autoBagging: Learning to Rank Bagging Workflows with Metalearning

Fábio Pinto (✉), Vítor Cerqueira, Carlos Soares and João Mendes-Moreira

INESC TEC/Faculdade de Engenharia, Universidade do Porto
Rua Dr. Roberto Frias, s/n
Porto, Portugal 4200-465
fhpinto@inesctec.pt vmac@inesctec.pt csoares@fe.up.pt jmoreira@fe.up.pt

Abstract. Machine Learning (ML) has been successfully applied to a wide range of domains and applications. One of the techniques behind most of these successful applications is Ensemble Learning (EL), the field of ML that gave birth to methods such as Random Forests or Boosting. The complexity of applying these techniques together with the market scarcity on ML experts, has created the need for systems that enable a fast and easy drop-in replacement for ML libraries. Automated machine learning (autoML) is the field of ML that attempts to answers these needs. We propose *autoBagging*, an autoML system that automatically ranks 63 bagging workflows by exploiting past performance and metalearning. Results on 140 classification datasets from the OpenML platform show that *autoBagging* can yield better performance than the Average Rank method and achieve results that are not statistically different from an ideal model that systematically selects the best workflow for each dataset. For the purpose of reproducibility and generalizability, *autoBagging* is publicly available as an R package on CRAN.

Keywords: automated machine learning, metalearning, bagging, classification

1 Introduction

Ensemble learning (EL) has proven itself as one of the most powerful techniques in Machine Learning (ML), leading to state-of-the-art results across several domains. Methods such as bagging, boosting or Random Forests are considered some of the favourite algorithms among data science practitioners. However, getting the most out of these techniques still requires significant expertise and it is often a complex and time consuming task. Furthermore, since the number of ML applications is growing exponentially, there is a need for tools that boost the data scientist's productivity.

The resulting research field that aims to answers these needs is Automated Machine Learning (autoML). In this paper, we address the problem of how to automatically tune an EL algorithm, covering all components within it: generation (how to generate the models and how many), pruning (which technique should be used to prune the ensemble and how many models should be discarded) and

integration (which model(s) should be selected and combined for each prediction). We focus specifically in the bagging algorithm [2] and four components of the algorithm: 1) the number of models that should be generated 2) the pruning method 3) how much models should be pruned and 4) which dynamic integration method should be used. For the remaining of this paper, we call to a set of these four elements a bagging workflow.

Our proposal is *autoBagging*, a system that combines a learning to rank approach together with metalearning to tackle the problem of automatically generate bagging workflows. Ranking is a common task in information retrieval. For instance, to answer the query of a user, a search engine ranks a plethora of documents according to their relevance. In this case, the query is replaced by new dataset and *autoBagging* acts as ranking engine. Figure 1 shows an overall schema of the proposed system. We leverage the historical predictive performance of each workflow in several datasets, where each dataset is characterised by a set of metafeatures. This metadata is then used to generate a metamodel, using a learning to rank approach. Given a new dataset, we are able to collect metafeatures from it and feed them to the metamodel. Finally, the metamodel outputs an ordered list of the workflows, taking into account the characteristics of the new dataset.

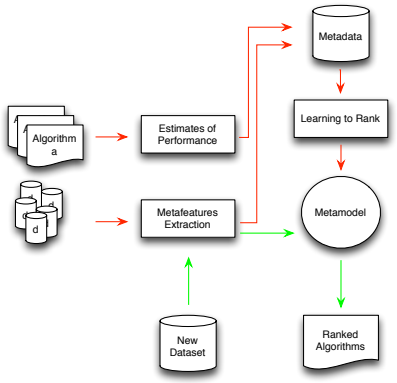

Fig. 1. Learning to Rank with Metalearning. The red lines represent offline tasks and the green ones represent online ones.

We tested the approach in 140 classification datasets from the OpenML platform for collaborative ML [12] and 63 bagging workflows, that include two pruning techniques and two dynamic selection techniques. Results show that *autoBagging* has a better performance than two strong baselines, Bagging with 100 trees and the average rank. Furthermore, testing the top 5 workflows recommended by *autoBagging* guarantees an outcome that is not statistically different from the Oracle, an ideal method that for each dataset always selects the best workflow. For the purpose of reproducibility and generalizability, *autoBagging* is available as an R package.[1]

[1] `https://github.com/fhpinto/autoBagging`

2 autoBagging

We approach the problem of algorithm selection as a learning to rank problem [7]. Lets take $\mathcal{D}$ as the dataset set and $\mathcal{A}$ as the algorithm set. $\mathcal{Y} = \{1, 2, ..., l\}$ is the label set, where each value represents a relevance score, which represents the relative performance of a given algorithm. Therefore, $l \prec l-1 \prec ... \prec 1$, where $\prec$ represents an order relationship.

