Fast Automated Selection of Learning Algorithm And its Hyperparameters by Reinforcement Learning

Valeria Efimova VEFIMOVA@CORP.IFMO.RU
Andrey Filchenkov AFILCHENKOV@CORP.IFMO.RU
Viacheslav Shalamov SHALAMOV@RAIN.IFMO.RU
ITMO University, Russia, St. Petersburg, Kronverksky pr. 49

Abstract

To solve a data analysis problem, we have to choose a proper algorithm and tune its hyperparameters to optimize the quality of solution. We consider a classification task and propose a reinforcement learning based approach, which involves a strategy of sharing time between learning models for optimizing their hyperparameters. Results show that the choice of reinforcement learning algorithm has a limited influence on results shown by our approach that are better than results shown by Auto-WEKA and TPOT.

Keywords: algorithm selection, hyperparameter optimization, multi-armed bandit, reinforcement learning, q-learning

1. Introduction

The majority of the machine learning applications include solving the supervised learning problem. A lot of learning algorithms exist for building such a model, especially in classification. Unfortunately, it turns out, that these algorithms demonstrate different performance, depending on the data. According to the No Free Lunch theorem (Ho and Pepyne, 2001), a single supervised algorithm can not build the best model for all possible problem instances without using instance specificity. Even more, each of these algorithms has a set of hyperparameters, which significantly affect the performance. Selection of both learning algorithm and its hyperparameters requires a lot of time and does not always lead to good performance.

Usually, this problem is divided into two individual problems to be solved separately: *algorithm selection* and *hyperparameter optimization*. The first is focused on selecting algorithm from a given set of algorithms (algorithm portfolio). The second one about tuning model hyperparameters in order to achieve the best possible performance of the model.

In this work, we analyze a method for simultaneous selection of learning algorithm and its hyperparameters. Our algorithm is faster than the exhaustive search without affecting solution quality and is based on reinforcement learning approach.

We use a solution of multi-armed bandit problem, POKER, and Q-learning. It is compared with the result from previous papers, based on Softmax and UCB1 algorithms for multi-armed bandit solutions. Also we use suggested reward function and show, that it improves multi-armed bandit results.

2. Related works

We use the following notation. We are given a dataset $D = \{d_1, d_2, \dots, d_n\}$, where $d_i = (x_i, y_i) \in X \times Y$, and a set of classification algorithms $\mathcal{A} = \{A^1, A^2, \dots, A^m\}$. We assume that each learning algorithm A^i has a related hyperparameter space Λ^i. We will denote the algorithm A with chosen hyperparameter vector $\lambda \in \Lambda$ as A_λ. Therefore, we should choose an algorithm $A^*_{\lambda^*}$ which is the most efficient with respect to a quality measure Q. For instance, Q can be hold-out empirical risk. Then we represent the problem as the quality measure maximization:

$$A^*_{\lambda^*} \in \underset{A^j \in \mathcal{A}, \lambda \in \Lambda^j}{\operatorname{argmax}} Q(A^j_\lambda, D).$$

2.1. Algorithm selection

In this subproblem we are given a set of algorithms with chosen hyperparameters $\mathcal{A} = \{A^1_{\lambda_1}, \dots A^m_{\lambda_m}\}$. We should choose a parametrized algorithm $A^*_{\lambda^*}$ that is the most effective with respect to a quality measure Q. The algorithm selection problem thus is stated as the empirical risk minimization problem:

$$A^*_{\lambda_*} \in \underset{A^j_{\lambda_j} \in \mathcal{A}}{\operatorname{argmin}} Q(A^j_{\lambda_j}, D).$$

In practice, it is typically solved as follows. All the algorithms with fixed hyperparameters are compared with each other and the algorithm showed the best performance is chosen. Other methods are also in use, such as selecting algorithms randomly or by some heuristics. However, these methods are applicable only in a restricted number of cases.

From the theoretical point of view, this problem has been in the scope of interest for decades. In earliest papers written several decades ago, decision rules were used (Aha, 1992). As an example, such rules are used to choose from 8-algorithms portfolio in (Ali and Smith, 2006). Nowadays, more effective approach exists for algorithm selection, such as meta learning (Giraud-Carrier et al., 2004; Abdulrahman et al., 2015). This approach involves reducing the algorithm selection to supervised learning (Filchenkov and Pendryak, 2015; Castiello et al., 2005). More efficient approach involves ranking each algorithm instead of labeling only the best one (Brazdil et al., 2003; Sun and Pfahringer, 2013).

