Parallel Hyper-parameter Optimization for Loan Default Prediction

1st Zhiwei Ruan
Zhejiang University of Science & Technology
Hangzhou, China
rnzhiw@163.com

2nd Zhe Chen
Nanjing University
Nanjing, China
wztxy89@163.com

3nd Yulai Zhang
Zhejiang University of Science & Technology
Hangzhou, China
zhangyulai@zust.edu.cn

***Abstract*—Hyper-parameter selection can significantly impact the performance of the machine learning model. Due to the large scale of data, parallel hyper-parameter selection is necessary for practical applications. Compared with the widely-used grid search and random search, Bayesian optimization is a global-wise method proposed in recent years with fewer iterations. We consider these three methods for hyper-parameter selection in their parallel implementations. In many real-world applications such as Internet financial lending, delayed loan review often hurts business efficiency, thus faster processing is required. To remedy this problem, we propose a unified framework for parallel hyper-parameter search, and prove the effectiveness by comparing the accuracy and time cost under different numbers of CPU cores in parallel.**

***Index Terms*—Bayesian optimization, distributed system, loan default prediction, machine learning**

I. Introduction

Hyper-parameter search methods [1] [2] have been widely used in automatic machine learning. However, due to the large-scale and immediacy of data, the hyper-parameter selection is always the most time-consuming part of the entire machine learning process.

Suppose the evaluation of the machine learning model is defined as a mapping from hyper-parameters to generalization performance. In that case, hyper-parameters selection can be regarded as an optimization problem for this black box function with a high evaluation cost. In traditional machine learning engineering, simple methods such as grid search [3], random search [4], and genetic algorithm [5] are common used. Grid search and random search rely on the computing power of the machine by conducting a large number of experiments under different combinations of hyper-parameters. Moreover, genetic algorithms [5] require enough initial sample points, and the optimization efficiency is not satisfied, so they are unsuitable for hyper-parameter selection.

Recently, Bayesian optimization [6] was proposed to deal with the hyper-parameters selection problem of machine learning. The probabilistic proxy model [7] in Bayesian optimization is adopted to approximate the current black-box objective function, and the acquisition function [8] is adopted to estimate the most likely position of the best point under the current data conditions. Compared with prior methods, Bayesian optimization utilizes limited sampling values to obtain the optimal value of a complex function in fewer evaluations. However, this kind of algorithm with sequential relationships is still inefficient without parallelism.

Parallel Bayesian optimization utilizes multiple threads to perform hyper-parameter search simultaneously, making it robust to non-convex problems with concurrency. Parallel computing [9] refers to the use of multiple components to execute computing tasks, mainly for large-scale data processing. Its basic component is a high-performance single-core CPU or multi-core CPU computer system. Typical parallel computing involves executing a program by utilizing multiple computers or processors at the same time. Ideally, many engines are running when processing in parallel, making the program run efficiently. Even machines with single-core processors can connect to others through the network for parallel processing.

In the process of hyper-parameter search, parallelism can enable the program to run on multiple threads or multi-cores, which significantly reduces time consumption. For this technique, the core issues are task scheduling and how to optimize threads from the perspective of the number of threads, data contention, thread-safety, storage, and data organization [10].

When the global economy is reeling from the COVID-19 pandemic, the probability of nonperforming loans is soaring. To avoid the occurrence of loan default, banks and other financial institutions will evaluate the credit risk of borrowers when issuing loans and decide whether to give loans according to the results. The model of loan default prediction needs to be trained with the daily updated data, and the training time becomes the bottleneck of loan issuing speed. Therefore, it is essential to employ a parallel Bayesian optimization algorithm to predict loan default and improve the efficiency of the banking business.

The rest of this paper is organized as follows: The preliminaries of Bayesian optimization and parallelism are described in Section 2. Section 3 introduces the proposed method in detail. Section 4 validates the performance of the algorithm through experiments. Conclusions are offered in Section 5.

