Adapting Decision DAGs for Multipartite Ranking[⋆]

José Ramón Quevedo, Elena Montañés, Oscar Luaces and Juan José del Coz
{quevedo,elena,oluaces,juanjo}@aic.uniovi.es

Artificial Intelligence Center, University of Oviedo at Gijón (Spain)

Abstract. Multipartite ranking is a special kind of ranking for problems in which classes exhibit an order. Many applications require its use, for instance, granting loans in a bank, reviewing papers in a conference or just grading exercises in an education environment. Several methods have been proposed for this purpose. The simplest ones resort to regression schemes with a pre- and post-process of the classes, what makes them barely useful. Other alternatives make use of class order information or they perform a pairwise classification together with an aggregation function. In this paper we present and discuss two methods based on building a Decision Directed Acyclic Graph (DDAG). Their performance is evaluated over a set of ordinal benchmark data sets according to the C-Index measure. Both yield competitive results with regard to state-of-the-art methods, specially the one based on a probabilistic approach, called PR-DDAG.

1 Introduction

Multipartite ranking has been recently named [11] as an extension of the traditional bipartite ranking from the binary to the multiclass case. Bipartite ranking aims to learn a model whose performance is evaluated according to its ability of sorting positive before negative examples. Such ability is commonly assessed in terms of the AUC, which is the area under the Receiver Operating Characteristic (ROC) curve [6]. In fact, multipartite ranking has also been called ordinal ranking, since it relates an ordinal classification and a ranking. Ordinal classification means to perform a classification whose classes display an order. Ordinal ranking goes further and also provides a ranking of the examples within the same class. Obviously, a good multipartite ranker is expected to place examples of the higher classes before examples of the lower ones.

Many applications may take advantage of this special kind of ranking. For instance, a banker commonly classifies customers that ask for a mortgage into classes as high risk, moderate risk, low risk, no risk. Imagine now that a certain number of mortgages are able to grant according to the interests of the bank

[⋆] This research has been partially supported by Spanish Ministerio de Ciencia e Innovación (MICINN) grants TIN2007-61273 and TIN2008-06247 and by FICYT, Asturias, Spain, under grant IB09-059-C2.

and that such number falls below the number of customers classified as no risk. Obviously, the bank has to decide within the no risk customers to who give the mortgage. Hence, an order within the examples of each class must be provided. The same demand happens in many other environments, for instance in job vacancies, paper reviews in a conference or waiting list in hospitals according to the urgency degree of the disease.

The main goal of this paper is to explore the use of a Decision Directed Acyclic Graph (DDAG) [24] to address the problem of multipartite ranking. DDAGs have been successfully applied before, but to the best of our knowledge for classification purposes rather than for ranking [3,7,23,24,27]. Unlike other approaches, our proposal makes use of the class order information to lead the build of the graph. We present two different methods that exploit the structure of a DDAG to produce a ranking. The first one follows a crisp approach and produces a global ranking by concatenating a set of consecutive bipartite rankings. The second one is a probabilistic approach where each example is propagated through all possible paths with a cumulative probability attached. The performance of both methods is as good as some state-of-the-art techniques, outperforming them in some situations. In addition, they are computationally more efficient than other algorithms used for the same purpose.

The rest of the paper is organized as follows. Section 2 includes an overview of some related work. In Section 3, the multipartite ranking problem is stated and the main state-of-the-art approaches are described. Then, Section 4 presents the two different DDAG-based methods proposed in this paper to tackle multipartite ranking. In Section 5 results of the experiments over benchmark data sets are described and analyzed. Finally, Section 6 draws some conclusions.

2 Related Work

Multipartite ranking is closely related to other fields that have been extensively studied. Ordinal classification, a research field between classification and regression, is one of the closest. In fact, it shares properties with the former that a specific number of classes is stated, and with the latter that such classes are ordered. This is the reason why most of the research is focused on adapting either classification or regression techniques to cope with ordinal classification. However, none of them seem completely adequate, since the former discards class order information, whereas the latter exploits that order relation but making strong assumptions about the distances between classes [19]. Recently, some work has been done to exploit class order information. In [2], authors reduce the problem to the standard two-class problem using both support vector machines and neural networks also providing generalization bounds. Two new support vector approaches for ordinal regression, which optimize multiple thresholds to define parallel discriminant hyperplanes for the ordinal scales are proposed in [4]. Frank and Hall [8] present a simple approach encoding the order information in class attributes allowing the use of any classifier that provides class probability estimates. Also, Herbrich et al. [14] proposed a distribution independent risk formulation of ordinal regression. It allows to derive an uniform convergence bound whose application configures a large margin algorithm based on a mapping from objects to scalar utility values, thus classifying pairs of objects.

Label ranking is another field also closely related to multipartite ranking, since a required order for the classes is stated. In this framework the goal is to learn a mapping from instances to rankings over a finite number of labels. However, the labels are ranked instead of the instances, as in multipartite ranking. Some research deal with this discipline, as the work in [17], where authors learn a ranking by pairwise comparison or in [1], in which they propose a sophisticated k-NN framework as an alternative to previous binary decomposition techniques.

No so much research can be found about multipartite ranking. In [26], the authors derive distribution-free probabilistic bounds on the expected error of a ranking function learned by a k-partite ranking algorithm. Waegeman et al. [29] generalize the Wilcoxon-Mann-Whitney statistic to more than two ordered categories in order to provide a better evaluation system in ordinal regression. One of the most recent work that copes with the problem itself is [11], in which the use of binary decomposition techniques are explored. On one hand, the authors adapt for this purpose an ordinal classification method [8] that converts the original m class problem into a $m-1$ binary problems. On the other hand, they analyze the use of pairwise classification [10] to induce also a multipartite ranker. The main problem that arises in decomposition approaches is that after such decomposition, an aggregation scheme must be adopted to compose again the original problem. In multipartite ranking combining scoring functions of each model is a better practice than combining the rankings yielded by each model [11].

3 Multipartite Ranking

As commented above, multipartite ranking provides a ranking of instances in ordinal classification tasks. Let us include some definitions to formally state the problem.