2.2. Hyperparameter optimization

Hyperparameter optimization is the process of selecting hyperparameters $\lambda^* \in \Lambda$ of a learning algorithm A to optimize its performance. Therefore, we can write:

$$\lambda^* \in \underset{\lambda \in \Lambda}{\operatorname{argmin}} Q(A_\lambda, D).$$

There are several algorithms for solving the second subproblem automatically: Grid Search (Bergstra and Bengio, 2012), Random Search (Hastie et al., 2005), Stochastic Gradient Descent (Bottou, 1998), Tree-structured Parzen estimator (Bergstra et al., 2011), and the Bayesian Optimization including Sequential Model-Based Optimization (SMBO) (Snoek et al., 2012). Sequential model-based algorithm configuration (SMAC), which is based on SMBO algorithm, is introduced in (Hutter et al., 2011). It is important to note that all these approaches can be expressed in a sequential manner, in which an algorithm decides on each step, in which point of hyperparameter domain the model performance should be evaluated given a history of previous evaluations.

Another idea is using meta-learning approach (Mantovani et al., 2015) for predicting the best hyperparameters. Reinforcement-based approach was used in (Jamieson and Talwalkar, 2015).

2.3. Simultaneous selection

Solution for selecting algorithm and tuning its hyperparameters simultaneously is highly important for machine learning applications. However, only a few papers exist devoted to this topic.

One of the possible approaches is to build a set of algorithms with fixed hyperparameters and select one from the set. This was introduced in (Leite et al., 2012). Authors used a set of about 300 algorithms to choose from. However, such selection-only approach cannot guarantee high performance: the set may simply not include suitable hyperparameter vector.

Another possible solution is to optimize hyperparameters for all algorithms and pick the best one. Such a solution was implemented in the Auto-WEKA library (Thornton et al., 2012). It allows automatically choosing from the 27 base learning algorithms, 10 meta-algorithms and 2 ensemble algorithms and optimize hyperparameters with SMAC method. However, this method takes enormous amount of time and may be referred to as exhaustive search: SMAC, which takes a significant amount of time by itself, has to be run on each of learning algorithm.

Several papers were devoted to full-model search, such as Escalante et al. (2009); Gorissen et al. (2009); Rosales-Pérez et al. (2014). Most of them exploit genetic algorithms for performing search within space of models, that usually include prepossessing technique applications, but hardly ever — hyperparameter optimization. Recently, Tree-based Pipeline Optimization Tool (TPOT) was introduced for this problem (Olson et al., 2016). It is based on *pipelines* organized in a tree-like structure. Each node of tree-based pipeline is a *pipeline operator*. There are four classes of pipeline operators: preprocessing, decomposition, feature selection and modeling. Every pipeline begins with one or more copies of the input dataset as the leaves of the tree, which is then fed into operators according to pipeline structure. Data is modified by node's operator. If there are multiple copies of dataset in the pipeline, they are combined by special combining operators. Authors use genetic programming to obtain better pipelines from existing pipelines.

3. Suggested approach

We associate a sequential hyperparameter optimization process π_i with each learning algorithm A_i. The solution of the initial problem can be achieved by running all these processes. However, instead of running all these processes one-by-one, a more practical approach is to share available computational resources(time) between processed in runtime.

It can be formally described as follows. We denote by T a time limitation for solving the main selection problem. We split T into several *timeslots* $T = t_1 + \dots + t_m$ such that

$t_i \geq 0\, \forall i$. Then we run process π_i within a *timeslot* t_i trying to get a maximal quality:

$$\max_j Q(A^j_{\lambda_j}, D) \xrightarrow[(t_1,\ldots,t_m)]{} \min,$$

where $A^j \in \mathscr{A}, \lambda_j = \pi_j(t_j, A^j, \emptyset)$ and $t_1 + \ldots + t_m = T$; $t_i \geq 0\, \forall i$ (Efimova et al., 2016).

