Mining Association Rules for Label Ranking

Cláudio Sá[1], Carlos Soares[1,2], Alípio Mário Jorge[1,3], Paulo Azevedo[5], and Joaquim Costa[4]

[1] LIAAD-INESC Porto L.A., Rua de Ceuta 118-6, 4050-190, Porto, Portugal
[2] Faculdade de Economia, Universidade do Porto
[3] DCC - Faculdade de Ciências, Universidade do Porto
[4] DM - Faculdade de Ciências, Universidade do Porto
[5] CCTC, Departamento de Informática, Universidade do Minho
claudio@liaad,up.pt, csoares@fep.up.pt, amjorge@fc.up.pt, pja@uminho.pt, jpcosta@fc.up.pt

Abstract. Recently, a number of learning algorithms have been adapted for label ranking, including instance-based and tree-based methods. In this paper, we continue this line of work by proposing an adaptation of association rules for label ranking based on the APRIORI algorithm. Given that the original APRIORI algorithm does not aim to obtain predictive models, two changes were needed for this achievement. The adaptation essentially consists of using variations of the support and confidence measures based on ranking similarity functions that are suitable for label ranking. Additionally we propose a simple greedy method to select the parameters of the algorithm. We also adapt the method to make a prediction from the possibly conflicting consequents of the rules that apply to an example. Despite having made our adaptation from a very simple variant of association rules for classification, partial results clearly show that the method is making valid predictions. Additionally, they show that it competes well with state-of-the-art label ranking algorithms.

1 Introduction

Label ranking is an increasingly popular topic in the machine learning literature [10, 6, 22]. Label ranking studies the problem of learning a mapping from instances to rankings over a finite number of predefined labels. It can be considered as a variant of the conventional classification problem, where only a single label is requested instead of a ranking of all labels [6]. In contrast to a classification setting, where the objective is to assign examples to a specific class, in label ranking we are interested in assigning a complete preference order of the labels to every example.

There are two main approaches to the problem of label ranking that we may refer to as decomposition and direct methods. *Decomposition methods* decompose the problem into several simpler problems (e.g., multiple binary problems). *Direct methods* adapt existing algorithms or develop new ones to treat the rankings as target objects without any transformation. An example of the former is the ranking by pairwise comparisons of [10]. Examples of algorithms that were

adapted to deal with rankings as the target objects include decision trees [20, 6], k-Nearest Neighbor [4, 6] and the linear utility transformation [11, 7]. This second group of algorithms can be divided into two approaches. The first one contains methods (e.g., [6]) that are based on statistical distributions of rankings, such as Mallows [14]. The other group of methods are based on measures of similarity or correlation between rankings (e.g., [20, 2]).

In this paper, we propose an adaptation of the association rule mining algorithm APRIORI for label ranking based on similarity measures. Association rules mining is a very important and successful task in data mining. Although the original algorithm was developed for descriptive tasks, several adaptations have been proposed for predictive problems.

The paper is organized as follows: sections 2 and 3 introduce the label ranking problem and the task of association rule mining, respectively; section 4 describes the algorithm proposed here; section 5 presents the experimental setup and discusses the results; finally, section 6 concludes this paper.

2 Label Ranking

The formalization of the label ranking problem given here follows the one provided in [6].[6] Label ranking can be considered as a variant of the conventional classification task. In classification, given an instance x from the instance space $\mathbb{X}$, the goal is to predict the label (or class) λ from a pre-defined set $\mathcal{L} = \{\lambda_1, \ldots, \lambda_k\}$ to which x belongs. In label ranking the goal is to predict the ranking of the labels in $\mathcal{L}$ that are associated with x. We assume that the ranking is a total order over $\mathcal{L}$ defined on the permutation space Ω. Alternatively, we may say that $\lambda_a \succ_x \lambda_b$ indicates that λ_a is preferred to λ_b given the instance x. A total order $\succ_x$ can be seen as a permutation π of the set $\{1, \ldots, k\}$, such that $\pi(a)$ is the position of λ_a in π.[7] Let us also denote $\overline{\pi}$ as the result of inverting the order in π. As in classification, we do not assume the existence of a deterministic $\mathbb{X} \to \Omega$ mapping. Instead, every instance is associated with a *probability distribution* over Ω. This means that, for each $x \in \mathbb{X}$, there exists a probability distribution $P(\cdot|x)$ such that, for every $\pi \in \Omega$, $P(\pi|x)$ is the probability that π is the ranking associated with x. The goal in label ranking is to learn the mapping $\mathbb{X} \to \Omega$. The training data is a set of instances $T = (T_x, T_\pi) = \{x_i, \pi_i\}, i = 1, \ldots, n$, where x_i are the independent variables describing instance i and π_i is the corresponding target ranking.