Definition 1 *Let be $\mathcal{L} = \{\ell_1, ..., \ell_p\}$ a set of classes. Then, a natural order relation denoted by $\prec$ is defined over $\mathcal{L}$ such that it holds $\ell_i \prec \ell_j$ if ℓ_i is semantically below ℓ_j, with $i, j \in \{1, ..., p\}$.*

Definition 2 *Let be $\mathcal{L} = \{\ell_1, ..., \ell_p\}$ a set of classes that satisfy a natural order relation denoted by $\prec$. Let also be $\mathcal{S} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_m\}$ a set of m instances from an input space $\mathcal{X}$, in which each instance $\boldsymbol{x}_i$ is labeled as $\ell_{\boldsymbol{x}_i} \in \mathcal{L}$. Then, the goal of multipartite ranking consists of obtaining a ranking function $f : \mathcal{X} \rightarrow \mathbb{R}$ such that as many as possible $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}$ such that $\ell_{\boldsymbol{x}_i} \prec \ell_{\boldsymbol{x}_j}$ it must satisfy that $f(\boldsymbol{x}_i) < f(\boldsymbol{x}_j)$.*

Thus, $f(\cdot)$ must place all the instances classified under the class ℓ_i before any instance classified under the class ℓ_j whenever $\ell_i \prec \ell_j$, with $i, j \in 1, ..., p$.

A well-known approach to address this task proposes to convert the original problem into a single binary classification problem. It involves including several pairwise constraints according to the order relation defined over $\mathcal{L}$, each one being a new instance that will feed the binary classifier. Formally, since for all $i, j \in \{1, ..., m\}$ such that $\ell_{\boldsymbol{x_i}} \prec \ell_{\boldsymbol{x_j}}$ it must be held that $f(\boldsymbol{x_i}) < f(\boldsymbol{x_j})$ and assuming that f is linear, then it holds that $f(\boldsymbol{x_i} - \boldsymbol{x_j}) < 0$. Analogously, $f(\boldsymbol{x_k} - \boldsymbol{x_l}) > 0$ for all $k, l \in \{1, ..., m\}$ such that $\ell_{\boldsymbol{x_l}} \prec \ell_{\boldsymbol{x_k}}$. Hence, $\mathcal{S}^+ = \{\boldsymbol{x_i} - \boldsymbol{x_j} / \ell_{\boldsymbol{x_i}} \prec \ell_{\boldsymbol{x_j}}\}$ is the set of positive examples of the binary classification task and $\mathcal{S}^- = \{\boldsymbol{x_k} - \boldsymbol{x_l} / \ell_{\boldsymbol{x_l}} \prec \ell_{\boldsymbol{x_k}}\}$ conforms the negative ones. The main disadvantage of this approach is that the number of instances for such binary classification problem is the order of $\mathcal{O}(m^2)$.

Decomposition methods involve several binary learning problems instead of a single one, but with the advantage that such problems are smaller and do not require to increase the number of instances. Some approaches fall into this kind of decomposition previously used for classification [8,10] and recently adapted to multipartite ranking [11]. They differ in the selection of the binary problems they solve.

The Frank and Hall (FH) approach [8] defines $p-1$ binary problems, where the i-th problem consists of obtaining a model $\mathcal{M}_i$ able to separate the class $\mathcal{C}_i^+ = \{\ell_1, ..., \ell_i\}$ from the class $\mathcal{C}_i^- = \{\ell_{i+1}, ..., \ell_p\}$, when i ranges from 1 to $p-1$. This model provides the probability, denoted by $P(\ell_i \prec \ell_{\boldsymbol{x}})$, of an instance $\boldsymbol{x}$ of belonging to a class higher than ℓ_i in the order relation defined over $\mathcal{L}$. Such probabilities in turn define one ranking per model. These rankings must be aggregated to obtain a global one. The aggregation function must guarantee that instances of lower classes keep lower in the global ranking and so does the function defined as follows

$$\mathcal{M}(\boldsymbol{x}) = \sum_{i=1}^{p-1} \mathcal{M}_i(\boldsymbol{x}) = \sum_{i=1}^{p-1} P(\ell_i \prec \ell_{\boldsymbol{x}}). \tag{1}$$

The learning by pairwise comparison (LPC) or round robin learning [10] defines $p(p-1)/2$ binary problems, where the i, j-th problem consists of separating the class ℓ_i and ℓ_j, when i, j range from 1 to p and $\ell_i \prec \ell_j$. This approach is also used in other learning tasks, as in multiclass classification, where the output of each induced model $\mathcal{M}_{i,j}$ for an example $\boldsymbol{x}$ is accounted as a vote for the predicted class ℓ_i or ℓ_j; then the most voted class is predicted following the MaxWins voting scheme [9]. But, this method is not suitable in multipartite ranking, since it is not able to induce a ranking. In [11], the authors propose two alternative aggregation functions, assuming that binary models yield normalized scores, i.e. $\mathcal{M}_{i,j}(\boldsymbol{x}) \in [0, 1]$, which is the probability that $\boldsymbol{x}$ belongs to class ℓ_j. Both sum the predictions in favor of the higher class,

$$\mathcal{M}(\boldsymbol{x}) = \sum_{1 \le i < j \le p} \mathcal{M}_{i,j}(\boldsymbol{x}), \tag{2}$$

but in the second one those predictions are weighted by the relative frequencies, p_i, p_j of the classes ℓ_i, ℓ_j in the training set,

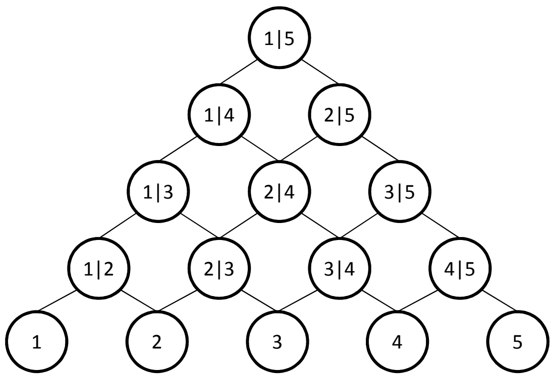

Fig. 1. A Decision Directed Acyclic Graph

$$\mathcal{M}(\boldsymbol{x}) = \sum_{1 \le i < j \le p} p_i p_j \mathcal{M}_{i,j}(\boldsymbol{x}). \tag{3}$$

4 Two DDAG-based methods for multipartite ranking

In this section we will present and discuss two different methods that are based on a common idea, building a DDAG [24] to cope with the multipartite ranking problem. The first one, called CR-DDAG, produces a global ranking by combining a set of consecutive bipartite rankings. The second one, called PR-DDAG, adds a probabilistic framework based on the structure of a DDAG. Since both methods share the same DDAG structure, we will first describe and motivate it.