II. PRELIMINARIES

A. Grid Search

The grid search method (GSM) is an approach to find the optimal parameter combination. The search area is divided into grids, the intersections are verified in turn, and the optimal parameter combination is selected [11]. GSM will divide the

search area in advance. Its disadvantage is that the more grids divided, the more combinations of hyper-parameters yielded. With the rise of calculation, the optimization time will also increase. If the mesh is not fine enough, the best combination of hyper-parameters may not be found.

The grid search algorithm [12] was applied to search the hyper-parameter of the support vector machine in 2003. However, the most basic grid search wastes a lot of time on many unnecessary combinations of hyper-parameters. To reduce the computational cost, Hsu et al. [13] presented a two-step grid search algorithm for hyper-parameter selection. Specifically, they divided the grid search into two steps: coarse search and fine search. Later, Bao et al. [14] proposed to find the path of maximum error reduction in model training, to optimize the grid search path, and achieved good results in time series prediction. Ou et al. [15] adopted a certain mechanism to reduce the size of the training set, to speed up the efficiency of grid search.

B. Random Search

The random search algorithm does not traverse all hyper-parameter combinations but selects a random value of each hyper-parameter for random combinations, thus greatly reducing the computational amount of hyper-parameter search and reducing the search time. It can be seen that random search uses random sampling in the parameter space [16] to replace the grid search in the form of grid search. For continuously changing hyper-parameters, the random search takes it as a distribution for sampling. Therefore, random search is suitable for a large number of hyper-parameters, and its performance is improved compared to grid search.

C. Bayesian optimization

Bayesian optimization is a distribution estimation algorithm [17] proposed by Pelikan et al. [18] in 2002. This is a black-box optimization algorithm for solving extreme value problems of functions whose expressions are unknown. This algorithm predicts the probability distribution of the function value at any point by constructing the objective function f for the function value at a set of sampling points, which is implemented by random forest logistic regression [19]. According to the regression results, it constructs a collection function to measure whether each point is worth exploring and solve the extreme value of the collection function to determine the next sampling point. Finally, the extreme value of this set of sampling points is returned as the extreme value of the function. This algorithm is used in the auto-machine learning algorithm to determine the hyper-parameter of the machine learning algorithm.

Some network architecture search (NAS) algorithms [20] also adopted the Bayesian optimization algorithms. Because the Bayesian optimization framework is efficient, and it is useful when the evaluation cost of the function f is expensive, and the derivative of the non-convex multimodal f cannot be obtained. There are two core elements in the Bayesian optimization framework, one is the probabilistic proxy model [21], which is composed of a prior distribution, and the other is a collection function that selects one of the best advantages from the proxy model for evaluation.

Let $f(x)$ be the mapping from the hyper-parameter vector $\boldsymbol{x}$ to the generalization performance of the model, where $x \in X$, $x \in R^d$, and X is the hyper-parameter space with dimension d. The goal of hyper-parameter selection is to find the d-dimensional hyper-parameter x^* that maximizes the generalization performance of the model in the hyper-parameter space.

Take finding the maximum value as an example:

$$x^* = \arg\max f(x), \tag{1}$$

where $f(x)$ expresses the measurement of model hyper-parameters such as generalization accuracy and other model generalization indicators. Because it takes a large amount of calculation and a long time to conduct model training and evaluation with a set of hyper-parameters, $f(x)$ is a black-box objective function with high evaluation cost.

Assuming that the known data is $D(1:t) = (x_i, y_i), i = 1, 2, \ldots, t$. y_i is the test set accuracy of the model trained with the hyper-parameter x_i. In the following, the current existing evaluation data will be abbreviated as D. We hope to estimate the optimal value of the hyper-parameters with fewer data (i.e., a smaller t).

D. Random Forest

Random forest is an ensemble algorithm [22], which belongs to the Bagging type. It combines several weak classifiers [23], by voting or averaging, to achieve higher accuracy and generalization performance. Its success is largely due to randomness and the forest structure, one of which makes it resistant to overfitting, the other makes it more accurate.

The weak classifier of random forest is a classification and regression tree (CART) [24]. When the dependent variable of the dataset is a constant value, the tree algorithm is a regression tree, and the mean value observed by the leaf nodes can be used as the predicted value. Otherwise, when it is a discrete value, the tree algorithm is a classification tree. Therefore, it can solve the classification problem significantly.

