Machine Learning, 50, 251–277, 2003

Ranking Learning Algorithms: Using IBL and Meta-Learning on Accuracy and Time Results

PAVEL B. BRAZDIL pbrazdil@liacc.up.pt
CARLOS SOARES csoares@liacc.up.pt
LIACC/Faculty of Economics, University of Porto, Portugal

JOAQUIM PINTO DA COSTA jpcosta@liacc.up.pt
LIACC/Faculty of Science, University of Porto, Portugal

Editors: David Aha

Abstract. We present a meta-learning method to support selection of candidate learning algorithms. It uses a k-Nearest Neighbor algorithm to identify the datasets that are most similar to the one at hand. The distance between datasets is assessed using a relatively small set of data characteristics, which was selected to represent properties that affect algorithm performance. The performance of the candidate algorithms on those datasets is used to generate a recommendation to the user in the form of a ranking. The performance is assessed using a multicriteria evaluation measure that takes not only accuracy, but also time into account. As it is not common in Machine Learning to work with rankings, we had to identify and adapt existing statistical techniques to devise an appropriate evaluation methodology. Using that methodology, we show that the meta-learning method presented leads to significantly better rankings than the baseline ranking method. The evaluation methodology is general and can be adapted to other ranking problems. Although here we have concentrated on ranking classification algorithms, the meta-learning framework presented can provide assistance in the selection of combinations of methods or more complex problem solving strategies.

Keywords: algorithm recommendation, meta-learning, data characterization, ranking

1. Introduction

Multistrategy learning is concerned with the combination of different strategies to solve complex data analysis or synthesis tasks. A task may involve a number of steps, each requiring one or more methods or strategies to be tried out. Some may be concerned with pre-processing (e.g., discretization of numeric values), others involve transformation of the given problem into a different one (e.g., division of a problem into interrelated subproblems). These are followed by the stage of model generation (e.g., generation of a classifier for a subset of data).

Clearly, the best combination of strategies depends on the problem at hand and methods that help the user to decide what to do are required (Mitchell, 1997; Brachman & Anand, 1996). It would be useful to have a method that could not only present all different options at each step, but also rank them according to their potential utility. In this paper we present an approach to this problem, focussing on one stage only, model generation. In particular we examine a framework that provides us with a ranking of candidate classification algorithms

to be used. This is a special case of the problem of obtaining a recommendation on the utility of different combinations of solution strategies. We have chosen to deal with the simpler problem because it does provide us with enough challenges and needs to be resolved first.

1.1. Support for automatic classifier recommendation

Ideally, we would like to be able to identify or design the single best algorithm to be used in all situations. However, both experimental results (Michie, Spiegelhalter, & Taylor, 1994) and theoretical work (Wolpert & Macready, 1996) indicate that this is not possible. Therefore, the choice of which algorithm(s) to use depends on the dataset at hand and systems that can provide such recommendations would be very useful (Mitchell, 1997).

We could reduce the problem of algorithm recommendation to the problem of performance comparison by trying all the algorithms on the problem at hand. In practice this is not usually feasible because there are too many algorithms to try out, some of which may be quite slow. The problem is exacerbated when dealing with large amounts of data, as it is common in Knowledge Discovery in Databases.

One approach to algorithm recommendation involves the use of meta-knowledge, that is, knowledge about the performance of algorithms. This knowledge can be either of theoretical or of experimental origin, or a mixture of both. The rules described by Brodley (1993) for instance, captured the knowledge of experts concerning the applicability of certain classification algorithms. More often the meta-knowledge is of experimental origin, obtained by meta-learning on past performance information of the algorithms (i.e., performance of the algorithms on datasets used previously) (Aha, 1992; Brazdil, Gama, & Henery, 1994; Gama & Brazdil, 1995). Its objective is to capture certain relationships between the measured dataset characteristics and the performance of the algorithms. As was demonstrated, meta-knowledge can be used to give useful predictions with a certain degree of success.

1.2. Recommending individual classifiers or ranking?

An important issue concerns the type of recommendation that should be provided. Many previous meta-learning approaches limit themselves to suggesting one algorithm or a small group of algorithms that are expected to perform well on the given problem (Todorovski & Džeroski, 1999; Kalousis & Theoharis, 1999; Pfahringer, Bensusan, & Giraud-Carrier, 2000). We believe this problem is closer in nature to ranking tasks like the ones commonly found in Information Retrieval and recommender systems. In those tasks, it is not known beforehand how many alternatives the user will actually take into account. For instance, the first suggestion given by a search engine may be ignored in favor of the second one because the latter site is faster and contains similar information. The same can be true of algorithm selection. The user may decide to continue using his favorite algorithm, if its performance is slightly below the topmost one in the ranking. Furthermore, when searching a topic on the web, one may investigate several links. The same can also apply to a data analysis task, if enough resources (time, CPU power, etc.) are available to try out more than one algorithm. Since we do not know how many algorithms the user might actually want to select, we

provide a ranking of all the algorithms. Algorithm recommendation using meta-learning was first handled as a ranking task by Brazdil, Gama, & Henery (1994). Later Nakhaeizadeh & Schnabl (1998) and more recently Keller, Paterson, & Berrer (2000) and also Brazdil & Soares (2000) used a similar approach. Here we cover some of these issues in greater depth.

1.3. Meta-learning algorithm

The issue concerning which method should be used for meta-learning does not yet have a satisfactory answer. Here we chose to use the instance-based learning (IBL) approach for the reasons explained next. In meta-learning the amount of data available (dataset descriptions, including algorithm performance) is usually quite small. Hence the task of inducing models that are general is hard, especially with algorithms that generate crisp thresholds, like decision tree and rule induction algorithms usually do. IBL has also the advantage that the system is extensible; once a new experimental result becomes available, it can be easily integrated into the existing results without the need to reinitiate complex re-learning. This property is relevant for algorithm selection because, typically, the user starts with a small set of meta-data but this set increases steadily with time.

Existing IBL approaches have been used either for classification or regression. Given that we are tackling a different learning problem, ranking, we had to adapt this methodology for this aim. We opted for a simple algorithm, the k-Nearest Neighbor (Mitchell, 1997, ch. 8), discussed in Sections 2 and 3. A distance function based on a set of statistical, information theoretic and other dataset characterization measures is used to select the most similar neighbors, that is, the datasets whose performance is expected to be relevant for the dataset at hand. The prediction, i.e., the recommended ranking, is constructed by aggregating performance information for the given candidate algorithms on the selected datasets. There are various ways how we can do that. We have shown earlier that a ranking method based on the relative performance between pairs of algorithms, assessed using success rate ratios, competes quite well with other alternative approaches (Brazdil & Soares, 2000).

1.4. Multicriteria assessment of performance

The evaluation measure that is commonly used in prediction problems is success rate.[1] However, it has been argued that it is important to consider several measures, especially in KDD, where the final user of the model is often not a data analyst (Nakhaeizadeh & Schnabl, 1997). We may consider, in addition to success rate, the interpretability of the model and also the speed of the algorithm. Interpretability is a highly subjective criterion because it depends on the expertise of the user as well as on its preferences. As for speed, some people could argue that it is not so important due to the increasing computational power available nowadays. In our view this objection does not hold in general because some algorithms may simply run for too long on the volumes of data gathered. Therefore, we have decided to investigate how to take into account the information regarding accuracy and speed (measured by the total execution time). It represents one important step in the direction of multicriteria evaluation of classification algorithms.

The ML and KDD communities usually ignore the issue of multicriteria evaluation, with the noteworthy exception of Nakhaeizadeh & Schnabl (1997, 1998). There are obviously many different ways of doing that but none has been widely adopted. Two important issues should affect the design of multicriteria evaluation. Firstly, it should be taken into account that the compromise between the criteria involved is often defined by the user. Thus, it should be defined in a simple and clear way. Secondly, multicriteria measures should yield values that can be interpreted by the user. We have tried to take these issues into account in the measure presented in Section 3.1.

1.5. Evaluation of ranking methods

Ranking can be seen as an alternative ML task, similar to classification or regression, which must therefore have an appropriate evaluation method. We describe a methodology for evaluating and comparing ranking methods and how this is applied to our situation. This is done in Sections 4 and 5, respectively. The rankings recommended by the ranking methods are compared against the observed rankings using Spearman's rank correlation coefficient (Neave & Worthington, 1992). To compare different ranking methods we use a combination of Friedman's test and Dunn's Multiple Comparison Procedure (Neave & Worthington, 1992).

2. Selection of relevant datasets by the IBL meta-learner

Given a new problem (*query dataset*), we wish to generate a ranking of the given set of candidate algorithms that would be related to their actual performance on that dataset, without actually executing them. As it is not possible to determine a unique ranking that would be valid for all datasets, we proceed as follows. We select, from a set of previously processed datasets (*training datasets*), those that are similar to the given query dataset. We expect that if two datasets are quite similar, so should be the corresponding performance of a given candidate algorithm. Then we build the ranking based on this information. Selection is performed with a simple instance-based learner, the k-Nearest Neighbor (k-NN) algorithm (Mitchell, 1997). Given a case, this algorithm simply selects k cases that are nearest to it according to some distance function.

