Effect of Incomplete Meta-dataset on Average Ranking Method

Salisu Mamman Abdulrahman SALISU.ABDUL@GMAIL.COM
LIAAD - INESC TEC/Faculdade de Ciências da Universidade do Porto

Pavel Brazdil PBRAZDIL@INESCPORTO.PT
LIAAD - INESC TEC/Faculdade de Economia, Universidade do Porto

Abstract

One of the simplest metalearning methods is the average ranking method. This method uses metadata in the form of test results of a given set of algorithms on a given set of datasets and calculates an average rank for each algorithm. The ranks are used to construct the average ranking. We investigate the problem of how the process of generating the average ranking is affected by incomplete metadata involving fewer test results. This is relevant, as such situations are often encountered in practice. In this paper we describe a relatively simple average ranking method that is capable of dealing with incomplete metadata. Our results show that the proposed method is relatively robust to omissions in the test results. This finding could be of use in a future design of experiments. As the incomplete metadata does not affect the final results much, we can simply conduct fewer tests and thus save computation time.

Keywords: Average Ranking, Aggregation of Rankings, Incomplete Metadata

1. Introduction

A large number of data mining algorithms exist, rooted in the fields of machine learning, statistics, pattern recognition and artificial intelligence. The task to recommend the most suitable algorithm(s) has thus become rather challenging. The algorithm selection problem, originally described by Rice (1976), has attracted a great deal of attention, as it endeavours to select and apply the best or near best algorithm(s) for a given task (Brazdil et al. (2008); Smith-Miles (2008)). We address the problem of robustness of one particular version of an average ranking method that uses incomplete rankings as input. These arise if we have incomplete test results in the meta-dataset. We have investigated how much the performance degrades in such circumstances.

The remainder of this paper is organized as follows. In the next section we present an overview of the existing work in related areas. Section 3 provides details about the proposed aggregation method for incomplete metadata, the experimental results and future work.

2. Related Work

In this paper we are addressing a particular case of the algorithm selection problem, oriented towards the selection of classification algorithms. Various researchers addressed this problem over the last 25 years. One approach to algorithm selection/recommendation relies on metalearning. The simplest method uses just performance results on different datasets in the form of rankings. Some commonly used measures of performance are accuracy, AUC or A3R that combines accuracy and runtime Abdulrahman and Brazdil (2014).

The rankings are then aggregated to obtain a single aggregated ranking. The aggregated ranking can be used as a simple model to identify the top algorithms to be used. This strategy is sometimes referred to as the *Top-N* strategy (Brazdil et al. (2008)).

A more advanced approach often considered as the *classical metalearning approach* uses, in addition to performance results, a set of measures that characterize datasets (Pfahringer et al. (2000); Brazdil et al. (2008); Smith-Miles (2008)). Other approaches exploit estimates of performance based on past tests in so-called active testing method for algorithm selection Leite et al. (2012). Hutter et al. (2011) used so-called *surrogate models* to suggest the next test to carry out in the hyper-parameter optimization task.

Aggregation of *complete rankings* is a simple matter. Normally it just involves calculating the average rank for all items in the ranking (Lin, 2010). Complete rankings are those in which k items are ranked N times and no value in this set is missing. Incomplete rankings arise when only some ranks are known in some of the rankings. These arise quite often in practice. Many diverse methods exist for the aggregation of incomplete rankings. According to Lin (2010), methods applicable to long lists can be divided into three categories: Heuristic algorithms, Markov chain methods and stochastic optimization methods. The last category includes, for instance, the *Cross Entropy Monte Carlo, CEMC* method.

Some of the approaches require that the elements that do not appear in list L_i of k elements be attributed a concrete rank (e.g. $k+1$). This does not seem to be correct. We should not be forced to assume that some information exists, if in fact we have none. We have considered using the R package *RankAggreg* (Pihur et al. (2014)), but unfortunately we would have to attribute a concrete rank (e.g. $k+1$) to all missing elements. We have therefore developed a simple heuristic method based on Borda's method described in Lin (2010).