Given the ranking $\hat{\pi}$ predicted by a label ranking model for an instance x, which is, in fact, associated with the true label ranking π, we need to evaluate the accuracy of the prediction. For that, we need a loss function on Ω. One such function is the number of discordant label pairs,

$$D(\pi, \hat{\pi}) = \#\{(i, j) | \pi(i) > \pi(j) \wedge \hat{\pi}(i) < \hat{\pi}(j)\} \tag{1}$$

[6] An alternative formalization can be found in [21].

[7] In this paper, we will use the two notations interchangeably.

which, if normalized into the interval $[-1, 1]$, will give the Kendall's τ coefficient. The latter is as a correlation measure where $D(\pi, \pi) = 1$ and $D(\pi, \overline{\pi}) = -1$. We obtain a loss function by averaging this function over a set of examples. We will use it as evaluation measure in this paper, as it has been used in recent studies [6]. However other distance measures could have been used, like Spearman's rank [18] correlation.

3 Association Rules Mining

An association rule (AR) is an implication of the form:

$$A \rightarrow C$$

where $A, C \subseteq \mathbb{X}$ and $A \bigcap C = \emptyset$.[8] Association rules are typically chacterized by two measures, support and confidence. The support of rule $A \rightarrow C$ in T is *sup* if *sup*% of the cases in it contain A and C. Additionally, it has a confidence $conf$ in T if $conf$% of cases in T that contain A also contain C.

The original method for induction of AR is the APRIORI algorithm that was proposed in 1994 [1]. We describe this algorithm in the following section. Association rules were originally proposed for descriptive purposes. However, they have been adapted for predictive tasks such as classification (e.g., [15]). Given that label ranking is a predictive task, we describe some useful notation from an adaptation of AR for classification in Section 3.2.

3.1 APRIORI Algorithm

APRIORI identifies all AR that have a support and confidence higher than a given minimal support threshold ($minsup$) and a minimal confidence threshold ($minconf$), respectively. Thus, the model generated by APRIORI is a set of AR of the form $A \rightarrow C$, where $A, C \subseteq \mathbb{X}$, and $sup(A \cup C) \geq minsup$ and $sup(A \cup C)/sup(A) \geq minconf$.

APRIORI mining process can be divided into three steps: *Candidate generation*, *Candidate support count* and *Rule Generation*. The first, *Candidate generation*, takes advantage of a simple restriction called the *downward closure property* which states that an *itemset* is considered frequent if, and only if, all of its *sub-itemsets* are frequent. Hence, it uses one generation of frequent *itemsets* (of length k) to build the next generation (of length $k+1$). This is why the algorithm was named APRIORI: at each step, it works with from candidates obtained *a priori*. The pseudo-code is given in Algorithm 1.

The second, *Candidate support count* scans the data to determine the support of the generated *itemsets*. For simplicity reasons, support is here expressed as

[8] To simplify notation, we assume that an itemset, usually referred to in the AR literature as I, is equivalent to an instance space $\mathbb{X}$ in label ranking. Although this is not exactly true, a simple parallel can be established between the two concepts, and thus, this does not affect the correctness of the paper.

Algorithm 1 APRIORI Candidate generator - APRIORIGen

Require: F_k
  $C_{k+1} = \varnothing$
  for all $f_1, f_2 \in F_k$ **do**
    if $f_1 = (i_1, \ldots, i_{k-1}, i_k)$ and $f_2 = (i_1, \ldots, i_{k-1}, i^*) : \forall i_k < i^*$ **then**
      $c = f_1 \cup f_2 = (i_1, \ldots, i_{k-1}, i_k, i^*)$
      if $c - \{i\} \in F_k, \forall i \in c$ **then**
        $C_{k+1} = C_{k+1} \cup c$
      end if
    end if
  end for
  return C_{k+1}

support count, which is simply the number of times an *itemset* can be found in the dataset. First, it scans the data to count the *1-itemsets*, $i \in T_x$. If one ore more exceeds *minsup* then the item becomes a frequent *1-itemset*.

In summary, the APRIORI algorithm consists of, at each step k, using the *APRIORIGen* step to generate $k - candidates$, and the support count setp to prune the set of itemsets with respect to the user defined *minsup*. The algorithm is summarized in Algorithm 2.