The current popular method for feature selection of random forest is the Gini coefficient. The Gini coefficient is selected based on whether each child node achieves the highest purity. For a general decision tree, if there are K classes in total and the probability that the sample belongs to the k class is P_k. Then, the Gini index of the probability distribution is:

$$Gini(p) = \sum_{k=1}^{K} p_k(1 - p_k) = 1 - \sum_{k=1}^{K} p_k^2. \tag{2}$$

For the CART tree, since it is a binary tree, it can be expressed as:

$$Gini(p) = 2p(1 - p). \tag{3}$$

When we traverse each dividing point of each feature, the feature $A = a$ is used to separate D into two parts, termed $D1$ (the set of samples satisfying $A = a$) and $D2$ (the set

of samples not satisfying $A = a$). When we traverse each segmentation point of each feature, when using feature $A = a$, divide D into two parts, namely $D1$ (a sample set that satisfies $A = a$), and $D2$ (a sample set that does not satisfy $A = a$). Then, under the condition of characteristic $A = a$, the Gini index of D is:

$$Gini(p) = \frac{|D_1|}{D} Gini(D_1) + \frac{|D_2|}{D} Gini(D_2). \quad (4)$$

The modeling process of random forest is described as follows:

Step 1. Using Bootstrap sampling method [25], M datasets with the same number of samples are generated from the original training set S, and each dataset contains P features.

Step 2. For each dataset, a CART decision tree is constructed. In constructing the sub-tree, not all features are used to select node fields, but p feature dimensions are randomly selected.

Step 3. Let each decision tree grow as fully as possible.

Step 4. For the RF model composed of M CART trees, the category with the highest votes is used for the final judgment result using the voting method.

The RF modeling process is demonstrated in Fig. 1.

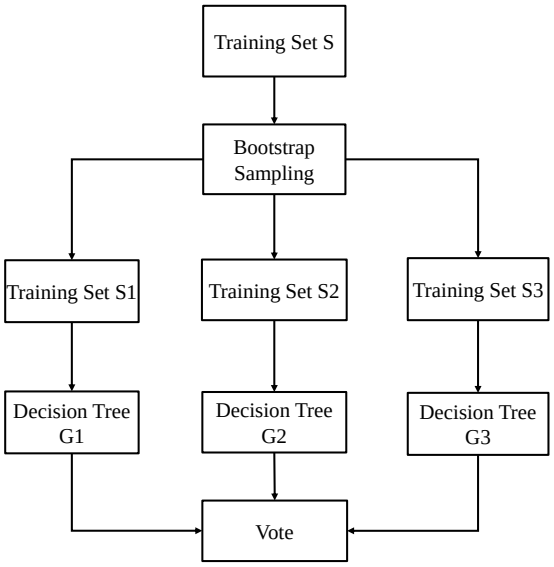

Fig. 1. RF modeling flowchart.

III. METHOD

In this paper, we apply the parallelization of the grid search, random search, and Bayesian optimization to the random forest. The search space of grid search and random search consists of the number of decision trees and the number of candidate split attributes in the random forest algorithm. The parameter search space of parallel Bayesian optimization algorithm based on Gaussian process is continuous or numerical variables. However, in machine learning, many parameters are of discrete type. Therefore, the parallel Bayesian optimization algorithm based on random forest can solve this problem greatly. Parallel Bayesian optimization mainly includes two parts, probability proxy model and acquisition function.

A. Algorithm Description of Bayesian optimization

The probabilistic proxy model is determined by the following formula:

$$p(f|D_{1:t}) = \frac{p(D_{1:t})p(f)}{p(D_{1:t})}, \quad (5)$$

where f is the unknown objective function. $D_{1:t} = \{(x_1, f_1), (x_2, f_2), \cdots, (x_t, f_t)\}$ represents the collection of collected sample points. f_t is the value of current sample point. $P(D_{1:t}|f)$ is the likelihood distribution of y. $p(f)$ is the prior probability distribution model [26] of f. $p(D_{1:t} : t)$ is the marginal likelihood distribution [27] of marginalized f. $p(f|D_{1:t})$ is the posterior probability distribution of f, which is the confidence of the unknown function after the prior probability distribution is modified.