As Lin (2010) pointed out simple methods often compare quite favourably to other more complex approaches.

3. Effect of Incomplete Meta-Data on Average Ranking Method

Our aim is to investigate the issue of how the generation of the average ranking is affected by incomplete test results in the meta-data available. Although other measures, such as AUC, could have been used instead, here we focus on the *ranking* obtained on the basis of a combined measure of accuracy and runtime (Abdulrahman and Brazdil (2014)):

$$A3R^{d_i}_{a_{ref},a_j} = \frac{\frac{SR^{d_i}_{a_j}}{SR^{d_i}_{a_{ref}}}}{\sqrt[N]{T^{d_i}_{a_j}/T^{d_i}_{a_{ref}}}} \tag{1}$$

Here, $SR^{d_i}_{a_j}$ and $SR^{d_i}_{a_{ref}}$ represent the *success rates* (accuracies) of algorithms a_j and a_{ref} on dataset d_i, where a_{ref} represents a given *reference algorithm*. Similarly, $T^{d_i}_{a_j}$ and $T^{d_i}_{a_{ref}}$ represent the runtime of the algorithms, in seconds. To trade off the importance of runtime and accuracy, A3R includes the N^{th} root parameter. This is motivated by the observation that runtime vary much more than accuracies. It is not uncommon that one particular algorithm is three orders of magnitude slower (or faster) than another one. Obviously, we do not want the time ratios to completely dominate the equation. If we take the N^{th} root of the runtime, we will get a number that goes to 1 in the limit.

We wish to see how robust the method is to omissions in the meta-data. This issue is relevant because meta-data gathered by researchers are very often incomplete. Here we consider two different ways in which the meta-data can be incomplete: First, the test results on some datasets may be completely missing. Second, there may be a certain proportion of omissions in the test results of some algorithms on each dataset.

The expectation is that the performance of the average ranking method would degrade when less information is available. However, an interesting question is how grave the degradation is. The answer to this issue is not straightforward, as it depends greatly on how diverse the datasets are and how this affects the rankings of algorithms. If the rankings are very similar, then we expect that the omissions would not make much difference. So the issue of the effects of omissions needs to be relativized. To do this we will investigate the following issues:

- Effects of missing test results on X% of datasets (alternative MTD);
- Effects of missing X% of test results of algorithms on each dataset (alternative MTA).

If the performance drop of alternative MTA were not too different from the drop of alternative MTD, then we could conclude that X% of omissions is not unduly degrading the performance and hence the method of average ranking is relatively robust. Each of these alternatives is discussed in more detail below.

Missing all test results on some datasets (alternative MTD): This strategy involves randomly omitting all test results on a given proportion of datasets from our meta-dataset. An example of this scenario is depicted in Table 1. In this example the test results on datasets D_2 and D_5 are completely missing. The aim is to show how much the average ranking degrades due to these missing results.

Table 1: Missing test results on a certain percentage of datasets (MTD)

Algorithms	D_1	D_2	D_3	D_4	D_5	D_6
a_1	0.85		0.77	0.98		0.82
a_2	0.95		0.67	0.68		0.72
a_3	0.63		0.55	0.89		0.46
a_4	0.45		0.34	0.58		0.63
a_5	0.78		0.61	0.34		0.97
a_6	0.67		0.70	0.89		0.22

Missing some algorithm test results on each dataset (alternative MTA): Here the aim is to drop a certain proportion of test results on each dataset. The omissions are simply distributed uniformly across all datasets. That is, the probability that the test result of algorithm a_i is missing is the same irrespective of which algorithm is chosen. An example of this scenario is depicted in Table 2. The proportion of test results on datasets/algorithms omitted is a parameter of the method. Here we use the values shown in Table 3. The metadata used in this study is described further in Section 3.2. This dataset was used to obtain a new one in which the test results of some datasets chosen at random, would be omitted. The resulting dataset was used to construct the average ranking. Each ranking was then used to construct a *loss-time curve* described in Section 3.2. The whole process was repeated 10 times. This way we would obtain 10 loss-time curves, which would be aggregated into a single loss-time curve. Our aim is to upgrade the average ranking method to be able to deal with incomplete rankings. The enhanced method (AR-MTA-H) is described in the next section. It can be characterized as a heuristic method of aggregation of incomplete rankings that uses weights of ranks. Later it is compared to the classical approach (AR-MTA-B), that serves as a baseline here. This method is based on the original Borda method described by Lin (2010) and is commonly used by many researchers.