A Directed Acyclic Graph (DAG) is a graph whose edges have an orientation and no cycles. A special case is the DDAG presented in [24]. In that paper, the authors describe and analyze a method to solve multiclass classification problems employing a DAG to combine the set of binary classifiers yielded by a decomposition scheme identical to the one used by LPC (see previous section). Thus, DDAG also trains $p(p-1)/2$ classifiers, one for each pair of classes. However, LPC and DDAG differ in the way they combine the predictions of those classifiers. In the latter, the structure of the DDAG determines such combination.

More in detail, the nodes of a DDAG are arranged in a triangle (see Figure 1) with the single root node at the top, two nodes in the second layer and so on until the final layer of p leaves. The k-th node in layer $l < p$ is connected to the k-th and $(k+1)$-th node in the $(l+1)$-th layer. Except for the leaves, each node has an associated model, namely $\mathcal{M}_{i,j}$, aimed at separating the classes ℓ_i and ℓ_j, and two successors which will be two leaves when $i = j - 1$, or two decision nodes with models $\mathcal{M}_{i,j-1}$ and $\mathcal{M}_{i+1,j}$ respectively. The prediction procedure of a DDAG works as follows (see the example in Figure 1). Starting from the root (model $\mathcal{M}_{1,5}$), a DDAG decides at each node, applying model $\mathcal{M}_{i,j}$, which of the two classes ℓ_i and ℓ_j is preferred for a certain instance $\boldsymbol{x}$. If the former is the winner, then the instance $\boldsymbol{x}$ is evaluated over its left child node (model $\mathcal{M}_{i,j-1}$ in the example), so class ℓ_j is discarded. Otherwise, the instance $\boldsymbol{x}$ is evaluated using its right child node (model $\mathcal{M}_{i+1,j}$), then class ℓ_i is discarded. The process continues until a leave is reached whose label is returned for the instance $\boldsymbol{x}$. Thus, an instance is evaluated over $p-1$ different models. Notice that a DDAG works as a list in which a class is discarded after each evaluation.

There is no consensus about which classes must be considered first and which ones in last place for each branch of the tree. For instance, in the foundational paper [24] the choice is arbitrary, in [3] a binary decision diagram and Huffman code [30] construction is employed and Jaakkola-Haussler [28] error bound is proposed in [7]. In this paper, we propose an alternative idea. As it can be easily proved, a DDAG predicts incorrectly the true class of an example $\boldsymbol{x}$ if one *competent* model in the prediction path followed by $\boldsymbol{x}$ fails. As it was defined in other papers [11], a binary model $\mathcal{M}_{i,j}$ is only competent to classify examples that belong to classes ℓ_i or ℓ_j. Thus, it is quite dangerous to locate at the root a model $\mathcal{M}_{i,j}$ if the classes ℓ_i and ℓ_j are hard to separate, because that model will evaluate all examples of these classes. Hence, placing such kind of models at lower levels is preferable, since they will classify less examples.

A simple adaptation of this idea to multipartite ranking (and in general to ordinal classification tasks) is based on the intuitive assumption, validated experimentally in [16], that the ordinal structure of the set of classes is also reflected in the topology of the input space. Thus, it seems that the lowest (ℓ_1) and highest (ℓ_p) classes of the order defined over $\mathcal{L}$ are those likely to be separated best. So, model $\mathcal{M}_{1,p}$ is proposed to be located at the root and, therefore, either the lowest or the highest class is discarded in the first layer. Recursively applying the same idea, our DDAG will place at each node the model between the two extreme classes of the ordered subset of classes that have not been discarded by its ancestor models.

Assuming that the order of the classes is $\ell_1 \prec \ell_2 \prec \ell_3 \prec \ell_4 \prec \ell_5$, Figure 1 shows the structure of a DDAG employed for both methods proposed in this paper. In symbols, if $\mathcal{M}_{i,j}$ corresponds to the k-th node in layer $l < p$, then $\mathcal{M}_{i,j-1}$ and $\mathcal{M}_{i+1,j}$ correspond to the k-th and $(k+1)$-th node in the $(l+1)$-th layer. For instance, in Figure1 the node 2|4 is the 2-nd node of the 3-rd layer that correspond to the model $\mathcal{M}_{2,4}$ which separates classes ℓ_2 and ℓ_4 and it is connected to the 2-nd and 3-rd nodes (respectively 2|3 and 3|4) of the 4-th layer that respectively correspond to models $\mathcal{M}_{2,3}$ and $\mathcal{M}_{3,4}$.

4.1 Consecutive Rankings DDAG (CR-DDAG)

At first sight, it is not trivial how to adapt a DDAGs to obtain a multipartite ranking. Originally, they were designed to deal with multiclass classification [24], which is a quite different task. Our first proposal works under the hypothesis that a multipartite ranking can be broken down into a set of ordered bipartite rankings. For instance, for a problem with 5 classes, i.e. $\mathcal{L} = \{\ell_1, ..., \ell_5\}$, then the whole multipartite ranking can be formed concatenating the consecutive bipartite rankings $\{1-2, 2-3, 3-4, 4-5\}$. Notice that these consecutive rankings correspond to the last level before the leaves in the structure of the DDAG described before (see Figure 1).

This method relies on the assumption that if each bipartite ranker orders well the examples of its two classes, then concatenating those consecutive binary rankings will lead to a good overall ranking. But, first of all a classification procedure is required to decide which bipartite ranker will be used for each instance. Here is when the DDAG structure plays its role. Taking Figure 1 as reference, the proposal consists of employing the remaining models, displayed in the higher levels of the DDAG, to select the bipartite ranker that must be applied. Thus, our DDAG is divided in two parts: the upper levels take charge of classifiying examples, and the lower level of ordering them. Hence, a mechanism to merge the consecutive rankings into a global ranking is required.

