Automatic Machine Learning Method for Hyper-parameter Search

Minglan Su a *, Baolin Liang b, Sicong Ma c, Chao Xiang d, Chaoying Zhang e and Jianxiu Wang f

China Telecom Corporation Limited Beijing Research Institute, Beijing 102209, China.

*, a Corresponding author e-mail: asuml@chinatelecom.cn,
b liangbl6@chinatelecom.cn, c masc@chinatelecom.cn,
d xiangchao@chinatelecom.cn, e zhangchy50@chinatelecom.cn,
f wangjianxiu@chinatelecom.cn

Abstract. Automatic Machine Learning (AutoML) uses automated data-driven methods to realize the selection of hyper-parameters, neural network architectures, regularization methods, etc., making machine learning techniques easier to apply and reducing dependence on experienced human experts. And hyper-parameter search based on automatic machine learning is one of the current research hotspots in the industry and academia. We mainly introduce the hyper-parameter search framework based on automatic machine learning and the common hyper-parameter search strategies. Combined with specific data sets, the classification accuracy of the model under different hyper-parameter search strategies is compared to find the model parameter configuration that can maximize the classification accuracy. Compared with the experience-based parameter adjustment method, the hyper-parameter search based on automatic machine learning can reduce labor costs, improve training efficiency, and automatically construct a dedicated convolutional neural network to maximize the model effect.

Keywords: Automatic Machine Learning, Hyper-parameter Search Framework, Hyper-parameter Search Strategies, Classification.

1. Introduction

In recent years, artificial intelligence technologies including face recognition[4], voice recognition[5], and machine learning have achieved initial results in a variety of business scenarios, and realized the goals of reducing costs and improving efficiency to a large extent[6]. However, the implementation of these applications is not fully automated. Because any application algorithm cannot achieve the same performance on all tasks. Therefore, in all aspects of machine learning applications, such as feature engineering, network architecture search, parameter selection, etc., parameters need to be reconfigured. This project is usually completed by experienced experts, and their selection is also impossible copy. This has caused the threshold of artificial intelligence technology to be too high, and it has restricted the
speed of its application in enterprises to a large extent. Therefore, automatic machine learning technology came into being[7][8][9].

Automatic machine learning technology uses a data-driven approach to realize automatic machine modeling and automatic parameter adjustment. It automates the entire machine learning process and reduces the participation of human experts in the entire machine learning process. The overall idea of this concept is to use various intelligent search and optimization algorithms to replace humans to find specific data processing and recognition solutions suitable for the task. The automatic machine learning algorithm is shown in Fig. 1, covering every step of the machine learning workflow, including data preparation, feature engineering, network architecture search[2].

Feature engineering[10] is a series of processes that make the model obtain better prediction effects on unknown data sets by transforming the original data set, including feature extraction, feature selection, feature construction, and feature learning. It involves a lot of experimentation, in-depth domain knowledge, and what machines are not good at: intuition. The purpose of automated feature engineering is to iteratively create new feature sets until the machine learning model reaches a satisfactory accuracy score.

Network architecture search[11] is to construct a new convolutional neural network by automatically combining various layers and connecting channels to achieve an optimal network design, reducing the influence of human factors and making the network design more reasonable. Weng Y[1] et al. automatically constructed a Convolutional Neural Networks(CNN) based on a differential architecture search method, and reduced the Top-5 error by 5.16% in the Tiny-ImageNet competition. In the network architecture search, hyper-parameter search is a critical step.

There are generally two types of parameters in the learning model. One can be obtained from the learning process, and the other cannot be estimated from data and needs to be set by human experience. The latter is the hyper-parameter. The choice of hyper-parameters has a great influence on the final display effect of the model. For example, a complex model may have better expressive ability to handle different types of data, but it may also be impossible to train because of too many layers, which may cause the gradient to disappear. Another example is that if the learning rate is too large, the convergence effect may be poor, and if the learning rate is too small, the convergence speed may be too slow. However, with limited computing power, money, and time resources, the hyper-parameter search space is huge, and it is impossible to optimize the hyper-parameter combination directly by establishing an objective function. Therefore, adopting a suitable strategy to automatically optimize the hyper-parameters and selecting a set of optimal hyper-parameters for the model can not only improve the performance and effect of learning, but also save a lot of computing resources and labor costs[3].