Measuring the similarity between datasets is not a simple task. Here we follow the approach of characterizing datasets using general, statistical and information theoretic measures (*meta-attributes*), described by Henery (1994). Examples of these three types of measures are *number of attributes*, *mean skewness* and *class entropy*, respectively. These kinds of measures have frequently been used in meta-learning (Brazdil, Gama, & Henery, 1994; Kalousis & Theoharis, 1999; Lindner & Studer, 1999). However, the number of meta-attributes available is relatively large considering that the performance information available includes relatively few examples (datasets). This problem is exacerbated by the fact that the nearest-neighbor algorithm is rather sensitive to irrelevant attributes (Mitchell, 1997). Therefore we selected *a priori* a small subset that, we believe, provides information about properties that affect algorithm performance. Below, we present a summary of those measures and the properties which they are expected to represent:

- The *number of examples (n.examples)* discriminates algorithms according to how scalable they are with respect to this measure.
- The *proportion of symbolic attributes (prop.symb.attrs)* is indicative of the aptitude or inadequacy of the algorithm to deal with symbolic or numeric attributes.
- The *proportion of missing values (prop.missing.values)* discriminates algorithms according to how robust they are with respect to incomplete data. This measure was later eliminated, as explained below.
- The *proportion of attributes with outliers (prop.attr.outliers)* discriminates algorithms according to how robust they are to outlying values in numeric attributes, which are possibly due to noise.[2] An attribute is considered to have outliers if the ratio of the variances of mean value and the α-trimmed mean is smaller than 0.7. We have used $\alpha = 0.05$.
- The *entropy of classes (class.entropy)* combines information about the number of classes and their frequency, measuring one aspect of problem difficulty.
- The *average mutual information of class and attributes (mut.info)* indicates the amount of useful information contained in the symbolic attributes. This measure was later dropped, as explained below.
- The *canonical correlation of the most discriminating single linear combination of numeric attributes and the class distribution (can.cor)* indicates the amount of useful information contained in groups of numeric attributes.

More details can be found in (Henery, 1994). Next, we performed a visual analysis of this set of measures with the aim of identifying measures that seem to provide little useful information. This was done by analyzing the correlation between values of a particular meta-attribute chosen and the performance of each algorithm. For each meta-attribute and algorithm pair, we plotted the values of the given meta-attribute and the ranks of the algorithm for all the datasets considered (Section 3.3). Figure 1 shows the graphs for algorithm C5.0 (tree) and the meta-attributes proportion of symbolic attributes and proportion of missing values. In the graph on the left-hand side, two clusters of points can be observed, on the top-left and bottom-right corners. This indicates that C5.0 performs better on datasets with more symbolic attributes and hence that this attribute should be kept. On the other hand, we cannot observe clear patterns in the graph on the right-hand side, concerning the proportion of missing values. This indicates that this meta-attribute may not be so useful. We performed this analysis for all pairs of meta-attributes and algorithms, and decided to drop two of the measures: proportion of missing values and average mutual information of class and attributes.

The distance function used here is the unweighted L_1 norm (Atkeson, Moore, & Schaal, 1997).

$$dist(d_i, d_j) = \sum_x \frac{\left| v_{x,d_i} - v_{x,d_j} \right|}{\max(v_{x,d_k}) - \min(v_{x,d_k})}$$

where d_i and d_j are datasets, and v_{x,d_i} is the value of meta-attribute x for dataset d_i. The distance is divided by the range to normalize the values, so that all meta-attributes have the same range of values.

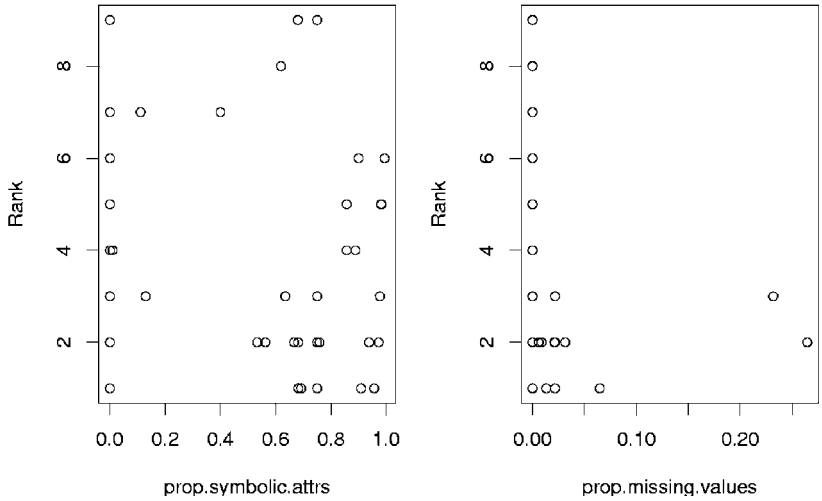

Figure 1. Plots of two meta-attribute (proportion of symbolic attributes and proportion of attributes with missing values) values against the rank of C5.0 (tree) for 53 datasets (see Section 3.3).

It may be the case that a meta-attribute is not applicable to a particular dataset. If dataset d_i has *no numeric attributes*, then it makes no sense to calculate, for instance, the canonical correlation meta-attribute. For such an attribute, dataset d_j can be considered *close* to dataset d_i if it also does not have any numeric attributes and their distance is 0. On the other hand, if dataset d_j does have numeric attributes, the dataset is considered to be different from d_i in this respect. So, the distance is the maximum possible, i.e., 1, after normalization.

3. Ranking based on accuracy and time

The method described further on, referred to as the *adjusted ratio of ratios* (ARR) ranking method, aggregates information concerning accuracy and time. It can be seen as an extension of the *success rate ratios* (SRR) method. This method is presented in (Brazdil & Soares, 2000) together with two other basic ranking methods, *average ranks* (AR) and *significant wins* (SW). The methods differ in the way performance information is aggregated to generate a ranking and are commonly used in literature on comparative studies.

The argument that one algorithm is better than another is often supported by *ratios* of success rates. This motivated us to consider the SRR method. Others prefer to provide information about how the algorithms are *ranked* on different datasets. Finally, some researchers prefer to count on how many datasets one method is *significantly better* than another. These approaches provided a motivation to consider the AR and the SW ranking methods.

We could ask the question of which one is "better" in a given situation. We have investigated this issue earlier (Brazdil & Soares, 2000). As the SRR method competed well with the other two on the datasets considered, we have adopted it in further studies. This method has an additional advantage: it can be easily extended to also incorporate time, as we shall see in the next section.

3.1. Combining success rates and time

The ARR multicriteria evaluation measure combines information about the accuracy and total execution time of learning algorithms. ARR is defined as:

$$ARR^{d_i}_{a_p,a_q} = \frac{\frac{SR^{d_i}_{a_p}}{SR^{d_i}_{a_q}}}{1 + AccD * \log\left(\frac{T^{d_i}_{a_p}}{T^{d_i}_{a_q}}\right)} \quad (1)$$

where $SR^{d_i}_{a_p}$ and $T^{d_i}_{a_p}$ represent the success rate and time of algorithm a_p on dataset d_i, respectively, and $AccD$, which is explained below, represents the relative importance of accuracy and time, as defined by the user.

The ARR measure can be related to the *efficiency measure* used in Data Envelopment Analysis (DEA), an Operations Research technique (Charnes, Cooper, & Rhodes, 1978) that has recently been used to evaluate classification algorithms (Nakhaeizadeh & Schnabl, 1997, 1998; Keller, Paterson, & Berrer, 2000). The efficiency measure of DEA identifies the set of *efficient* algorithms, lying on a frontier (*efficiency frontier*). The philosophy underlying both measures is the same. The efficiency measure of DEA uses a ratio of *benefits* and *costs* to assess the overall quality of a given candidate, when compared to others. In the SRR method presented earlier, the ratio of success rates, $SR^{d_i}_{a_p}/SR^{d_i}_{a_q}$, can be seen as a measure of the advantage of algorithm a_p over algorithm a_q (i.e., a benefit). The equivalent ratio for time, $T^{d_i}_{a_p}/T^{d_i}_{a_q}$, can be seen as a measure of the disadvantage of algorithm a_p over algorithm a_q (i.e., a cost). Thus, like in DEA, if we take the ratio of the benefit and the cost, we obtain a measure of the overall quality of algorithm a_p. However, we note that time ratios have, in general, a much wider range of possible values than success rate ratios. If a simple time ratio were used it would dominate the ratio of ratios. This effect can be controlled by re-scaling. We use $\log(T^{d_i}_{a_p}/T^{d_i}_{a_q})$ that provides a measure of the order of magnitude of the ratio. The relative importance between accuracy and time is taken into account by multiplying this expression by the $AccD$ parameter. This parameter is provided by the user and represents the amount of accuracy he/she is willing to trade for a 10 times speedup or slowdown. For example, $AccD = 10\%$ means that the user is willing to trade 10% of accuracy for 10 times speedup/slowdown. Finally, we also add 1 to yield values that vary around 1, as happens with the success rate ratio.

There are a few differences between ARR and the efficiency measure of DEA. The ARR measure captures the relative performance of two algorithms for a given compromise between the criteria. To compare an algorithm to others, some aggregation must be performed, as it will be shown in the next section. Furthermore, the weights are assumed to have predefined values. In DEA the efficiency of an algorithm is obtained by comparing it to others candidates and the compromise between the criteria is set automatically.