According to the parameter types of the model, the probabilistic agent model can be divided into parametric agent model and non-parametric agent model. During the process of optimization and update, the number of parameters of the parameter proxy model remains unchanged. On the contrary, the non-parametric proxy model has many parameters and can better describe the unknown objective function due to its greater flexibility and extensibility. Among the latter, Gaussian process [28] is widely used and can achieve the fitting effect of countless multi-layer neural networks.

The Gaussian process is a set of random variables. For the RF model to be optimized, Gaussian process is the parameter combination of RF model, which can be expressed as follows:

$$\begin{cases} f(x) \sim gp(m(x), k(x, x')) \\ m(x) = E[f(x)] \\ k(x, x') = E[(f(x) - m(x))(f(x') - m(x'))] \end{cases}, \quad (6)$$

in the formula, $m(x)$ is a mean value function. To simplify the calculation, $m(x)$ is always set to 0. $k(x, x')$ is a covariance function, then $f(x) \sim gp(0; k(x, x'))$. The prior distribution of the unknown function can be expressed as $p(f_{1:t}|D_{1:t}) \sim N(0, K_t)$. $f_{1:t}$ is the f corresponding to the sampling point. The value set of $\{f_1, f_2, \cdots, f_t\}$, K_t is the covariance matrix formed by the covariance function:

$$K_t = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_t) \\ \vdots & \ddots & \vdots \\ k(x_t, x_1) & \cdots & k(x_t, x_t) \end{bmatrix}. \quad (7)$$

When a new set of evaluation samples (x_{t+1}, f_{t+1}) are added to the set of all evaluation points, the covariance matrix is updated to.

$$\begin{cases} K_t = \begin{bmatrix} K_t k_t^T \\ k_t k(x_{t+1}, x_{t+1}) \end{bmatrix} \\ k_t = \left[k(x_{t+1}, x_1), k(x_{t+1}, x_2), \cdots, k(x_{t+1}, x_t)\right] \end{cases}. \quad (8)$$

Then, we use the updated covariance matrix to estimate the posterior probability distribution of f_{t+1}, as shown in the following formula:

$$\begin{cases} p(f_{t+1}|D_{1:t+1}, x_{t+1}) \sim N(\mu, \sigma^2) \\ \mu = k_{t+1}^T K_{t+1}^{-1} f_{1:t+1} \\ \sigma^2 = k_{t+1}(x_{t+1}, x_{t+1}) - k_{t+1}^T K_{t+1}^{-1} k_{t+1} \\ k_{t+1} = \left[k(x_{t+1}, x_1), k(x_{t+1}, x_2), \cdots, k(x_{t+1}, x_{t+1})\right] \end{cases}. \tag{9}$$

The collection function searches the next evaluation point purposefully from the parameter space, mainly including three methods: probability of improvement (PI), expected improvement (EI), and upper confidence bound (UCB). When PI is selected as the collection function, PI represents the possibility that the next sample point may improve the optimal objective function. The formula is shown as follows:

$$\alpha_t(x : D_{1:t}) = p(f(x) \leq v^* - \varepsilon) = \varphi(\frac{v^* - \varepsilon - \mu_t(x)}{\sigma_t(x)}), \tag{10}$$

where v^* is the optimal value of the current objective function. φ is the cumulative density function of the standard normal distribution. ε is the balance parameter. By adjusting the size of ε, it can avoid falling into the situation of local optimal and implement the global search for the optimal value.

