axis) considered in the ensemble.

- Neither LRT nor W-LRT and F-LRT are competitive with respect to the ensembles, even when using a small number of models. This fact can be observed in Tables 3, 4 and 5 and Figures 2, 3 and 4.

- In the complete case IBLR shows a very good performance. The ensembles require a big number of models to be competitive with it. Things are quite different in the incomplete cases, where IBLR is clearly defeated by the ensembles, even when using a small/moderate number of models. In this case it seems that solving the rank aggregation problem by using a two-level process (first at the level of each tree, and then aggregating the output of the trees) helps to reduce the uncertainty introduced by the partial rankings in the training set.

- The best ensemble is the one using the original LRT algorithm as base classifier, although it is also the most expensive one in terms of time. Ensembles based on W-LRT and F-LRT also show a good performance.

However, as mentioned above, in order to be in a position of extracting sound conclusions, for each of the three scenarios we have carried out a statistical analysis according to the standard procedure described in [38, 39] by using the software available in [40]. Following the recommendations included in those articles, the number of datasets in the statistical study should be at least twice the number of algorithms. Thus, we include in the study the two base classifiers from [1] LRT and IBLR, the two LRT-based unsupervised discretization techniques W-LRT and F-LRT, and the ensembles based on LRT, W-LRT and F-LRT with $b = 25$ and $b = 100$ models. The procedure can be summarized in two steps:

- First, we perform a Friedman test [41]. By using a significance level of 5% the test rejects the null hypothesis that all the tested algorithms are equivalent with p-values 5.6971e^{-23} (complete case), 3.9116e^{-25} (incomplete case, 30%) and 1.2632e^{-27} (incomplete case, 60%).

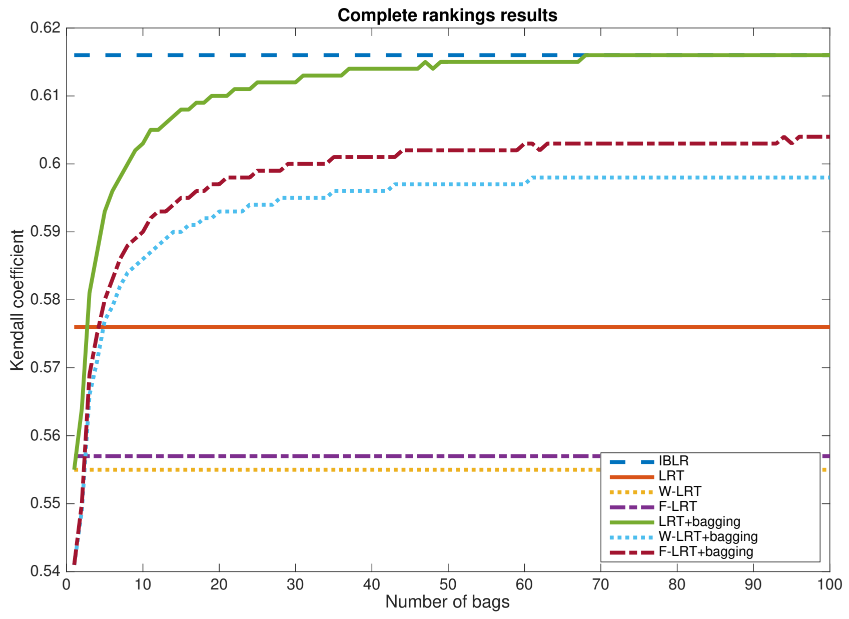

Figure 2: Accuracy of each algorithm averaged over all the datasets (complete rankings)