Table 2: Missing test results on a certain percentage of algorithms (MTA)

Algorithms	D_1	D_2	D_3	D_4	D_5	D_6
a_1	0.85	0.77	▨	0.98	▨	0.82
a_2	▨	055	0.67	0.68	0.66	▨
a_3	0.63	▨	0.55	0.89	▨	0.46
a_4	0.45	0.52	0.34	▨	0.44	0.63
a_5	0.78	0.87	0.61	0.34	0.42	▨
a_6	▨	0.99	▨	0.89	▨	0.22

Table 3: Percentages of omissions and the numbers of datasets and algorithms used

Omissions %	0	5	10	20	50	90	95
No of datasets used in MTD	38	36	34	30	19	4	2
No of tests per dataset in MTA	53	50	48	43	26	5	3

3.1. Aggregation Method for Incomplete Rankings (AR-MTA-H)

Before describing the method, let us consider a motivating example (see Table 4), illustrating why we cannot simply use the usual average ranking method (Lin (2010)), often used in comparative studies in machine learning literature. Let us compare the rankings R_1 (Table 4a) and R_2 (Table 4b). We note that algorithm a_2 is ranked 4th in ranking R_1, but has rank 1 in ranking R_2. If we used the usual method, the final ranking of a_2 would be the mean of the two ranks, i.e. (4+1)/2=2.5. This seems intuitively wrong, as the information in ranking R_2 is incomplete. If we carry out just one test and obtain ranking R_2 as a result, this information is obviously inferior to ranking that include results of more test(e.g R_1) leading to ranking R_1. This suggests that the number of tests should be taken into account to set the weight of the individual elements of the ranking.

Table 4: An example of two rankings R_1 and R_2 and the aggregated ranking R^A

(a)

R_1	Rank
a_1	1
a_3	2
a_4	3
a_2	4
a_6	5
a_5	6

(b)

R_2	Rank
a_2	1
a_1	2

(c)

R^A	Rank	Weight
a_1	1.67	1.2
a_3	2	1
a_4	3	1
a_2	3.5	1.2
a_6	5	1
a_5	6	1

In our method the weight is calculated using the expression $(N-1)/(N_{max}-1)$, where N represents the number of filled in elements in the ranking and N_{max} the maximum number of elements that could be filled in. So, for instance, in the ranking R_1, $N = 6$ and $N_{max} = 6$. Therefore, the weight of each element in the ranking is $5/5 = 1$. We note that $N - 1$ (i.e. 5), represents the number of non-transitive relations in the ranking, namely $a_1 > a_3$, $a_3 > a_4$, .. , $a_6 > a_5$. The transitive relations are not considered. Here $a_i > a_j$ is used to indicate that a_i is preferred to a_j.

Let us consider the incomplete ranking R_2. Suppose we know a priori that the ranking could include 6 elements and so $N_{max} = 6$, as in the previous case. Then the weight of each element will be $(N - 1)/(N_{max} - 1) = 1/5 = 0.2$. The notion of *weight* captures the fact that ranking R_2 provides less information than ranking R_1. We need this concept in the process of calculating the average ranking.

Our upgraded version of the aggregation method for incomplete rankings involves initializing the average ranking R^A with the first ranking. Then in each subsequent step, another ranking is read and aggregated with the average ranking, producing an updated average ranking. The aggregation iterates over the elements in the ranking. If the element appears in both the aggregated ranking and the new ranking, its rank is recalculated as a weighted average of the two ranks:

$$r_i^A := r_i^A * w_i^A/(w_i^A + w_i^j) + r_i^j * w_i^j/(w_i^A + w_i^j) \tag{2}$$

where r_i^A represents the rank of element i in the aggregated ranking and r_i^j the rank of element i in the ranking j and w_i^A and w_i^j represent the corresponding weights. The weight is updated as follows:

$$w_i^A := w_i^A + w_i^j \tag{3}$$

If the element appears in the aggregated ranking, but not in the new ranking, both the rank and the weight are kept unchanged.