Therefore, we explore a variety of hyper-parameter search strategies based on automatic machine learning in combination with computer vision classification task. Define the hyper-parameter search space, and realize the search optimization of hyper-parameters such as learning rate, hidden layer
number, batchsize, etc. through different strategies. Construct a classification model according to the hyper-parameters obtained from the search, and compare the classification accuracy of the model to compare the pros and cons of the search strategy. This article mainly discusses the framework of automatic machine learning, hyper-parameter search strategies, and the implementation of hyper-parameter search schemes combined with specific classification tasks. Finally, summarizes and prospects.

2. Related work

2.1. Research Status of Hyperparameter Search

The performance of the model is closely related to the setting of hyper-parameters[12]. Hyper-parameters can determine the complexity of the model. For example, the parameters of the hidden layer can determine the number of cascading convolution layers; the hyper-parameters can determine the training effect and final performance of the model, such as learning rate and batchsize. According to the specific algorithm model, the number of hyper-parameters varies from a few to dozens. Especially with the development of deep learning technology, neural networks are becoming more and more complex, and the hyper-parameters involved are also increasing. And different models will have different optimal hyper-parameter combinations. Taking the convolutional neural network as an example, the setting of network structure hyper-parameters such as the number of layers of the model and the number of neurons directly affects the performance of the network. The traditional hyper-parameter selection method is generally based on experience or random methods to taste the hyper-parameter settings, which makes a lot of manpower cost to be spent on the experiment of hyper-parameter adjustment, rather than the innovation of the network structure. In addition, it also makes some models difficult to reproduce and expand, and some technical achievements are difficult to promote and land[13]. Therefore, efficient hyper-parameter search strategies have become one of the hot spots that academia and industry pay attention to.

In the industry, the more commonly used methods are the grid method and the random method. Researchers usually perform repeated sampling based on experience, combining grid search and random search to find the optimal solution or the approximate optimal solution. It has been proved in the literature that in some hyper-parameter search scenarios, the average efficiency of the random method is better than that of the grid search method. However, for high-dimensional hyper-parameter search scenarios, the random search strategy is largely blind. This leads to a considerable amount of calculation, so the random search strategy is not an efficient hyper-parameter tuning method.

Lisha Li and Kevin Jamieso et al.[15] formulated hyper-parameter optimization as a non-random pure exploration problem, in which predefined resources (such as iterations, data samples, or features) are allocated to randomly sampled configurations, achieving a search acceleration of dozens of times. Aaron Klein[24] designed a Bayesian optimization program that uses model loss and training time as optimization indicators to optimize the hyper-parameters of support vector machines and deep learning networks. Experiments show that the optimization speed of the program is the same as other Bayesian optimization algorithms ten to one hundred times.

Microsoft launched an automated machine learning competition called "ChaLearn" in 2014. The competition lasted for three years. The theme of the competition was to develop a fully automatic "black box" learning system to achieve feature-based classification and regression problems. In the end, under the leadership of Frank Hutter, the AAD Freiburg team won the challenge. The team developed and released the AUTO-SKLEARN tool. In order to achieve the goal of the competition, the team used Bayesian optimization methods to optimize the hyper-parameters of various algorithms in the machine learning library[16]. In addition, Google's open source framework AutoML also implements model selection and hyper-parameter tuning functions. AutoML mainly uses exhaustive methods and some strategies for optimal model construction[17].
2.2. Classification Algorithm Based on Deep Learning

Convolutional neural networks are usually used in visual image tasks to achieve target classification. Convolutional neural network is a special multi-layer perceptron, which is very robust to translation, zoom, tilt etc. Similar to traditional neural networks, convolutional neural networks are also composed of an input layer, a hidden layer, and an output layer. The difference is that the hidden layer of a convolutional neural network consists of a convolutional layer, an activation function, a pooling layer, and a fully connected layer. Similar to traditional neural networks, convolutional neural networks are also composed of an input layer, a hidden layer, and an output layer. The difference is that the hidden layer of a convolutional neural network consists of a convolutional layer, an activation function, a pooling layer, and a fully connected layer. An input image can be regarded as a matrix composed of pixels. The convolution kernel slides sequentially on the matrix and convolves the picture pixels at the corresponding positions to achieve the purpose of extracting the characteristics of different frequency bands of the picture. The classic classification convolutional neural network is shown in the Fig. 2.

