Empirical Evaluation of Ranking Trees on Some Metalearning Problems

Carla Rebelo
LIAAD-INESC Porto LA
Universidade do Porto, Portugal
crebelo@liaad.up.pt

Carlos Soares
Faculdade de Economia do Porto
LIAAD-INESC Porto LA
Universidade do Porto, Portugal
csoares@fep.up.pt

Joaquim Pinto da Costa
Faculdade de Ciências
Universidade do Porto, Portugal
jpcosta@fc.up.pt

Abstract

The problem of learning rankings is receiving increased attention from several research communities. In this paper we empirically evaluate an adaptation of the algorithm of learning decision trees for rankings. Our experiments are carried out on some metalearning problems, which consist of relating characteristics of learning problems to the relative performance of learning algorithms. We obtain positive results which, somewhat surprisingly, indicate that the method predicts more accurately the top ranks.

Introduction

The prediction of the *class* to which an object (i.e., an *example*) belongs based on a set of measures that describe it (i.e., *attributes*) is a problem investigated in several scientific areas, including pattern recognition, statistics and machine learning (Mitchell 1997). As an example, the objects could be customers, the attributes could be socio-demographic variables (such as age or income) and the class could be the product that he/she will buy, from the portfolio of a company. Many different applications exist in areas such as medicine, retail, banking and finance, e-business and government.

Many algorithms exist to induce from existing data, models for prediction problems, which can be referred to as *learning algorithms*. One of the simplest and most successful learning algorithms are decision trees. Decision trees represent a set of classification rules from the root node to the terminal nodes (leaves), which provide the classification for the corresponding objects or examples. Each node of the tree specifies a test of some attribute used to describe the example, and each branch of the node corresponds to one of the possible values for this attribute. The prediction for an example is generated by first applying the test of the root node of the tree and, based on the value for the corresponding attribute, following one of the branches of the node. This process is repeated for the sub-tree rooted at the node the branch leads to. Generally, a decision tree is a disjunction of conjunctions of restrictions on the value of attributes. Each path from the root to a leaf node is a combination of tests of attributes, and the tree is a disjunction of these conjunctions (Mitchell 1997).

In some cases, the prediction may be more useful if provided in the form of ranking. In the example above, rather than recommending a single product, it may be more useful to provide a ranking of the products in decreasing order of expected customer preference, i.e., the probability that he/she will buy it. Other examples include medical diagnosis, prediction of diseases in agricultural products, prediction of the ranking of financial analysts and metalearning.

In general, a ranking represents a preference relation on a set of items (Fürnkranz and Hüllermeier 2005). In the case of predicting the ranking of financial analysts, analyst A is preferable to analyst B if the A's recommendations have been more accurate than the recommendations of analyst B.

Due to its simplicity and popularity, it is no surprise that decision trees were one of the first algorithms adapted for the task of learning rankings (Todorovski, Blockeel, and Dzeroski 2002). A ranking tree is a decision tree where the leaves do not predict one from a set of possible class values but, instead, predict a ranking of the set of class values. The adaptation of this algorithm to the task of predicting rankings implies several adaptation, including the splitting criterion and generation of the prediction.

In this paper, we empirically evaluate ranking trees on the problem of metalearning (Brazdil, Soares, and Costa 2003). This problem consists of providing recommendations concerning which algorithm(s) will obtain the best results on learning problems.

This paper is structured as follows. We start by presenting the problem of recommending learning algorithms and discuss why it can be better handled as a ranking problem. Next, we describe the problem of learning rankings and discuss three evaluation measures. The following section describes the ranking trees algorithm, identifying the issues involved in its adaptation for this purpose. The results are presented in the following section the paper ends with some conclusions and ideas future work.

Recommendation of Learning Algorithms

Many different learning algorithms are available to data analysts nowadays. For instance, decision trees, neural networks, linear discriminants, support vector machines among others can be used in classification problems. The goal of

	a_1	a_2	a_3	a_4
d_1	90%	61%	82%	55%
d_2	84%	86%	60%	79%

Table 1: Accuracy of four learning algorithms on two classification problems.

data analysts is to use the one that will obtain the best performance on the problem at hand. Given that the performance of learning algorithms varies for different datasets, data analysts must select carefully which algorithm to use for each problem, in order to obtain satisfactory results.