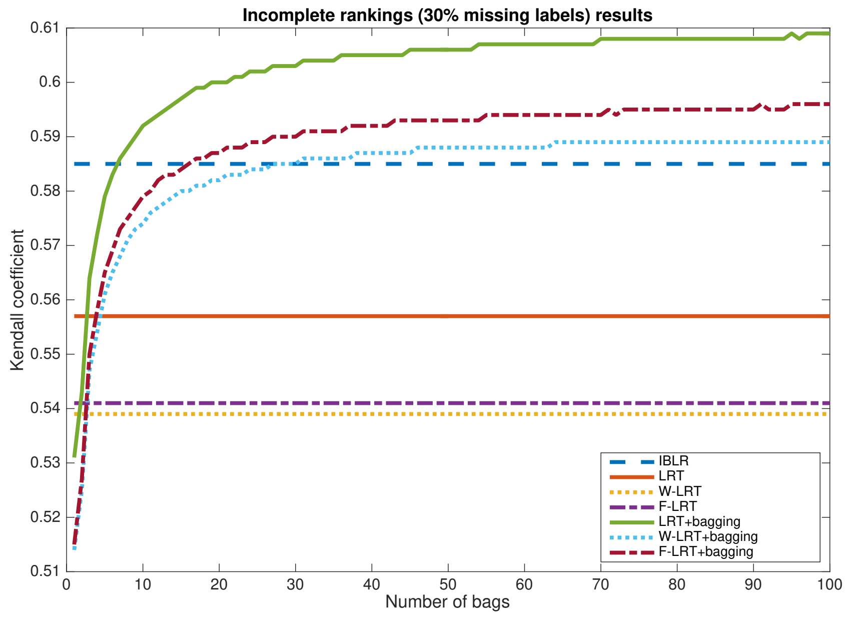

Figure 3: Accuracy of each algorithm averaged over all the datasets (rankings with 30% missing labels)

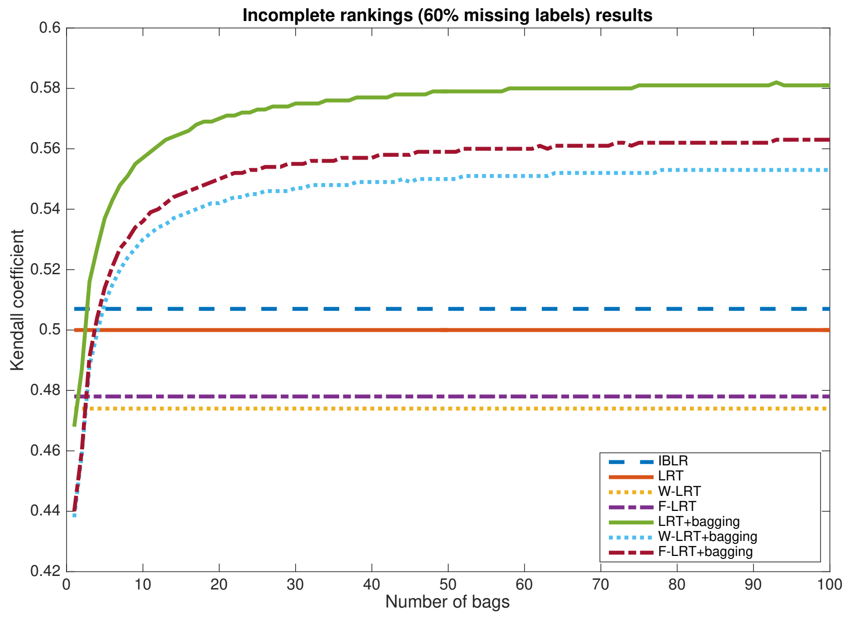

Figure 4: Accuracy of each algorithm averaged over all the datasets (rankings with 60% missing labels)

- Next, we perform a post-hoc test by applying Holm's procedure [42] also using a 5% significance level. This test compares all the algorithms against the one having the best mean rank (the first in the ranking computed during the Friedman test) which is taken as control.

The results of the post-hoc tests are shown in Tables 6, 7 and 8 for the three scenarios considered. We show the ranking computed by the Friedman test and the adjusted p-value using Holm's procedure with a 5% significance level. The win-tie-loss numbers (W T L) in row A are referred to the relative results between the control algorithm and the one in row A. Boldfaced results correspond to non-rejected hypotheses.