The convolution output of layer $l$ is:

$$a^l = f \left( \sum_{k=1}^{K} a^{l-1}_k * W^l_k + b^l \right)$$  \hspace{1cm} (1)

Where $K$ is the number of feature maps, $f$ is the activation function, $W$ is the convolution kernel, and $b$ is the bias.

In order to reduce the amount of calculation of the convolutional neural network, the pooling layer is introduced into the convolutional neural network. The pooling layer can keep important information in the image while reducing the size of the image, which greatly improves the computational efficiency of the convolutional neural network. After the input image undergoes multiple convolution-pooling processes, the output feature map must be fully connected and Softmax function. The fully connected layer can be expressed as:

$$a^L = f \left( W^L * a^{L-1} + b^L \right)$$  \hspace{1cm} (2)

The output feature vector is classified using the Softmax function, as following,

$$\sigma(z_j) = \frac{e^{z_j}}{\sum_{j=1}^{J} e^{z_j}}$$  \hspace{1cm} (3)

3. Hyperparameter search

In this section, we introduce the framework of hyper-parameter search and related search strategies.

3.1. Hyper-parameter Search Framework

The hyper-parameter search framework is shown in Fig. 3. First define the search space of each parameter, and then select a hyper-parameter search strategy to sample in the search space to obtain a set of initial hyper-parameters. Construct the network based on the initial parameters, and complete the
training process on the training set. Choose appropriate evaluation indicators to evaluate the model. If the accuracy of the model reaches the threshold, this set of parameters will be output as the optimal hyper-parameter configuration. Otherwise, resample in the search space, initialize the parameters, and continue the above process. If the number of iterations reaches the upper limit, the set of hyper-parameter configurations with the highest accuracy in the output process.

3.2. Hyper-parameter Search Strategy

Grid search[18] is to loop through all candidate parameter selections, try every possibility, and the best performing parameter is the final result. It arranges and combines the possible values of each parameter, and lists all possible combined results to generate a "mesh". Through loop traversal, the value of the constraint function and the objective function is calculated for each grid. For the points that meet the constraints, compare the values of the objective function one by one, discard the bad points, keep the good ones, and finally get the approximate solution of the optimal solution.

![Hyperparameter search framework.](image)

### Table 1. Hyperparameter search results of classifiers under different search strategies

| Dataset | Search Strategy | Batch size | Hidden size | lr | momentum | Average Time(s) | Acc |
|---------|----------------|------------|-------------|----|----------|-----------------|-----|
| MNIST   | Rand           | 16         | 256         | 0.01 | 0.001344410243357348 | 260.5 | 99.17 |
|         | TPE            | 32         | 512         | 0.01 | 0.2866708729774593  | 61.6  | 99.23 |
|         | Anneal         | 32         | 1024        | 0.01 | 0.4318147371048311  | 373.6 | 99.27 |
|         | Evolution      | 64         | 256         | 0.1  | 0.2731678906839663  | 143.6 | 99.22 |

Random search[14] is a method of using random numbers to find the optimal solution of function approximation, which is different from grid search. In a certain interval, it continuously generates random points randomly rather than tending to it, and calculates the value of its constraint function and
objective function. For points that meet the constraint conditions, compare the value of the objective function one by one, and discard the bad points. Keep good points, and finally get the approximate solution of the optimal solution. This method is based on probability theory. The more random points are taken, the greater the probability of obtaining the optimal solution. This method has the problem of poor accuracy, but the efficiency of finding the approximate optimal solution is higher than that of grid search. Random search is generally used for rough selection or census.

Naïve Evolution[22] comes from Large-Scale Evolution of Image Classifiers. It randomly initializes a population-based on search space. For each generation, it chooses better ones and does some mutation (e.g., change a hyper-parameter, add/remove one layer) on them to get the next generation. Naïve Evolution requires many trials to work, but it's very simple and easy to expand new features.

This annealing algorithm[23] begins by sampling from the prior, but tends over time to sample from points closer and closer to the best ones observed. This algorithm is a simple variation on the random search that leverages smoothness in the response surface. The annealing rate is not adaptive.