3.2. Generation of rankings

A ranking of the candidate algorithms is built by calculating the ARR value for each of them, expressing their relative quality. For that purpose we aggregate the performance information

selected by the k-NN algorithm for each algorithm in the following way. First we calculate the geometric mean across all datasets and then the arithmetic mean across algorithms:

$$ARR_{a_p} = \frac{\sum_{a_q} \sqrt[n]{\prod_{d_i} ARR^{d_i}_{a_p,a_q}}}{m} \tag{2}$$

where n represents the number of datasets and m the number of algorithms. The higher the ARR value of an algorithm, the higher the corresponding rank.[3] The ranking is derived directly from this measure.

We used a geometric mean because we prefer that the relative performance of ap and aq across several datasets, $ARR_{ap,aq}$, verifies the following property: $ARR_{ap,aq} = 1/ARR_{aq,ap}$. This would not be true if the arithmetic mean was used.

In the following section we present the experimental setting and a simple example illustrating the use of this ranking method.

3.3. Experimental setting and an example of rankings

The set of meta-data used here was obtained from the METAL project (METAL Consortium, 2002). It contains estimates of accuracy and time for 10 algorithms on 53 datasets, using 10-fold cross-validation.[4] The algorithms include three decision tree classifiers, C5.0, boosted C5.0 (Quinlan, 1998) and Ltree, which is a decision tree algorithm that can induce oblique decision surfaces (Gama, 1997). Two rule-based systems were also used, C5.0 rules and RIPPER (Cohen, 1995), as well as two neural networks from the SPSS Clementine package, Multilayer Perceptron (MLP) and Radial Basis Function Network (RBFN). We also included the instance-based learner (IB1) and the naive Bayes (NB) implementations from the MLC++ library (Kohavi et al., 1994). Finally, an implementation of linear discriminant (LD) (Michie, Spiegelhalter, & Taylor, 1994) was also used. All algorithms were executed with default parameters which is clearly a disadvantage for some of them (e.g., MLP and RBFN).

The 53 datasets used included all datasets from the UCI repository (Blake, Keogh, & Merz, 1998) with more than 1000 cases,[5] plus the Sisyphus data[6] and two confidential datasets provided by DaimlerChrysler.

To illustrate the ARR ranking method presented earlier, we apply it to the full set of meta-data, for three different settings of the compromise between accuracy and time, $AccD \in \{0.1\%, 1\%, 10\%\}$. The setting $AccD = 0.1\%$ ($AccD = 10\%$) represents a situation where accuracy (time) is the most important criterion. The setting $AccD = 1\%$ represents a more balanced compromise between the two criteria.

The rankings obtained (Table 1) represent an overall picture of the performance of the algorithms on the 53 datasets. We observe that, as expected, the ranks of faster algorithms (e.g., Ltree and LD) improve as time is considered more important (i.e., $AccD = 10\%$), while the opposite occurs for slower ones (e.g., boosted C5.0 and RIPPER).

To illustrate how we can generate a ranking using the k-NN meta-learning method, we present an example. Suppose we want to obtain a ranking of the given algorithms on a given dataset, `abalone`, without conducting tests on that dataset. We must, thus, use only

Table 1. Overall rankings based on the 53 datasets for three different settings of the compromise between accuracy and speed. The dominant criterion is indicated next to the *AccD* value.

	0.1% (accuracy)		1%		10% (time)	
AccD rank	a_p	ARR_{a_p}	a_p	ARR_{a_p}	a_p	ARR_{a_p}
1	boosted C5.0	1.13	boosted C5.0	1.14	C5.0 (tree)	1.26
2	C5.0 (rules)	1.11	C5.0 (rules)	1.12	C5.0 (rules)	1.20
3	C5.0 (tree)	1.10	C5.0 (tree)	1.11	Ltree	1.18
4	Ltree	1.10	Ltree	1.10	boosted C5.0	1.17
5	IB1	1.06	IB1	1.06	LD	1.11
6	RIPPER	1.00	RIPPER	1.00	IB1	1.05
7	LD	0.98	LD	0.99	NB	1.03
8	NB	0.95	NB	0.96	RIPPER	0.99
9	MLP	0.88	MLP	0.87	MLP	0.77
10	RBFN	0.79	RBFN	0.78	RBFN	0.68

information about the performance of the algorithms on the remaining 52 *training datasets* in the process. Table 2 presents a ranking generated by the ARR with the 5-NN method ($AccD = 0.1\%$), which selected the following datasets: `vowel`, `pendigits`, `vehicle`, `satimage`, and `letter`.

The obvious questions is how good is this ranking? The overall ranking (Table 1) can be used as a baseline for the purpose of this comparison. We note that this ranking is somewhat different from the one shown in Table 2. In the following Section we explain how different ranking methods can be evaluated and compared.

4. Evaluation of ranking methods

The framework for the empirical evaluation of rankings is based on a leave-one-out procedure. For each dataset (*test dataset*), we do the following: (1) build a *recommended ranking* by applying the ranking method under evaluation to all but the test dataset (*training datasets*), (2) build an *ideal ranking* for the test dataset, and (3) Measure the *agreement* between the two rankings. The score of a ranking method is expressed in terms of the mean agreement between the recommended ranking and the ideal ranking.

The agreement is a measure of the quality of the recommended ranking and thus also of the ranking method that generated it.

4.1. Generation of the ideal ranking

The ideal ranking should represent, as accurately as possible, the correct ordering of the algorithms on the test dataset. It is constructed on the basis of their performance as estimated by conducting tests on that dataset. However, the ranking obtained simply by ordering the estimates may not capture well the notion of a true situation.

Table 2. Recommended ranking for `abalone` dataset using 5-NN for $AccD = 0.1\%$.

Rank	a_p	ARR_{a_p}
1	boosted C5.0	1.13
2	IB1	1.13
3	Ltree	1.07
4	C5.0 (rules)	1.06
5	C5.0 (tree)	1.05
6	RIPPER	1.01
7	MLP	1.01
8	LD	0.93
9	RBFN	0.87
10	NB	0.82

In one of the tests carried out we have observed that, on the `glass` dataset, C5.0 (error 30%) and Ltree (error 32%) were ranked 2nd and 3rd, respectively. However, their performance was *not significantly different* and the ideal ranking should reflect this. One possibility would be to represent this tie explicitly, e.g., assign rank 2.5 to both. Our approach here is different, and exploits the fact that in such situations these algorithms often swap positions in different folds of the N-fold cross-validation procedure. Therefore, we use N orderings to represent the true ideal ordering, instead of just one. The ideal ordering corresponding to fold j of dataset d_i is constructed by ordering the algorithms a_p according to $(\sum_{aq} ARR^{d_i,j}_{a_p,a_q})/m$, where m is the number of algorithms and $ARR^{d_i,j}_{a_p,a_q}$ is calculated in a similar manner to Eq. (1), but using the performance information in the fold j rather than the average.

4.2. *Evaluation of the recommended ranking*

To measure the agreement between the recommended ranking and each of the N orderings that represent the ideal ranking, we use Spearman's rank correlation coefficient (Neave & Worthington, 1992)

$$r_s = 1 - \frac{6\sum_{i=1}^{m}(rr_i - ir_i)^2}{m^3 - m} \tag{3}$$

where rr_i and ir_i are the recommended and ideal ranks of algorithm i respectively, and m is the number of algorithms. An interesting property of this coefficient is that it is basically the sum of squared errors, which can related to the commonly used error measure in regression. Furthermore, the sum is normalized to yield more meaningful values: the value of 1 represents perfect agreement and -1, perfect disagreement. A correlation of 0 means that the rankings are not related, which would be the expected score of the random ranking method.

Table 3. Effect of rank error $e = |rr_i - ir_i|$ in the value of the Spearman's rank correlation coefficient.

e	1	2	3	4	5	6	7	8	9
$d(e)$	0.006	0.024	0.055	0.097	0.152	0.218	0.297	0.388	0.491

To support interpretation of the values obtained, we can use the following function:

$$d(e) = \frac{6e^2}{m^3 - m}$$

which calculates the difference in the value of Spearman's rank correlation if an algorithm is incorrectly ranked by e positions, i.e., $e = |rr_i - ir_i|$. For example, suppose that the algorithms in the 3rd and 5th positions have been swapped, in a ranking of 10 algorithms. Therefore, for each of them $e = |3 - 5| = 2$ and the total difference in the correlation value is $2 * d(2) = 2 * (6 * 2^2/(10^3 - 10)) = 0.048$. If all other algorithms were ranked correctly the correlation would be $1 - 0.048 = 0.952$. The inverse of this function can be used to interpret correlation values obtained. Suppose, for instance, that a correlation value of 0.952 was obtained in a ranking of 10 algorithms. As the previous calculations show, this difference means that 2 algorithms two ranks apart were swapped. Table 3 lists values of $d(e)$ for rankings with 10 algorithms.

We illustrate the use of Spearman's rank correlation coefficient (Eq. (3)) for recommended ranking evaluation with a simplified example (Table 4). Given that the number of algorithms is $m = 6$, we obtain $r_s = 1 - \frac{6*10.5}{6^3-6} = 0.7$. Note that naive Bayes and linear discriminant share the second place in the recommended ranking, so they are both assigned rank $\frac{2+3}{2} = 2.5$, following the method in (Neave & Worthington, 1992).[7]

As explained in the previous section, these calculations are repeated for all the folds, permitting us to calculate the score of the recommended ranking as the average of the individual coefficients or simply, as *average correlation*, $\overline{r_S}$.