Therefore, we can say that a performance measure establishes a preference relation between learning algorithms for each problem. For instance, Table 1 illustrates the preference relations between four classification algorithms (a_i) on two datasets (d_j) defined by estimates of the classification accuracy of those algorithms on those datasets.

Selecting the algorithm by trying out all alternatives is generally not a viable option, as explained in (Todorovski, Blockeel, and Dzeroski 2002):

> In many cases, running an algorithm on a given task can be time consuming, especially when complex tasks are involved. It is therefore desirable to be able to predict the performance of a given algorithm on a given task from description and without actually running the algorithm.

The learning approach to the problem of algorithm recommendation consists of using a learning algorithm to model the relation between the characteristics of learning problems (e.g., application domain, number of examples, proportion of symbolic attributes) and the relative performance of a set of algorithms (Brazdil, Soares, and Costa 2003). We refer to this approach as *metalearning* because we are learning about the performance of learning algorithms.

Metalearning approaches commonly cast the algorithm recommendation problem as a classification task. Therefore, the recommendation provided to the user consists of a single algorithm. However, this is not the most suitable form of recommendation. Although the computational cost of executing all the algorithms is very high, it is often the case that it is possible to run a few of the available algorithms. Therefore, it makes more sense to provide recommendation in the form of a ranking. The user can then execute the algorithms in the suggested order, until no computational resources (or time) are available.

Learning Rankings

The problem of predicting rankings has similarities with the problem of supervised classification. In classification we have a set of examples, caracterized by attributes and each one is assigned to one of a set of classes. Given a new example, described by the values of the attributes, the objective in supervised classification is to predict the class it belongs to. On the other hand, in ranking the goal is to predict the order of the classes as applicable to each example. Thus, the input to a problem of learning rankings is a set of examples as described by a set of attributes and with a known ranking of the classes (the target ranking). The goal is to obtain a model that, given a new example, generates a ranking of all the classes (or items).

In general, a ranking represents a *preference function* over a set of items. Therefore, given a set of n items,

$$X = (X_1, X_2, ..., X_{n-1}, X_n)$$

we define a ranking as a vector,

$$R = (R_1, R_2, ..., R_{n-1}, R_n)$$

where R_i is the rank of item X_i and the item with rank R_i is preferred to item with rank R_j if $R_i < R_j$ (Soares 2004).

A number of methods have been proposed for learning rankings (Fürnkranz and Hüllermeier 2005). Some of these methods are based on existing learning algorithms such as the k-Nearest Neighbors (KNN) (Brazdil, Soares, and Costa 2003) and Decision Trees (Todorovski, Blockeel, and Dzeroski 2002), which is the method tested in this work.

The generalization ability of ranking methods, i.e., their ability to accurately predict the rankings on unseen examples, can be estimated using the same strategies that are used for other learning problems. Here we have used *Leave-one-out Cross Validation*, which consists of iteratively, for each example, computing the accuracy of the prediction made for the selected example of a model obtained on all the remaining examples (Witten and Frank 2000).

We assess the accuracy of the predictions by comparing the ranking predicted by the method for a given example with the corresponding target ranking. In classification problems there is a natural measure of accuracy. The classifier provides a class for each example: if it is correct, it is counted as a success; otherwise, it is counted as a mistake. Thus, the accuracy is the proportion of correct predictions relative to the number of examples (Witten and Frank 2000). We have used three measures of ranking accuracy: Spearman's Rank Correlation Coefficient, the Weighted Rank Correlation coefficient and the Log Ranking Accuracy which are presented below.

Spearman Coefficient (CS)

Spearman's rank correlation coefficient, r_S, has been proposed in the early $20th$ century by Charles Spearman and is given by the expression:

$$r_S = 1 - \frac{6 \sum_{i=1}^{n} (R(X_i) - R(Y_i))^2}{n^3 - n} \quad (1)$$

where X and Y are two sets of n values and $R(X_i)$ represents the rank of element i in the series X. Nothing is assumed about the distribution of values of the variables. The coefficient simply evaluates the monotonicity of two sets of values, i.e., if their variations are related. If they tend to increase or decrease together, the variables are positively correlated. However, if one tends to increase while the other decreases then they are negatively correlated. This is more general than other coefficients, such as Pearson's because it does not assume that the relationship between the two variables is represented by a particular type of function (Neave and Worthington 1992).