Furthermore, $D_m = \{d_1, d_2, ..., d_m\}$ is the set of datasets for training and d_i is the i-th dataset, $A_i = \{a_{i,1}, a_{i,2}, ..., a_{i,n_i}\}$ is the set of algorithms associated with dataset d_i and $\mathbf{y}_i = \{y_{i,1}, y_{i,2}, ..., y_{i,n_i}\}$ is the set of labels associated with dataset d_i, where n_i represents the sizes of A_i and $\mathbf{y}_i$; $a_{i,j}$ represents the j-th algorithm in A_i; and $y_{i,j} \in Y$ represents the j-th label in $\mathbf{y}_i$, representing the relevance score of $a_{i,j}$ with respect to d_i. Finally, the meta-dataset is denoted as $S = \{(d_i, A_i), \mathbf{y}_i\}_{i=1}^{m}$.

We use metalearning to generate the metafeature vectors $x_{i,j} = \phi(d_i, a_{i,j})$ for each dataset-algorithm pair, where $i = 1, 2, ..., m$; $j = 1, 2, ..., n_i$ and ϕ represents the metafeatures extraction functions. These metafeatures can describe d_i, $a_{i,j}$ or even the relationship between both. Therefore, taking $\mathbf{x}_i = \{x_{i,1}, x_{i,2}, ..., x_{i,n_i}\}$ we can represent the meta-dataset as $S' = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{m}$.

Our goal is to train a meta ranking model $f(d, a) = f(x)$ that is able to assign a relevance score to a given new dataset-algorithm pair d and a, given x.

2.1 Metafeatures

We approach the problem of generating metafeatures to characterize d and a with the aid of a framework for systematic metafeatures generation [10]. Essentially, this framework regards a metafeature as a combination of three components: meta-function, a set of input objects and a post-processing function. The framework establishes how to systematically generate metafeatures from all possible combinations of object and post-processing alternatives that are compatible with a given meta-function. Thus, the development of metafeatures for a MtL approach simply consists of selecting a set of meta-functions (e.g. entropy, mutual information and correlation) and the framework systematically generates the set of metafeatures that represent all the information that can be obtained with those meta-functions from the data.

For this task in particular, we selected a set of meta-functions that are able to characterize the datasets as completely as possible (measuring information regarding the target variable, the categorical and numerical features, etc) the algorithms and the relationship between the datasets and the algorithms (who can be seen as landmarkers [9]). Therefore, the set of meta-functions used is: *skewness*, *Pearson's correlation*, *Maximal Information Coefficient* (MIC [11]), *entropy*, *mutual Information*, *eta squared* (from ANOVA test) and *rank* of each algorithm [1].

Each meta-function is used to systematically measure information from all possible combination of input objects available for this task. We defined the input objects available as: *discrete descriptive data of the datasets*, *continuous*

descriptive data of the datasets, *discrete output data of the datasets*, *five sets of predictions (discrete predicted data) for each dataset* (naive bayes, decision tree with depth 1, 2 and 3, and majority class).

For instance, if we take the example of using Entropy as meta-function, it is possible to measure information in discrete descriptive data, discrete output data and discrete predicted data (if the base-level problem is a classification task). After computing the entropy of all these objects, it might be necessary to aggregate the information in order to keep the tabular form of the data. Take for the example the aggregation required for the entropy values computed for each discrete attribute. Therefore, we choose a palette of aggregation functions to capture several dimensions of these values and minimize the loss of information by aggregation. In that sense, the post-processing functions chosen were: *average*, *maximum*, *minimum*, *standard deviation*, *variance* and *histogram binning*.

Given these meta-functions, the available input objects and post-processing functions, we are able to generate a set of 131 metafeatures. To this set we add eight metafeatures: the number of examples of the dataset, the number of attributes and the number of classes of the target variable; and five landmarkers (the ones already described above) estimated using accuracy as error measure. Furthermore, we add four metafeatures to describe the components of each workflow: the number of trees, the pruning method, the pruning cut point and the dynamic selection method. In total, *autoBagging* uses a set of 143 metafeatures.

2.2 Metatarget

In order to be able to learn a ranking meta-model $f(d, a)$, we need to compute a metatarget that represents a score z to each dataset-algorithm pair (d, a), so that: $\mathcal{F} : (\mathcal{D}, \mathcal{A}) \rightarrow \mathcal{Z}$, where $\mathcal{F}$ is the ranking meta-models set and $\mathcal{Z}$ is the metatarget set.