Table 6: Results of the post-hoc test for the mean accuracy using complete rankings

method	p-value	rank	win	tie	loss
LRT100	-	1.93	-	-	-
LRT25	**1.0292e-01**	3.45	19	2	0
F-LRT100	**9.6939e-02**	3.93	18	2	1
IBLR	**9.6939e-02**	3.93	10	1	10
W-LRT100	3.7376e-02	4.36	17	1	3
F-LRT25	5.3682e-04	5.55	20	1	0
W-LRT25	1.9635e-04	5.81	19	0	2
LRT	6.7266e-09	7.64	21	0	0
F-LRT	6.1747e-14	9.19	21	0	0
W-LRT	5.6785e-14	9.21	21	0	0

Table 7: Results of the post-hoc test for the mean accuracy using rankings with 30% missing labels

method	p-value	rank	win	tie	loss
LRT100	-	1.50	-	-	-
F-LRT100	**9.9492e-02**	3.14	18	0	3
LRT25	**9.9492e-02**	3.33	21	0	0
W-LRT100	5.6346e-03	4.40	20	0	1
F-LRT25	4.2945e-04	5.12	20	1	0
IBLR	4.6265e-05	5.64	16	0	5
W-LRT25	3.4422e-05	5.74	20	0	1
LRT	1.4405e-10	7.76	21	0	0
W-LRT	5.2710e-15	9.05	21	0	0
F-LRT	5.7307e-16	9.31	21	0	0

In the light of these results we can conclude that:

Table 8: Results of the post-hoc test for the mean accuracy coefficient using rankings with 60% missing labels

method	p-value	rank	win	tie	loss
LRT100	-	1.29	-	-	-
LRT25	**1.4083e-01**	2.95	21	0	0
F-LRT100	**1.4083e-01**	2.98	17	1	3
W-LRT100	5.6346e-03	4.19	20	1	0
F-LRT25	4.2945e-04	4.90	21	0	0
W-LRT25	9.4333e-06	5.74	21	0	0
IBLR	3.0271e-09	7.10	19	0	2
LRT	6.7721e-12	7.95	21	0	0
W-LRT	2.8084e-15	8.90	21	0	0
F-LRT	1.3525e-15	9.00	21	0	0

- LRT, W-LRT and F-LRT algorithms, taken as basis for our all proposals, are always ranked in the last positions. However, their performance clearly improve when applying bagging. We also observe that, as expected, the use of bagging boosts the performance of the two weak classifiers (W-LRT and F-LRT), given rise to faster ensembles.
- The algorithm LRT100 is ranked in the first position in the three scenarios and so taken as control for the post-hoc tests.
- In the complete case, the post-hoc analysis reveals that IBLR, LRT25 and F-LRT100 show no statistically significant difference with respect to LRT100. These results confirm the good performance of IBLR in the complete case, being competitive with the ensemble methods. Furthermore, although LRT100 gets the first position in the ranking, two more faster ensembles show no statistical difference with respect to it.
- In the incomplete-30 and the incomplete-60 cases the post-hoc analysis yields a similar result to the complete one, but now, IBLR is no competitive anymore. This fact emphasizes how the ensemble-based methods cope better with the uncertainty of the incomplete cases than IBLR.

5.4.2 Tree size

In [1] the authors compare the tree size, measured as number of tree nodes, between the tree learnt by LRT and the one obtained by standard classification[7] tree induction algorithm (J48 Weka implementation [43] of C4.5 [26]) for the same dataset. The authors report that the size of trees learnt by LRT is similar or sometimes even smaller than the corresponding ones learnt by J48.

Here we have compared the size of the trees learnt by the original LRT algorithm and the one of the trees learnt by our two simpler proposals: W-LRT and F-LRT. The results, normalized by the LRT tree size, are shown for each dataset (and also on average) in Table 9. As can be observed, the size of the trees obtained by F(W)-LRT is always between 0.7 and 1.5 times the size of the tree obtained by LRT. In fact, if we look the average over the 21 datasets, there is no difference between the size of the trees obtained by the three algorithms (LRT, W-LRT and F-LRT). Therefore, the use of less informed thresholds does not have a significant impact on the size of the obtained trees. This result is important, as when using the bagging approach, b trees must be simultaneously stored in memory.

Table 9: Mean tree size of W-LRT and F-LRT normalized by LRT tree size.

