Automating Multi-element Subspace Exploration via Reinforcement Learning

Yi Sun
School of Software
Xi'an Jiaotong University
Xi'an, China
e-mail: xwrong@stu.xjtu.edu.cn

YinXiao Liu
Data Technology and Product Department
Alibaba Group
Hangzhou, China
e-mail: yinxiao.lyx@alibaba-inc.com

ZhongYao Wang
Data Technology and Product Department
Alibaba Group
Hangzhou, China
e-mail: zhongyao.wangzy@alibaba-inc.com

BaoLong Niu
School of Software
Xi'an Jiaotong University
Xi'an, China
e-mail: bugdragon@stu.xjtu.edu.cn

***Abstract*—Machine learning enters many aspects of our lives and brings us great convenience. However, building an effective machine learning model for a specific task requires not only expertise but also a lot of time and resources. In order to solve this problem, more and more research projects focus on automated machine learning (AutoML). In this paper, we propose an algorithm that can simultaneously optimize the space of multiple datasets, multiple models, and multiple hyperparameters. We call this an automating multi-element subspace exploration algorithm. We first formalize this problem as a reinforcement learning problem and then we define the state, action and well-designed reward function in reinforcement learning system. In addition, we use some skills and experience to accelerate the entire optimization process. Finally, our experimental results on multiple tasks demonstrate that our method is effective.**

Keywords-machine learning; reinforcement learning; AutoML; model selection; hyperparameter optimization

I. INTRODUCTION

In recent years, as machine learning has succeeded in some areas, more and more industries are researching and deploying their own machine learning systems, but this process is not easy because users not only have an understanding of industry knowledge, but also be proficient in data preprocessing and choose the right model, features and hyperparameters.

In order to reduce the requirements for industry users, new ideas for automating the process of machine learning have been proposed, called automated machine learning (AutoML). There are many versions of AutoML definitions. In general, AutoML should be an end-to-end automated process that includes data preprocessing, feature engineering, model selection, and model evaluation to solve specific tasks. In [1], AutoML is defined as a combination of machine learning methods and automated methods. In other words, AutoML provides an automated pipeline that enables a class of machine learning problems to be solved with a given resource.

Reinforcement learning is an important method in the study of AutoML. It is a powerful algorithm that continuously optimizes by interacting with the environment to maximize long-term returns. For example, the famous NAS[2], ENAS[3], are all RL-base algorithms. However, NASs in traditional machine learning tasks, such as sales prediction, age prediction, etc. are far less effective than CV and NLP tasks.

In this paper, we mainly solve the automatic learning problem of a class of traditional machine learning tasks. Unlike other problems, the goal of this problem is to solve the selection of multi-dataset, multi-model, and multi-hyperparameter simultaneously which is called the multi-element subspace optimization problem. The reason why we want to solve this multi-element optimization problem simultaneously is that the correlation between these factors can be fully considered and can be integrated into a common framework. The reason why we use reinforcement learning is that reinforcement learning is highly scalable and its process is very similar to the manual optimization process and it is often used to solve the optimization process of complex spaces. Our challenge comes from five aspects. First, we have multiple datasets, but we don't know which datasets might help predict the results. Perhaps when we predict a task, only one dataset is helpful for the prediction result, and it is also possible that a single dataset does not help the prediction result, but the joint characteristics of several datasets will greatly help the prediction result. If we use all the datasets directly, there are two risks. On the one hand, the model has serious over-fitting risk. On the other hand, the calculation amount of the model will increase. Second, we have a series of base models, such as LR, DeepFM, DCN, etc. Some tasks use LR better, some tasks use DCN better, we want the algorithm to tell us which model is more beneficial to our task. Third, the model's hyperparameters also have an impact on the prediction results, such as learning rate, regularization coefficient, and so on. Forth, the amount of data in multiple datasets is very large, so our AutoML algorithm must be able to find the optimal solution

efficiently. Finally, how to integrate the above problems into a unified framework.

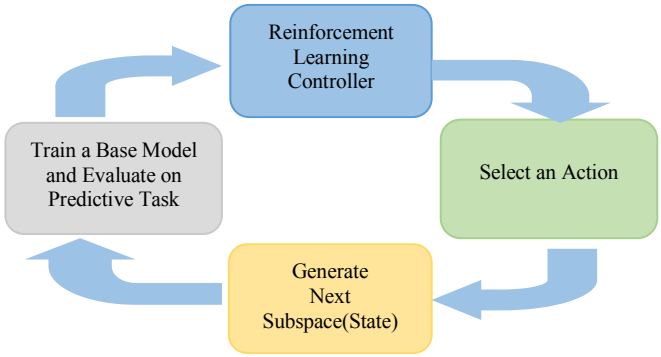

Figure 1. General process of reinforcement learning algorithm.

