Ranking Algorithms by Performance

Lars Kotthoff

INSIGHT Centre for Data Analytics

1 Introduction

The Algorithm Selection Problem [8] is to select the most appropriate algorithm for solving a particular problem. It is especially relevant in the context of algorithm portfolios [2, 3], where a single solver is replaced with a set of solvers and a mechanism for selecting a subset to use on a particular problem. A common way of doing algorithm selection is to train a machine learning model and predict the best algorithm from a portfolio to solve a particular problem.

Several approaches in the literature, e.g. [4, 7], compute schedules for running the algorithms in the portfolio. Such schedules rely on a ranking of the algorithms that dictates when to run each algorithm and for how long. Despite this, no comparison of different ways of arriving at such a schedule has been performed to date. In this paper, we investigate how to predict a complete ranking of the portfolio algorithms on a particular problem. In machine learning, this is known as the label ranking problem. We evaluate a range of approaches to predict the ranking of a set of algorithms on a problem. We furthermore introduce a framework for categorizing ranking predictions that allows to judge the expressiveness of the predictive output. Our experimental evaluation demonstrates on a range of data sets from the literature that it is beneficial to consider the relationship between algorithms when predicting rankings.

While a complete ranking is not required to do algorithm selection, it can be beneficial. Predictions of algorithm performance will always have some degree of uncertainty associated with them. Being able to choose from among a ranked list of all portfolio algorithms can be used to mitigate the effect of this by selecting more than one algorithm.

2 Organizing predictions

We propose the following levels to categorise the predictive output of a model with respect to what ranking may be obtained from it.

Level 0 The prediction output is a single label of the best algorithm. It is not possible to construct a ranking from this and we do not consider it in this paper.

Level 1 The prediction output is a ranking of algorithms. The relative position of algorithms in the ranking gives no indication of the difference in performance.

Level 2 The prediction output is a ranking with associated scores. The difference between ranking scores is indicative of the difference in performance.

In the remainder of this paper, we will denote the framework $\mathcal{R}$ and level x within it $\mathcal{R}_x$. Higher levels strictly dominate the lower levels in the sense that their predictive output can be used to the same ends as the predictive output at the lower levels.

In the context of algorithm selection and portfolios, examples for the different levels are as follows. A $\mathcal{R}_0$ prediction is suitable for selecting a single algorithm. $\mathcal{R}_1$ allows to select the n best solvers for running in parallel on an n processor machine. $\mathcal{R}_2$ allows to compute a schedule where each algorithm is allocated resources according to its expected performance. Note that while it is possible to compute a schedule given just a ranking with no associated expected performances (i.e. $\mathcal{R}_1$), better-quality schedules can usually be obtained if some kind of performance score is predicted. The expected performance can be related directly to the time allocated the algorithm rather than allocating a fixed time that is oblivious of the expected performance.

2.1 Empirical evaluation

We evaluate the following ten ways of ranking algorithms, five from $\mathcal{R}_1$ and five from $\mathcal{R}_2$. The difference between some of these approaches lies in what kind of predictive models are learned from the same training data.

Order The ranking of the algorithms is predicted directly as a label. The label consists of a concatenation of the ranks of the algorithms. This approach is in $\mathcal{R}_1$. [6] use a conceptually similar approach to compute the ranking with a single prediction step.

Order score For each training example, the algorithms in the portfolio are ranked according to their performance. The rank of an algorithm is the quantity to predict. We used both regression and classification approaches. The ranking is derived directly from the predictions. These two approaches are in $\mathcal{R}_1$.

Faster than classification A classifier is trained to predict the ranking as a label similar to Order score given the predictions of which is faster for each pair of algorithms. This approach is in $\mathcal{R}_1$.

Faster than difference classification A classifier is trained to predict the ranking as a label given the predictions for the performance differences for each pair of algorithms. This approach is in $\mathcal{R}_1$.

Solve time The time to solve a problem is predicted and the ranking derived directly from this. In addition to predicting the time itself, we also predicted the log. These approaches are in $\mathcal{R}_2$. Numerous approaches predict the solve time to identify the best algorithm, for example [9].

Probability of being best The probability of being the best algorithm for a specific instance in a $[0, 1]$ interval is predicted. If an algorithm is the best on an instance, the probability should be 1, else 0. The ranking is derived directly from this. This approach is in $\mathcal{R}_2$.

Faster than majority vote The algorithms are ranked by the number of times they were predicted to be faster than another algorithm. This is the approach used to identify the best algorithm in recent versions of SATzilla [10]. This approach is in $\mathcal{R}_2$. While the individual predictions are simple labels (faster or not), the aggregation is able to provide fine-grained scores.

Faster than difference sum The algorithms are ranked by the sum over the predicted performance differences for each pair of algorithms. Algorithms that are often or by a lot faster will have a higher sum and rank higher. This approach is in $\mathcal{R}_2$.

Our evaluation uses four data sets taken from the literature. We use the SAT-HAN and SAT-IND SATzilla 2009 training data sets with 19 and 18 solvers, respectively. The third data set comes from the QBF solver evaluation 2010 with 5 solvers. Finally, we take the CSP data set from [1] with 2 solvers.

