Online Algorithm Selection

Hans Degroote *
KU Leuven, Department of Computer Science, Belgium
CODeS & imec-ITEC
Hans.Degroote@kuleuven.be

Abstract

Algorithm selection approaches have achieved impressive performance improvements in many areas of AI. Most of the literature considers the offline algorithm selection problem, where the initial selection model is never updated after training. However, new data from running algorithms on instances becomes available while an algorithm selection method is in use. In this extended abstract, the online algorithm selection problem is considered. In online algorithm selection, additional data can be processed, and the selection model can change over time. This abstract details the online algorithm setting, shows that it is a contextual multi-armed bandit, proposes a solution methodology, and empirically validates it.

1 Online Algorithm Selection

Many AI-problems are NP-complete: there exists no general efficient algorithm to solve them with. Nevertheless, the problems are often solved efficiently using heuristics. Such heuristics work well in some cases, but not in others. The idea of algorithm selection is to compose a set of complementary algorithms, with each algorithm performing well on different kinds of instances, and to predict for each new instance which algorithm is best suited to solve it. This problem of predicting which algorithm is best for each instance is known as the algorithm selection problem.

Algorithm selection methods use supervised learning techniques to build a selection mapping (λ), which maps each instance to the algorithm believed to be best for it. To do so, instances are characterised by a set of cheaply-computable features, correlated with their difficulty. The selection mapping is initialised based on offline training data, consisting of the performance of the algorithms on a set of training instances. Once created, the selection mapping is consulted to make predictions for new online problem instances, but it is never modified. If the training data did not accurately capture the problem, poor selections will be made, resulting in poor performance that will always remain poor.

*Work supported by the Belgian Science Policy Office (BELSPO) in the Interuniversity Attraction Pole COMEX (http://comex.ulb.ac.be)

Every time a prediction for a new online instance is made, the performance of the selected algorithm on that instance becomes known. Offline algorithm selection methods throw away this free data. The idea of online algorithm selection is to process it, aiming to improve the selection mapping. Online algorithm selection is most useful when training data is expensive to obtain or fails to accurately capture the problem.

Online algorithm selection is a generalisation of the offline problem, allowing the selection mapping to change when new data becomes available. A solution strategy for online algorithm selection (β) defines how to choose the selection mapping, based on all data available so far (H): both the training data and the online data. This data consists of records $\{i, \varphi, a, p\}$, with i an instance, φ its feature values, a an algorithm, and p the observed performance. Algorithm 1 shows the general procedure for online algorithm selection.

Algorithm 1 Online algorithm selection

```
Input: training data H_T
Input: online strategy β
H = H_T
for instance i do
    λ = β(H) //Get selection map, based on all data
    a = λ(i) //Make selection
    Solve i with a, observing performance p
    H = H ∪ {i, φ, a, p} //Add newly generated data
```

A common approach to offline algorithm selection is to learn a regression model for each algorithm in the portfolio. These regression models predict the performance of the corresponding algorithm on an instance, based on the instance's feature values. The algorithm with best predicted performance is then selected. These methods can easily take the online data into account, by updating the regression model of the selected algorithm after each online instance. This makes them a good candidate for an online strategy.

A popular alternative is to use a classifier to directly predict the best algorithm for a new instance. However, such a method cannot be directly applied to online algorithm selection, because it cannot process the online data, which consists of the performance of only one algorithm for each instance. Based on such incomplete data, it is impossible to know which algorithm is best, which is a requirement for the

classifier. [Malitsky, 2014] proposed a work-around for this issue. A clustering technique is used to initialise a selection mapping, selections are made, and the mapping remains the same as long as the online instances are similar to those in the training data. However, when a substantially different online instance is encountered, all algorithms are run on it, and the clustering is updated. The disadvantages of this method are that it cannot process the data generated by the similar instances, and that it needs to run all algorithms on the dissimilar instances. In contrast, regression-based method can use all online data and need never run additional experiments.

2 Online Algorithm Selection as a Contextual Multi-armed Bandit Problem

Online algorithm selection can be modelled as a contextual multi-armed bandit problem.