The classical problem of reinforcement learning (Sutton and Barto, 1998) is in the finding tradeoff between *exploitation* and *exploration*. On the one hand, we can *exploit* the algorithm A^i which happens currently to be the best by spending more timeslots on π_i. On the other hand, we can *explore* other algorithms which can possibly demonstrate better performance with properly tuned hyperparameters. Unfortunately, it is not known, which algorithm will be the best, so we need to find it out by spending time budget wisely.

Consider a much simpler problem, when we can use only one hyperparameter optimization algorithm (HPOA). In this case, all π_i are applications of HPOA to the learning algorithms. The question is, how to choose the next process or, which is the same, the next learning algorithm to be tuned during next time slot.

3.1. Multi-armed Bandit Problem

Consider the N-levered device. Using each of the levers (arm) grants a certain reward. This reward is chosen according to an unknown probability distribution. At each iteration k, an agent chooses an arm a_i and get a reward $r(i, k)$. In multi-armed bandit problem, the agent's goal is to minimize the total loss by time T. Algorithms for this problem are described in (Sutton and Barto, 1998).

We associate an arm a_i with sequential hyperparameter optimization process $\pi_{a_i}(t, A^i, \{\lambda_k\}_{k=0}^q) \rightarrow \lambda^i_{q+1} \in \Lambda^i$ for learning algorithms $\mathcal{A} = \{A^1, \ldots, A^m\}$. After *playing* an arm $i = a_k$ at iteration k, we assign time slot t to process π_{a_k} to optimize hyperparameters. When time slot ends and the selected process stops, we receive hyperparameter vector λ^i_k. After that, we evaluate the estimated quality for a process π_i at iteration k, which is $Q(A^i_{\lambda^i_k}, D)$.

Note, that the multi-armed bandit problem is maximization problem. As soon as process π_i optimizes hyperparameters for empirical risk minimization, we define an average reward function as follows:

$$\bar{r}_{i,(k)} = \frac{Q_{max} - E_{(k)}(Q(A^i_{\lambda^i_k}, D))}{Q_{max}}, \tag{3.1}$$

where Q_{max} is the maximal empirical risk that was achieved on a given dataset, $E_t(Q(A^i_{\lambda^i_k}, D))$ is empirical risk expectation at iteration k.

POKER strategy. Price of Knowledge and Estimated Reward (POKER) introduced in (Vermorel and Mohri, 2005) is also ideologically similar to multi-armed bandit approach. It relies on three main ideas: pricing uncertainty, exploiting the lever distribution.

Unlike other algorithms from this paper, this algorithm uses notion of *horizon* that is the number of rounds that remains to be played. Amount of exploration highly depends on *horizon* value. With each *arm* of multi-armed bandit is associated some uniformly distributed cost, and the arm with maximal cost is chosen.

UCB1, Softmax. The question is how to define the reward function. In our preliminary research (Efimova et al., 2016), we present a reward function that is based on SMAC properties, showed in equation 3.1. Also we implement this approach and compare it to

Auto-WEKA. We use six common classification algorithms and the following algorithms solving multi-armed bandit problem: UCB1, $\varepsilon-$greedy, Softmax (Sutton and Barto, 1998) with naïve reward function and the suggested reward function. The experiments show that the average efficiency of the suggested approach outperforms average efficiency of Auto-WEKA library with 40%.

However, this approach is not universal due to the reward function distribution which must be the same at any step. It is hard to achieve this property for every HPOA. That is why Q-learning may be more preferable.

Q-learning. In the algorithm and its hyperparameters selection problem we have linear ordered states. These states describe empirical risk achieved at some moment and satisfy Markov decision process definition. Therefore, we assume that the states are hidden and associate actions with a sequential hyperparameter optimization processes $\mathcal{C} = \{\pi_i(t, A^i, \{\lambda_k\}_{k=0}^{q}) \rightarrow \lambda_{q+1}^{i} \in \Lambda^i\}_{i=0}^{m}$ for a learning algorithms $\mathcal{A} = \{A^1, \dots, A^m\}$.

It is very important for original algorithm, from which states transition is made. In our problem, source of transition is not important. Thus, instead of 2-D array we use 1-D array, which defines target states.

4. Experiments

As a baseline for comparison, we chose Auto-WEKA and TPOT. Experiments were made on 10 different datasets from UCI repository[1].