Complete rankings

method	aut	bod	cal	cpu	ele	fri	gla	hou	iri	pen	seg	sto	veh	vow	win	wis	spo	hea	dtt	col	dia	avg.
W-LRT	1.174	0.958	0.896	0.89	0.936	0.971	0.968	0.795	1.23	0.94	1.105	0.912	1.05	0.94	1.259	0.768	0.915	0.951	0.959	0.966	0.938	0.977
F-LRT	0.999	0.926	0.962	0.913	0.953	0.733	0.916	1.011	1.321	0.905	1.24	0.962	0.989	0.949	1.334	0.985	0.825	0.894	1.028	1.029	1.04	0.996

Rankings with 30% missing labels

method	aut	bod	cal	cpu	ele	fri	gla	hou	iri	pen	seg	sto	veh	vow	win	wis	spo	hea	dtt	col	dia	avg.
W-LRT	1.28	0.984	0.902	0.885	0.941	1.033	1.01	0.806	1.296	0.962	1.15	0.963	1.117	0.945	1.331	0.771	0.91	0.948	0.961	0.956	0.943	1.004
F-LRT	1.078	0.959	0.971	0.916	0.949	0.788	1.02	1.012	1.413	0.924	1.291	1.013	1.088	0.956	1.458	0.976	0.825	0.89	1.025	1.021	1.039	1.029

Rankings with 60% missing labels

method	aut	bod	cal	cpu	ele	fri	gla	hou	iri	pen	seg	sto	veh	vow	win	wis	spo	hea	dtt	col	dia	avg.
W-LRT	1.561	0.957	0.935	0.862	0.957	1.255	1.064	0.821	1.163	0.98	1.29	1.104	1.256	0.96	1.464	0.789	0.902	0.935	0.962	0.958	0.946	1.053
F-LRT	1.333	0.949	1.0	0.932	0.958	1.01	1.084	1.035	1.297	0.939	1.444	1.156	1.192	0.97	1.379	0.939	0.823	0.884	1.019	1.017	1.042	1.067

5.4.3 Time

In this paper we deal with two different paradigms of machine learning, instance-based and model-based, which unevenly distribute the time requirements between the learning and inference phases. Therefore, for the sake of a fair CPU-time based comparison we consider the time used in the whole process: learning from the training set and validating the test set. Table 10 shows the CPU time (seconds) required by IBLR, LRT, F-LRT and W-LRT for carrying out a 5×10 cv over each dataset. Hence, the time needed to learn and validate a single model is about 1/50 of the amount reported. As can be observed, LRT is the most expensive algorithm in terms of CPU time,

[7]To obtain a standard classification problem, only the first label in the ranking is taken as class label.

needing 40% more time than IBLR on average, although they perform quite similar when the number of instances grows. Regarding the two weak tree-based classifiers, they run two orders of magnitude faster than the original LRT algorithm. To obtain (roughly) the time needed by the bagging-based versions of the tree-based algorithms, we can multiply the time shown in Table 10 by the number of b bags considered.

Therefore, we can conclude that:

- Even though LRT-based bagging classifiers obtain very good accuracy, they are not affordable in practice when the number of instances grows, due to the required amount of CPU time.
- Bagging F-LRT and W-LRT with $b \in \{50, 100\}$ needs at most the same CPU time than LRT and IBLR, in particular in the largest datasets (cal, ele and fri).
- The combination of these observations with those obtained in Section 5.4.1 clearly points out to F-LRT as the algorithm with the best trade-off accuracy-time, as it is always in the *top* group of algorithms according to accuracy, but it is also much faster than the other algorithms in such group (LRT100 and LRT50).