Suppose the aim is to aggregate the rankings R_1 and R_2 shown before. The new rank of a_2 will be r_2^A = 4 * 1/1.2 + 1*0.2/1.2 = 3.5. The weight will be w_2^A = 1 + 0.2 = 1.2. The final aggregated ranking of rankings R_1 and R_2 is shown in Table 4c.

3.2. The Effects of Omissions in the Meta-data

Meta dataset used: The data used in the experiments involves the meta-dataset constructed from evaluation results retrieved from OpenML (Vanschoren et al. (2014)), a collaborative science platform for machine learning. This dataset contains the results of 53 parameterized classification algorithms from the Weka workbench (Hall et al. (2009)) on 39 datasets[1]. In leave-one-out cross-validation, 38 datasets are used to generate the model (e.g. average ranking), while the dataset left out is used for evaluation.

Using leave-one-out cross-validation helps to gain confidence that the average ranking can be effectively transferred to new datasets and produce satisfactory outcomes.

Characterization of the meta-dataset: We were interested to analyze different rankings of classification algorithms on different datasets used in this work and, in particular, how these differ for different pairs of datasets. If two datasets are very similar, the algorithm rankings will also be similar and consequently the correlation coefficient will be near 1. On the other hand, if the two datasets are quite different, the correlation will be low. In the extreme case, the correlation coefficient will be -1 (i.e. one ranking is the inverse of the other). The distribution of pairwise correlation coefficients provides an estimate of how difficult the meta-learning task is. Fig.1 shows a histogram of correlation values. The histogram is accompanied by the *expected value*, *standard deviation* and *coefficient of variation* calculated as the ratio of standard deviation to the expected value (mean) (Witten and Frank (2005)). These measures are shown in Table 5.

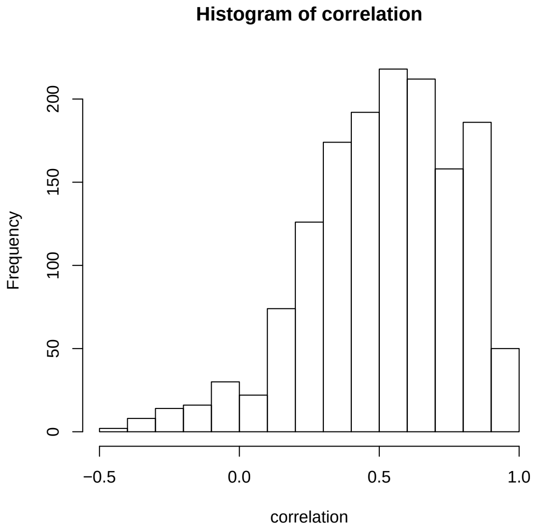

Figure 1: Spearman correlation coefficient between rankings of pairs of datasets.

Evaluation using loss-time curves: Our aim was to investigate how certain omissions in the meta-datasets affect the performance. The results are presented in the form of loss-time curves (van Rijn et al. (2015)) which show how the *performance loss* depends on

[1] Full details: http://www.openml.org/project/tag/ActiveTestingSamples/u/1

Table 5: Measures characterizing the histogram of correlations in Fig. 1

Measure %	Expected Value	Standard Deviation	Coefficient of Variation
Value	0.5134	0.2663	51.86%

time. The *loss* is calculated as the difference between the performance of the best algorithm identified using the ranking to the ideal choice. Each loss-time curve can be characterized by a number representing the *mean loss* in a given interval. This characteristic is similar to AUC, but there is an important difference. When talking about AUCs, the values fall in the 0-1 interval. Our loss-time curves span the interval T_{min} - T_{max} which is specified by the user. Typically the user only worries about run times when they exceed a minimum value. In the experiments here we have set T_{min} to 10 seconds. The value of T_{max} was set to 10^4 seconds, i.e. about 2.78 hours.