Although our approach allows to use any hyperparameter optimization method, we decided to use SMAC method that is used by Auto-WEKA, to perform comparison in the most fair way. We consider six well-known classification algorithms to choose from: k Nearest Neighbors (4 categorical and 1 numerical hyperparameters), Support Vector Machine (4 and 6), Logistic Regression (0 and 1), Random Forest (2 and 3), Perceptron (5 and 2), and C4.5 Decision Tree (6 and 2).

The whole available time budget T we split into small equal time slots t_j. We assign an available timeslot to a selected process π_i at each iteration.

We compare the method performance for different time slot t sizes to find the optimal value. We consider time slot sizes from 10 to 60 seconds with 3 second step. After that we run the suggested method on datasets Car, German Credits, KRvsKP. For the above comparison, we use 4 multi-armed bandit based algorithms: UCB1, 0.4-greedy, 0.6-greedy, Softmax. We run each configuration 3 times. The results showed no clear pattern, and we assumed time budget t to be 30 seconds (Efimova et al., 2016). In the quality comparison, we consider suggested method with one multi-armed bandit problem solutions, $POKER_{E(Q)}$, with the suggested reward function and Q-learning. Time budget on iteration is $t = 30$ seconds, the general time limitation for Auto-Weka is $T = 3$ hours $= 10800$ seconds and for POKER and Q-learning $T = 25$ minutes $= 1500$ seconds. We run each configuration 12 times with random seeds of SMAC algorithm. TPOT is not limited by specified algorithms and by time. The experiment results are shown in Table 1. We added $UCB1_{E(Q)}$, $Softmax_{E(Q)}$ results from (Efimova et al., 2016) for more expressive comparison.

As it can be seen from results, TPOT performs significantly worse that Auto-Weka. Q-learning happens to be close enough to Auto-Weka, showing better results in 2 cases of

1. http://www.cs.ubc.ca/labs/beta/Projects/autoweka/datasets/

Table 1: Comparison of Auto-WEKA and TPOT with suggested methods for selecting classification algorithm and its hyperparameters for the given dataset. We performed 12 independent runs of each configuration and report the smallest empirical risk Q achieved by Auto-WEKA, TPOT and variations of the suggested method. We highlight with bold entries that are minimal for the given dataset.

Dataset	AutoWEKA	TPOT	Q-learning	$POKER_{E(Q)}$	$UCB1_{E(Q)}$	$Softmax_{E(Q)}$
Car	0.3305	13.32	**0.1836**	**0.1836**	**0.1836**	**0.1836**
Yeast	34.13	45.43	36.28	33.65	**29.81**	**29.81**
KR-vs-KP	0.2976	5.471	**0.1488**	**0.1488**	**0.1488**	**0.1488**
Semeion	4.646	16.8	4.027	**1.786**	**1.786**	**1.786**
Shuttle	**0.00766**	0.1321	0.0118	**0.00766**	**0.00766**	**0.00766**
Dexter	7.143	8.631	5.0	2.381	2.381	**0.16**
Waveform	11.28	13.93	11.42	**8.286**	**8.286**	**8.286**
Secom	4.545	12.19	4.545	**3.636**	**3.636**	**3.636**
Dorothea	6.676	7.453	4.938	5.602	4.32	**2.469**
German Credits	19.29	27.67	15.71	15.71	**14.29**	**14.29**

10 and worse results also in 2 cases. POKER happens to be better than Auto-Weka in all 10 cases, but in 4 cases it is slightly worse than previously proposed $Softmax_{E(Q)}$ algorithm, which use the suggested reward function, achieved the smallest empirical risk in most cases.

Moreover, the suggested approach does not restrict hyperparameters space of a learning algorithm. It is significant, that the suggested method allows to select a learning algorithm with hyperparameters, whose quality is not worse than Auto-WEKA outcome quality.

5. Conclusions

In this paper, we tried to improve previously suggested approach to solve the problem of simultaneous algorithm selection and its hyperparameters optimization. This approach is based on reinforcement learning. We used Q-learning, which happens to be worse than previously suggested algorithms.

Also, we used a multi-armed bandit problem solution with the previously suggested reward function. This reward function improves multi-armed bandit for every algorithm we took. Performance improvement of this reward function happens to be significantly higher, that the difference between classical multi-armed bandit solutions.

Out method can be enhanced by applying meta-learning: for a new dataset we might want to predict in advance, what learning algorithm will perform better, and give more timeslots to it.