In the standard multi-armed bandit problem, a gambler has access to a set of slot machines (bandits) and he must decide on a strategy in which order to pull their arms. His goal is to realise as much profit as possible. Each time an arm is pulled, the gambler receives a random reward sampled from a distribution belonging to the selected arm. Initially, all distributions are unknown, but as the gambler gambles on he obtains more information about the distributions of the available arms, and can make better-informed choices.

The contextual multi-armed bandit problem generalises the multi-armed bandit problem by introducing side information: before pulling an arm, the gambler sees a context vector. This context vector describes the current situation, and the reward of each arm depends on it. The gambler's goal is again to maximise profit, but in order to do so he has to learn how the context vector relates to the rewards.

The online algorithm selection problem is a contextual multi-armed bandit problem: each algorithm corresponds to an arm and pulling an arm is the equivalent of selecting an algorithm. Features are used as basis for the selection, which is the equivalent of seeing a context vector. Maximising profit is the equivalent of maximising performance.

Multi-armed bandit literature shows that the simple greedy strategy of always pulling the predicted best arm can perform poorly. Instead, a predicted non-best arm should occasionally be explored, to increase the probability that it is indeed worse, and had not not simply been unlucky before.

3 Empirical Validation

Experiments were run to test the validity of online algorithm selection, using the greedy strategy described at the end of section 1 and two exploring strategies: ϵ-greedy and a UCB-variant. All strategies used random forest as regressor. Tests were run on 18 scenarios of the algorithm selection benchmark ASLIB [Bischl *et al.*, 2016]. The online setting was simulated by splitting each scenario's instances in three subsets: 10% used to initialise the models, 80% used as online instances, and 10% held out to measure how model quality evolves over time. As a baseline, a regression forest trained on 10% of the data but never updated was used. This is a classic offline strategy. As a final reference, the benchmark

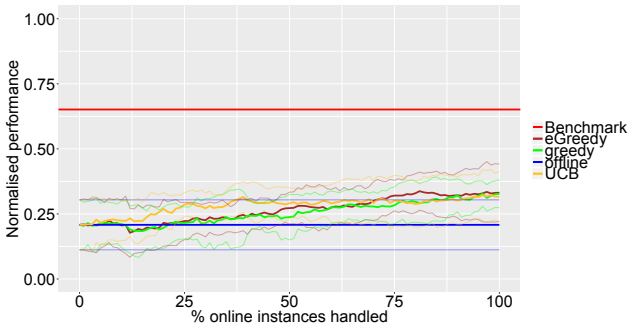

Figure 1: Evolution of model-quality over time aggregated over all scenarios. Opaque lines show the mean and semi-transparent lines one sd above and below.

performance obtained by running regression forest on 90% of the data and testing it on the remaining 10% was used.

The greedy strategy performed better than the offline baseline in 15 of the 18 scenarios. Both exploring strategies always performed worse than the greedy strategy, and performed worse than the baseline in about half. The cost of exploring was not compensated sufficiently by later gains.

Figure 1 plots selection-mapping-quality in function of the percentage of online instances handled, averaged over all scenarios. To aggregate, the performances of each experiment were normalised in relation to the single best solver (always selecting the same algorithm, performance 0) and the virtual best solver (selecting the best algorithm for each instance, performance 1). This plot shows that all online strategies learned better models over time, but that the exploring strategies did not learn better models than greedy.

4 Conclusions and Future Work

The contributions of the research carried out so far are that the problem of online algorithm selection has been formally defined, that is has been shown to be a contextual bandit, that a solution methodology has been proposed, and that an extensive empirical study has been performed on benchmark data, illustrating the validity of the method and providing a framework for future experiments.

Future work is to investigate why the exploratory strategies did not learn better models than the online greedy strategy. Other future work is to apply the methodology to a concrete application, to obtain better insight than possible with benchmark data, by having access to unlimited instance generation.

References

[Bischl *et al.*, 2016] Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Marius Lindauer, Yuri Malitsky, Alexandre Fréchette, Holger Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, et al. Aslib: A benchmark library for algorithm selection. *Artificial Intelligence*, 237:41–58, 2016.

[Malitsky, 2014] Yuri Malitsky. Evolving instance-specific algorithm configuration. In *Instance-Specific Algorithm Configuration*, pages 93–105. Springer, 2014.