The expression above is valid only if there are no ties (the same numerical value in two or more observations), although in the case of a small number of ties it can still be applied (after using average ranks for the tied observations). If the number of ties is very large, then it is better to use the expression of Pearson's correlation coefficient of the two vectors of ranks, as this is an alternative way of finding the Spearman's coefficient in all situations.

$$r_S(X,Y) = \frac{\sum_{i=1}^{n}(R(X_i) - R(\bar{X}_i))(R(Y_i) - R(\bar{Y}_i))}{\sqrt{\sum_{i=1}^{n}(R(X_i) - R(\bar{X}_i))^2 \sum_{i=1}^{n}(R(Y_i) - R(\bar{Y}_i))^2}} \quad (2)$$

It is, however, computationally less efficient that the one above.

Weighted Rank Correlation (WRC)

Given a ranking of learning algorithms, it can be expected that the higher an algorithm is ranked, the higher the probability that it will be executed by the user. Similar scenarios are expected in other ranking applications. Generally, this is true in ranking problems where the prediction is only used as a recommendation. Therefore, the evaluation of ranking algorithms should assign greater importance to higher ranks. However, Spearman's coefficient treats all ranks equally. An alternative coefficient is the Weighted Rank Correlation Coefficient (Pinto da Costa and Soares 2005; da Costa and Roque 2006). Let $d_i^2 = (R(X_i) - R(Y_i))^2$ and

$$W_i^2 = d_i^2((n - R(X_i) + 1) + (n - R(Y_i) + 1)) \quad (3)$$

The first term of this product d_i^2 represents the quadratic error of rank, exactly as in r_S, and represents the distance between $R(X_i)$ and $R(Y_i)$. The second term weighs the error of rank by the importance of the two ranks involved $R(X_i)$ and $R(Y_i)$. Based on these expressions, the Weighted Rank Correlation coefficient is defined as:

$$r_W(X,Y) = 1 - \frac{6\sum_{i=1}^{n} W_i^2}{n^4 + n^3 - n^2 - n} \quad (4)$$

Log Ranking Accuracy (LRA)

Another weighted measure of accuracy, that gives even more importance to higher ranks than r_W is the Log Ranking Accuracy (Soares 2004):

$$r_{log}(X,Y) = 1 - 2 * \frac{6\sum_{i=1}^{n} log_{1+R(X_i)}(1 + R(X_i) - R(Y_i))^2)}{\sum_{i=1}^{n} log_{1+i}(1 + (i - (n - i + 1))^2)} \quad (5)$$

Ranking Trees

One of the advantages of tree-based models is how they can clearly express information about the problem, because their structure is relatively easy to interpret even for people without a background on learning algorithms. It is also possible to obtain information about the importance of the various attributes for the prediction depending on how close to the root they are used. The Top-Down Induction of Decision Trees (TDIDT) algorithm is commonly used for induction of decision trees (Mitchell 1997). It is a recursive partitioning algorithm that iteratively splits data into smaller subsets which are increasingly more homogeneous in terms of the target variable (Figure 1).

TDIDT (D)
- BestSplit ← Test of the attributes that optimizes the *splitting criterion*
- IF *stopping criterion* is TRUE
  - Determine the leaf prediction based on the target values of the examples in *D*
  - Return a leaf node with the corresponding *leaf prediction*
- ELSE
  - LeftSubtree ← TDIDT($D_{\neg BestSplit}$)
  - RightSubtree ← TDIDT($D_{BestSplit}$)
  - Return (BestSplit, LeftSubtree, RightSubtree)

Figure 1: TDIDT algorithm.