In summary, we developed an enhanced reinforcement learning framework for the multi-element subspace exploration problem. Specifcally, our contributions are as follows:(i)We integrate the multi-element subspace exploration problem into a reinforcement learning framework for the first time. In other AutoML systems, these parts are always designed separately. (ii) We propose a new form of reward, taking into account the accuracy of the model and the minimum number of datasets.

II. Related Work

In the study of this multi-element subspace exploration problem, there are several important directions, i.e. feature selection, model selection, hyperparameter optimization.

A. *Feature Selection*

Feature selection can be divided into three types: filter method, wrapper method and embedded method[4]. The filter method first sorts the features from high to low according to the relevance score, and then selects the top ranked features. The typical filter methods are feature selection based on correlation [5, 6] and feature selection based on univariate [7, 8]. The advantage of the filter method is that the calculation is simple and fast, so it is effective for high-dimensional features. However, they have the disadvantage of ignoring the relationship between feature selection and subsequent predictors and the dependence between features. The wrapper method treats prediction performance as an objective function [9]. The most typical wrapper method is the branch and bound algorithm [10, 11]. The embedded method has the best effect on the combination of predictors. Typical embedded methods include decision tree [12], LASSO [13]. Feature selection algorithms are also widely used in gene chip classification problems. The correlation-based feature selection algorithm used in [14] can not only improve the efficiency of the algorithm, but also improve its accuracy. [15] identified a compact feature set that consists of genre discriminative features.

B. *Hyperparameter Optimization*

There are five methods for hyperparameter optimization. First, Grid search and random search are the most widely used strategies. Grid search [16] – [18] has good scalability, convenient parallelism, and good results, but it also has some disadvantages. The search space increases exponentially with the number of parameters, and it is difficult to determine the change interval without knowing which parameter changes are more important. If the interval division is too small, the speed will be slow, but if the interval division is too large , it may miss the optimal solution. Bergstra and Bengio[19] show that random search is more practical and efficient than grid search. However, random search has a problem that it is hard to determine whether the best set of hyperparameters is found. As an improvement on this method, Li and Jamieson et al. [20] proposed the hyperband, which is a trade-off between resource budgets and performance. Second, NAS [2] introduced reinforcement learning for the first time, and they trained recurrent neural networks (RNN) to automatically generate network architectures using reinforcement learning (RL) technology. And later ENAS [3] in order to improve efficiency. Third, Evolutionary algorithm is a heuristic optimization algorithm based on population. Compared with calculus-based methods and exhaustive methods, the evolutionary algorithm has the advantages of wide applicability and strong robustness, and is a global optimization algorithm. Fourth, In terms of grid search, random search, and evolutionary algorithm, each test is independent, that is, it may test multiple times in poorly-performing regions. Bayesian optimization establishes a probability model of the objective function, then uses the probability model to select the most promising hyperparameters , and uses the real objective function to evaluate the selected hyperparameters. Therefore, Bayesian optimization uses previous evaluation results to continuously update the probability model. Probability model maps hyperparameters to probability of score on objective function. Bayesian optimization algorithms can be divided into Gaussian Processes [21], [22], Tree Parzen Estimators [23], and Random Forests [24] according to different probability models. Fifth, the previous method needs to evaluate many sets of parameters requires amount of time and resources. Gradient descent-based methods can reduce much time spent searching for hyperparameters, but increase the consumption of GPU memory.

C. *Model Selection*

Model selection can be divided into traditional model selection and NAS. Traditional model selection refers to selecting the best performing one from the base models of machine learning. From the perspective of model structure, NAS can be divided into four categories: Entire structure model, Cell-based structure model, Hierarchical Structure model, Network Morphism based structure model, [3] belongs to Cell-based structure model.