Algorithm 2 APRIORI

Require: *minsup*
  C_k: Candidate itemset of size k
  F_k: Frequent itemset of size k
  T_x: Transactions in the database
  $F_1 = \{$frequent itemsets of lenght 1 in $T_x\}$
  for $k = 1; F_k \neq \varnothing; k++$ **do**
    $F_{k+1} = \varnothing$
    $C_{k+1} =$ APRIORIGen(F_k)
    for all $t \in T$ **do**
      for all $c \in C_{k+1}$ **do**
        if $c \subseteq t$ **then**
          $sup(c) = sup(c) + 1$
        end if
      end for
    end for
    $F_{k+1} = \{c : c \in C_{k+1} \wedge sup(c) \geq minsup\}$
  end for
  return $\cup_k F_k, \forall k$

The final part *Rule Generation* (Algorithm 3), splits every frequent *itemset* into all possible combinations of two non-empty *sub-itemsets* in order to obtain rules from it. This will result in an *antecedent* and a *consequent* of a rule, which

will have its confidence value ($conf$) compared to the user specified threshold $minconf$.

Algorithm 3 Rule generation

Require: $minconf$ and F_k
  for all $f \in F_k; k \geq 2$ **do**
    for all $a \subseteq f; length(a) \leq k-1$ **do**
      $c = f \setminus \{a\}$
      $conf(a \rightarrow c) = \frac{sup(f)}{sup(a)}$
      if $conf(a \rightarrow c) \geq minconf$ **then**
        return $a \rightarrow c$
      end if
    end for
  end for

Despite the usefulness and simplicity of APRIORI, it runs a time consuming candidate generation process and needs space and memory proportional to the number of possible combinations in the database. Additionally it needs multiple scans of the database and typically generates a very large number of rules. Because of this, many new pruning methods were proposed in order to avoid that. Such as the hashing technique [16], dynamic itemset counting [5], parallel and distributed mining [17], relational database systems integrated with mining [19] which reduces the number of database scans.

3.2 Class Association Rules

As stated earlier, AR were originally proposed for descriptive purposes. However, given that label ranking is a prediction task, we need to adapt their definition for that purpose. Our adaptation is based on Classification Association Rules (CAR), proposed as part of the Classification Based on AR (CBA) algorithm [15].

A class association rule (CAR) is an implication of the form:

$$A \rightarrow \lambda$$

where $A \subseteq \mathbb{X}$, and $\lambda \in \mathcal{L}$, which is the class label. The rules can also be represented as $\langle condset, \lambda \rangle$, where $condset = A$.[9] A rule $A \rightarrow \lambda$ holds in T with confidence $conf$ if $conf\%$ of cases in T that contain A are labelled with class λ. And with support sup in T if $sup\%$ of the cases in it contain A and are labelled with class λ.

CBA takes a tabular data set $T = (T_x, T_\lambda) = \{x_i, \lambda_i\}$, where x_i is a set of items and λ_i the corresponding class, and look for all $ruleitems$ of the form $\langle condset, \lambda \rangle$. The algorithm aims to choose a set of high accuracy rules R to

[9] We will use the two notations interchangeably.

match T_x. An instance $x_i \in T_x$ is considered matched by R if at least one rule $(A \rightarrow \lambda) \in R$, with $A \subseteq x_i$, and $\lambda \in \mathcal{L}$. If the rules can't classify all examples, a default class is given to them (e.g., the majority class in the training data).

4 Association Rules for Label Ranking

We define a *Label Ranking Association Rule* (LRAR) as a straightforward adaptation of class association rules (CAR):

$$A \rightarrow \pi$$

where $A \subseteq \mathbb{X}$, and $\pi \in \Omega$. The only difference is that the label $\lambda \in \mathcal{L}$ is replaced by the ranking of the labels, $\pi \in \Omega$. Similar to what the predicton made in CBA, when an exemple matches the rule $A \rightarrow \pi$, the predicted ranking is π.

4.1 Direct Implementation

A straightforward application of the CBA algorithm to the label ranking method can be made by considering each unique permutation $\pi \in \Omega$ as a class. More generally, the number of classes will be the number of distinct $\pi \in \Omega$. Under this assumption we are able to directly use the CBA, or, in fact, any other classification algorithm, on label ranking data sets.

An example of how the data can be displayed for this problem is given in Table 1.