It starts by determining the split that optimizes a given splitting criterion. A split is a test on one of the attributes that divides the dataset into two disjoint subsets. For instance, given a numerical attribute A_2, a split could be $A_2 \geq 5$. One of the problems with the most simple version of the TDIDT algorithm is that it only stops when the nodes are pure, i.e., when the value of the target attribute is the same for all examples in the node. This usually leads the algorithm to overfit, i.e., to generate models that fit not only to the patterns in the data but also to the noise. One approach to address this problem is to introduce a stopping criterion in the algorithm that tests whether the best split is significantly improving the quality of the model. If not, the algorithm stops and returns a leaf node. This node is represented by the prediction that will be made for new examples that fall into that node. This prediction is generated by a rule that solves potential conflicts in the set of training examples that are in the node. In classification, the prediction rule is usually the most frequent class among the training examples. If the stopping criterion is not verified, then the algorithm is executed recursively for the subsets of the data obtained based on the best split.

An adaptation of the TDIDT algorithm for the problem of learning rankings has recently been proposed (Todorovski, Blockeel, and Dzeroski 2002), called *Ranking Trees*. This algorithm is based on the Clustering Trees algorithm (Blockeel, Raedt, and Ramon 1998). Adaptation of this algorithm for ranking involves a few issues, including the splitting criterion, the stopping criterion and the prediction rule. Before discussing these issues, we note that a training example in the problem of learning rankings is described using a set of m attributes $(A_1, A_2, \ldots, A_m)$ and is associated with a ranking $(R_1, R_2, \ldots, R_n)$ as defined earlier.

Splitting Criterion

The splitting criterion is a measure that quantifies the quality of a given partition of the data. It is usually applied to all the possible splits of the data that can be made based on individual tests of the attributes.

In *Ranking Trees* the goal is to obtain leaf nodes that contain examples with target rankings as similar between them-

Attribute	Condition		Negated condition	
	values	rank corr.	values	rank corr.
A_1	a	0.3	b, c	-0.2
	b	0.2	a, c	0.1
	c	0.5	a, b	0.2
A_2	< 5	-0.1	≥ 5	0.1

Table 2: Illustration of the splitting criterion

Ranking	R_1	R_2	R_3	R_4
e_1	1	3	2	4
e_2	2	1	4	3
Mean rank	1.5	2	3	3.5
Predicted	1	2	3	4

Table 3: Illustration of the prediction rule, where e_i represents training example i

selves as possible. To assess the similarity between the rankings of a set of training examples, we compute the mean correlation between them, using Spearman's correlation coefficient. The quality of the split is given by the weighted mean correlation of the values obtained for the subsets, where the weight is given by the number of examples in each subset.

The splitting criterion of ranking trees is illustrated both for nominal and numerical attributes in Table 2. The nominal attribute A_1 has three values (a, b and c). Therefore, three splits are possible. For the numerical attribute A_2, a split can be made in between every pair of consecutive values. In this case, the best split is $A_1 = c$, with a mean correlation of 0.5 for the training examples that verify the test and a mean correlation of 0.2 for the remaining, i.e., the training examples for which $A_1 = a$ or $A_1 = b$.

Stopping Criterion

The stopping criterion is used to determine if it is worthwhile to make a split or if there is a danger of overfitting. In the original implementation of *Ranking Trees* the criterion is not described. Here we define that a split should only be made if the similarity between examples in the subsets should increase significantly. Let S_{parent} be the similarity between the examples in the parent node, D, and S_{split} the weighted mean similarity in the subsets obtained with the best split. The stopping criterion is defined as follows:

$$(1 + S_{parent}) \geq \gamma(1 + S_{split}) \quad (6)$$

Note that the significance of the increase in similarity is controlled by the parameter γ

Prediction Rule

The prediction rule is a method to generate a prediction from the (possibly conflicting) target values of the training examples in a leaf node. In *Ranking Trees*, the method that is used to aggregate the rankings that are in the leaves is based on the mean ranks of the items in the training examples that fall into the corresponding leaf. Table 3 illustrates the prediction rule used in this work.