Variants the of ranking methods used: Ranking methods use a particular performance measure to construct an ordering of algorithms. Some commonly used measures of performance are accuracy, AUC or A3R that combines accuracy and runtime (Abdulrahman and Brazdil, 2014). In this study here we have opted for A3R, because it leads to good results when loss time curves are used in the evaluation. All the ranking variants used in the experiments described here use A3R.

Results: Table 6 presents the results for the alternatives *AR-MTD*, *AR-MTA-H* and *AR-MTA-B* in terms of mean interval loss (MIL). All loss-time curves start from the initial loss of the default classification. This loss is calculated as the difference in performance between the best algorithm and the default accuracy for each dataset. The default accuracy is calculated in the usual way, by simply predicting the majority class for the dataset in question. The values for the ordinary average ranking method, *AR-MTA-B*, are also shown, as this method serves as a baseline. Fig. 2 shows the loss-time curves for the three alternatives when the number of omissions is 90%. Not all loss-time curves are shown, as the figure would be rather cluttered.

Table 6: Mean interval loss (MIL) values for different percentages of omissions

Omission% Method	0	5	10	20	50	90	95
AR-MTD	0.531	0.535	0.535	0.536	0.550	1.175	1.633
AR-MTA-H	0.531	0.534	0.537	0.542	0.590	1.665	2.042
AR-MTA-B	0.531	0.536	0.537	0.544	0.593	2.970	3.402
AR-MTA-H/AR-MTD	1.00	1.00	1.00	1.01	1.07	1.42	1.25

Our results show that the proposed average ranking method *AR-MTA-H* achieves better results than the baseline method *AR-MTA-B*. We note also that although the proposed method *AR-MTA-H* achieves comparable results to *AR-MTD* for many values of the percentage drop (all values up to about 50%) and only when we get to rather extreme values, such as 90%, the difference is noticeable. Still the differences between our proposed variant *AR-MTA-H* and *AR-MTD* are smaller than the differences between *AR-MTA-B* and *AR-*