Table 1. An example of a label ranking dataset to be processed by a AR-based classification algorithm. TID represents the identifier of the example (the transaction in AR terminology)

TID	A1	A2	A3	λ_i
1	L	XL	S	(2,3,1)
2	XXL	XS	S	(2,1,3)
3	L	XL	XS	(1,3,2)

The support count of *condset* ($sup(condset)$)is the number of cases in T_x that contain the *condset*, and for *ruleitem* ($sup(\langle condset, \pi >)$) is the number of cases in T that contain the *condset* and have the ranking π associated. Thus, the *confidence* will be:

$$conf(\langle condset \rightarrow \pi \rangle) = \frac{sup(\langle condset, \pi \rangle)}{sup(condset)} \tag{2}$$

A rule $A \rightarrow \lambda$ holds in T with confidence $conf$ if $conf\%$ of cases in T that contain A are labelled with class λ. And with support sup in T if $sup\%$ of the cases in it contain A and are labelled with class λ.

Drawbacks This approach has two important problems. First, the number of classes can be extremely large, up to a maximum of $k!$, where k is the length of the set of labels, $\mathcal{L}$. For instance, if the number of labels is 5, the number of permutations is $5! = 120$. This means that the amount of data required to learn a reasonable mapping $\mathbb{X} \rightarrow \Omega$ is too big.

The second disadvantage is that this approach does not take into account the differences in nature between label rankings and classes. In classification, two examples either have the same class or not. In this regard, label ranking is more similar to regression than to classification. Given an example with a target value of 5, another example with a target value of 5.1 is more similar to it than one with a target value of 4 (at least, in terms of the target values).

This property can be used in the induction of prediction models. Again, let us consider the case of regression. A large number of observations with a given target value, say 5.3, increases the probability of observing similar values, say 5.4 or 5.2, but not so much for very different values, say -3.1 or 100.2. A similar reasoning can be made in label ranking. Let us consider the case of a data set in which ranking $\pi_a = \{A, B, C, D, E\}$ occurs in 1% of the examples. Treating rankings as classes would mean that $P(\pi_a) = 0.01$. Let us further consider that the rankings $\pi_b = \{A, B, C, E, D\}, \pi_c = \{B, A, C, D, E\}$ and $\pi_d = \{A, C, B, D, E\}$ occur in 50% of the examples. Taking into account the stochastic nature of these rankings [6], $P(\pi_a) = 0.01$ seems to underestimate the probability of observing π_a. In other words it is expected that the observation of π_b, π_c and π_d increases the probability of observing π_a and vice-versa, because they are similar to each other.

This affects even rankings which are not observed in the available data. For example, even though $\pi_e = \{A, B, D, C, E\}$ is not present in the data set it would not be entirely unexpected to see it in future data.

4.2 Support and Confidence for Label Ranking

To take this characteristic into account, we can argue that the support of a ranking π increases with the observation of similar rankings and that the variation is proportional to the similarity. Given a measure of similarity between rankings $s(\pi_a, \pi_b)$, we can adapt the concept of support of the *ruleitem* $\langle condset, \pi \rangle$ as follows:

$$sup_{lr}(\langle condset, \pi \rangle) = \frac{\sum_{i:condset \subseteq x_i} s(\pi_i, \pi)}{n} \tag{3}$$

Essentially, what we are doing is assigning a weight to each target ranking in the training, π_i, data that represents its contribution to the probability that π may be observed. Some *itemsets* give full contribution to the support count (i.e., 1), while others give partial or even a null contribution.

Any function that measures the similarity between two rankings or permutations can be used, such as Kendall's τ [13] or Spearman's ρ [18]. The function

used here is of the form:

$$s(\pi_a, \pi_b) = \begin{cases} s'(\pi_a, \pi_b) & \text{if } s'(\pi_a, \pi_b) \geq \theta \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where s' is a similarity function. This general form assumes that below a given threshold, θ, is not useful to discriminate between different similarity values, as they are so different from π_a. This means that, the support *sup* of the $ruleitem = \langle condset_a, \pi_a \rangle$ will have contributions from all the *ruleitems* of the form $\langle condset_a, \pi_b \rangle$, for all π_b where $s'(\pi_a, \pi_b) > \theta$). Again, many functions can be used as s'.

The confidence of a rule $condset \rightarrow \pi$ is obtained simply by replacing the measure of support with the new one.

$$conf_{lr}(\langle condset, \pi \rangle) = \frac{sup_{lr}(\langle condset, \pi \rangle)}{sup(condset)} \tag{5}$$

Given that the loss function that we aim to minimize is known beforehand, it makes sense to use it to measure the similarity between rankings. Therefore, we use Kendall's τ. In this case, we think that $\theta = 0$ would be a reasonable value, givent that it separates the negative from the positive contributions. Table 2 shows an example of a label ranking dataset represented following this approach.