For that purpose some modifications in DDAG are carried out in order to obtain a CR-DDAG. First, the leaves are ruled out. Secondly, the models of the layer immediately before the leaves, i.e. the set of consecutive bipartite rankers ($\mathcal{M}_{i,i+1}$), must yield a value in the interval $(0, 1)$. Finally, the final ranking score is computed according to the following expression

$$\mathcal{M}(\boldsymbol{x}) = i + \mathcal{M}_{i,i+1}(\boldsymbol{x}), \tag{4}$$

where $\mathcal{M}_{i,i+1}$ correspond to the model selected by the cascade classification process carried out by the higher nodes of the DDAG. Adding the class index i to the output of the bipartite ranker guarantees that an instance ranked by $\mathcal{M}_{i,i+1}$ is placed in the global ranking higher than an instance ranked by $\mathcal{M}_{j,j+1}$ with $j < i$.

The main disadvantage of CR-DDAG is that if a competent model for an instance of class ℓ_j fails during the classification process, then such instance will follow a path ending in node whose model will be $\mathcal{M}_{i,i+1}$ with $j \notin \{i, i+1\}$. Then, if $j < i$ (respectively $j > i+1$), the instance may be placed much higher (respectively much lower) in the global ranking than it should be. Therefore, CR-DDAG has two potential sources of errors. On one hand, the classification process can choose the incorrect ranking model, and, on the other hand, the bipartite rankers may commit mistakes. The first one is more damaging in the sense that it could produce bigger losses in ranking evaluation measures.

4.2 Probabilistic Ranking DDAG (PR-DDAG)

In order to diminish some of the undesirable drawbacks of CR-DDAG, we propose a Probabilistic Ranking Decision Directed Acyclic Graph in which examples are propagated through every edge with a probability attached, so no irrevocable decisions are taken. In fact, PR-DDAG shares some ideas with the LPC approach. First, it exploits the redundancy considering the outcomes of several models to rank an instance. And second, it relies on the assumption that the order of the output space is also reflected on the topology of the input space, in the sense that the prediction of $\mathcal{M}_{i,k}(\boldsymbol{x})$ for an instance of class ℓ_j will be higher if $k \leq j$ and lower if $j \leq i$. However, the main difference with LPC is that PR-DDAG computes the posterior probabilities using the structure of the DDAG. The goal is that the contribution of competent models will be higher in the global ranking, reducing the effect of the so-called *non-competence problem*. In the next section, we will explain deeply this property.

PR-DDAG suposses that every model $\mathcal{M}_{i,j}(\boldsymbol{x})$ predicts the probability of $\boldsymbol{x}$ of belonging to the positive class (ℓ_j). Obviously, the probability of being of the negative class (ℓ_i) will be $1 - \mathcal{M}_{i,j}(x)$. These probabilities are successively propagated through the graph from the root to the leaves, in such a way that an instance will reach a node $i|j$ with a probability $P_{i,j}(\boldsymbol{x})$ computed as follows

$$P_{i,j}(\boldsymbol{x}) = \begin{cases} 1, & i = 1, j = p, \\ P_{i,j+1}(\boldsymbol{x}) \cdot (1 - \mathcal{M}_{i,j+1}(\boldsymbol{x})), & i = 1, j < p, \\ P_{i-1,j}(\boldsymbol{x}) \cdot \mathcal{M}_{i-1,j}(\boldsymbol{x}), & i > 1, j = p, \\ P_{i-1,j}(\boldsymbol{x}) \cdot \mathcal{M}_{i-1,j}(\boldsymbol{x}) + P_{i,j+1}(\boldsymbol{x}) \cdot (1 - \mathcal{M}_{i,j+1}(\boldsymbol{x})), & i > 1, j < p. \end{cases} \quad (5)$$

Notice that the method states that $P_{1,p}(\boldsymbol{x}) = 1$, since any instance $\boldsymbol{x}$ must arrive to the root node with probability 1. At the end, this propagation process provides a probability distribution of the classes for an instance $\boldsymbol{x}$, that is, $\{P_{i,i}(\boldsymbol{x}) : i = 1, \ldots, p\}$.

Finally, in order to produce the global ranking, PR-DDAG employes the aggregation function proposed in [20]:

$$\mathcal{M}(\boldsymbol{x}) = \sum_{i=1}^{p} T(i) \cdot P_{i,i}(\boldsymbol{x}), \quad (6)$$

where $T(i)$ is some monotone increasing function of the relevance level i. According to [20], it seems that complex functions do not provide better rankings, hence, the simple $T(i) = i$, called *expected relevance*, can be a sensible and stable choice. In this case, the product by the index of the class in the summation enhances the value of the probability as the index of the class increases.

4.3 Analyzing the properties of CR-DDAG and PR-DDAG

Table 1 shows a summary of the main properties of all approaches described above (FH, LPC, CR-DDAG and PR-DDAG). All methods solve $p(p-1)/2$ binary problems except FH which solves just $p-1$, which is related to the training set used in each binary problem they solve. Particularly, FH employs the whole data set, whereas only instances of two classes are used in the rest of the approaches. All of them take each instance $(p-1)$ times, and assuming an uniform distribution of the instances through the classes, LPC, CR-DDAG and PR-DDAG only handle m/p instances in each binary problem against m that FH uses. Then, taking into account the training size and that a $\mathcal{O}(m^\alpha)$ binary classifier (typically with $\alpha > 1$) is chosen, FH has a complexity $\mathcal{O}(pm^\alpha)$, whereas LPC, CR-DDAG and PR-DDAG reduce it to $\mathcal{O}(p^{2-\alpha}m^\alpha)$. Thus, those methods are more efficient than FH whenever a base learner with super-linear

Table 1. Binary problems and evaluations required by each method

	FH	LPC	CR-DDAG	PR-DDAG
Binary problems	$p-1$	$p(p-1)/2$	$p(p-1)/2$	$p(p-1)/2$
Classes in training	All classes	Only two	Only two	Only two
Whole Training size	$(p-1)m$	$(p-1)m$	$(p-1)m$	$(p-1)m$
Binary Training size	m	m/p	m/p	m/p
Binary complexity	$\mathcal{O}(m^\alpha)$	$\mathcal{O}(p^{-\alpha}m^\alpha)$	$\mathcal{O}(p^{-\alpha}m^\alpha)$	$\mathcal{O}(p^{-\alpha}m^\alpha)$
Whole complexity	$\mathcal{O}(pm^\alpha)$	$\mathcal{O}(p^{2-\alpha}m^\alpha)$	$\mathcal{O}(p^{2-\alpha}m^\alpha)$	$\mathcal{O}(p^{2-\alpha}m^\alpha)$
Evaluations	$p-1$	$p(p-1)/2$	$p-1$	$p(p-1)/2$
Non-competence problem	No	Yes	No	Yes
Redundancy	No	Yes	No	Yes

complexity is applied. Concerning to the evaluation stage (see again Table 1) FH is comparable to CR-DDAG, which only need to compute $p-1$ evaluations, while LPC and PR-DDAG evaluate all models.