Hyperband[15] refers to factors such as time and computing resources as Budget, denoted by $B$. And suppose that the number of candidate hyper-parameter configurations at the beginning is $n$, then the budget allocated to each hyper-parameter group is $B/n$. The hyperband algorithm is a tradeoff between $n$ and $B/n$. That is, hyperband tries to use limited resources to explore as many configurations as possible and returns the most promising ones as a final result.

Population based training (PTB)[19] is an asynchronous automatic hyper-parameter adjustment optimization method. It perfectly combines parallel search and sequential optimization. The former executes many optimization tasks with different hyper-parameters in parallel, and the advantage is that it can use computing resources in parallel to find the optimal solution faster; the latter needs to use the previous information to perform the next hyper-parameter optimization, so it can only be executed serially, but generally can get a better solution. PBT perfectly combines the two methods and has the advantages of both.

BOHB (Bayesian Optimization + Hyperband)[20] is a hyper-parameter optimization algorithm that combines the advantages of bayesian optimization and hyperband. The hyperband algorithm has high computational efficiency, but the input hyper-parameter configuration is randomly selected. BOHB relies on HB (Hyperband) to determine how many sets of parameters are run each time and how many resources (budget) are allocated for each set of parameters. Once the parameters generated by bayesian optimization reach the number of configurations required for the iteration, these configurations are used to start the standard continuous halving process. Observe the performance of these parameters under different resource allocations (budgets) and use them as benchmark data for bayesian optimization model selection parameters in subsequent iterations. BOHB adopts model-based bayesian optimization algorithm to select hyper-parameter configuration as the input of hyperband algorithm. Its performance is better than bayesian optimization and hyperband, and it has higher computational efficiency.

The tree-structured Parzen Estimator (TPE)[21] is a sequential model-based optimization (SMBO) method. The SMBO method establishes a sequential model that estimates the performance of hyper-parameters based on historical measurement values, and then selects new hyper-parameters to be tested based on the model.

4. Experimental

In order to verify the important role of search strategy in the hyper-parameter optimization process, we compared the performance of several different search strategies.

4.1. Experimental Details

We verify the hyper-parameter search strategy on the classification task. The classification network consists of a convolutional network. We set four hyper-parameters, namely batch_size, hidden_size, learning rate (lr), momentum. Batch_size represents the number of training examples included in each batch, and its search space is [16, 32, 64, 128]. Hidden_size represents the number of convolutions contained in the hidden layer, and its search space is [128, 256, 512, 1024]. The learning rate represents
the initial learning speed of the training process, and its search space is [0.0001, 0.001, 0.01, 0.1]. Momentum is the parameter in the optimizer, the search space is [0, 1]. Both the training data set and the test data set are from MNIST Dataset. MNIST is a dataset of handwriting digits with labels. The dataset contains 60,000 examples for training and 10,000 examples for testing. These numbers have been standardized in size and are located in the center of the image. These images are 28×28 pixels grayscale images with values from 0 to 9.

4.2. Experimental Results
To fit a machine/deep learning model into different tasks/problems, hyper-parameters always need to be tuned. To get the optimal hyper-parameter setting, we adopt different strategies for hyper-parameter search. The experimental results are shown in Table 1. It proves that different search strategies will indeed bring different search results, and the efficiency of different search strategies is also very different.

5. Conclusion
At present, artificial intelligence and deep learning are becoming more popular, and the choice of hyper-parameters has a great influence on the final effect of the model. Machine learning algorithms such as SVM have hyper-parameters such as gamma, kernel to adjust, while neural network models have more hyper-parameters such as learning rate, optimizer, L1/L2 normalization, etc., which can be adjusted. Therefore, understanding and mastering better hyper-parameter tuning methods is very valuable in scientific research and engineering. We mainly introduce the hyper-parameter search strategy based on automatic machine learning, and compare the search results under different strategies on the classification task, and prove that the hyper-parameter search based on automatic machine learning can indeed improve the search efficiency and at the same time reduce the human intervention in the parameter search process, lower the threshold of deep learning, and make the deep learning network convenient in academia and industry. application.