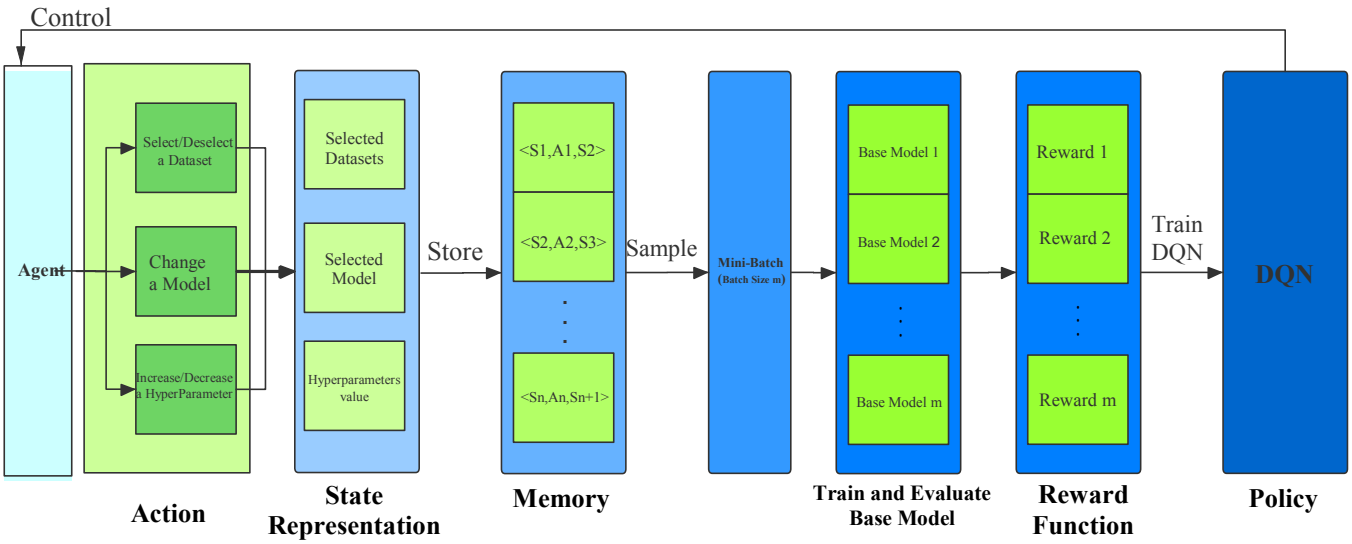

Figure 2. Framework.The framework consists of two stages. In the training stage, the policy is trained via samples from the memory. In the control stage, agent select/deselect a dataset or change a model or increase/decrease a hyperparameter based on the policy.

III. Problem Formulation

Essentially, the problem is equivalent to how to find an optimal subspace in the multi-element space. In other words, we study the problem of multi-element subspace exploration, which is formulated as a reinforcement learning task. Fig. 1 shows an overview of reinforcement learning algorithm. Specifically, the components of our joint space reinforcement learning framework include state, agent, reward, and agent action.

A. State

In our design, the state is a subspace of the multi-element space. It represents the datasets we selected, the model and the corresponding hyperparameter values. Whenever an agent selects an action, the state changes.

B. Agent

Agent is designed to make the selection decision for the corresponding state.

C. Reward

We designed a measurement to quantify the reward R generated by the multi-element subspace, which is defined as the weighted sum of (i) the predictive accuracy of the model corresponding to this subspace, (ii) the number of selected datasets P.

D. Action

For the agent, we define three types of actions, the first is to select/deselect a dataset, the second is to change a model, and the third is to increase/decrease the value of a hyperparameter.

IV. Proposed Method

We propose the reinforcement learning framework for automated multi-element subspace exploration. Then we discuss how to measure the reward function and how to accelerate the reinforcement learning process.

A. Framework Overview

Fig. 2 shows our proposed framework consists of many multi-element subspace exploration steps. Each exploration step includes two stage, i.e., agent control stage and model training stage.

In the agent control stage, the agent takes action according to its policy network, the input of the policy network is the current state, and the output is a long-term reward for taking different actions. Then select an action based on the greedy-ε method to get the next state. After that, we add the resulting triplet <st,at,st+1> to the memory of the experience replay.

In the training stage, there are two training processes. The first is to train the base model corresponding to the state to get reward, and the second is to train the agent's policy network via experience replay. In general, at time t, our training process is that we use experience replay memory to extract a mini-batch, which is a triplet form <st, at, st+1>, indicating the current state of st, the action at, and the next state st+1. Then, we train the base model corresponding to each state of this mini-batch, get the accuracy of the validation dataset ACC and the number of selected datasets P, and then we can get reward based on the weight sum Equation:

$$R = \alpha * ACC + \beta * P. \quad (1)$$

α, β are constant weight factor.