Table 10: Mean CPU time (s) used by the base algorithms

Complete rankings																					
method	aut	bod	cal	cpu	ele	fri	gla	hou	iri	pen	seg	sto	veh	vow	win	wis	spo	hea	dtt	col	dia
IBLR	4.46	0.95	1637.15	269.13	1522.89	8306.15	0.70	5.20	18.17	636.52	24.58	7.41	7.72	6.73	8.21	8.57	38.43	40.99	33.04	35.61	39.80
LRT	27.10	3.22	1768.33	556.70	1537.55	8331.00	2.61	6.14	0.74	760.15	304.20	14.07	12.27	39.78	2.10	7.99	1460.43	745.38	716.06	530.90	992.36
W-LRT	4.86	0.10	13.07	6.05	27.02	57.99	0.14	0.25	0.09	29.36	3.41	0.71	1.10	0.54	0.75	0.25	6.88	5.50	6.68	6.79	6.34
F-LRT	4.38	0.15	11.69	5.47	24.39	53.97	0.12	0.27	0.09	26.04	3.89	1.59	1.13	0.46	1.26	0.36	5.86	5.05	5.01	5.25	5.58
Rankings with 30% missing labels																					
method	aut	bod	cal	cpu	ele	fri	gla	hou	iri	pen	seg	sto	veh	vow	win	wis	spo	hea	dtt	col	dia
IBLR	10.84	5.55	1638.48	265.08	1414.88	7294.81	7.77	10.09	8.92	675.28	29.81	12.08	11.51	13.49	10.89	12.49	32.39	38.75	31.60	38.98	70.96
LRT	25.64	4.09	2098.66	627.69	1720.57	9797.26	3.55	6.81	0.75	1059.86	362.64	12.97	13.44	53.34	3.20	13.36	2442.06	818.01	756.56	636.01	1197.88
W-LRT	4.76	0.14	14.13	7.14	28.93	58.03	0.69	0.26	0.08	33.12	4.04	0.50	1.18	0.73	0.19	0.22	9.30	6.18	7.38	6.89	7.18
F-LRT	4.20	0.19	12.53	6.80	26.69	56.07	0.20	0.29	0.12	31.97	4.83	0.50	1.24	0.59	0.16	0.50	7.47	5.45	5.59	5.53	6.11
Rankings with 60% missing labels																					
method	aut	bod	cal	cpu	ele	fri	gla	hou	iri	pen	seg	sto	veh	vow	win	wis	spo	hea	dtt	col	dia
IBLR	15.18	11.69	1652.02	271.43	1390.15	7240.44	11.48	12.38	7.52	664.03	28.81	8.02	18.63	18.52	7.95	9.85	31.63	37.30	31.88	35.04	50.33
LRT	19.58	4.85	2156.29	609.18	1691.78	9466.61	3.30	5.98	2.86	955.61	295.43	12.82	13.06	43.67	3.23	10.21	2247.73	818.31	997.7	778.34	1108.6
W-LRT	4.3	0.19	13.78	6.31	29.22	56.03	0.34	0.39	0.09	30.61	3.7	0.51	1.13	0.55	0.2	0.34	8.88	6.44	7.58	6.61	7.17
F-LRT	3.55	0.38	11.82	6.43	26.91	53.34	0.18	0.37	0.10	28.63	4.07	0.49	1.09	0.56	0.35	0.67	7.06	5.83	6.07	5.67	6.12

5.4.4 Some comments on scalability

As a complement to the analysis of the CPU time required by the studied algorithms, here we carry out the analysis of the theoretical complexity orders reported in Section 4.1. As will be observed, the analysis is in line with the results shown in the previous section. Nevertheless, here we put emphasis on the scalability to larger problems.

Regarding the tree-based models, LRT is linear in the number of predictive attributes and sub-cubic ($N^2log(N)$) regarding the number of instances. On the other hand the two weak learners, W-LRT and F-LRT, are sub-quadratic ($Nlog(N)$) in the number of instances. Therefore, both algorithms show a good scalability behaviour when the number of predictive attributes grows (high dimensionality) but LRT is poorly scalable when the number of instances grows significantly (large samples). Actually, as aforementioned, the weak learners are (in theory) N times faster than LRT.

With respect to the ensembles, the complexity of the learning step increases proportionally to the number of bags b. Therefore, the same comments as for the base algorithms apply to the ensemble case. However, in favour of bagging it must be pointed out that a coarse-grain parallel implementation for the ensemble can be easily produced, so dividing the time required by the number of available nodes/cores. Furthermore, it is also of interest to observe that as usually $b << N$, the ensemble of weak learners can be learnt even faster than a single LRT. This is of course a valuable feature in favour of bagging F-LRT when facing large sample problems, specially if we also take into account that it obtains more accurate models.