B. Algorithm Framework of Random Forest

Random forest algorithm is a combination of learning algorithms based on decision tree [29]. In constructing a single tree, random forest ensures the independence of these trees by randomly selecting some variables or features to participate in the division of tree nodes. After the random forest is obtained, each decision tree in the forest makes a judgment on the sample when a new sample is fed, and then makes a prediction according to their votes. The algorithm framework is as follows:

Algorithm 1: Random Forest

Data: Training set T_{train}, number of decision trees in the forest N_{tree}, number of predictive variables in each sample M, number of variables in each tree node involved in partitioning M_{tree}, size of bootstrap Sample $S_{sampsize}$.
Result: Prediction result of decision tree.

for $i_{tree} \leftarrow 0$ **to** N_{tree} **do**
  Using the training set T_{train} to generate Bootstrap data samples with a size of $S_{sampsize}$.
  Using the Bootstrap data generated above to construct an unpruned tree i_{tree}. In the process of generating tree i_{tree}, randomly select M_{tree} variables from M and select the best variable according to the Gini value to branch.
end
return prediction result of decision tree

C. Algorithm Framework of Parallel Bayesian optimization

In this method, the random forest algorithm is used as the agent model. In each iteration, the existing observations (Hyperparameter evaluation value) are used to fit into a random forest model, and according to this model, the evaluation value of the unknown hyper-parameter combination is predicted. Each tree in the random forest will give an evaluation value, so we can build a Gaussian distribution [30] according to the evaluation value of N trees and use computer multithreading to fit the model. The algorithm framework is as follows:

Algorithm 2: Parallel Bayesian optimization

Data: Randomly initialize n_0 points in the forest.
Result: The point x^* that maximizes f(x) in the evaluated data.

Using the initialized n_0 points, and get n_0 point results$\{(x_1, f(x_1)), (x_2, f(x_2)), \cdots, (x_{n0}, f(x_{n0})\}$, determine the maximum number of iterations.
Using Algorithm 1 to train the random forest model on multiple threads.
while $n \leq N$ **do**
  Generate candidate points through large-scale random sampling [31], bring the candidate points into the trained random forest to obtain information such as the mean and standard deviation, and calculate the acquisition function $a_n(x)$
  Choose the point that maximizes $a_n(x)$ as x_{n+1}
  Get the new point x_{n+1}, and bring it in to get $f(x_{n+1})$ Update $n = n + 1$, retrain the random forest on multiple threads for the current n points
  return The point x^* that maximizes f(x) in the evaluated data
end
return prediction result of decision tree

IV. EXPERIMENTS

Starting from the assumption that multi-threaded parallel hyper-parameter search can improve the performance, we perfectly implement the application of loan default prediction on the basis of the random forest algorithm. All the models are trained on an Nvidia Tesla K80 GPU. We conduct experiments on a 12-core, 48-thread server from one core to 10 cores. The results show that, under the parallel condition, the accuracy of these three methods significantly improved, and the time consumption significantly decreased.

A. Datasets and Implementation Details

The dataset used in this paper comes from a competition project on the Kaggle data science competition platform. The dataset contains 250,000 samples, of which 150,000 samples are used as the training set, and 100,000 samples are used as the test set. The training set has 150,000 historical data of borrowers, of which the 10,026 default samples, accounting for

6.684% of the total sample. The loan default rate is 6.684%, and 139,974 non-default samples account for 93.316% of the total sample. We can observe that this dataset is extremely unbalanced. This dataset includes information such as the borrower's age, income, family, and loan status. There are 11 variables in total, among which "SeriousDlqin2yrs" is the label, and the other 10 variables are the predicted features.

We first train a random forest model using the dataset. Table I shows the parameter settings of the random forest model. We run grid search, random search, and Bayesian optimization under different CPU cores, change the value of "n_jobs", and observe the prediction accuracy and running time of the model. For precise comparison, these experiments are repeated 10 times.

TABLE I
PARAMETER SETTINGS OF RANDOM FOREST MODEL

Name	Value
n_estimators	100
oob_score	True
min_samples_split	2
min_samples_leaf	50
class_weight	balanced_subsample
bootstrap	True

The model evaluation indexes used in this experiment are the accuracy and time of prediction. Our dataset adopts a three-fold cross-validation method. According to the experimental results, the hyper-parameter search method of parallel Bayesian optimization has the highest efficiency, and when "n_jobs" is 10, the prediction accuracy of the program is the largest, and the running time is the shortest.