Table 2. An example of a label ranking dataset to be processed by the APRIORI-LR algorithm.

				π_1	π_2	π_3
TID	A1	A2	A3	$(1,3,2)$	$(2,1,3)$	$(2,3,1)$
1	L	XL	S	0.33	0.00	1.00
2	XXL	XS	S	0.00	1.00	0.00
3	L	XL	XS	1.00	0.00	0.33

To present a more clear interpretation, the example given in table 1, the *condset* (L, XL, S) (TID=1) contributes to the the support count of the itemset (L, XL, S, π_1) with 1. The same example, in the table 2 will also give a small contribution of 0.33 to the support count of the itemset (L, XL, S, π_3), given their similarity. On the other hand, in both cases, no contribution is given to the count of the itemset's (L, XL, S, π_2) support, which are clearly different.

4.3 APRIORI-LR Algorithm

Using the definitions of support and confidence proposed, adaptation of APRIORI for label ranking is simple. Given a training set $T = (T_x, T_\pi) = \{x_i, \pi_i\}, i = 1, \ldots, n$, frequent *itemsets* are generated with Algorithm 4 on T. These consist of both regular itemsets and itemsets with one complete ranking included. Then,

the adapted version of the Rule generator, the Algorithm 5, will be used to make the rules based on this last group. The first group will be ignored.

For the generation of frequent itemsets it was used the CAREN [3] software.

Algorithm 4 APRIORI-LR - APRIORI for Label Ranking pseudo-code

Require: *minsup*
  C_k: Candidate itemset of size k
  F_k: Frequent itemset of size k
  $T = (T_x, \pi_i)$: Transactions in the database
  $F_1 = \{$frequent itemsets of lenght 1 in $T_x\}$
  for $k = 1; F_k \neq \varnothing; k++$ **do**
    $F_{k+1} = \varnothing$
    $C_{k+1} =$ APRIORIGen(F_k)
    for all $t \in T$ where $t = \{t_x, \pi_x\}$ **do**
      for all $c \in C_{k+1}$ **do**
        if $c \subseteq t_x$ **then**
          $sup(c) = sup(c) + 1$
        else if $c \subseteq t$ **then**
          $sup(c) = sup(c) + sup_{lr}(c)$
        end if
      end for
    end for
    $F_{k+1} = \{c : c \in C_{k+1} \wedge sup(c) \geq minsup\}$
  end for
  return $\cup_k F_k, \forall k$

Algorithm 5 Rule generation for APRIORI-LR

Require: $minconf$ and F_k
  for all $f \in F_k; k \geq 2$ **do**
    if $anyi \subseteq f : i \in \Omega$ **then**
      $f_x = f \setminus \{\pi\}$
      $conf(f_x \rightarrow \pi) = \frac{sup(f_x \rightarrow \pi)}{sup(f_x)}$
      if $conf(f \rightarrow \pi) \geq minconf$ **then**
        return $f \rightarrow \pi$
      end if
    end if
  end for

Let $\mathcal{R}_{lr}$ be the set of all the generated *label ranking association rules*. The algorithm aims to create a set of high accuracy rules $r_{lr} \in \mathcal{R}_{lr}$ to cover T_x. The classifier has the following format:

$$< r_{lr_1}, r_{lr_2}, \ldots, r_{lr_n} >$$

However, if these are insufficient to rank the given examples, a $default\_ranking$ will be used. The default ranking can be the average ranking [4].

This approach has two problems. The first is that it can only predict rankings which were present in the training set. The second problem is that it solves conflicts between rankings without taking into account the "continuous" nature of rankings, which was illustrated earlier. The problem of generating a single permutation from a set of conflicting rankings has been studied in the context of *consensus rankings*.

It has been shown in [12] that a ranking obtained by ordering the average ranks of the labels across all rankings minimizes the euclidean distance to all those rankings. In other words, it maximizes the similarity according to Spearman's ρ [18]. Given m rankings π_i $(i = 1, \ldots, m)$ we aggregate them by computing for each item j $(j = 1, \ldots, k)$

$$\overline{r_j} = \frac{\sum_{i=1}^{m} \pi_{i,j}}{m} \tag{6}$$

The predicted ranking $\hat{\pi}$ is obtained by ranking the itens according to the value of $\overline{r_j}$.

We can take advantage of this in the ranker builder in the following way: the final predicted label ranking is the consensus of all the label rankings in the consequent of the rules r_{lr} triggered by the test example.