Experimental Results

We empirically tested the *Ranking Trees* algorithm on some ranking problems obtained from metalearning applications. The value of the γ parameter used was 0.995. This choice was based on the number of nodes generated. Figure 2 shows that lower values will generally lead to trees with a single node. On the other hand, for $\gamma > 1$ (i.e., when splits

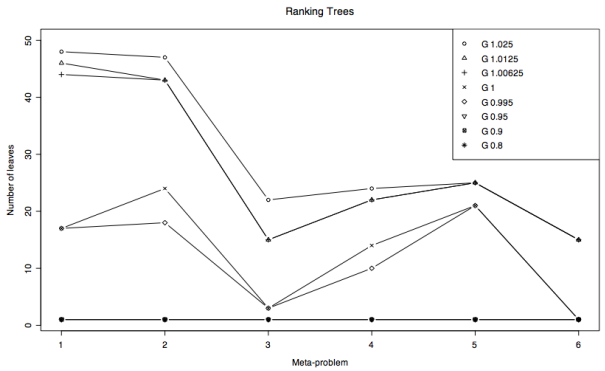

Figure 2: Variation of the number of leaves with the value of γ.

generating nodes that are less homogeneous than the parent may be accepted), the number of leaves is close to the number of examples of the datasets, indicating that there is overfitting. For comparison purposes, the k-Nearest Neighbor (KNN) algorithm, which was previously applied on the same problems was also implemented (Brazdil, Soares, and Costa 2003). Based on the results reported in that work, we used a single neighbor ($k = 1$). Additionally, we compared the results with a simple baseline, the *default ranking*, which is the mean ranking over all training examples (Brazdil, Soares, and Costa 2003), which is essentially the application of the decision rule on all the training rankings. The code for all the examples in this paper has been written in R (www.r-project.org).

The performance of the methods was estimated using leave-one-out because of the small size of the datasets. Both algorithms were evaluated using the three ranking accuracy measures described earlier. However, given that the results obtained with Spearman's correlation coefficient and the Weighted Rank coefficient are similar, we only present the former.

Datasets

We used the following meta-learning problems in our experiments:

- Classification: these data represent the performance of ten algorithms on a set of 57 classification tasks (datasets). The metafeatures describing the datasets are based on (Henery 1994) and (Kalousis 2002). The same authors also provide descriptions of the meta-features in the other two sets, referred to as *Met-* and *Stat-set*.

- Regression: these data represent the performance of nine algorithms on a set of 42 regression tasks (datasets). Also in this case, two different sets of metafeatures were used.
- SVM: these data represent the performance of different variants of the Support Vector Machines algorithm on the same 42 regression datasets as in the previous set (Soares, Brazdil, and Kuba 2004). Two sets of metafeatures were also used in this metalearning problem.

Results and Discussion

Comparing ranking trees with the KNN algorithm in terms of Spearman's coefficient (CS), we observe that the latter generally obtains better results (Figure 3). This may be explained with the smal size of the dataset (a few dozen examples), which makes the induction of a model with good generalization abilities hard. This is also supported by previous results, in which the use of KNN on these problems with larger values of k leads to worse results (Soares 2004). On the other hand, these results are somewhat contradictory with a previous comparison between ranking trees and KNN, in which better accuracy is reported for the former algorithm (Todorovski, Blockeel, and Dzeroski 2002). The difference may be explained by the different experimental setup that is used.

Additionally, the comparison with the default ranking indicates that none of these methods (ranking trees and KNN) are able to predict the ranking of algorithms on new datasets very accurately.

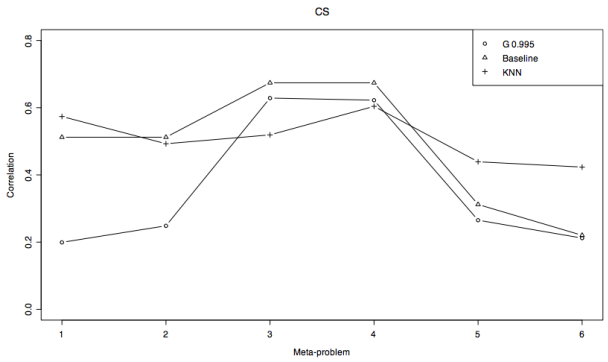

Figure 3: Comparison of ranking accuracy measured with the Spearman's coefficient of Ranking Trees (black) with the KNN algorithm (grey) and the default ranking baseline (white).