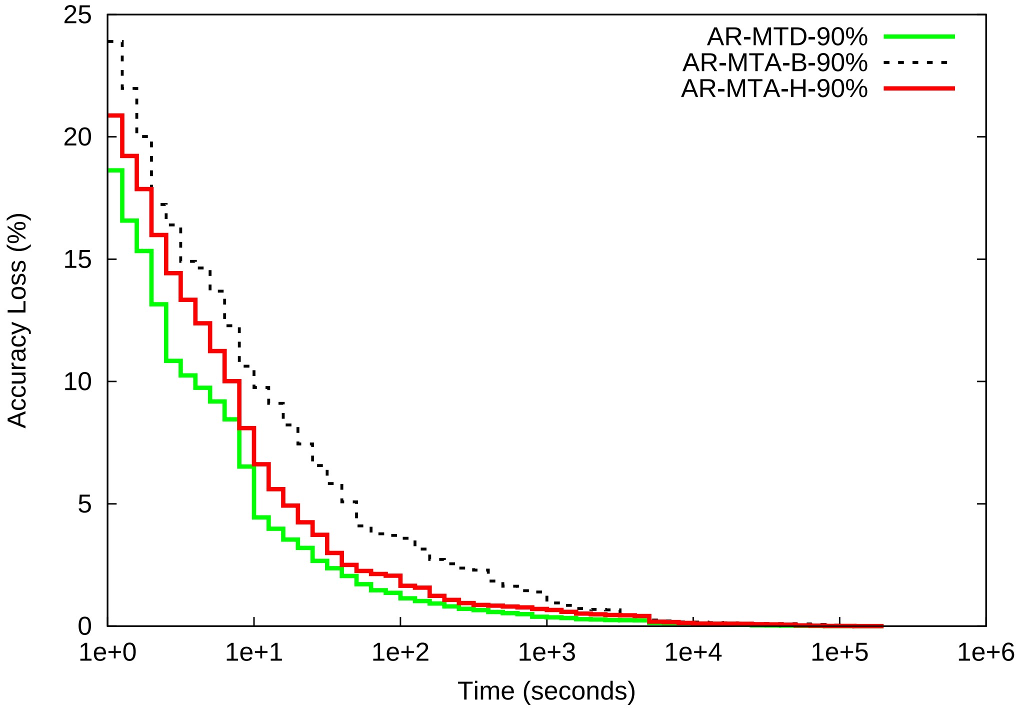

Figure 2: Comparison of *AR-MTA-H* with *AR-MTD* and the baseline method *AR-MTA-B* for 90% of omissions.

MTD. These results indicate that the proposed average ranking method is relatively robust to omissions.

4. Conclusion:

In this paper we have investigated the problem of how the process of generating the average ranking is affected by incomplete test results. We have shown that a percentage drop of up to 50% does not make much difference. This can be exploited in the future - we can simply opt for fewer tests when gathering metadata.

We have described a relatively simple method, *AR-MTA-H*, that permits to aggregate incomplete rankings. We have also proposed a methodology that is useful in the process of evaluating our aggregation method. This involves using a standard aggregation method *AR-MTD* on a set of complete rankings, but whose number is decreased following the proportion of omissions in the incomplete rankings. As we have shown, the proposed aggregation method achieves quite comparable results and is relatively robust to omissions in test results in the test data.

Future work: As the incomplete meta-dataset does not affect much the final ranking and the corresponding loss, this could be explored in future design of experiments, when gathering new test results. We could investigate approaches that permit to consider also the costs (runtime) of off-line tests. Their cost (runtime) could be set to some fraction of the cost of on-line test (i.e. tests on a new dataset), but not really ignored altogether.

Acknowledgements
This work is supported by TETFund 2012 Intervention for Kano University of Science and Technology, Wudil, Kano State, Nigeria for PhD Overseas Training from the Federal Government of Nigeria. The authors also acknowledge the support of project *NanoSTIMA: Macro-to-Nano Human Sensing: Towards Integrated Multimodal Health Monitoring and Analytics/NORTE-01-0145-FEDER-000016*, which is financed by the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF).

References

S. M. Abdulrahman and P. Brazdil. Measures for Combining Accuracy and Time for Meta-learning. In *Meta-Learning and Algorithm Selection Workshop at ECAI 2014*, pages 49–50, 2014.

Pavel Brazdil, Christophe Giraud-Carrier, Carlos Soares, and Ricardo Vilalta. *Metalearning: Applications to data mining*. Springer Science & Business Media, 2008.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA Data Mining Software: An Update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. *International Conference on Learning and Intelligent Optimization*, pages 507–523, 2011.

R. Leite, P. Brazdil, and J. Vanschoren. Selecting Classification Algorithms with Active Testing. In *Machine Learning and Data Mining in Pattern Recognition*, pages 117–131. Springer, 2012.

S. Lin. Rank aggregation methods. *WIREs Computational Statistics*, 2:555–570, 2010.

B. Pfahringer, H. Bensusan, and Ch. Giraud-Carrier. Tell me who can learn you and I can tell you who you are: Landmarking various learning algorithms. In *Proc. of the 17th Int. Conf. on Machine Learning*, pages 743–750, 2000.

V. Pihur, S. Datta, and S. Susmita Datta. RankAggreg, an R package for weighted rank aggregation. Department of Bioinformatics and Biostatistics, University of Louisville, http://vpihur.com/biostat, 2014.

J. R. Rice. The Algorithm Selection Problem. *Advances in Computers*, 15:65–118, 1976.

K. A. Smith-Miles. Cross-disciplinary Perspectives on Meta-Learning for Algorithm Selection. *ACM Computing Surveys (CSUR)*, 41(1):6:1–6:25, 2008.

J. N. van Rijn, S. M. Abdulrahman, P. Brazdil, and J. Vanschoren. Fast Algorithm Selection using Learning Curves. In *Advances in Intelligent Data Analysis XIV*. Springer, 2015.

J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo. OpenML: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.