To compute z, we use a cross validation error estimation methodology (4-fold cross validation in the experiments reported in this paper, Section 3), in which we estimate the performance of each bagging workflow for each dataset using Cohen's kappa score [4]. On top of the estimated kappa score, for each dataset, we rank the bagging workflows. This ranking is the final form of the metatarget and it is then used for learning the meta-model.

3 Experiments

Our experimental setup comprises 140 classification datasets extracted from the OpenML platform for collaborative machine learning [12]. We limited the datasets extracted to a maximum of 5000 instances, a minimum of 300 instances and a maximum of 1000 attributes, in order to speed up the experiments and exclude datasets that could be too small for some of bagging workflows that we wanted to test.

Regarding bagging workflows, we limited the hyperparameters of the bagging workflows to four: number of models generated, pruning method, pruning cut

point and dynamic selection method. Specifically, each hyperparameter could take the following values:

- Number of models: 50, 100 or 200. Decision trees was chosen as learning algorithm.
- Pruning method: Margin Distance Minimization(MDSQ) [8], Boosting-Based Pruning (BB) [8] or none.
- Pruning cut point: 25%, 50% or 75%.
- Dynamic integration method: Overall Local Accuracy (OLA), a dynamic selection method [13]; K-nearest-oracles-eliminate (KNORA-E) [6], a dynamic combination method; and none.

The combination of all these hyperparameters described above generated 63 valid workflows. We tested these bagging workflows in the datasets extracted from OpenML with 4-fold cross validation, using Cohen's kappa as evaluation metric. We used the XGBoost learning to rank implementation for gradient boosting of decision trees [3] to learn the metamodel as described in Section 2.

As baselines, at the base-level, we use 1) bagging with 100 decision trees 2) the average rank method, which basically is a model that always predicts the bagging workflow with the best average rank in the meta training set and the 3) oracle, an ideal model that always selects the best bagging workflow for each dataset. As for the meta-level, we use as baseline the average rank method.

As evaluation methodology, we use an approach similar to the leave-one-out methodology. However, each test fold consists of all the algorithm-dataset pairs associated with the test dataset. The remaining examples are used for training purposes. The evaluation metric at the meta-level is the Mean Average Precision at 10 (MAP@10) and at the base-level, as mentioned before, we use Cohen's kappa. The methodology recommended by Demšar [5] was used for statistical validation of the results.

3.1 Results

Figure 2 shows a loss curve, relating the average loss in terms of performance with the number of workflows tested following the ranking suggested by each method. The loss is calculated as the difference between the performance of the best algorithm ranked by the method in comparison with the ground truth ranking. The loss for all datasets is then averaged for aggregation purposes. We can see, as expected, that the average loss decreases for both methods as the number of workflows tested increases.

In terms of comparison between *autoBagging* and the Average Rank method, it is possible to visualize that *autoBagging* shows a superior performance for all the values of the x axis. Interestingly, this result is particularly noticeable in the first tests. For instance, if we test only the top 1 workflow recommended by *autoBagging*, on average, the kappa loss is half of the one we should expect from the suggestion made by the average rank method.

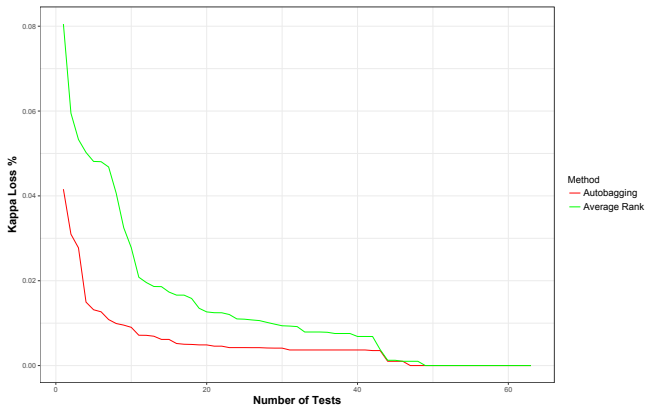

Fig. 2. Loss curve comparing autoBagging with the Average Rank baseline.

We evaluated these results to assess their statistical significance using Demšar's methodology. Figures 3 and 4 show the Critical Difference (CD) diagrams for both the meta and the base-level.