Table 4. Simplified example of the evaluation of a recommended ranking using Spearman's rank correlation coefficient.

i	rr_i	ir_i	$(rr_i - ir_i)^2$
C5.0 (tree)	4	4	0
Ltree	1	1	0
IB1	6	6	0
LD	2	2.5	0.25
NB	5	2.5	6.25
boosted C5.0	3	5	4
$\sum_i^m (rr_i - ir_i)^2$			10.5
r_s			0.7

One important consequence of having several orderings to represent the ideal ranking is the following. The correlation coefficient will be 1 only if all the rankings are exactly the same. In other words, the maximum average correlation will be less than 1 if at least one of the orderings is different from the others. This sets a limit for the maximum achievable value for the recommended ranking.

The critical values (Neave & Worthington, 1992) provide, for a given number of items and confidence level, the threshold which determines whether two rankings are significantly correlated or not. The critical value for a ranking of ten algorithms and for a 95% confidence level (one-sided test) is 0.564. So, if the value is higher, it indicates that a significant correlation might exist. As we are dealing with average values of the coefficient, strictly speaking, no definite conclusions should be drawn. However, the critical value provides a reference value that can be used to assess the quality of the generated rankings.

4.3. Results

The k-NN approach to ranking was evaluated empirically using two values of the number of neighbors (k), 1 and 5. We chose 1-NN because it is known to perform often well (Ripley, 1996). The 5 neighbors represent approximately 10% of the 52 training datasets, and this has lead to good results in a preliminary study (Soares & Brazdil, 2000). Finally we have evaluated a simple ranking method consisting of applying ARR to *all* the training datasets (i.e., 52-NN), which will be referred to as ARR. The ARR method can be considered as a baseline to assess the improvements obtained due to meta-learning with k-NN. Three different settings of the compromise between accuracy and time were considered, namely $AccD \in \{0.1\%, 1\%, 10\%\}$.

The three variants of the method were evaluated following the leave-one-out procedure defined at the beginning of Section 4. The recommended rankings were compared to each of the individual ideal orderings using Spearman's rank correlation coefficient and the average correlation obtained, $\overline{r_S}$, represents the score of the corresponding method on that iteration.

The results are presented in Table 5. The most important conclusion is that using meta-learning with k-NN improves the results of the baseline ranking method, ARR. Furthermore, the difference increases with the importance of accuracy. This is expected, given that the selection of meta-attributes was made mainly having accuracy in mind. Besides, the baseline

Table 5. Mean average correlation ($\overline{r_S}$) obtained with ARR with 1-NN, 5-NN and all data for different values of *AccD* on the METAL meta-data. The $+/-$ column indicates the number of datasets where the corresponding method has higher/lower correlation than ARR. The best results for each setting are emphasized. The dominant criterion is indicated next to the *AccD* value.

Accuracy/time compromise	ARR with 1-NN		ARR with 5-NN		ARR
AccD	mean $\overline{r_S}$	$+/-$	mean $\overline{r_S}$	$+/-$	mean $\overline{r_S}$
0.1% (accuracy)	**0.619**	32/21	0.543	28/24	0.524
1%	**0.649**	31/22	0.587	30/23	0.569
10% (time)	**0.759**	30/23	0.758	31/22	0.736

value for ARR when time is considered important ($AccD = 10\%$) is relatively high (0.736), and it is obviously more difficult to improve this value further.

We also observe that there is a clear positive correlation between the recommended rankings generated and the ideal rankings. As mentioned earlier, Table 3 can be used to provide an approximate interpretation of the values obtained. One such interpretation for the score of 0.759 obtained by 1-NN for $AccD$=10% is that, on average, this method approximately swapped one pair of algorithms by two positions and another one by four ($2 * 0.024 + 2 * 0.097 = 0.242$). Another interpretation for the same situation is that it misplaced two pairs of algorithms by three positions and another two by one rank ($4 * 0.055 + 4 * 0.006 = 0.242$).

One surprising result is the good performance of the baseline method, ARR. The expected performance of this method—in an unrealistic setting where the distribution of the ranks of algorithms was uniform—would be equal to the expected performance of a random ranker, i.e., 0. In our experiments, the values obtained ranged from 0.524 to 0.737. The explanation for this is that the rank distribution in our meta-data is very uneven. The algorithm boosted C5.0 is very often in the first position. This was unmatched, say, by RBFN with default parameters. Therefore, the task of improving the performance of the baseline ranking method is quite difficult.

The performance of 1-NN is particularly good, with an advantage of almost 0.10 when accuracy is the dominant criterion. In summary, our results show that the IBL approach proposed together with an adequate selection of meta-features performs better than the baseline, despite the good performance of the latter. To determine whether this improvement is statistically significant rather than caused by chance, we conducted appropriate statistical tests which are presented in the next section.

5. Comparison of ranking methods

To test whether the results presented in the previous section are statistically significant, we have used Friedman's test, a distribution-free hypothesis test on the difference between more than two population means (Neave & Worthington, 1992). The reasons for this choice are (1) we have no information about the distribution of the average correlation coefficient in the population of datasets, so a distribution-free test is required, (2) the number of methods we wish to compare (i.e., samples) is larger than 2, and (3) the samples are *related* because the methods are evaluated on the same set of datasets. According to Neave & Worthington (1992) not many distribution-free methods can compete with Friedman's test with regard to both power and ease of computation.

Here, the hypotheses are:

H_0: There is *no difference* in the mean average correlation coefficients, $\overline{r_S}$, for the three ranking methods (ARR with 1-NN, ARR with 5-NN and ARR with all data).

H_1: There are *some differences* in the mean average correlation coefficients, $\overline{r_S}$, for the three ranking methods.

To illustrate Friedman's test we compare four fictitious ranking methods ($j = 1, \ldots, 4$) on simulated meta-data consisting of three datasets (Table 6). For the sake of simplicity,

Table 6. Some steps in the application of Friedman's test to compare ranking methods.

	dataset 1		dataset 2		dataset 3			
Method j	r_s	R_j^1	r_s	R_j^2	r_s	R_j^3	$\bar{R}_j$	$(\bar{R}_j - \bar{R})^2$
1	0.357	1	−0.171	2	0.630	1	1.3	1.36
2	0.314	2	−0.086	1	0.577	2	1.7	0.69
3	0.301	3	−0.214	3	0.552	3	3	0.25
4	0.295	4	−0.401	4	0.218	4	4	2.25

we assume that the ideal ranking consists of a single ordering rather than N, one for each fold of the cross-validation procedure (Section 4). First, we rank the correlation coefficients obtained by the ranking methods for each dataset. We thus obtain $R_j^{d_i}$, representing the rank of the correlation obtained by ranking method j on dataset d_i, when compared to the corresponding correlations obtained by the other methods on the same dataset. Next, we calculate the *mean rank* for each method, $\bar{R}_j = \frac{\sum_i R_j^{d_i}}{n}$, where n is the number of points in the sample (datasets in the present case) and the *overall mean rank* across all methods, $\bar{R}$. As each method is ranked from 1 to k, where k is the number of methods being compared, we know that $\bar{R} = \frac{k+1}{2} = 2.5$. Then, we calculate the sum of the squared differences between the mean rank for each method and the overall mean rank, $S = \sum_j (\bar{R}_j - \bar{R})^2$. Finally, we calculate Friedman's statistic, $M = \frac{12nS}{k(k+1)}$. In this simple example, where $n = 3$ and $k = 4$, we obtain $S = 4.56$ and $M = 8.2$.

The rationale behind this test is that if H_0 is true, that is, if all methods perform equally well on average, then the distribution of ranks should be approximately uniform for all methods. In other words, each method should be ranked first approximately as many times as it is ranked second, and so on. If this were true, the mean rank for each method would be similar to the overall mean rank, i.e., $\bar{R}_j \simeq \bar{R}$ and the sum of the squared difference between those values would be small, $S \simeq 0$. Consequently, Friedman's statistic, M, which can be seen as a normalized value of S, taking into account the size of the data, will also be small. The larger the differences in performance, the larger the value of M. The null hypothesis will be rejected if $M \geq$ *critical value*, where the critical value is obtained from the appropriate table, given the desired confidence level, the number of samples (i.e., methods), k, and the number of points (i.e., datasets), n. If this were the case, we can claim, with the given confidence level, that the methods have different performance. In the example of Table 6, where $n = 3$ and $k = 4$, the critical value is 7.4 for a confidence level of 95%. As the test $8.2 \geq 7.4$ is true, we reject the null hypothesis that the average performance of the four ranking methods is the same.

Dealing with Ties: When applying this test ties may occur, meaning that two ranking methods have the same correlation coefficient. In that case, the average rank value is assigned to all the methods involved, as explained earlier for Spearman's correlation coefficient. When the number of ties exceeds a limit, the M statistic must be corrected.