In classification, the non-competence problem does actually not matter so much for LPC provided all competent models make correct predictions; the same is true for DDAGs. As explained in [11], however, this property is lost for LPC in the case of multipartite ranking. Interestingly, it seems to be maintained for CR-DDAG: as long as all competent binary classifiers make correct decisions, an instance from class ℓ_i must end up either in the bipartite models $\mathcal{M}_{i-1,i}$ or $\mathcal{M}_{i,i+1}$, and these are in turn handled by competent bipartite rankers.

PR-DDAG reduces the influence of non-competent models. Let us explain it with a simple example. Imagine a problem with 3 classes, so the model $\mathcal{M}_{1,3}$ will be at the root, and the models $\mathcal{M}_{1,2}$ and $\mathcal{M}_{2,3}$ at the second level. If we evaluate an instance of class 2, the prediction of $\mathcal{M}_{1,3}$ will distribute its input probability (1, since it is the root of the DDAG) between its child nodes. Applying recursively Equation (5), the posterior probability for class 2 will be:

$$P_{2,2}(\boldsymbol{x}) = (1 - \mathcal{M}_{1,3}(\boldsymbol{x})) \cdot \mathcal{M}_{1,2}(\boldsymbol{x}) + \mathcal{M}_{1,3}(\boldsymbol{x}) \cdot (1 - \mathcal{M}_{2,3}(\boldsymbol{x})).$$

Notice that the sum of the input probabilities of models $\mathcal{M}_{1,2}$ and $\mathcal{M}_{2,3}$ is 1. Thus, the role of the non-competent model $\mathcal{M}_{1,3}$ is to distribute the importance of models $\mathcal{M}_{1,2}$ and $\mathcal{M}_{2,3}$ in order to compute that posterior probability. But, since both models are competent to that task, the decision of $\mathcal{M}_{1,3}$ is not so important. In fact, due to the design of the DDAG, it can be proved that, for any class, the sum of the input probabilities of all its competent models coming from non-competent models is always 1. This means that the role of the non-competent models placed above them on the DDAG is to distribute those input probabilities. At the end, posterior probabilities will depend heavily on competent models.

On the other hand, LPC is the most redundant method. In fact, it is the redundancy obtained from applying such number of models what seems to

Table 2. Properties of the data sets used in the experiments. All of them were taken from the WEKA repository (`http://www.cs.waikato.ac.nz/~ml/weka/index_datasets.html`)

Data set	#examples	#numeric feat.	#nominal feat.	#classes
asbestos	83	1	2	3
balance-scale	625	4	0	3
cmc	1473	2	7	3
pasture	36	21	1	3
post-operative	90	0	8	3
squash (unst.)	52	20	3	3
car	1728	0	6	4
grub-damage	155	8	6	4
nursery	1000	0	9	4
swd	1000	100	0	4
bondrate	57	4	7	5
eucalyptus	736	14	5	5
lev	1000	4	4	5
era	1000	4	4	9
esl	488	4	4	9

thwart the misclassification errors produced by the non-competence problem. PR-DDAG also provides some kind of redundancy, due to the evaluations over all competent models, but in a lower degree compared to LPC.

5 Experiments

In this section we report the results of the experiments performed to evaluate the approaches proposed in this paper to tackle multipartite ranking. This set of experiments was designed to address three main goals. Firstly, CR-DDAG and PR-DDAG were compared with the main state-of-the-art binary decomposition approaches for mutipartite ranking, FH and LPC (see Section 3). Secondly, we studied the influence of the learning algorithm used to build each binary classifier on the compared multipartite ranking methods. Finally, since our aim is to predict a ranking, we included a base learner able to optimize the AUC measure in a bipartite ranking task.

Before discussing the experimental results, the next subsection describes the settings used in the experiments: learning algorithms, data sets, procedures to set parameters, and the measure to estimate the scores.

5.1 Experimental settings

Due to the lack of ordinal benchmark data sets, several previous works have used discretized regression data sets. Despite this can be reasonable, here we wanted to study the performance of the different approaches only on truly ordinal data sets. Thus, the experiments were performed over several ordinal data sets whose main properties are shown in Table 2. This group of data sets were previously used in [16].

We compared the five multipartite ranking algorithms described through the paper, namely, FH (Eq. 1), the two different aggregation methods for LPC approach: the unweighted variant (Eq. 2), called LPCU in the following, and the weighted version LPCW (Eq. 3), and finally the two proposed DDAG-based methods, CR-DDAG (Eq. 4) and PR-DDAG (Eq. 6).

All of them were implemented with three different base learners. First, we employed a binary SVM (*libsvm* implementation [31]) with probabilistic outputs, the second one was the *logistic regression* of [21], and finally, the implementantion of SVM$_{perf}$ presented in [18], setting the target maximization function to be the AUC. In this last case, since its output is not a probability, the algorithm reported in [25] was used to map it into a probability [22]. SVM$_{perf}$ and *libsvm* were used with the linear kernel. In all cases, the regularization parameter C was established for each binary problem performing a grid search over the interval $C \in [10^{-2}, \ldots, 10^{2}]$ optimizing the accuracy in the cases of *libsvm* and *logistic regression* and the AUC in the case of SVM$_{perf}$. Both, accuracy and AUC, were estimated by means of a 2-fold cross validation repeated 5 times.