Finally, if we focus on IBLR, the first thing we must notice is that even being a lazy learner, it actually performs a learning stage in the way of model/parameter selection. In fact, one of the key points of its good accuracy is the selection of the number of neighbours to be used for each dataset. Thus, by assuming that distances computation dominates consensus computation, a lower bound for the learning step complexity is $O(N^2n)$, that is, quite similar to LRT. However, while LRT requires $O(log(N))$ for prediction, IBLR needs $O(Nn)$. Although this is a known fact for lazy learners, it represents a clear disadvantage, because models are learnt from data only once but *queried* many times. Therefore, in the large sample case, and taking into account the learning time complexity, the preference order among the compared algorithms becomes Bagging(W/F-LRT) $\succ$ IBLR $\succ$ LRT $\succ$ Bagging(LRT). Nonetheless, if we also take into account the complexity of doing inference over a large number of instances, then IBLR may be the least preferred algorithm.

6 Conclusions

In this paper we deal with the label ranking problem. Inspired by the decision tree algorithm (LRT) proposed in [1], we design two weak classifiers which can be learnt more efficiently.

From the experimental study we can conclude that bagging the weak learner using unsupervised frequency-based discretization to select the split point (F-LRT), is competitive with the ensemble of LRT in terms of accuracy. Actually, bagging F-LRT is also competitive in accuracy to the state-of-the-art IBLR algorithm.

We have also studied the problem of dealing with partial information. In this scenario our proposal significantly outperform IBLR algorithm, being the difference bigger as the number of missing labels grows.

As future research we plan to use different techniques to build the ensemble, e.g. boosting and/or stacking. Furthermore, we also plan to use simpler unstable classifiers as base models, as could be the case of NB-like hybrid probabilistic classifiers.

7 Acknowledgments

This study has been partially funded by FEDER funds, the Spanish Government (MINECO) and the Regional Government (JCCM) through projects TIN2013-46638-C3-3-P and PEII-2014-049-P. The authors would like to thank Prof. Eyke Hüllermeier because of his help with some issues related to paper [1].

References

[1] W. Cheng, J. C. Huhn, E. Hüllermeier, Decision tree and instance-based learning for label ranking, in: Proceedings of the 26th Annual International Conference on Machine Learning, ICML09, 2009, pp. 161–168.

[2] J. Fürnkranz, E. Hüllermeier, Preference Learning, Springer-Verlag New York, Inc., 2010.

[3] M. de Gemmis, L. Iaquinta, P. Lops, C. Musto, F. Narducci, G. Semeraro, Learning preference models in recommender systems, in: Preference Learning., 2010, pp. 387–407.

[4] T.-Y. Liu, Learning to Rank for Information Retrieval, Springer Berlin Heidelberg, 2011.

[5] S. Henzgen, E. Hüllermeier, Mining Rank Data, Springer International Publishing, 2014, pp. 123–134.

[6] E. Frank, M. Hall, A simple approach to ordinal classification, in: Proceedings of the 12th European Conference on Machine Learning (ECML'01), 2001, pp. 145–156.

[7] T.-Y. Liu, The listwise approach, in: Learning to Rank for Information Retrieval, Springer Berlin Heidelberg, 2011, pp. 71–88.

[8] W. Cheng, S. Henzgen, E. Hüllermeier, Labelwise versus pairwise decomposition in label ranking, in: Proceedings of the Workshop on Lernen, Wissen & Adaptivität (LWA2013), 2013, pp. 129–136.

[9] S. Destercke, M.-H. Masson, M. Poss, Cautious label ranking with labelwise decomposition, European Journal of Operational Research 246 (3) (2015) 927 – 935.

[10] E. Hüllermeier, J. Fürnkranz, W. Cheng, K. Brinker, Label ranking by learning pairwise preferences, Artificial Intelligence 172 (16-17) (2008) 1897–1916.

[11] M. Gurrieri, P. Fortemps, X. Siebert, Alternative Decomposition Techniques for Label Ranking, Springer International Publishing, 2014, pp. 464–474.

[12] S. Har-Peled, D. Roth, D. Zimak, Constraint classification for multiclass classification and ranking, in: Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada], 2002, pp. 785–792.

[13] C. L. Mallows, Non-null ranking models. I, Biometrika 44 (1-2) (1957) 114–130.

[14] F. Schalekamp, A. van Zuylen, Rank aggregation: Together we're strong, in: Proceedings of the 11th Workshop on Algorithm Engineering and Experiments (ALENEX), 2009, pp. 38–51.