4.4 Matching Maximization

Due to the intrinsic nature of each different dataset, or even of the pre-processing methods used to prepare the data (e.g., the discretization method), the maximum *minsup/minconf* needed to obtain a rule set $\mathcal{R}_{lr}$ that matches all or at least most of the examples, may vary significantly.

Before the experiment tests we use a greedy method to define the minimum confidence. As stated earlier, a rule set $\mathcal{R}_{lr}$ matches an example if at least one rule $(A \rightarrow \lambda) \in \mathcal{R}_{lr}$, with $A \subseteq x_i$. Then, our goal is to obtain a rule set $\mathcal{R}_{lr}$ that maximizes the number of examples that are matched. Additionally, we want that $\mathcal{R}_{lr}$ contains the best rules. In other words, we want the rules with high confidence values.

The matching maximization method (Algorithm 6) will determine the $minconf$ that obtains the rule set according to those criteria. Then the 10-fold cross validation will be started taking into account this value. The 5% step is somewhat arbitrary. From the efficiency point of view, a suitable $minconf$ must be found as soon as possible. On the other hand, this very same value should be as high as possible. As a result, 5% seems a reasonable step.

The ideal value for the *minsup*, is the nearest to 1However, in some datasets, namely those with a great number of attributes, the frequent itemset generation can be a heavy time consuming task. In this case, the results wont have the *minsup* set to 1%. In this work, one such example is the *authorship*, wich has 70 attributes.

Algorithm 6 Matching Maximization Algorithm

$M = 0$
$minconf = 100\%$
$minsup = 1$
while $M < 100\%$ **do**
  $minconf = minconf - 5\%$
  Run the algorithms 4 and 5 with $minsup$ and $minconf$ and determine the percentage of examples matched, M
end while
return minsup, minconf

This procedure has the important advantage that it does not take into account the accuracy of the rule sets generated.

5 Experimental Results

Here, we start by describing the datasets used, then we present the experimental setup and, finally, we present and discuss results.

5.1 Datasets

The data sets in this work were taken from KEBI Data Repository in the Philipps University of Marburg, which were also used in the experimental work presented in [6]. Some information is provided in Table 3.

Two types of methods were used to generate these label data sets: (type A) the target ranking is a permutation of the classes of the original target attribute, derived from the probabilities generated by a naive Bayes classifier; (type B) the target ranking is derived for each example from the order of the values of a set of numerical variables, which are no longer used as independent variables. Despite it may seem a little arbitrary, the original attributes are correlated so the remaining independent variables will also contain information about these artificial created rankings.

Continuous variables were discretized by two distinct methods: (1) *recursive minimum entropy partitioning* criterion ([9]) with the *minimum description length* (MDL) as stopping rule, motivated by [8] and (2) *equal width* bins.

The first method, in some cases, was not able to find a partition for some attributes. As we are dealing with AR discovery, these attributes would origin a $sup = 100\%$. This means that they are present in all frequent item sets. Despite the fact that it does not affect the accuracy of the algorithm, it would slow down the process of finding ARs, so they were removed. For the cases were all attributes were removed due to this reason, the results are not applicable (na).

5.2 Experimental Setup

The evaluation measure is Kendall's τ and the performance of the method was estimated using ten-fold cross-validation.

Table 3. Summary of the datasets

Datasets	type	#examples	#labels	#attributes
autorship	A	841	4	70
bodyfat	B	252	7	7
calhousing	B	20640	4	4
cpu-small	B	8192	5	6
elevators	B	16599	9	9
fried	B	40769	5	9
glass	A	214	6	9
housing	B	506	6	6
iris	A	150	3	4
pendigits	A	10992	10	16
segment	A	2310	7	18
stock	B	950	5	5
vehicle	A	846	4	18
vowel	A	528	11	10
wine	A	178	3	13
wisconsin	B	194	16	16

The performance of APRIORI-LR is compared with a baseline method, the default ranking explained earlier. Additionally, we compare the performance of our algorithm with the results obtained with constraint classification (CC), instance-based label ranking (IBLR) and ranking trees (LRT), that were presented in [6]. However, we note that we did not run experiments with these methods and simply compared our results with the published results of the other methods. Thus, they were probably obtained with different partitions of the data and can not be compared directly. However, they provide some indication of the quality of our method, when compared to the state-of-the-art.

5.3 Results

In Tables 4 and 5, the *minsup* and *minconf* presented are the values obtained when the M reached approximately 100% as shown in column M. Some of them were stopped with a value of M slightly less then that because of the computational costs in terms of time consumption.