B. Data Processing and Analysis

1) Data Dictionary and Missing Value Statistics:

We analyze the attributes in the loan default prediction dataset, and all attributes are shown in Table II.

TABLE II
PROPERTIES IN THE LOAN DEFAULT PREDICTION DATASET

Name	Type
SeriousDlqin2yrs	0/1
RevolvingUtilizationOfUnsecuredLines	Percentage
Age	Integer
NumberOfTime30-59DaysPastDueNotWorse	Integer
DebtRatio	Percentage
MonthlyIncome	Integer
NumberOfOpenCreditLinesAndLoans	Integer
NumberOfTimes90DaysLate	Integer
NumberRealEstateLoansOrLines	Integer
NumberOfTime6089DaysPastDueNotWorse	Integer
NumberOfDependents	Integer

We use Numpy and Pandas to count the missing values of each attribute column. Combined with the meaning of attributes with missing values, we decide to use the average value of the column to fill them. The number of missing attributes is illustrated in Table III.

TABLE III
STATISTICS ON THE NUMBER OF MISSING ATTRIBUTES OF THE DATASET

Name	Missing Value
SeriousDlqin2yrs	0
RevolvingUtilizationOfUnsecuredLines	0
Age	0
NumberOfTime30-59DaysPastDueNotWorse	0
DebtRatio	0
MonthlyIncome	29731
NumberOfOpenCreditLinesAndLoans	0
NumberOfTimes90DaysLate	0
NumberRealEstateLoansOrLines	0
NumberOfTime6089DaysPastDueNotWorse	0
NumberOfDependents	3924

2) Data Type Distribution:

By categorizing the defaults of borrowers, we can observe that the classification is obviously unbalanced. The default statistics are reported in Fig. 2.

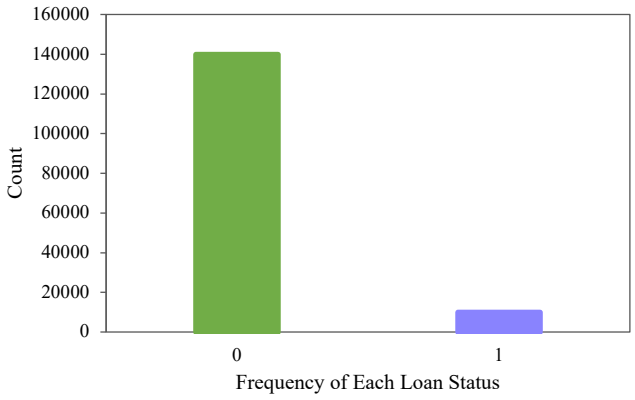

Fig. 2. Statistics on the number of loan defaults. Horizontal-axis: Whether the loan is in default. Vertical-axis: Number of loan defaults.

3) Feature Correlation Analysis:

We analyze the correlation between each feature attribute and label attribute, divide the attribute column into segments, and separately count the proportion of people in each segment and the default rate. Now take "age" as an example. Other correlation analysis methods are similar. We can be seen from Fig. 3 that the feature age has a strong correlation with the label.

The relationship between the "NumberOfOpenLoans" and the default rate is shown in Table IV. The standard deviation of the default rate is 0.986, and the coefficient of variation CV is 0.986. Therefore, this feature is weakly correlated with the label and can be deleted.

C. Parallel Grid Search

In this section, we use parallel grid search to predict loan default based on random forest. The process is as follows:

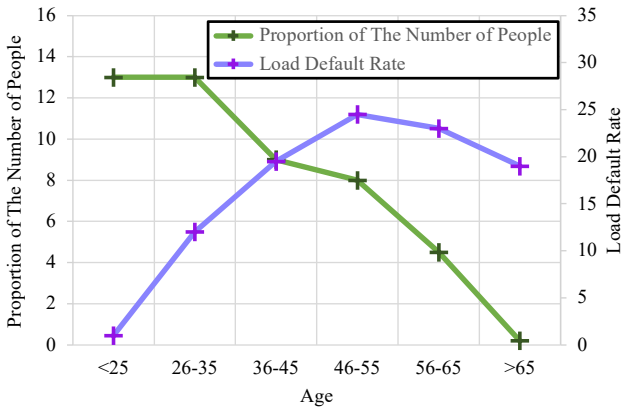

Fig. 3. Correlation analysis diagram of feature age and label attributes. Horizontal-axis: Age; Main vertical-axis: Proportion of the number of people; Secondary vertical-axis: Load default rate.