The ranking performance of the methods was measured in terms of the C-index, estimated using a 5-fold cross validation. C-index is a metric of concordance [12] commonly used in statistics and equivalent to the pairwise ranking error [14]. It has been recently used in [11] for multipartite ranking as an estimation of the probability that a randomly chosen pair of instances from different classes is ranked correctly. If $\mathcal{M}$ is a model, $\mathcal{S}$ the whole training set, p the number of different classes and $\mathcal{S}_k$ with $k = 1, ..., p$ the instances of $\mathcal{S}$ of the class k, then the C-index is defined by

$$C(\mathcal{M}, \mathcal{S}) = \frac{1}{\sum_{i<j} |\mathcal{S}_i||\mathcal{S}_j|} \sum_{1 \leq i < j \leq p} |\mathcal{S}_i||\mathcal{S}_j| AUC(\mathcal{M}, \mathcal{S}_i \cup \mathcal{S}_j). \qquad (7)$$

This metric considers that each class contribution is proportional to its size. An analogous metric that grants the same importance for all classes is the Jonckheere-Terpstra statistic [15], proposed in [13] as another multiclass extension of the AUC. However, conclusions reported in [11] show little differences between them.

Finally, according to the recommendations exposed in [5], a two-step statistical test procedure is carried out. The first step consists of a Friedman test of the null hypothesis that all rankers have equal performance. Then, in case that this hypothesis is rejected, a Nemenyi test to compare learners in a pairwise way is conducted. The average ranks over all data sets are computed and shown at the last row of every table. The ranks of each data sets are indicated in brackets close to the corresponding C-index. In case of ties, average ranks are shown. Since we are comparing 5 algorithms over 15 data sets, the critical rank differences are 1.58 and 1.42 for significance levels of 5% and 10%, respectively.

Table 3. C-index for all approaches using *libsvm* as the base learner

	FH		LPCU		LPCW		CR-DDAG		PR-DDAG	
asbestos	84.83	(1.00)	75.79	(4.00)	76.48	(2.00)	72.78	(5.00)	76.41	(3.00)
balance-scale	97.91	(4.00)	97.95	(3.00)	97.90	(5.00)	97.99	(2.00)	98.03	(1.00)
cmc	68.59	(1.00)	66.41	(5.00)	66.88	(4.00)	67.24	(3.00)	67.68	(2.00)
pasture	90.40	(4.00)	92.13	(1.50)	92.13	(1.50)	88.67	(5.00)	90.53	(3.00)
post-operative	53.78	(4.00)	50.95	(5.00)	54.48	(2.00)	54.23	(3.00)	57.43	(1.00)
squash (unst.)	89.14	(3.00)	89.59	(1.00)	88.89	(4.50)	88.89	(4.50)	89.41	(2.00)
car	98.59	(2.00)	88.37	(5.00)	98.44	(3.00)	98.92	(1.00)	98.30	(4.00)
grub-damage	73.21	(1.00)	70.67	(2.00)	69.68	(3.00)	68.39	(5.00)	69.13	(4.00)
nursery	98.13	(4.00)	97.03	(5.00)	98.33	(2.00)	98.52	(1.00)	98.30	(3.00)
swd	81.03	(3.00)	81.19	(2.00)	81.01	(4.00)	80.64	(5.00)	81.44	(1.00)
bondrate	66.66	(1.00)	52.76	(5.00)	56.57	(4.00)	61.73	(3.00)	61.98	(2.00)
eucalyptus	93.72	(1.00)	89.47	(5.00)	90.37	(4.00)	93.44	(3.00)	93.63	(2.00)
lev	86.36	(3.00)	86.32	(4.00)	86.46	(1.00)	86.03	(5.00)	86.38	(2.00)
era	73.91	(1.00)	73.81	(3.00)	73.88	(2.00)	72.90	(5.00)	73.66	(4.00)
esl	95.58	(4.00)	95.62	(3.00)	96.16	(1.00)	96.13	(2.00)	95.48	(5.00)
Avg. rank		(2.47)		(3.57)		(2.87)		(3.50)		(2.60)

5.2 Experimental results

Table 3, Table 4 and Table 5 show the ranking performance in terms of the C-index for all approaches when *libsvm*, *logistic regression* and SVM$_{perf}$ were respectively adopted as base learners. Analyzing the obtained results using *libsvm* to learn each binary model (see Table 3), we observe that the best method is FH, following by PR-DDAG, LPCW, CR-DDAG and LPCU. However, none of the methods is significantly better using a Nemenyi test. Between our approaches, PR-DDAG wins in 11 out of 15 data sets. The same happens between LPCW and LPCU, the former outperforms the latter 10 times and it is only worse in 4 data sets.

These first results were quite surprising because they contradict in some way the experimental results reported in [11]. In that work, FH was significantly better than LPCW and LPCU. The reasons that can explain these quite opposite conclusions can be: i) we used only truly ordinal data sets, and ii) the base learner was different (in [11] *logistic regression* was used). Motivated for this second reason, we also employed *logistic regression* as the learning algorithm (see Table 4). In this case, the obtained results were similar to those presented in [11]. Now, our approach PR-DDAG obtains better performance, followed by FH. According to a Nemenyi test, PR-DDAG is only significantly better than LPCU (for a significance level of 5%), the worst method in this experiment. The difference between PR-DDAG and LPCW (1.4) is very close to the critical difference (1.42), but it is not significant (see Figure 2). In this case, CR-DDAG obtains better results than both approaches based on LPC. It seems that *logistic regression* is not a good choice as base learner for LPC, since both versions offer the worst performance. However, it keeps the fact that LPCW provides better

Table 4. C-index for all approaches using *logistic regression* as the base learner