The agent uses its corresponding mini-batch samples and the reward to train its Deep Q Network (DQN), in order to obtain the maximum long-term reward based on the Bellman Equation:

$$Q(st, at|\theta t) = Rt + \gamma * \max Q(st+1, at+1|\theta t+1) \quad (2)$$

where θ is the parameter set of DQN, Rt is the instance reward at time t, st is the state at time t, at is the the action taken at time t, and γ is the discount factor between 0 and 1.

TABLE I. ACCURACY COMPARISON OF DIFFERENT ALGORITHM ON DIFFERENT TASKS

Task	OURS		Random Search		Manual Search
	Top-3 Average Accuracy	Top-10 Average Accuracy	Top-3 Average Accuracy	Top-10 Average Accuracy	
Baby gender	**0.7888**	0.7864	0.7834	0.7826	0.7803
Baby age level	**0.6534**	0.6509	0.6448	0.6413	0.6435
Age level	**0.6664**	0.6647	0.6550	0.6534	0.6562
Has child	**0.9079**	0.9068	0.9023	0.9015	0.9004
Married	**0.8518**	0.8473	0.8422	0.8382	0.8426

TABLE II. ACCURACY COMPARISON OF DIFFERENT REWARD FUNCTION ON DIFFERENT TASKS

Task	Reward Function Based on (1)			Reward Function Based on Accuracy		
	Top-3 Average Accuracy	Top-10 Average Accuracy	Average Number of Selected Datasets	Top-3 Average Accuracy	Top-10 Average Accuracy	Average Number of Selected Datasets
Baby gender	**0.7888**	0.7864	**5.7**	0.7850	0.7836	9.0
Baby age level	**0.6534**	0.6509	**7.7**	0.6473	0.6462	10.0

B. *Measuring Reward*

In the control stage, instead of using the quads <st,at,st+1,rt> directly like other articles, we use the triples <st,at,st+1>. This is because training base model is the most time-consuming part of the whole process, and the reward rt must be obtained through the trained base model. We postpone the rewards until the training stage is sampled, saving unnecessary computational costs.

We use (1) to measure rewards. The first item is that we use state st to train a base model and get its accuracy on the validation dataset. The main purpose of this item is to make the reinforcement learning algorithm optimize the policy network in the direction of high accuracy. The second item is the number of datasets selected by state st. The main purpose of this item is to make the reinforcement learning algorithm optimize the policy network in the direction of selecting as few datasets as possible. Obviously, the factor of the first term should be a positive number and the second factor should be a negative number. Their absolute size is the size of importance.

C. *Accelerating Training Process*

Accelerating the optimization process of the AutoML system, whether in the application of actual business scenarios or in the research of this direction, is very important. There are two main aspects to accelerate the training process. On the one hand, to accelerate the training process of the policy network, we mainly use the experience replay [25,26].In the experience replay, the agent takes samples from its memory that stores different training samples to train the policy network. The two advantages of doing this are: (i) Deep neural network as a supervised learning model, requiring data to satisfy independent and identical distributions. (ii) Breaking the correlation between data due to timing relationships. On the other hand, to accelerate the training process of the base model, we mainly use the small model to train on the small dataset, and then the method of migrating the optimal model to the big dataset. In order to find the optimal configuration, it is usually necessary to train hundreds of base models, and training the base model is the most time-consuming part, so speeding up this part is crucial for the entire optimization process.

V. EXPERIMENTAL RESULTS

A. *Data Description*

We used a total of 10 different datasets, each with a different number of features, ranging from about five hundreds to about ten thousands. We tested our method on five tasks to verify the robustness of the method and applicability.

B. *Overall Performances*

Our criterion is the top-3 and top-10 average accuracy of the model in the validation dataset. The input layer of DQN is a 14-dimensional state vector. The first layer and the second layer have 32 and 16 nodes respectively, and the output layer also has 16 nodes. The hidden layers are all activated using ReLU. In the experiments, we set mini-batch size to 32 and use AdamOptimizer with a learning rate of 0.001. The training process is to train about 100 base models on a 1080ti GPU. The entire optimization process takes about 9 hours. Table 1 shows that our method exceed random search and manual search in total five tasks.

C. *Reward Function*

We study the impact of the reward function design. We compare two cases. One is to use only the accuracy of the validation dataset as the reward, and the other is (1). Table 2 shows that reward function based on (1) is better than just using accuracy as the reward function. From the results, we can see that not only is average number of selected datasets less than the latter, but the average accuracy is also higher than the latter. In this experiment, we set α to 1.0 and β to -0.02.