A different perspective is given by the LRA measure. According to this measure, the rankings predicted with KNN are clearly better than the ones generated by ranking trees as well as by the default ranking. The fact that better results are generally obtained in terms of the LRA measure than with Spearman's coefficient indicates that the method is better at predicting the top rank than the lower ones. This is a good result as top ranks are more important than lower ones. For instance, in the metalearning problem it may be expected that the user will most probably execute the algorithms that are recommended at the top ranks than the others.

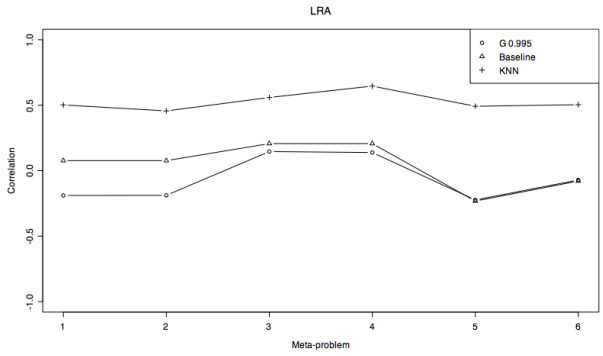

Figure 4: Comparison of ranking accuracy measured with the Log Ranking Accuracy measure of Ranking Trees (black) with the KNN algorithm (grey) and the default ranking baseline (white).

Conclusions

In this paper, we empirically evaluate ranking trees on some metalearning problems. The results indicate that the method does not achieve better results than the KNN algorithm and also a baseline method. However, these results may be explained by the small number of examples in the datasets. It is, thus, necessary to test the methods on larger datasets.

Concerning the algorithm presented here, we plan to evaluate the effect of changing the parameter of the stopping criterion. We also plan to evaluate alternative splitting criteria, prediction rules and stopping criteria. In terms of the splitting criterion, we will test measures of similarity that give more importance to the top ranks. Finally, we plan to test the method on different ranking problems.

Acknowledgements

This work was partially supported by project *Rank!*, funded by Fundação para a Ciência e Tecnologia (PTDC/EIA/81178/2006).

References

Blockeel, H.; Raedt, L. D.; and Ramon, J. 1998. Top-down induction of clustering trees. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, 55–63. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Brazdil, P. B.; Soares, C.; and Costa, J. P. D. 2003. Ranking learning algorithms: Using ibl and meta-learning on accuracy and time results. *Mach. Learn.* 50(3):251–277.

da Costa, J. P., and Roque, L. 2006. Limit distribution for the weighted rank correlation coefficient, r_w. *REVSTAT - Statistical Journal* 4(3).

Fürnkranz, J., and Hüllermeier, E. 2005. Preference learning. *Kunstliche Intelligenz* 19(1):60–61.

Henery, R. 1994. Methods for comparison. In Michie, D.; Spiegelhalter, D.; and Taylor, C., eds., *Machine Learning, Neural and Statistical Classification*. Ellis Horwood. chapter 7, 107–124.

Kalousis, A. 2002. *Algorithm Selection via Meta-Learning*. Ph.D. Dissertation, University of Geneve, Department of Computer Science.

Mitchell, T. M. 1997. *Machine Learning*. McGraw-Hill Higher Education.

Neave, H., and Worthington, P. 1992. *Distribution-Free Tests*. Routledge.

Pinto da Costa, J., and Soares, C. 2005. A weighted rank measure of correlation. *Australian & New Zealand Journal of Statistics* 47(4):515–529.

Soares, C.; Brazdil, P.; and Kuba, P. 2004. A meta-learning method to select the kernel width in support vector regression. *Machine Learning* 54:195–209.

Soares, C. 2004. *Learning Rankings of Learning Algorithms*. Ph.D. Dissertation, Department of Computer Science, Faculty of Sciences, University of Porto. Supervisors: Pavel Brazdil and Joaquim Pinto da Costa.

Todorovski, L.; Blockeel, H.; and Dzeroski, S. 2002. Ranking with predictive clustering trees. In *ECML '02: Proceedings of the 13th European Conference on Machine Learning*, 444–455. London, UK: Springer-Verlag.

Witten, I. H., and Frank, E. 2000. *Data mining: practical machine learning tools and techniques with Java implementations*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.