[15] C. R. de Sa, C. Soares, A. M. Jorge, P. J. Azevedo, J. P. da Costa, Mining association rules for label ranking, in: Advances in Knowledge Discovery and Data Mining - 15th Pacific-Asia Conference, PAKDD 2011, Shenzhen, China, May 24-27, 2011, Proceedings, Part II, 2011, pp. 432–443.

[16] G. Ribeiro, W. Duivesteijn, C. Soares, A. J. Knobbe, Multilayer perceptron for label ranking, in: Artificial Neural Networks and Machine Learning - ICANN 2012 - 22nd International Conference on Artificial Neural Networks, Lausanne, Switzerland, September 11-14, 2012, Proceedings, Part II, 2012, pp. 25–32.

[17] W. Cheng, K. Dembczynski, E. Hüllermeier, Label ranking methods based on the plackett-luce model, in: Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel, 2010, pp. 215–222.

[18] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140.

[19] C. Dwork, R. Kumar, M. Naor, D. Sivakumar, Rank aggregation methods for the web, in: Proceedings of the 10th International Conference on World Wide Web, ACM, 2001, pp. 613–622.

[20] J. L. Kemeny, J. G. Snell, Mathematical models in the social sciences, Blaisdell, New York.

[21] J. Borda, Memoire sur les elections au scrutin, Histoire de l'Academie Royal des Sciences.

[22] A. Ali, M. Meila, Experiments with kemeny ranking: What works when?, Mathematical Social Sciences 64 (1) (2012) 28–40.

[23] P. Emerson, The original borda count and partial voting, Social Choice and Welfare 40 (2) (2013) 353–358.

[24] M. A. Fligner, J. S. Verducci, Distance based ranking models, Journal of the Royal Statistical Society 48 (3) (1986) 359–369.

[25] J. Hernández-González, I. Inza, J. A. Lozano, Weak supervision and other non-standard classification problems: A taxonomy, Pattern Recognition Letters 69 (2016) 49–55.

[26] J. R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, 1993.

[27] L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone, Classification and Regression Trees, Wadsworth, 1984.

[28] J. R. Quinlan, Learning with continuous classes, in: 5th Australian Joint Conference on Artificial Intelligence, World Scientific, Singapore, 1992, pp. 343–348.

[29] Y. Wang, I. H. Witten, Induction of model trees for predicting continuous classes, in: Poster papers of the 9th European Conference on Machine Learning, Springer, 1997.

[30] J. A. Aledo, J. A. Gámez, D. Molina, Using extension sets to aggregate partial rankings in a flexible setting, Applied Mathematics and Computation 290 (2016) 208–223.

[31] L. I. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, Wiley-Interscience, 2004.

[32] I. H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems), Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[33] J. Ceberio, Solving Permutation-based Combinatorial Optimization Problems with Estimation of Distribution Algorithms and Extensions Thereof, PhD Thesis, University of the Basque Country (EHU), 2014.

[34] C. R. de Sa, C. Soares, A. J. Knobbe, P. J. Azevedo, A. M. Jorge, Multi-interval discretization of continuous attributes for label ranking, in: Discovery Science, 2013, pp. 155–169.

[35] R. L. Plackett, The analysis of permutations, Journal of the Royal Statistical Society. Series C (Applied Statistics) 24 (2) (1975) 193–202.

[36] D. Luce, Individual Choice Behavior, Wiley, 1959.

[37] W. Cheng, K. Dembczynski, E. Hüllermeier, Label ranking methods based on the plackett-luce model, in: ICML, 2010, pp. 215–222.

[38] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.

[39] S. García, F. Herrera, An extension on ?statistical comparisons of classifiers over multiple data sets? for all pairwise comparisons, Journal of Machine Learning Research 9 (2008) 2677–2694.

[40] J. Arias, J. Cózar, ExReport: Fast, reliable and elegant reproducible research (2015).
URL `http://exreport.jarias.es/`

[41] M. Friedman, A Comparison of Alternative Tests of Significance for the Problem of m Rankings, The Annals of Mathematical Statistics 11 (1) (1940) 86–92.

[42] S. Holm, A simple sequentially rejective multiple test procedure, Scandinavian Journal of Statistics 6 (1979) 65–70.

[43] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten, The weka data mining software: An update, SIGKDD Exploration Newsletter 11 (1) (2009) 10–18.