VI. CONCLUSION

In this paper, we study the problem of automated multi-element subspace exploration. We first integrate this multi-element subspace exploration problem into a reinforcement learning framework. The agent can decide to select or delete a dataset, change a base model, increase or decrease one of the hyperparameters. The reward function we propose is a weighted sum of accuracy and the number of selected datasets. In order to accelerate the subspace exploration, we use experiments replay and training methods with small dataset and small model. Finally, we conducted extensive experiments on real datasets to demonstrate the effectiveness of the proposed method.

REFERENCES

[1] Yoshua Bengio et al. 2009. Learning deep architectures for AI. Foundations and trends® in Machine Learning 2, 1 (2009), 1–127.

[2] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning." [Online]. Available: http://arxiv.org/abs/1611.01578

[3] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," vol. ICML. [Online]. Available: http://arxiv.org/abs/1802.03268

[4] Xiucai Ye, Kaiyang Ji, and Tetsuya Sakurai. Unsupervised feature selection with correlation and individuality analysis. International Journal of Machine Learning and Computing, 6(1):36–41, 2016.

[5] Mark A Hall. 1999. Feature selection for discrete and numeric class machine learning. (1999)

[6] Lei Yu and Huan Liu. 2003. Feature selection for high-dimensional data: A fast correlation-based flter solution. In Proceedings of the 20th international conference on machine learning (ICML-03). 856–863.

[7] George Forman. 2003. An extensive empirical study of feature selection metrics for text classifcation. Journal of machine learning research 3, Mar (2003), 1289– 1305

[8] Yiming Yang and Jan O Pedersen. 1997. A comparative study on feature selection in text categorization. In Icml, Vol. 97. 412–420.

[9] Dihua Guo, Hui Xiong, Vijay Atluri, and Nabil Adam. 2007. Semantic feature selection for object discovery in high-resolution remote sensing imagery. In Pacifc-Asia Conference on Knowledge Discovery and Data Mining. Springer, 71– 83.

[10] Ron Kohavi and George H John. 1997. Wrappers for feature subset selection. Artifcial intelligence 97, 1-2 (1997),

[11] Patrenahalli M. Narendra and Keinosuke Fukunaga. 1977. A branch and bound algorithm for feature subset selection. IEEE Transactions on computers 9 (1977), 917–922.

[12] V Sugumaran, V Muralidharan, and KI Ramachandran. 2007. Feature selection using decision tree and classifcation through proximalsupport vector machine for fault diagnostics of roller bearing. Mechanical systems and signal processing 21, 2 (2007), 930–942.

[13] Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological) (1996), 267–288.

[14] Al-Batah MS, Zaqaibeh BM, Alomari SA, Alzboon MS. Gene Microarray Cancer Classification using Correlation Based Feature Selection Algorithm and Rules Classifiers. International Journal of Online and Biomedical Engineering (iJOE). International Association of Online Engineering (IAOE); 2019 May 14; 15(08):62. https://doi.org/10.3991/ijoe.v15i08.10617

[15] J. Yoon, H. Lim, D. Kim, "Music Genre Classification Using Feature Subset Search", International journal of Machine Learning and Computing, 2016.

[16] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation," in Proceedings of the 24th international conference on Machine learning. ACM, 2007, pp. 473–480.

[17] H. H. Hoos, Automated Algorithm Configuration and Parameter Tuning, 2011.

[18] I. Czogiel, K. Luebke, and C. Weihs, Response surface methodology for optimizing hyper parameters. Universitatsbibliothek Dortmund, ¨ 2006.

[19] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," p. 25.

[20] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization." [Online]. Available: http://arxiv.org/abs/1603.06560

[21] J. Gonzalez, "Gpyopt: A bayesian optimization framework in python," http://github.com/SheffieldML/GPyOpt, 2016.

[22] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in Advances in neural information processing systems, 2012, pp. 2951–2959.

[23] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," p. 9.

[24] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model based optimization for general algorithm configuration," in Learning and Intelligent Optimization, C. A. C. Coello, Ed. Springer Berlin Heidelberg, vol. 6683, pp. 507–523. [Online]. Available: http://link.springer.com/10.1007/978-3-642-25566-3 40

[25] Long-Ji Lin. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. Machine learning 8, 3-4 (1992), 293–321.

[26] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. Nature 518, 7540 (2015), 529.