Moreover, we can use solutions of a contextual multi-armed bandit problem and add a *context vector* to hyperparameters optimization process. We can use the predicted empirical risk estimate and use it as a context.

Acknowledgments

The research was supported by the Government of the Russian Federation (grant 074-U01) and the Russian Foundation for Basic Research (project no. 16-37-60115).

References

Salisu Mamman Abdulrahman, Pavel Brazdil, Jan N van Rijn, and Joaquin Vanschoren. Algorithm selection via meta-learning and sample-based active testing. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases; International Workshop on Meta-Learning and Algorithm Selection*. University of Porto, 2015.

David W Aha. Generalizing from case studies: A case study. In *Proc. of the 9th International Conference on Machine Learning*, pages 1–10, 1992.

Shawkat Ali and Kate A Smith. On learning algorithm selection for classification. *Applied Soft Computing*, 6(2):119–138, 2006.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, 2012.

James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554, 2011.

Léon Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9):142, 1998.

Pavel B Brazdil, Carlos Soares, and Joaquim Pinto Da Costa. Ranking learning algorithms: Using ibl and meta-learning on accuracy and time results. *Machine Learning*, 50(3): 251–277, 2003.

Ciro Castiello, Giovanna Castellano, and Anna Maria Fanelli. Meta-data: Characterization of input features for meta-learning. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 457–468. Springer, 2005.

Valeria Efimova, Andrey Filchenkov, and Anatoly Shalyto. Reinforcement-based simultaneous algorithm and its hyperparameters selection. https://arxiv.org/abs/1611.02053, 2016.

Hugo Jair Escalante, Manuel Montes, and Luis Enrique Sucar. Particle swarm model selection. *Journal of Machine Learning Research*, 10(Feb):405–440, 2009.

Andrey Filchenkov and Arseniy Pendryak. Datasets meta-feature description for recommending feature selection algorithm. In *Artificial Intelligence and Natural Language and Information Extraction, Social Media and Web Search FRUCT Conference (AINL-ISMW FRUCT), 2015*, pages 11–18. IEEE, 2015.

Christophe Giraud-Carrier, Ricardo Vilalta, and Pavel Brazdil. Introduction to the special issue on meta-learning. *Machine learning*, 54(3):187–193, 2004.

Dirk Gorissen, Tom Dhaene, and Filip De Turck. Evolutionary model type selection for global surrogate modeling. *Journal of Machine Learning Research*, 10(Sep):2039–2078, 2009.

Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

Yu-Chi Ho and David L Pepyne. Simple explanation of the no free lunch theorem of optimization. In *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, volume 5, pages 4409–4414. IEEE, 2001.

Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization*, pages 507–523. Springer, 2011.

Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. *JMLR*, 41:240–248, 2015.

Rui Leite, Pavel Brazdil, and Joaquin Vanschoren. Selecting classification algorithms with active testing. In *Machine Learning and Data Mining in Pattern Recognition*, pages 117–131. Springer, 2012.

Rafael Gomes Mantovani, André LD Rossi, Joaquin Vanschoren, André Carlos Ponce de Leon Carvalho, et al. Meta-learning recommendation of default hyper-parameter values for svms in classifications tasks. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases; International Workshop on Meta-Learning and Algorithm Selection*. University of Porto, 2015.

Randal S. Olson, Nathan Bartley, Ryan J. Urbanowicz, and Jason H. Moore. Evaluation of a tree-based pipeline optimization tool for automating data science. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, pages 485–492, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4206-3. doi: 10.1145/2908812.2908918. URL http://doi.acm.org/10.1145/2908812.2908918.

Alejandro Rosales-Pérez, Jesus A Gonzalez, Carlos A Coello Coello, Hugo Jair Escalante, and Carlos A Reyes-Garcia. Multi-objective model type selection. *Neurocomputing*, 146: 83–94, 2014.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

Quan Sun and Bernhard Pfahringer. Pairwise meta-rules for better meta-learning-based algorithm ranking. *Machine learning*, 93(1):141–161, 2013.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 1998.

Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Auto-weka: Automated selection and hyper-parameter optimization of classification algorithms. *CoRR, abs/1208.3719*, 2012.

Joannes Vermorel and Mehryar Mohri. Multi-armed bandit algorithms and empirical evaluation. In *ECML*, volume 3720, pages 437–448. Springer, 2005.