The tables show that the method obtains results that are clearly better than the ones obtained by the baseline method with both discretization methods. This means that the APRIORI-LR is identifying valid patterns that can predict label rankings.

As stated earlier, we compare APRIORI-LR with state-of-the-art methods, based on results published in [6], just to get a rough idea of the quality of the proposed method. These results show that APRIORI-LR is a competitive method. We note that APRIORI-LR is a simple adaptation of APRIORI for label

Table 4. Results obtained with *minimum entropy* discretization

	tau	baseline	minsup(%)	minconf(%)	#rules	M
authorship	.57	.57	50	50	9.0	100%
bodyfat	.07	-.04	1	15	3203.3	97%
calhousing	.29	.05	1	35	207.9	96%
cpu-small	.44	.23	1	35	2643.6	100%
elevators	.64	.29	1	60	1779.5	97%
fried	.77	-.01	1	35	1910.1	97%
glass	.85	.67	1	85	465.1	99%
housing	.76	.06	1	60	2502.9	97%
iris	.96	.09	1	90	114.4	100%
pendigits	na	-	-	-	-	-
segment	.90	.37	1	85	633123.6	100%
stock	.89	.09	1	80	1126.6	99%
vehicle	.75	.18	1	85	2535.5	100%
vowel	.68	.20	1	70	20642.8	99%
wine	.84	.33	15	95	5877.0	100%
wisconsin	.04	-.03	1	0	1227.0	93%

Table 5. Results obtained with *equal width* discretization with 3 bins for each attribute

	tau	baseline	minsup(%)	minconf(%)	#rules	M
authorship	NA	-	-	-	-	-
bodyfat	.16	-.04	1	25	14971.2	99%
calhousing	.14	.05	1	20	829.5	100%
cpu-small	.26	.23	1	30	1445.4	100%
elevators	.62	.29	1	60	16911.1	100%
fried	.72	-.01	3	35	652.7	99%
glass	.79	.68	1	75	11374.5	100%
housing	.51	.06	1	50	3459.8	97%
iris	.88	.09	1	80	67.6	100%
pendigits	.69	.45	10	75	16524.3	90%
segment	.50	.37	35	75	4265.6	49%
stock	.83	.09	2	65	872.1	100%
vehicle	.44	.18	26	75	175.7	46%
vowel	.72	.19	1	70	137115.5	99%
wine	.91	.33	1	95	165263.1	100%
wisconsin	.28	-.03	5	20	404773.2	100%

Table 6. Comparison of APRIORI-LR with state-of-the-art methods. APRIORI-LR results obtained with *equal width* discretization with 3 bins for each attribute

	APRIORI-LR				
	EW	ME	CC	IBLR	LRT
authorship	NA	0.57	0.920	0.936	0.882
bodyfat	0.16	0.07	0.281	0.248	0.117
calhousing	0.14	0.29	0.250	0.351	0.324
cpu-small	0.26	0.44	0.475	0.506	0.447
elevators	0.62	0.64	0.768	0.733	0.760
fried	0.72	0.77	0.999	0.935	0.890
glass	0.79	0.85	0.846	0.865	0.883
housing	0.51	0.76	0.660	0.745	0.797
iris	0.88	0.96	0.836	0.966	0.947
pendigits	0.69	NA	0.903	0.944	0.935
segment	0.5	0.90	0.914	0.959	0.949
stock	0.83	0.89	0.737	0.927	0.895
vehicle	0.44	0.75	0.855	0.862	0.827
vowel	0.72	0.68	0.623	0.900	0.794
wine	0.91	0.84	0.933	0.949	0.882
wisconsin	0.28	0.04	0.629	0.506	0.343

ranking. We expect that the results can be significantly improved, for instance, by implementing more complex pruning methods.

6 Conclusions

In this paper we present a simple adaptation of the APRIORI algorithm for label ranking. This adaptation essentially consists of 1) enforcing rules to have label rankings in their consequent, 2) using variations of the support and confidence measures that are suitable for label ranking and 3) generating the model with parameters selected by a simple greedy algorithm.

These results clearly show that this is a viable label ranking method. It clearly outperforms a simple baseline, which means that, despite its simplicity, it is inducing useful patterns.

Additionally, the results obtained indicate that the choice of the discretization method and the number of bins per attribute, play an important role in the efficiency. The tests indicate that the supervised discretization method, (*minimum entropy*, gives better results than the *equal width* partitioning. This is, however, not the main focus of this work.