Assuming, for the sake of argument, that the figures in Table 6 were as follows. Suppose that the correlation of methods 2, 3, and 4 on dataset 1 was 0.314 and the correlation of

methods 3 and 4 on dataset 2 was -0.401, we thus would have ties in dataset 1 among methods 2, 3, and 4 and in dataset 2 between methods 3 and 4. In the former case, rank $(2+3+4)/3 = 3$ is assigned to all the tied methods and, in the latter, rank $(3+4)/2 = 3.5$ to methods 3 and 4. Next, we calculate Friedman's statistic as before, $M = 5.5$. Then, for each dataset, we calculate $t^* = t^3 - t$, where t is the number of methods contributing to a tie. Given that three methods are tied on dataset 1, we obtain $t = 3$ and $t^* = 24$. On dataset 2, we obtain $t = 2$ and $t^* = 6$ and on dataset 3, $t = t^* = 0$. Next, we obtain $T = 30$ by adding up all t^*'s. The correction factor is $C = 1 - \frac{T}{n(k^3-k)}$, yielding 0.83 in our example. The modified statistic is $M^* = M/C = 6.6$. The critical values for M^* are the same as for M, so the null hypothesis cannot be rejected for a confidence level of 95% because $6.6 \geq 7.4$ is false.

5.1. *Results of Friedman's test*

In the example used above, we have assumed, for simplicity sake, that the data consists of n correlation values for each ranking method, where n is the number of datasets. However, as explained in Section 4, our evaluation methodology generates $n * N$ points for each method, where N is the number of folds in the cross-validation procedure used. Our comparison is based on these detailed results. The only change required to the test described is to replace n by $n * N$. Applying Friedman's test to the results of Section 4.3, we observe that there are statistically significant differences at a 99% confidence level between the ranking methods compared (Table 7).

5.2. *Which method is better?*

In the previous section we have shown that there are significant differences between the $k = 3$ methods. Naturally, we must determine which methods are different from one another. To answer this question we use *Dunn's multiple comparison* technique (Neave & Worthington, 1992). Using this method we test $p = \frac{1}{2}k(k-1)$ hypotheses of the form:

Table 7. Results of Friedman's test. The dominant criterion is indicated next to the *AccD* value.

Accuracy/time compromise (*AccD*)	0.1% (accuracy)	1%	10% (time)
$\bar{R}_{1\text{-}NN}$	1.90	1.89	1.95
$\bar{R}_{5\text{-}NN}$	2.00	2.00	1.88
$\bar{R}_{ARR}$	2.10	2.11	2.17
M	10.0	13.8	24.4
k		3	
nN		530	
Critical value (99%)		9.21	
Significantly different?	Yes	Yes	Yes

$H_0^{(i,j)}$: There is no difference in the mean average correlation coefficients between methods i and j.
$H_1^{(i,j)}$: There is some difference in the mean average correlation coefficients between methods i and j.

Again we refer to the example in Table 6 to illustrate how this procedure is applied. First, we calculate the rank sums for each method. In this case they are 4, 5, 9 and 12 respectively for methods 1 to 4. Then we calculate the normalized differences of rank sums, $T_{i,j} = D_{i,j}/stdev$ for each pair of ranking methods, where $D_{i,j}$ is the difference in the rank sums of methods i and j, and $stdev = \sqrt{\frac{nk(k+1)}{6}}$. As before, k is the number of methods and n is the number of datasets. In the example, where $n = 3$ and $k = 4$ we obtain $stdev = 3.16$. To compare, for example, method 1 to methods 2 and 4, we get $D_{1,2} = -1$ and $D_{1,4} = -8$ and finally, we obtain $T_{1,2} = -0.32$ and $T_{1,4} = -2.53$. These calculations are repeated for all pairs of methods.

The values of $T_{i,j}$, which approximately follow a normal distribution, are used to reject or accept the corresponding null hypothesis at an appropriate confidence level. As we are doing multiple comparisons simultaneously, we have to carry out the Bonferroni adjustment. This technique redefines the significance level[8] to be used in individual tests by dividing the overall significance level, α, by the number of tests. It aims to prevent the chance of obtaining significant conclusions by chance. Here we are doing pairwise comparisons between k methods, so the adjusted significance level is $\alpha' = \alpha/k(k-1)$. However, given that the risk of obtaining false significant differences is somewhat reduced due to the previous application of Friedman's test, Neave & Worthington (1992) suggest a rather high overall significance level, α, (between 10% and 25%). If we use an overall significance level $\alpha = 25\%$, we obtain $\alpha' = \alpha/k(k-1) = 2.08\%$ for our example where $k = 4$. Consulting the appropriate table we obtain the corresponding critical value, $z = 2.03$. If $|T_{i,j}| \geq z$ then the methods i and j are significantly different. Using the values calculated above, we can conclude that method 1 is significantly better than method 4 ($|T_{1,4}| \geq 2.03$ is true) but not significantly better than method 2 ($|T_{1,2}| \geq 2.03$ is false).

5.3. *Results of Dunn's multiple comparisons procedure*

Given that Friedman's test has shown that there are significant differences between the three methods compared, 1-NN, 5-NN and ARR, we have used Dunn's multiple comparison procedure to determine which ones are significantly different. Given that three methods are being compared, the number of hypotheses being tested is $p = 3$. The overall significance level is 25% which, after the Bonferroni adjustment, becomes 4.2%, corresponding to a critical value of 1.73. We observe (Table 8) that the 1-NN method is significantly better than the baseline method, ARR, on all settings of the accuracy/time compromise we have tested. On the other hand, 5-NN is also significantly better than the baseline, except when accuracy is very important ($AccD = 0.1\%$), although the p-value (1.67) is quite close the critical value (1.73). The performance of 1-NN and 5-NN is only significantly different when $AccD = 1\%$, although, as before, the p values are also close to the critical value. These results confirm that IBL improves the quality of the rankings when compared to the baseline.

Table 8. Results of Dunn's multiple comparison test. The symbol $\gg$ represents the "significantly better" relation. The dominant criterion is indicated next to the *AccD* value.

Accuracy/time compromise (*AccD*)	0.1% (accuracy)	1%	10% (time)
$T_{1\text{-}NN,ARR}$	3.16	3.71	3.55
$T_{5\text{-}NN,ARR}$	1.67	1.90	4.74
$T_{1\text{-}NN,5\text{-}NN}$	1.49	1.81	1.20
Critical value (25%)		1.73	
1-NN $\gg$ *ARR*	Yes	Yes	Yes
5-NN $\gg$ *ARR*	No	Yes	Yes
1-NN $\gg$ *5-NN*	No	Yes	No

6. Discussion and further work

We have shown that meta-learning with k-NN improves the quality of rankings in general, but the results raise a few questions that are addressed here.

6.1. Meta-learning within complex multistrategy learning systems

In this paper we have focussed our attention on the issue of how we can exploit meta-learning to pre-select and recommend one or more classification algorithms to the user. The choice of adequate methods in a multistrategy learning system may significantly improve its overall performance. The approach described here is a first step in this direction. As the meta-knowledge can be extended depending on which types of problem get encountered, the whole system can adapt to new situations. Adaptability is often seen as a desirable property and a crucial aspect of intelligent systems. Having a multiplicity of (possibly adaptable) methods is on one hand an advantage, but of course, a question arises which ones one should use when. One interesting problem in future should investigate how the methodology presented could be applied and/or extended to make complex multistrategy learning systems adaptable, so as to provide us with efficient solutions.

6.2. Meta-learning versus simpler strategies

Here we analyze the results of our meta-learning approach in terms of the trade-off between accuracy loss and time savings (figure 2). Two simple reference strategies for this purpose are cross-validation (CV) and the selection of the algorithm with the best average accuracy. Although CV may also fail, it is currently the best method for performance estimation and also for algorithm selection (Schaffer, 1993). However, it is rather impractical in a wide variety of situations due to the size of the data and the number of algorithms available.[9] In our experimental setting, it achieved an accuracy of 89.93%, taking on average approximately four hours to run all the algorithms on one dataset. As for the use of the algorithm with the best performance, it can be seen as the "default decision" for algorithm recommendation.

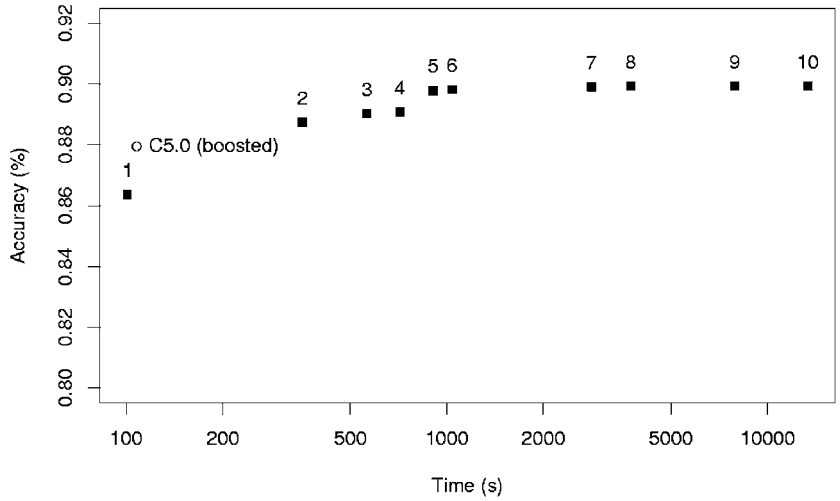

Figure 2. Average accuracy versus average execution time for the strategy of executing the top-N algorithms in the recommended ranking, for all possible values of N, and simpler strategies of cross-validation (Top-10) and selecting the algorithm which obtains the best accuracy, on average, (boosted C5.0).

