Information Fusion 35 (2017) 38–50

Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

Tackling the supervised label ranking problem by bagging weak learners

Juan A. Aledo[a], José A. Gámez[b,*], David Molina[c]

[a] Departamento de Matemáticas. Universidad de Castilla-La Mancha. Albacete, Spain
[b] Departamento de Sistemas Informáticos. Universidad de Castilla-La Mancha. Albacete, Spain
[c] Departamento de Matemáticas. Universidad de Castilla-La Mancha. Ciudad Real, Spain

ARTICLE INFO

Article history:
Received 18 March 2016
Revised 29 August 2016
Accepted 4 September 2016
Available online 7 September 2016

Keywords:
Supervised classification
Label ranking problem
Bagging
Ensemble
Mallows model
Decision tree

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.

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 prob-

* Corresponding author.
E-mail addresses: juanangel.aledo@uclm.es (J.A. Aledo), jose.gamez@uclm.es (J.A. Gámez), david.molina@uclm.es (D. Molina).

http://dx.doi.org/10.1016/j.inffus.2016.09.002
1566-2535/

lem 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(query,document) $\rightarrow$ 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 [15–17], 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:

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

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 [14,23,30]. In particular, given an incomplete ranking π we use a method that manages the uncertainty about the non-ranked items by taking into