This work uncovered several possibilities that could be better studied in order to improve the algorithm's performance. Some of the most important are improvements to the methods for prediction generation and matching maximization. Additionally, it is essential to implement some model pruning method. Improvements can also be achieved by adapting the discretization method, the

choice of the measure of similarity s in conjunction with the parameter θ and the use of different measures for the improvement of APRIORI-LR.

In terms of real world applications, these can be adapted to rank analysts, based on their past performance and also radios, based on user's preferences.

Acknowledgment

This work was partially supported by FCT project Rank! (PTDC/EIA/81178/2006) and Palco AdI project Palco3.0 financed by QREN and Fundo Europeu de Desenvolvimento Regional (FEDER). We thank the anonymous referees for useful comments.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proc. 20th Int. Conf. Very Large Data Bases, VLDB. vol. 1215, p. 487499. Citeseer (1994)
2. Aiguzhinov, A., Soares, C., Serra, A.P.: A similarity-based adaptation of naive bayes for label ranking. In: Discovery Science (2010)
3. Azevedo, P.: CAREN-A java based apriori implementation for classification purposes (2003)
4. Brazdil, P., Soares, C., Costa, J.: Ranking Learning Algorithms: Using IBL and Meta-Learning on Accuracy and Time Results. Machine Learning 50(3), 251–277 (2003)
5. Brin, S., Motwani, R., Ullman, J.D., Tsur, S.: Dynamic itemset counting and implication rules for market basket data. Proceedings of the 1997 ACM SIGMOD international conference on Management of data - SIGMOD '97 pp. 255–264 (1997), http://portal.acm.org/citation.cfm?doid=253260.253325
6. Cheng, W., Hühn, J., Hüllermeier, E.: Decision tree and instance-based learning for label ranking. In: ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning. pp. 161–168. ACM, New York, NY, USA (2009)
7. Dekel, O., Manning, C., Singer, Y.: Log-linear models for label ranking. Advances in Neural Information Processing Systems 16 (2003)
8. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. In: MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-. pp. 194–202. MORGAN KAUFMANN PUBLISHERS, INC. (1995), http://robotics.stanford.edu/users/sahami/papers-dir/disc.pdf
9. Fayyad, Irani: Multi-interval discretization of continuous-valued attributes for classification learning. In: International Conference on Machine Learning. pp. 1022–1027 (1993), http://www.cs.orst.edu/~bulatov/papers/fayyad-discretization.pdf
10. Fürnkranz, J., Hüllermeier, E.: Preference learning. KI 19(1), 60– (2005)
11. Har-Peled, S., Roth, D., Zimak, D.: Constraint classification: a new approach to multiclass classification. In: Proc. of the International Workshop on Algorithmic Learning Theory (ALT). pp. 135–150. Springer-Verlag (2002)
12. Kemeny, J., Snell, J.: Mathematical Models in the Social Sciences. MIT Press (1972)
13. Kendall, M., Gibbons, J.: Rank correlation methods (1970)
14. Lebanon, G., Lafferty, J.D.: Conditional Models on the Ranking Poset. In: NIPS. pp. 415–422 (2002)

15. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. Knowledge Discovery and Data Mining pp. 80–86 (1998), http://www.aaai.org/Library/KDD/1998/kdd98-012.php
16. Park, J.S., Chen, M.S., Yu, P.S.: An effective hash-based algorithm for mining association rules. ACM SIGMOD Record 24(2), 175–186 (May 1995), http://portal.acm.org/citation.cfm?doid=568271.223813
17. Park, J., Chen, M., Yu, P.: Efficient parallel data mining for association rules. of the fourth international conference on (1995), http://portal.acm.org/citation.cfm?id=221270.221320
18. Spearman, C.: The proof and measurement of association between two things. American Journal of Psychology 15, 72–101 (1904)
19. Thomas, S., Sarawagi, S.: Mining generalized association rules and sequential patterns using SQL queries. . Conf. on Knowledge Discovery and Data Mining (1998), http://www.aaai.org/Papers/KDD/1998/KDD98-062.pdf
20. Todorovski, L., Blockeel, H., Džeroski, S.: Ranking with Predictive Clustering Trees. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) Proc. of the 13th European Conf. on Machine Learning. pp. 444–455. No. 2430 in LNAI, Springer-Verlag (2002)
21. Vembu, S., Gärtner, T.: Label ranking algorithms: A survey. Preference Learning. Springer (2009), http://www-kd.iai.uni-bonn.de/pubattachments/399/VembuG09Book.pdf
22. Vembu, S., Gärtner, T.: Label Ranking Algorithms: A Survey. In: Johannes Fürnkranz, E.H. (ed.) Preference Learning. Springer–Verlag (2010)