In our experimental setting it is boosted C5.0, achieving an accuracy of 87.94% and taking less than two minutes to run, on average. One could argue that, with such a small margin for improvement (2%), it is not worthwhile to worry about algorithm selection: choosing boosted C5.0 will provide quite good results on average. However, in some business applications (e.g., cross-selling in a e-commerce site that sells thousands of items daily), an improvement of 2% or even less may be significant.

The strategy of executing the algorithm ranked in the first position is worse than executing boosted C5.0. However, if we use the full potential of a ranking method, and execute the top 2 algorithms in the ranking (strategy Top-2 in the figure), the time required is larger than boosted C5.0's, although still acceptable in many applications (less than 6 min.) and the loss of accuracy would be only 1.20%. Running one more algorithm (method Top-3) would provide further improvement in accuracy (0.90% loss) while taking only a little longer (less than 10 min.). CV performance is almost achieved by Top-5, (0.15% losses), still taking what would be an acceptable amount of time in many applications (approximately 15 min.). The overhead of meta-learning is negligible. On the 53 datasets used in this study, the time to calculate the measures was never higher than 1 min[10] and meta-learning using this Nearest-Neighbor algorithm is almost instantaneous.

6.3. Ranking versus ensemble methods

It could be argued that a suitable ensemble method could be used, eliminating, thus, the need to rank individual algorithms. This is an interesting possibility, but it should be noted that we included one ensemble method in our study, namely boosted C5.0. As figure 2 shows, this method, like any other ensemble method, is expected to work well in some situations

only (boosted C5.0 was the best method in 19 out of 53 datasets). We expect that similar situations will happen for other ensemble methods and, therefore, from our perspective, the information about the performance of ensemble methods in the past can be used to rank them, together with other algorithms.

6.4. Other meta-learning approaches

Meta-learning has been used to combine different biases in the same model. CAMLET iteratively searches for the best bias for a given problem (Suyama, Negishi, & Yamaguchi, 1999). The Model Class Selection system recursively chooses the bias that is most adequate for different subsets of the data (Brodley, 1993). One disadvantage of these methods is that they require reprogramming to include new biases, unlike our ranking method, which can be applied to off-the-shelf algorithms.

Meta Decision Trees (MDT) select a particular classifier for each case in a dataset, rather than providing a prediction directly (Todorovski & Džeroski, 2000, 2003). In general, these alternative meta-learning approaches perform better than the basic algorithms on some datasets and worse in others. Their performance could possibly be improved if information about the past performance of algorithms was used.

Another different approach to meta-learning is based on so-called self-modifying policies (Schmidhuber, Zhao, & Schraudolph, 1997). This methodology is used to build complex probabilistic algorithms based on a set of instructions, some of which are capable of changing other instructions as well as themselves. This enables the algorithm to adapt its bias to the problem at hand, not only in-between problems but also during their solution. However, it is assumed that the series of problems are similar and thus bias selection depends on long-term performance rather than focussing on the problems that are most similar to the one at hand.

The knowledge transfer in between problems has recently been discussed also by Hochreiter, Younger, & Conwell (2001), who proposed a method for meta-learning for recurrent neural networks.

6.5. Analysis of ranking methods performance

Our results show that no meta-learning method is universally better than the others, as would be expected. We analyzed the space of datasets, as defined by our characterization measures, with the aim of identifying regions where each method performs best. Since the set of features selected seems to be more adequate for discriminating accuracy, we focussed on the setting where this is the dominant criterion ($AccD = 0.1\%$). First, we standardized each of the data characteristics X: the values X_i were replaced with $(X_i - \bar{X})/\sigma_X$, where $\bar{X}$ and σ_X represent the average value and the standard error of X. Then, we separated the datasets into three groups, depending on which of the ranking methods obtained the best result, with 22 datasets on the 1-NN group, 13 in the 5-NN group and 19 in the baseline group.[11] Then, for each group, we calculated the average (standardized) value of each of the characteristics. Some striking trends can be observed for most data characteristics (figure 3). For instance, analyzing the proportion of symbolic attributes meta-feature,

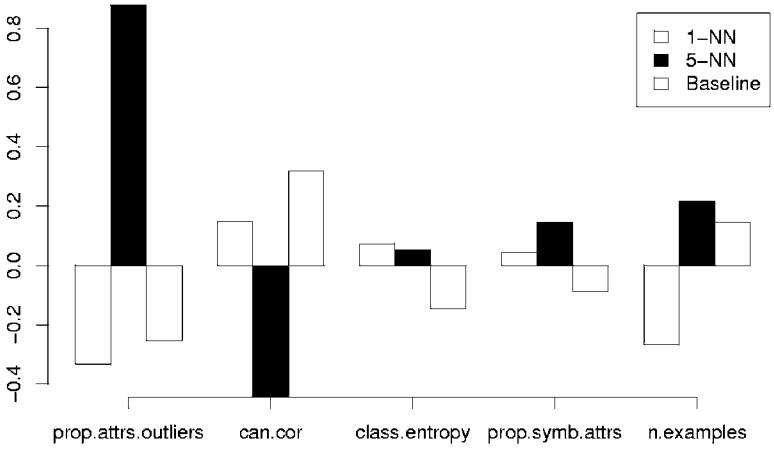

Figure 3. Average standardized value of data characterization measures, grouping the datasets by the method which performed best.

5-NN seems to be better for datasets with values above average while the baseline wins in exactly the opposite type of dataset. On the other hand, 1-NN has a tendency to be the best for datasets where the proportion of symbolic attributes is close to the average. Another interesting meta-feature is class entropy, where we observe that the baseline performs better for datasets with less-than-average values, while the opposite occurs for the IBL variants.

6.6. Meta-learning algorithm

In previous work where IBL was used for meta-learning, some positive (Gama & Brazdil, 1995; Lindner & Studer, 1999) and some negative (Kalousis & Hilario, 2000; Pfahringer, Bensusan, & Giraud-Carrier, 2000) results have been reported. These results are not contradictory, however, because the meta-learning problems addressed are different. Some address it as a regression problem, i.e., prediction of individual algorithm performance, while others as a classification problem, i.e., selection of the best algorithm from a set of candidate algorithms. We follow yet another approach to meta-learning, where the aim is to rank all the candidate algorithms (Section 1.2). Our choice of k-NN is due to some of its properties, like extensibility and ability to deal with small data (Section 1.3). The results obtained indicate that this was a good choice (Sections 4 and 5).

One common criticism of the IBL approach is that it does not provide explicit knowledge (e.g., a set of rules) to the user. However, each individual prediction can be explained quite simply by showing the instances on which the decision is based. This can be particularly useful in meta-learning because the user is probably more familiar with previous datasets than with some of the complex measures that are used to characterize them. However, the issue of whether a better ranking method can be devised remains open. We believe, however, that the use of other types of algorithms, like decision trees, depends on the availability of more meta-data (i.e., datasets), which may possibly be generated artificially. It is also

conceivable that the IBL approach could be improved (Atkeson, Moore, & Schaal, 1997), namely by weighting the contribution of each neighbor by its distance to the query dataset. The evaluation methodology presented here can be used to prove or disprove whether any new ranking method brings about any quantitative improvement.

6.7. Comparison with other ranking methods

In (Keller, Paterson, & Berrer, 2000), an IBL approach to meta-learning is also used to generate rankings of the algorithms. The main difference to our work is that the performance of the algorithms on the selected datasets is aggregated in a different way, using the concept of efficiency from Data Envelopment Analysis (Charnes, Cooper, & Rhodes, 1978). It is not trivial to compare these two ranking methods because the ideal rankings are not created in the same way. Measures that are independent of the ranking methods are required to enable a fair comparison (Berrer, Paterson, & Keller, 2000). We plan to do this in the future.

A different approach to ranking is proposed by Bensusan & Kalousis (2001). The authors proceed in two stages. In the first stage, the method predicts the performance of the algorithms using regression algorithms and then it generates a ranking by ordering the estimates. They test different regression algorithms and report better results for some of them when compared to IBL ranking. However, these results should be interpreted with caution given than the ideal ranking used consisted only of a single ordering based on average performance (Section 4.1).

An interesting system is presented by Bernstein & Provost (2001), called Intelligent Discovery Electronic Assistant (IDEA). IDEA consists of two components (1) a plan generator that uses an ontology to build a list of valid processes (i.e., a learning algorithm plus pre- and post-processing methods), and (2) a heuristic ranker that orders the valid processes based on some heuristic. The heuristics are knowledge-based and can take into account user preferences regarding accuracy and time, much like in the ARR measure presented here. Good results are presented, but a more thorough evaluation could be carried out, namely because the ideal ranking is again based on a single ordering only and the method is not compared to appropriate baselines. IDEA is independent of the ranking method and, thus, it would be interesting to replace their heuristic ranking method with ours, which is based on past performance.