References
[1] Weng Y, Zhou T, Liu L, et al. Automatic convolutional neural architecture search for image classification under different scenes[J]. IEEE Access, 2019, 7: 38495-38506.
[2] Yao Q, Wang M, Chen Y, et al. Taking human out of learning applications: A survey on automated machine learning[J]. arXiv preprint arXiv:1810.13306, 2018.
[3] Bergstra J S, Bardenet R, Bengio Y, et al. Algorithms for hyper-parameter optimization[C]//Advances in neural information processing systems. 2011: 2546-2554.
[4] Elmahmudi A, Ugail H. Deep face recognition using imperfect facial data[J]. Future Generation Computer Systems, 2019, 99: 213-225.
[5] Eljawad L, Aljamaeen R, Alsmadi M, et al. Arabic Voice Recognition Using Fuzzy Logic and Neural Network[J]. ELJAWAD, L., ALJAMAEEEN, R., ALSMADI, MK, AL-MARASHDEH, I., ABOUELMAID, H., ALSMADI, S., HADDAD, F., ALKHASAWNEH, RA, ALZUGHOUL, M. & ALAZZAM, MB, 2019: 651-662.
[6] Guo X, Shen Z, Zhang Y, et al. Review on the application of artificial intelligence in smart homes[J]. Smart Cities, 2019, 2(3): 402-420.
[7] Drori I, Krishnamurthy Y, Lourencio R, et al. Automatic machine learning by pipeline synthesis using model-based reinforcement learning and a grammar[J]. arXiv preprint arXiv:1905.10345, 2019.
[8] Santos A, Castelo S, Felix C, et al. Visus: An interactive system for automatic machine learning model building and curation[C]//Proceedings of the Workshop on Human-In-the-Loop Data Analytics. 2019: 1-7.
[9] Blasch E, Majumder U K, Rovito T, et al. Automatic machine learning for target recognition[C]//Automatic Target Recognition XXIX. International Society for Optics and Photonics, 2019, 10988: 109880L.
[10] Nargesian F, Samulowitz H, Khurana U, et al. Learning Feature Engineering for
Classification[C]/IJCAI. 2017: 2529-2535.
[11] Cai H, Chen T, Zhang W, et al. Efficient architecture search by network transformation[J]. arXiv preprint arXiv:1707.04873, 2017.
[12] Hutter F, Lücke J, Schmidt-Thieme L. Beyond manual tuning of hyperparameters[J]. Künstliche Intelligenz, 2015, 29(4): 329-337.
[13] Wainer J, Cawley G. Empirical evaluation of resampling procedures for optimising SVM hyperparameters[J]. The Journal of Machine Learning Research, 2017, 18(1): 475-509.
[14] Bergstra J, Bengio Y. Random search for hyper-parameter optimization[J]. The Journal of Machine Learning Research, 2012, 13(1): 281-305.
[15] Li L, Jamieson K, DeSalvo G, et al. Hyperband: A novel bandit-based approach to hyperparameter optimization[J]. The Journal of Machine Learning Research, 2017, 18(1): 6765-6816.
[16] Feurer M, Klein A, Eggensperger K, et al. Efficient and robust automated machine learning[C]/Advances in neural information processing systems. 2015: 2962-2970.
[17] Zoph B, Le Q V. Neural architecture search with reinforcement learning[J]. arXiv preprint arXiv:1611.01578, 2016.
[18] Ndiaye E, Le T, Fercoq O, et al. Safe grid search with optimal complexity[C]/International Conference on Machine Learning. 2019: 4771-4780.
[19] Jaderberg M, Dalibard V, Osindero S, et al. Population based training of neural networks[J]. arXiv preprint arXiv:1711.09846, 2017.
[20] Falkner S, Klein A, Hutter F. BOHB: Robust and efficient hyperparameter optimization at scale[J]. arXiv preprint arXiv:1807.01774, 2018.
[21] Kastner M, Nellen N, Jahn C. Model-based Optimisation with Tree-structured Parzen Estimation for Container Terminals[C]/ASIM 2019 Simulation in Produktion und Logistik. Wissenschaftliche Scripten, 2019: 489-498.
[22] Real E, Moore S, Selle A, et al. Large-scale evolution of image classifiers[J]. arXiv preprint arXiv:1703.01041, 2017.
[23] Bellio R, Ceschia S, Di Gaspero L, et al. Feature-based tuning of simulated annealing applied to the curriculum-based course timetabling problem[J]. Computers & Operations Research, 2016, 65: 83-92.
[24] Klein A, Falkner S, Bartels S, et al. Fast bayesian optimization of machine learning hyperparameters on large datasets[C]/Artificial Intelligence and Statistics. PMLR, 2017: 528-536.