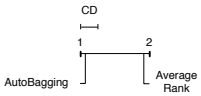

Fig. 3. Critical Difference diagram (with $\alpha = 0.05$) of the experiments at the meta-level.

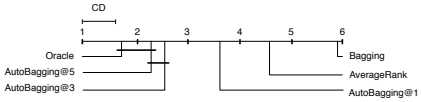

Fig. 4. Critical Difference diagram (with $\alpha = 0.05$) of the experiments at the base-level.

At the meta-level, using MAP@10 as evaluation metric, *autoBagging* presents a clearly superior performance in comparison with the Average Rank. The difference is statistically significant, as one can see in the CD diagram. This result is in accordance with performance that visualized in Figure 2 for both methods.

At the base-level, we compared *autoBagging* with three baselines, as mentioned before: bagging with 100 decision trees, the Average Rank method and the oracle. We test three versions of *autoBagging*, taking the top 1, 3 and 5 bagging workflows ranked by the meta-model. For instance, in *autoBagging@3*, we test the top 3 bagging workflows ranked by the meta-model and we choose the best.

Starting by the tail of the CD diagram, both the Average Rank method and *autoBagging@1* show a superior performance than Bagging with 100 decision trees. Furthermore, *autoBagging@1* also shows a superior performance than the Average Rank method. This result confirms the indications that we visualized in Figure 2.

The CD diagram shows also *autoBagging@3* and *autoBagging@5* have a similar performance. However, and we must highlight these results, *autoBagging@5* shows a performance that is not statistically different from the oracle. This is extremely promising since it shows that the performance of *autoBagging* excels if the user is able to test the top 5 bagging workflows ranked by the system.

4 Conclusion

This paper presents *autoBagging*, an *autoML* system that makes use of a learning to rank approach and metalearning to automatically suggest a bagging ensemble specifically designed for a given dataset. We tested the approach on 140 classification datasets and the results show that *autoBagging* is clearly better than the baselines to which was compared. In fact, if the top five workflows suggested by *autoBagging* are tested, results show that the system achieves a performance that is not statistically different from the oracle, a method that systematically selects the best workflow for each dataset. For the purpose of reproducibility and generalizability, autoBagging is publicly available as an R package.

Acknowledgements

This work was partly funded by the ECSEL Joint Undertaking, the framework programme for research and innovation horizon 2020 (20142020) under grant agreement number 662189-MANTIS-2014-1.

References

1. Brazdil, P., Carrier, C.G., Soares, C., Vilalta, R.: Metalearning: Applications to data mining. Springer Science & Business Media (2008)
2. Breiman, L.: Bagging predictors. Machine learning 24(2), 123–140 (1996)
3. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. pp. 785–794. KDD '16, ACM (2016)
4. Cohen, J.: A coefficient of agreement for nominal scales. Educational and psychological measurement 20(1), 37–46 (1960)
5. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. JMLR 7(Jan), 1–30 (2006)
6. Ko, A.H., Sabourin, R., Britto Jr, A.S.: From dynamic classifier selection to dynamic ensemble selection. Pattern Recognition 41(5), 1718–1731 (2008)
7. Liu, T.Y.: Learning to rank for information retrieval. Foundations and Trends® in Information Retrieval 3(3), 225–331 (2009)
8. Martínez-Muñoz, G., Hernández-Lobato, D., Suárez, A.: An analysis of ensemble pruning techniques based on ordered aggregation. IEEE Transactions on Pattern Analysis and Machine Intelligence 31(2), 245–259 (2009)
9. Pfahringer, B., Bensusan, H., Giraud-Carrier, C.: Tell me who can learn you and i can tell you who you are: Landmarking various learning algorithms. In: ICML. pp. 743–750 (2000)
10. Pinto, F., Soares, C., Mendes-Moreira, J.: Towards automatic generation of metafeatures. In: PAKDD. pp. 215–226. Springer (2016)
11. Reshef, D.N., Reshef, Y.A., Finucane, H.K., Grossman, S.R., McVean, G., Turnbaugh, P.J., Lander, E.S., Mitzenmacher, M., Sabeti, P.C.: Detecting novel associations in large data sets. Science 334(6062), 1518–1524 (2011)
12. Vanschoren, J., Van Rijn, J.N., Bischl, B., Torgo, L.: Openml: networked science in machine learning. ACM SIGKDD Explorations Newsletter 15(2), 49–60 (2014)
13. Woods, K., Kegelmeyer Jr, W.P., Bowyer, K.: Combination of multiple classifiers using local accuracy estimates. Transactions on Pattern Analysis and Machine Intelligence 19(4), 405–410 (1997)