6.8. Meta-attributes

As mentioned in Section 2, obtaining data characteristics that are good predictors of relative performance between algorithms is a very difficult task. Some of those measures may be irrelevant, others may not be adequately represented (e.g., the proportion of symbolic attributes is more informative than the number of symbolic attributes), while some important ones may be missing (one attribute which we could experiment with is concept complexity Vilalta (1999)). Furthermore, the number of data characteristics should not be too large for the amount of available meta-data. Contrary to previous work using similar measures (Gama & Brazdil, 1995; Lindner & Studer, 1999; Bensusan & Kalousis, 2001; Kalousis & Theoharis, 1999; Sohn, 1999), we selected a set of measures *a priori* with the aim

of representing some of the properties that affect the performance of algorithms. This set can be refined, e.g., by introducing measures of resilience to noise, etc. (Hilario & Kalousis, 1999). However, as our results show, most of the attributes we selected are mostly useful to discriminate algorithms on the basis of accuracy. To avoid this bias, one can use feature selection methods at the meta-level to choose the appropriate characteristics for a given multicriteria setting. Todorovski, Brazdil, & Soares (2000) show that the quality of rankings can be improved in this way. One important issue is that dataset characterization is relational in nature. For instance, skewness is calculated for each numeric attribute and the number of attributes varies for different datasets. The most common approach, which was also followed here, is to do some aggregation, e.g., calculate the mean skewness. Kalousis & Theoharis (1999) use a finer-grained aggregation, where histograms with a fixed number of bins are used to construct new meta-attributes, e.g., skewness smaller than 0.2, between 0.2 and 0.4, etc. Individual attribute information was used with ILP (Todorovski & Džeroski, 1999) and CBR (Hilario & Kalousis, 2001) approaches, but no conclusive results were achieved. A different type of data characterization is landmarking (Bensusan & Giraud-Carrier, 2000; Pfahringer, Bensusan, & Giraud-Carrier, 2000), which can be related to earlier work on yardsticks (Brazdil, Gama, & Henery, 1994). Landmarks are quick estimates of algorithm performance on a given dataset obtained using simple versions of the algorithms (Bensusan & Giraud-Carrier, 2000; Pfahringer, Bensusan, & Giraud-Carrier, 2000) or by sampling from the dataset (Fürnkranz & Petrak, 2001; Soares, Petrak, & Brazdil, 2001b).

6.9. Multicriteria evaluation

We believe that there are situations where the compromise between accuracy and time can be stated in the form of a percentage of accuracy the user is willing to trade for a certain speedup. The ARR measure fits those situations. It is important to extend ARR to include other performance criteria. However, some of these measures, e.g., novelty or understandability, are highly subjective and others, e.g., complexity, are hard to compare across different algorithms. Relatively little work has been dedicated to this issue (e.g., Nakhaeizadeh & Schnabl (1998)) without widespread use of the resulting measures, so we opted to concentrate on criteria which are commonly used. Nakhaeizadeh & Schnabl (1998) adapted DEA for this purpose where user preferences are stated in the form "criterion *A* is 50 times more important than criterion *B*." This is not as user-friendly as ARR because it implicitly assumes that criteria are measured in directly comparable units.

6.10. Ranking evaluation

Spearman's correlation coefficient (or alternative rank correlation coefficients like Kendall's tau (Neave & Worthington, 1992)) represents an adequate measure of the agreement between rankings of algorithms, but it does not distinguish between individual ranks. In practice, however, it seems that swapping, say, the 5th and 6th algorithms, is less important than swapping the first two. We could adopt the *weighted correlation* coefficient, discussed in (Soares,Costa, & Brazdil, 2001a), to solve this problem.

Although we have shown the advantages of representing the ideal ranking as N orderings, other possibilities could also be considered (Soares, Brazdil, & Costa, 2000). One such possibility is to use a partial order representation, based on pairwise comparisons between the algorithms. We should note that our ideal rankings are based on CV, which is not a perfect estimation method although it serves well for practical purposes (Schaffer, 1993).

6.11. Ranking reduction

With the growing number of algorithms, some of alternatives presented in a ranking may be redundant. Brazdil, Soares, & Pereira (2001) present a reduction method that eliminates certain items in the ranking. Evidence is presented that this is useful in general, and leads to time savings.

7. Conclusions

We have presented a meta-learning method to support the selection of learning algorithms that uses the k-Nearest Neighbor algorithm to identify the datasets that are most similar to the one at hand. The performance of the candidate algorithms on those datasets is used to generate a ranking that is provided to the user. The distance between datasets is based on a small set of data characteristics that represent a set of properties that affect the performance of the learning algorithms. Although we concentrated on classification algorithms only, this methodology can provide assistance in the selection of combinations of methods or more complex strategies.

The performance of algorithms is assessed using the Adjusted Ratio of Ratios (ARR), a multicriteria evaluation measure that takes accuracy and time into account. Other criteria, e.g., interpretability and complexity, could also be included in the future.

As it is not yet a general practice in ML/KDD to work with rankings, we had to identify and adapt existing statistical techniques to devise an appropriate evaluation methodology. This enabled us to show that ARR with k-NN leads to significantly better rankings in general than the baseline ranking method. The evaluation methodology is general and can be used in other ranking problems.

In summary, our contributions are (1) exploiting rankings rather than classification or regression, showing that is possible to adapt the IBL approach for that task, (2) providing an evaluation methodology for ranking, (3) providing a multicriteria evaluation measure that combines success rate and time, (4) identified a set of data characteristics that seem to be good predictors of relative performance of algorithms and (5) providing a ground work for ranking alternative solution strategies in a multistrategy system.

Appendix: The data mining advisor

The method for algorithm recommendation presented in this paper was developed as part of the METAL Esprit project (METAL Consortium, 2002). This method is incorporated in the publicly available Data Mining Advisor, at `www.metal-kdd.org`. We would like to thank

Dietrich Wettschereck, Stefan Müller and Adam Woznica for their contributions to this site.

Acknowledgments

We would like to thank anonymous referees of this paper for their constructive comments. Thanks also to all the METAL partners for a fruitful working atmosphere, in particular to Johann Petrak for providing the scripts to obtain the meta-data and to Jörg Keller, Iain Paterson, Helmut Berrer and Christian Köpf for useful exchange of ideas. We also thank DaimlerChrysler and Guido Lindner for providing us the data characterization tool. Finally, we thank Rui Pereira for implementing a part of the methods and his useful contributions. The financial support from ESPRIT project METAL, project ECO under PRAXIS XXI, FEDER, Programa de Financiamento Plurianual de Unidades de I&D and from the Faculty of Economics is gratefully acknowledged.

Notes

1. Given that this paper focus on classification tasks, we will use the term "success rate" and "accuracy" interchangeably.
2. We have no corresponding meta-attribute for symbolic attributes because none was readily available.
3. We represent higher ranks with smaller integers.
4. Not all experiments were executed on the same machine and so, a time normalization mechanism was employed.
5. Some preparation was necessary in some cases, so some of the datasets were not exactly the same as the ones used in other experimental work.
6. `research.swisslife.ch/kdd-sisyphus`.
7. The same reasoning is applied in the ideal rankings and when more than two algorithms are tied.
8. The significance level is (1—confidence level).
9. A few methods have been proposed to speed-up cross-validation (e.g. racing (Maron & Moore, 1994)). However, these methods achieve much smaller gains in time when compared to our approach and, thus, were not considered here.
10. We used the *Data Characterization Tool* (Lindner & Studer, 1999) that computes many measures other than the ones we used. This means that the value presented is an upper bound on the time that is really necessary to obtain them.
11. In one of the datasets we observed a tie between 1-NN and the baseline.

References

Aha, D. (1992). Generalizing from case studies: A case study. In D. Sleeman & P. Edwards (Eds.), *Proceedings of the Ninth International Workshop on Machine Learning (ML92)* (pp. 1–10). San Mateo, CA: Morgan Kaufmann.

Atkeson, C. G., Moore, A. W., & Schaal, S. (1997). *Locally Weighted Learning* (Vol. 11) (pp. 11–74). Boston: Kluwer.

Bensusan, H., & Giraud-Carrier, C. (2000). If you see la sagrada familia, you know where you are: Landmarking the learner space. Technical report, Department of Computer Science, University of Bristol.

Bensusan, H., & Kalousis, A. (2001). Estimating the predictive accuracy of a classifier. In P. Flach & L. de Raedt (Eds.), *Proceedings of the 12th European Conference on Machine Learning* (pp. 25–36). New York: Springer.

Bernstein, A., & Provost, F. (2001). An intelligent assistant for the knowledge discovery process. In W. Hsu, H. Kargupta, H. Liu, & N. Street (Eds.), *Proceedings of the IJCAI-01 Workshop on Wrappers for Performance Enhancement in KDD*.

Berrer, H., Paterson, I., & Keller, J. (2000). Evaluation of machine-learning algorithm ranking advisors. In P. Brazdil & A. Jorge (Eds.), *Proceedings of the PKDD2000 Workshop on Data Mining, Decision Support, Meta-Learning and ILP: Forum for Practical Problem Presentation and Prospective Solutions* (pp. 1–13).

Blake, C., Keogh, E., & Merz, C. (1998). Repository of machine learning databases. Available at http:/www.ics.uci.edu/∼mlearn/MLRepository.html

Brachman, R., & Anand, T. (1996). The process of knowledge discovery in databases. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, & R. Uthurusamy (Eds.), *Advances in Knowledge Discovery and Data Mining,* ch. 2 (pp. 37–57). AAAI Press/The MIT Press.

Brazdil, P., Gama, J., & Henery, B. (1994). Characterizing the applicability of classification algorithms using meta-level learning. In F. Bergadano & L. de Raedt (Eds.), *Proceedings of the European Conference on Machine Learning (ECML-94)* (pp. 83–102). Berlin: Springer-Verlag.

Brazdil, P., & Soares, C. (2000). A comparison of ranking methods for classification algorithm selection. In R. de Mántaras & E. Plaza (Eds.), *Machine Learning: Proceedings of the 11th European Conference on Machine Learning ECML2000* (pp. 63–74). Berlin: Springer.