TABLE IV
TABLE OF THE RELATIONSHIP BETWEEN THE NUMBER OF LOANS AND THE DEFAULT RATE

Frequency	Default rate
Less than 5	8.41812
between 6 and 10	5.53808
between 11 and 15	6.18147
between 16 and 20	6.86573
between 21 and 25	6.72298
between 26 and 30	7.89809
more than 30	7.34463

- Load training set, test set, and preprocessing data.
- Decompose the training data into training_new and test_new, where training_new is used for training and test_new is used for verification.
- Use Imputer to process data and use Mean to replace missing values.
- Use training_new data to build a random forest model.
- Deal with unbalanced data distribution.
- Use parallel grid search with CrossValidation to perform the hyper-parameter adjustment.
- Output the best model and make predictions on the test data.

We carried out three cross-validations on the dataset and repeated the experiment 10 times. In Fig. 4, we can see the prediction accuracy and prediction speed of loan repayment under different CPU core numbers.

D. Parallel Random Search

In this section, we use parallel random search to predict loan default based on random forest. The process is as follows:

- Load training set, test set, and preprocessing data.
- Decompose the training data into training_new and test_new, where training_new is used for training and test_new is used for verification.

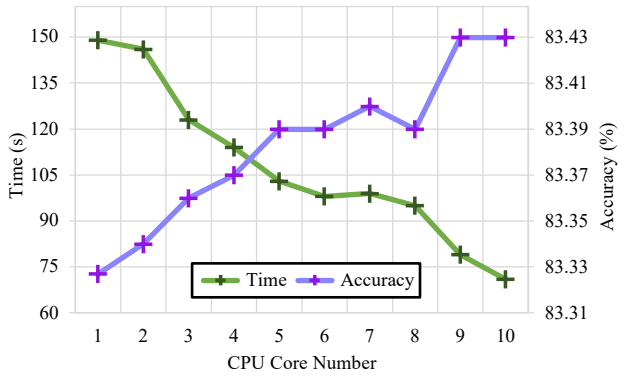

Fig. 4. Experimental results of loan default prediction by parallel grid search with different CPU core numbers. Horizontal-axis: the number of cores. Main vertical-axis: the time required to complete the loan default forecast. Secondary vertical-axis: the accuracy of the loan default forecast.

- Use Imputer to process data and use Mean to replace missing values.
- Use training_new data to build a random forest model.
- Deal with unbalanced data distribution.
- Use parallel random search with RandomizedSearchCV to perform the hyper-parameter adjustment.
- Output the best model and make predictions on the test data.

We carried out three cross-validations on the dataset and repeated the experiment 10 times. In Fig. 5, we can see the prediction accuracy and prediction speed of loan repayment under different CPU core numbers.

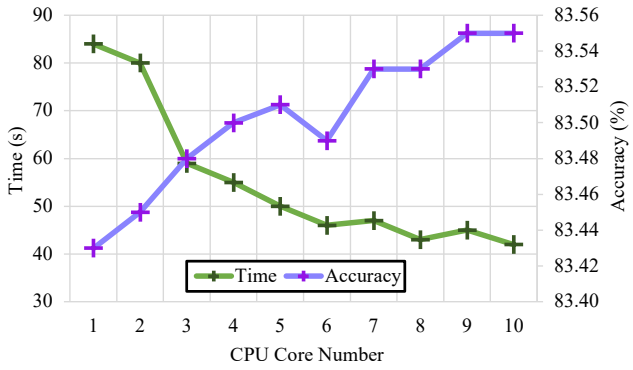

Fig. 5. Experimental results of loan default prediction by parallel random search with different CPU core numbers. Horizontal-axis: the number of cores. Main vertical-axis: the time required to complete the loan default forecast. Secondary vertical-axis: the accuracy of the loan default forecast.