	FH		LPCU		LPCW		CR-DDAG		PR-DDAG	
asbestos	82.96	(5.00)	83.50	(2.00)	83.10	(4.00)	83.51	(1.00)	83.46	(3.00)
balance-scale	97.66	(2.00)	97.63	(5.00)	97.65	(3.50)	97.90	(1.00)	97.65	(3.50)
cmc	67.99	(3.00)	67.51	(5.00)	67.98	(4.00)	68.54	(2.00)	68.74	(1.00)
pasture	85.73	(5.00)	86.53	(3.50)	86.53	(3.50)	87.47	(1.00)	86.80	(2.00)
post-operative	45.47	(3.00)	45.52	(2.00)	43.74	(4.00)	42.87	(5.00)	45.66	(1.00)
squash (unst.)	91.26	(2.00)	90.81	(3.00)	90.30	(5.00)	90.64	(4.00)	91.67	(1.00)
car	99.04	(3.00)	86.57	(5.00)	98.94	(4.00)	99.12	(1.00)	99.07	(2.00)
grub-damage	74.16	(1.00)	71.67	(3.00)	71.66	(4.00)	71.04	(5.00)	73.59	(2.00)
nursery	98.58	(1.00)	89.47	(4.00)	88.12	(5.00)	98.57	(2.00)	97.15	(3.00)
swd	81.10	(4.00)	81.14	(3.00)	81.17	(2.00)	81.07	(5.00)	81.49	(1.00)
bondrate	73.07	(3.00)	72.01	(4.00)	71.33	(5.00)	76.85	(1.00)	73.95	(2.00)
eucalyptus	93.99	(2.00)	88.35	(5.00)	89.84	(4.00)	93.76	(3.00)	94.06	(1.00)
lev	86.44	(2.50)	86.41	(4.00)	86.53	(1.00)	86.32	(5.00)	86.44	(2.50)
era	73.90	(2.00)	73.84	(4.00)	73.96	(1.00)	72.83	(5.00)	73.89	(3.00)
esl	96.18	(1.00)	95.39	(5.00)	96.17	(2.00)	95.91	(4.00)	96.08	(3.00)
Avg. rank		(2.63)		(3.83)		(3.47)		(3.00)		(2.07)

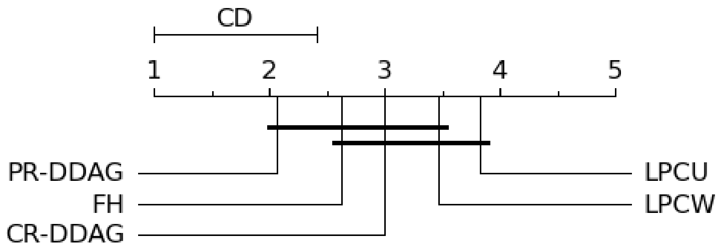

Fig. 2. Friedman-Nemenyi Test ($p < 0.1$) for all methods using *logistic regression*

results than LPCU, although the differences are less remarkable than in case of *libsvm* adopted as base learner.

In the last experiment the goal was to check if these multipartite ranking algorithms can benefit of using a ranking base learner, like SVM$_{perf}$ optimizing the AUC measure (see Table 5). In this case the differences between all the approaches are smaller, only 0.74 between the best (FH) and the worst (LPCU). Now, LPCW and PR-DDAG are almost tied, and the difference between PR-DDAG and CR-DDAG is the smallest of the three experiments. Again, LPCU attains worse results than the other methods. It seems that neither CR-DDAG nor PR-DDAG take advantage of optimizing the AUC. However, it is a good choice for LPCW, since it obtains better position in this case.

Summarizing all these results, we can draw some conclusions. In general, none of the methods is significantly better than the others. In fact, only in one

Table 5. C-index for all approaches using SVM$_{perf}$ as the base learner

	FH		LPCU		LPCW		CR-DDAG		PR-DDAG	
asbestos	82.34	(4.00)	82.62	(3.00)	81.97	(5.00)	84.37	(1.00)	83.69	(2.00)
balance-scale	98.06	(1.50)	97.91	(4.50)	97.91	(4.50)	98.06	(1.50)	97.92	(3.00)
cmc	68.15	(2.00)	67.32	(4.00)	67.72	(3.00)	67.10	(5.00)	68.44	(1.00)
pasture	90.40	(4.00)	93.20	(1.50)	93.20	(1.50)	83.07	(5.00)	90.53	(3.00)
post-operative	53.75	(5.00)	55.31	(4.00)	62.81	(1.00)	59.16	(2.00)	56.38	(3.00)
squash (unst.)	87.73	(1.00)	85.58	(4.50)	87.01	(3.00)	87.21	(2.00)	85.58	(4.50)
car	98.86	(3.00)	89.71	(5.00)	98.88	(1.00)	98.83	(4.00)	98.87	(2.00)
grub-damage	72.23	(1.00)	70.72	(3.00)	69.84	(4.00)	64.67	(5.00)	72.03	(2.00)
nursery	98.56	(2.00)	96.90	(5.00)	98.31	(4.00)	98.55	(3.00)	98.60	(1.00)
swd	80.57	(3.00)	81.08	(1.00)	81.03	(2.00)	80.56	(4.00)	80.51	(5.00)
bondrate	69.37	(2.00)	59.95	(5.00)	69.89	(1.00)	68.54	(3.00)	65.77	(4.00)
eucalyptus	93.19	(1.00)	91.11	(4.00)	91.78	(2.00)	88.46	(5.00)	91.46	(3.00)
lev	86.32	(4.00)	86.56	(1.00)	86.27	(5.00)	86.47	(2.00)	86.46	(3.00)
era	73.64	(4.00)	73.78	(2.00)	73.99	(1.00)	72.84	(5.00)	73.73	(3.00)
esl	96.03	(2.00)	95.74	(3.00)	93.90	(5.00)	96.07	(1.00)	95.34	(4.00)
Avg. rank		(2.63)		(3.37)		(2.87)		(3.23)		(2.90)

case, an algorithm (PR-DDAG) is significantly better than another (LPCU). However, it seems that FH, PR-DDAG and LPCW perform slightly better than the rest. Both PR-DDAG and FH are quite stable, no matter what base learner is used. On the other hand, the choice of the base learner is quite important for LPC methods. Particularly, in the case of using a *logistic regression*, applying LPC approaches is not the best choice.

Focusing on our proposals, PR-DDAG obtains quite similar results than FH and besides it is computationally more efficient. Despite its appealing idea, CR-DDAG performs worse, but not significantly, than PR-DDAG. One reason for this behavior may be the lack of redundancy CR-DDAG suffers from. Indeed, the performance is affected by the classification stage before the consecutive ranking combination takes place.

Finally, using a binary base learner that optimizes the AUC does not improve the results of decomposition multipartite methods. Following [5], we used the Wilcoxon signed ranks test to compare the performance of the same method using different base learners. Only twice significantly differences were found. For LPCU, SVM$_{perf}$ is better than *libsvm* at level $p < 0.05$, and for LPCW, *logistic regression* is better than SVM$_{perf}$ at level $p < 0.10$.