Brazdil, P., Soares, C., & Pereira, R. (2001). Reducing rankings of classifiers by eliminating redundant cases. In P. Brazdil & A. Jorge (Eds.), *Proceedings of the 10th Portuguese Conference on Artificial Intelligence (EPIA 2001)*. New York: Springer.

Brodley, C. (1993). Addressing the selective superiority problem: Automatic algorithm/model class selection. In P. Utgoff (Ed.), *Proceedings of the Tenth International Conference on Machine Learning* (pp. 17–24). San Mateo, CA: Morgan Kaufmann.

Charnes, A., Cooper, W., & Rhodes, E. (1978). Measuring the efficiency of decision making units. *European Journal of Operational Research, 2*, 429–444.

Cohen, W. (1995). Fast effective rule induction. In A. Prieditis & S. Russell (Eds.), *Proceedings of the 11th International Conference on Machine Learning* (pp. 115–123). San Mateo, CA: Morgan Kaufmann.

Fürnkranz, J., & Petrak, J. (2001). An evaluation of landmarking variants. In C. Giraud-Carrier, N. Lavrac, & S. Moyle (Eds.), *Working Notes of the ECML/PKDD 2000 Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning* (pp. 57–68).

Gama, J. (1997). Probabilistic linear tree. In D. Fisher (Ed.), *Proceedings of the 14th International Machine Learning Conference (ICML97)* (pp. 134–142). San Mateo, CA: Morgan Kaufmann.

Gama, J., & Brazdil, P. (1995). Characterization of classification algorithms. In C. Pinto-Ferreira & N. Mamede (Eds.), *Progress in Artificial Intelligence* (pp. 189–200). Berlin: Springer-Verlag.

Henery, R. (1994). Methods for comparison. In D. Michie, D. Spiegelhalter, & C. Taylor (Eds.), *Machine Learning, Neural and Statistical Classification,* ch. 7 (pp. 107–124). Ellis Horwood.

Hilario, M., & Kalousis, A. (1999). Building algorithm profiles for prior model selection in knowledge discovery systems. In *Proceedings of the IEEE SMC'99 International Conference on Systems, Man and Cybernetics*. New York: IEEE Press.

Hilario, M., & Kalousis, A. (2001). Fusion of meta-knowledge and meta-data for case-based model selection. In A. Siebes & L. de Raedt (Eds.), *Proceedings of the Fifth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD01)*. New York: Springer.

Hochreiter, S., Younger, A., & Conwell, P. (2001). Learning to learn using gradient descent. In G. Dorffner, H. Bischof, & K. Hornik (Eds.), *Lecture Notes on Comp. Sci. 2130, Proc. Intl. Conf. On Artificial Neural Networks (ICANN-2001)* (pp. 87–94). New York: Springer.

Kalousis, A., & Hilario, M. (2000). A comparison of inducer selection via instance-based and boosted decision tree meta-learning. In R. Michalski & P. Brazdil (Eds.), *Proceedings of the Fifth International Workshop on Multistrategy Learning* (pp. 233–247).

Kalousis, A., & Theoharis, T. (1999). NOEMON: Design, implementation and performance results of an intelligent assistant for classifier selection. *Intelligent Data Analysis, 3:5*, 319–337.

Keller, J., Paterson, I., & Berrer, H. (2000). An integrated concept for multi-criteria ranking of data-mining algorithms. In J. Keller & C. Giraud-Carrier (Eds.), *Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination*.

Kohavi, R., John, G., Long, R., Mangley, D., & Pfleger, K. (1994). MLC++: A machine learning library in c++. Technical report, Stanford University.

Lindner, G., & Studer, R. (1999). AST: Support for algorithm selection with a CBR approach. In C. Giraud-Carrier & B. Pfahringer (Eds.), *Recent Advances in Meta-Learning and Future Work* (pp. 38–47). J. Stefan Institute. Available at http://ftp.cs.bris.ac.uk/cgc/ICML99/lindner.ps.Z

Maron, O., & Moore, A. (1994). Hoeffding races: Accelerating model selection search for classification and function approximation. In J. Cowan, G. Tesauro, & J. Alspector (Eds.), *Advances in Neural Information Processing Systems* (pp. 59–66). San Mateo, CA: Morgan Kaufmann.

METAL Consortium (2002). Esprit project METAL (#26.357). Available at www.metal-kdd.org.

Michie, D., Spiegelhalter, D., & Taylor, C. (1994). *Machine Learning, Neural and Statistical Classification*. Ellis Horwood.

Mitchell, T. (1997). *Machine Learning*. New York: McGraw-Hill.

Nakhaeizadeh, G., & Schnabl, A. (1997). Towards the personalization of algorithms evaluation in data mining. In R. Agrawal & P. Stolorz (Eds.), *Proceedings of the Third International Conference on Knowledge Discovery & Data Mining* (pp. 289–293). AAAI Press.

Nakhaeizadeh, G., & Schnabl, A. (1998). Development of multi-criteria metrics for evaluation of data mining algorithms. In D. Heckerman, H. Mannila, D. Pregibon, & R. Uthurusamy (Eds.), *Proceedings of the Fourth International Conference on Knowledge Discovery in Databases & Data Mining* (pp. 37–42). AAAI Press.

Neave, H., & Worthington, P. (1992). *Distribution-Free Tests*. London: Routledge.

Pfahringer, B., Bensusan, H., & Giraud-Carrier, C. (2000). Tell me who can learn you and i can tell you who you are: Landmarking various learning algorithms. In P. Langley (Ed.), *Proceedings of the Seventeenth International Conference on Machine Learning (ICML2000)* (pp. 743–750). San Mateo, CA: Morgan Kaufmann.

Quinlan, R. (1998). *C5.0: An Informal Tutorial*. RuleQuest. Available at http://www.rulequest.com/see5-unix.html.

Ripley, B. (1996). *Pattern Recognition and Neural Networks*. Cambridge.

Schaffer, C. (1993). Selecting a classification method by cross-validation. *Machine Learning, 13:1*, 135–143.

Schmidhuber, J., Zhao, J., & Schraudolph, N. (1997). *Reinforcement Learning With Self-Modifying Policies* (pp. 293–309). Boston: Kluwer.

Soares, C., & Brazdil, P. (2000). Zoomed ranking: Selection of classification algorithms based on relevant performance information. In D. Zighed, J. Komorowski, & J. Zytkow (Eds.), *Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD2000)* (pp. 126–135). New York: Springer.

Soares, C., Brazdil, P., & Costa, J. (2000). Measures to compare rankings of classification algorithms. In H. Kiers, J.-P. Rasson, P. Groenen, & M. Schader (Eds.), *Data Analysis, Classification and Related Methods, Proceedings of the Seventh Conference of the International Federation of Classification Societies IFCS* (pp. 119–124). New York: Springer.

Soares, C., Costa, J., & Brazdil, P. (2001a). Improved statistical support for matchmaking: Rank correlation taking rank importance into account. In *JOCLAD 2001: VII Jornadas de Classificação e Análise de Dados* (pp. 72–75).

Soares, C., Petrak, J., & Brazdil, P. (2001b). Sampling-based relative landmarks: Systematically test-driving algorithms before choosing. In P. Brazdil & A. Jorge (Eds.), *Proceedings of the 10th Portuguese Conference on Artificial Intelligence (EPIA 2001)* (pp. 88–94). New York: Springer.

Sohn, S. (1999). Meta analysis of classification algorithms for pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 21:11*, 1137–1144.

Suyama, A., Negishi, N., & Yamaguchi, T. (1999). CAMLET: A platform for automatic composition of inductive applications using ontologies. In C. Giraud-Carrier & B. Pfahringer (Eds.), *Proceedings of the ICML-99 Workshop on Recent Advances in Meta-Learning and Future Work* (pp. 59–65).

Todorovski, L., Brazdil, P., & Soares, C. (2000). Report on the experiments with feature selection in meta-level learning. In P. Brazdil & A. Jorge (Eds.), *Proceedings of the Data Mining, Decision Support, Meta-Learning and ILP Workshop at PKDD2000* (pp. 27–39).

Todorovski, L., & Džeroski, S. (1999). Experiments in meta-level learning with ILP. In J. Rauch & J. Zytkow (Eds.), *Proceedings of the Third European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD99)* (pp. 98–106). New York: Springer.

Todorovski, L., & Džeroski, S. (2000). Combining multiple models with meta decision trees. In D. Zighed, J. Komorowski, & J. Zytkow (Eds.), *Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD00)* (pp. 54–64). New York: Springer.

Todorovski, L., & Džeroski, S. (2003). Combining classifiers with meta-decision trees. *Machine Learning Journal*. *50:3* pp. 223–249.

Vilalta, R. (1999). Understanding accuracy performance through concept characterization and algorithm analysis. In C. Giraud-Carrier & B. Pfahringer (Eds.), *Recent Advances in Meta-Learning and Future Work* (pp. 3–9). J. Stefan Institute.

Wolpert, D., & Macready, W. (1996). No free lunch theorems for search. Technical Report SFI-TR-95-02-010, The Santa Fe Institute. Available at http://lucy.ipk.fhg.de:80/~stephan/nfl/nfl.ps

Received October 20, 2000
Revised November 28, 2001
Accepted June 2, 2002
Final manuscript July 30, 2002