We use the Weka machine learning toolkit to train models and make predictions. We evaluated our approaches using the `AdaBoostM1 BayesNet`, `DecisionTable`, `IBk` with 1, 3, 5 and 10 neighbours, `J48`, `J48graft`, `JRip`, `LibSVM` with radial basis function kernel, `MultilayerPerceptron`, `OneR`, `PART`, `RandomForest`, `RandomTree`, `REPTree`, and `SimpleLogistic` algorithms for the approaches that use classification and the `AdditiveRegression`, `GaussianProcesses`, `LibSVM` with ε and ν kernels, `LinearRegression`, `M5P`, `M5Rules`, `REPTree`, and `SMOreg` algorithms for regression. We used the standard parameters in Weka.

Where several layers of machine learning algorithms are required, they are stacked as follows. The first layer is trained on the original training set with the features of the original problems. The prediction of the models of this first layer is used to train a model in a second layer that takes the predictions of the earlier layer as input. The output is the final prediction that we use to compute the ranking.

The performance of each approach on each data set is evaluated using stratified ten-fold cross-validation. We assess the quality of a predicted ranking by comparing it to the actual ranking (derived from the measured performance) using the Spearman correlation test.

3 Results and Conclusion

We present aggregate results in Table 1. For each instance, the Spearman rank correlation coefficient is computed between the predicted and the actual ranking. We show the median of the distribution of those coefficients for all data sets and rank prediction approaches. Only the values for the respective best machine learning model are shown. In addition to the scores for individual data sets, we show the sum over all data sets.

		CSP	QBF	SAT-HAN	SAT-IND	$\sum$
$\mathcal{R}_1$	Order	1	**1**	0.888	0.897	3.785
	Order score (classification)	1	0.4	0.823	0.759	2.981
	Order score (regression)	1	0.4	0.837	0.816	3.053
	Faster than classification	1	**1**	**0.891**	**0.899**	**3.79**
	Faster than difference classification	1	0.4	0.83	0.789	3.019
$\mathcal{R}_2$	Solve time	1	-0.15	0.453	0.424	1.727
	Solve time (log)	1	-0.1	0.791	0.752	2.444
	Probability of being best	1	0.1	0.114	0.352	1.566
	Faster than majority vote	1	0.8	0.888	0.878	3.566
	Faster than difference sum	1	0.1	0.472	0.43	2.002

Table 1. Median of the ranking quality scores for all data sets and rank prediction approaches for the respective best machine learning algorithm for a particular prediction approach. Higher scores are better. All numbers are rounded to three decimal places. The best value for each column is typeset in **bold**.

The overall best approach is the Faster than classification approach, followed by the Order approach. The Faster than majority vote, Order score (regression), and Faster than difference classification approaches exhibit good performance as well. The results clearly demonstrate that the relationship between the portfolio algorithms is important to take into account when predicting the ranking of algorithms. In general, the approaches that consider the algorithms only in isolation perform worse than the approaches that consider the portfolio as a whole or pairs of algorithms.

Overall, the approaches in $\mathcal{R}_1$ have better performance than those in $\mathcal{R}_2$. The likely reason for this is that the predictions in $\mathcal{R}_2$ are inherently more complex and there is more margin for error. The Faster than classification, Faster than majority vote and Order are the approaches that deliver the best overall performance. While some of these are complex and rely on layers of machine learning models, the Order approach is actually the simplest of those evaluated here. Its simplicity makes it easy to implement and an ideal starting point for researchers planning to predict rankings of algorithms. In addition to the approaches named above, predicting the order through a ranking score predicted by a regression algorithm achieved good performance.

This paper presented a first attempt at organising algorithm selection models with respect to how their predictive output can be used when computing rankings. We evaluated a number of different approaches and identified promising ones that deliver good performance in practice. An extended version that presents the results in more detail can be found in [5].

Acknowledgements Lars Kotthoff is supported by European Union FP7 grant 284715.

References

1. Gent, I.P., Jefferson, C., Kotthoff, L., Miguel, I., Moore, N., Nightingale, P., Petrie, K.: Learning when to use lazy learning in constraint solving. In: ECAI. pp. 873–878 (Aug 2010)
2. Gomes, C.P., Selman, B.: Algorithm portfolios. Artificial Intelligence 126(1-2), 43–62 (2001)
3. Huberman, B.A., Lukose, R.M., Hogg, T.: An economics approach to hard computational problems. Science 275(5296), 51–54 (1997)
4. Kadioglu, S., Malitsky, Y., Sabharwal, A., Samulowitz, H., Sellmann, M.: Algorithm selection and scheduling. In: 17th International Conference on Principles and Practice of Constraint Programming. pp. 454–469 (2011)
5. Kotthoff, L.: Ranking algorithms by performance. Tech. Rep. arXiv:1311.4319 (Nov 2013), `http://arxiv.org/abs/1311.4319`
6. Kotthoff, L., Gent, I.P., Miguel, I.: An evaluation of machine learning in algorithm selection for search problems. AI Communications 25(3), 257–270 (2012)
7. O'Mahony, E., Hebrard, E., Holland, A., Nugent, C., O'Sullivan, B.: Using case-based reasoning in an algorithm portfolio for constraint solving. In: Proceedings of the 19th Irish Conference on Artificial Intelligence and Cognitive Science (Jan 2008)
8. Rice, J.R.: The algorithm selection problem. Advances in Computers 15, 65–118 (1976)
9. Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: SATzilla: portfolio-based algorithm selection for SAT. J. Artif. Intell. Res. (JAIR) 32, 565–606 (2008)
10. Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Hydra-MIP: automated algorithm configuration and selection for mixed integer programming. In: Workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion. pp. 16–30 (2011)