6 Conclusions

This paper proposes two multipartite ranking approaches that exploits the order information the classes exhibits. Both have as the point of the departure the structure of a Decision Directed Acyclic Graph (DDAG) and include a pairwise decomposition technique. One of them, called Consecutive Ranking DDAG

(CR-DDAG) combines a set of consecutive bipartite rankings to obtain a global ranking, but first it performs a classification to decide which bipartite ranker must be applied. The other, called Probabilistic Ranking DDAG (PR-DDAG) includes a probabilistic framework based on propagating probabilities through the graph.

Several experiments over a benchmark ordinal data set were carried out using different base learners, including one algorithm that optimizes the AUC measure. None of the methods outperforms the others, but PR-DDAG exhibits competitive performance, in terms of the C-index measure, with regard to other state-of-the-art algorithms previously proposed in the literature for the same purpose. PR-DDAG also presents interesting theoretical properties, as its computational complexity and its capacity of reducing the non-competence problem.

References

1. Klaus Brinker and Eyke Hüllermeier. Case-based label ranking. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Machine Learning: ECML 2006*, volume 4212, pages 566–573. Springer Berlin Heidelberg, 2006.
2. Jaime S. Cardoso and Joaquim F. Pinto da Costa. Learning to classify ordinal data: The data replication method. *Journal of Machine Learning Research*, 8:1393–1429, 2007.
3. Peng Chen and Shuang Liu. An improved dag-svm for multi-class classification. *International Conference on Natural Computation*, 1:460–462, 2009.
4. Wei Chu and S. Sathiya Keerthi. New approaches to support vector ordinal regression. In Luc De Raedt and Stefan Wrobel, editors, *Proceedings of the ICML'05*, volume 119, pages 145–152. ACM, 2005.
5. Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.
6. Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.
7. Jun Feng, Yang Yang, and Jinsheng Fan. Fuzzy multi-class svm classifier based on optimal directed acyclic graph using in similar handwritten chinese characters recognition. In *Advances in Neural Networks - ISNN 2005, Second International Symposium on Neural Networks, 2005, Proceedings, Part I*, volume 3496, pages 875–880, 2005.
8. Eibe Frank and Mark Hall. A simple approach to ordinal classification. In *EMCL '01: Proceedings of the 12th European Conference on Machine Learning*, pages 145–156, London, UK, 2001. Springer-Verlag.
9. Jerome H. Friedman. Another approach to polychotomous classification. Technical report, Department of Statistics, Stanford University, 1996.
10. Johannes Fürnkranz. Round robin classification. *Journal of Machine Learning Research*, 2:721–747, 2002.
11. Johannes Fürnkranz, Eyke Hüllermeier, and Stijn Vanderlooy. Binary decomposition methods for multipartite ranking. In *ECML PKDD '09: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 359–374. Springer-Verlag, 2009.
12. Mithat Gonen and Glenn Heller. Concordance probability and discriminatory power in proportional hazards regression. *Biometrika*, 92(4):965–970, 2005.

13. David J. Hand and Robert J. Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine Learning*, 45(2):171–186, 2001.
14. Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Large margin rank boundaries for ordinal regression. In Smola, Bartlett, Schoelkopf, and Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press, Cambridge, MA, 2000.
15. J. Higgins. *Introduction to Modern Nonparametric Statistics*. Duxbury Press, 2004.
16. Jens C. Hühn and Eyke Hüllermeier. Is an ordinal class structure useful in classifier learning? *Int. J. of Data Mining Modelling and Management*, 1(1):45–67, 2008.
17. E. Hüllermeier, J. Fürnkranz, W. Cheng, and K. Brinker. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172(16-17):1897–1916, November 2008.
18. Thorsten Joachims. A support vector method for multivariate performance measures. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 377–384, New York, NY, USA, 2005. ACM.
19. Stefan Kramer, Gerhard Widmer, Bernhard Pfahringer, and Michael de Groeve. Prediction of ordinal classes using regression trees. In *ISMIS '00: Proceedings of the 12th International Symposium on Foundations of Intelligent Systems*, pages 426–434, London, UK, 2000. Springer-Verlag.
20. Ping Li, Christopher J. C. Burges, and Qiang Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *NIPS*. MIT Press, 2007.
21. Chih-Jen Lin, Ruby C. Weng, and S. Sathiya Keerthi. Trust region newton method for logistic regression. *Journal of Machine Learning Research*, 9:627–650, 2008.
22. Oscar Luaces, Francisco Taboada, Guillermo M. Albaiceta, Luis A. Domínguez, Pedro Enríquez, and Antonio Bahamonde. Predicting the probability of survival in intensive care unit patients from a small number of variables and training examples. *Artificial Intelligence in Medicine*, 45(1):63 – 76, 2009.
23. Cao D. Nguyen, Tran A. Dung, and Tru H. Cao. Text classification for dag-structured categories. In *Advances in Knowledge Discovery and Data Mining, 9th Pacific-Asia Conference, PAKDD 2005*, volume 3518 of *Lecture Notes in Computer Science*, pages 290–300, 2005.
24. John C. Platt, Nello Cristianini, and John Shawe-taylor. Large margin dags for multiclass classification. In *Advances in Neural Information Processing Systems*, pages 547–553. MIT Press, 2000.
25. John C. Platt and John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.
26. Shyamsundar Rajaram and Shivani Agarwal. Generalization bounds for k-partite ranking. In S. Agarwal, C. Cortes, and R. Herbrich, editors, *Proceedings of the NIPS 2005 Workshop on Learning to Rank*, pages 28–23, 2005.
27. Fumitake Takahashi and Shigeo Abe. Optimizing directed acyclic graph support vector machines. In *IAPR - TC3 International Workshop on Artificial Neural Networks in Pattern Recognition University of Florence, Italy, 2003*, 2003.
28. V. Vapnik and O. Chapelle. Bounds on error expectation for support vector machines. *Neural Computation*, 12(9):2013–2036, 2000.
29. Willem Waegeman, Bernard De Baets, and Luc Boullart. Roc analysis in ordinal regression learning. *Pattern Recognition Letters*, 29(1):1–9, 2008.
30. Mark Allen Weiss. *Data structures and algorithm analysis in C (2nd ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.
31. T. F. Wu, C. J. Lin, and R. C. Weng. Probability estimates for multi-class classification by pairwise coupling. *J. of Machine Learning Research*, 5:975–1005, 2004.