E. Parallel Bayesian Optimization

In this section, we use the parallel Bayesian optimization search method to predict loan default based on random forest. The process is as follows:

- Load training set, test set, and preprocessing data.

- Decompose the training data into training_new and test_new, where training_new is used for training and test_new is used for verification.
- Use Imputer to process data and use Mean to replace missing values.
- Use training_new data to build a random forest model.
- Deal with unbalanced data distribution.
- Use Bayesian optimization with BayesSearchCV to perform the hyper-parameter adjustments.
- Output the best model and make predictions on the test data.

We carried out three cross-validations on the dataset and repeated the experiment 10 times. The accuracy and speed of loan repayment under different CPU core numbers are demonstrated in Fig. 6.

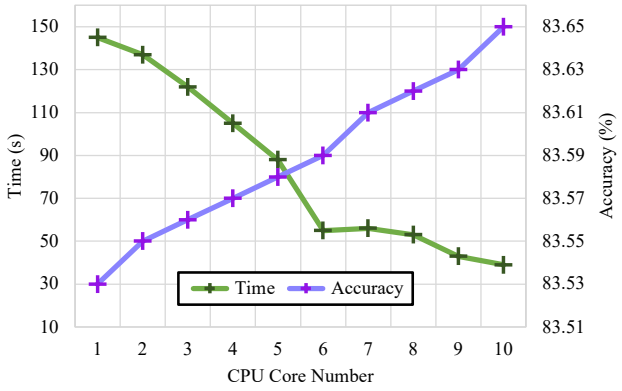

Fig. 6. Experimental results of loan default prediction by parallel random search with different CPU core numbers. Horizontal-axis: the number of cores. Main vertical-axis: the time required to complete the loan default forecast. Secondary vertical-axis: the accuracy of the loan default forecast.

V. CONCLUSION

In this paper, we propose the application of parallel hyper-parameter search to loan default prediction. Specifically, we implement parallel hyper-parameter search based on three methods, including grid search, random search, and Bayesian optimization. Compared with the other two methods, parallel Bayesian optimization can achieve better accuracy and speed, solving the problem of the slow speed of the original method in the high-dimensional search space. Experiments show that parallel Bayesian optimization has an excellent performance on the loan default prediction task. In the future, we will further improve the parallel Bayesian optimization algorithm and apply it to more scenarios.

ACKNOWLEDGMENT

This research was funded by the Young Scientists Fund of the National Natural Science Foundation of China (grant numbers: 61803337).

REFERENCES

[1] L. Li, K. Jamieson, A. Rostamizadeh, E. Gonina, M. Hardt, B. Recht, and A. Talwalkar, "A system for massively parallel hyperparameter tuning 2018," *arXiv preprint arXiv:1810.05934*, 2021.

[2] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in neural information processing systems*, vol. 24, 2011.

[3] G. Li, W. Wang, W. Zhang, Z. Wang, H. Tu, and W. You, "Grid search based multi-population particle swarm optimization algorithm for multimodal multi-objective optimization," *Swarm and Evolutionary Computation*, vol. 62, p. 100843, 2021.

[4] H. Mania, A. Guy, and B. Recht, "Simple random search provides a competitive approach to reinforcement learning," *arXiv preprint arXiv:1803.07055*, 2018.

[5] A. Klein, S. Falkner, N. Mansur, and F. Hutter, "Robo: A flexible and robust bayesian optimization framework in python," in *NIPS 2017 Bayesian optimization workshop*, 2017.

[6] S. Tripoppoom, X. Ma, R. Yong, J. Wu, W. Yu, K. Sepehrnoori, J. Miao, and N. Li, "Assisted history matching in shale gas well using multiple-proxy-based markov chain monte carlo algorithm: The comparison of k-nearest neighbors and neural networks as proxy model," *Fuel*, vol. 262, p. 116563, 2020.

[7] H. Wang, B. van Stein, M. Emmerich, and T. Back, "A new acquisition function for bayesian optimization based on the moment-generating function," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2017, pp. 507–512.

[8] G. Shomron, T. Horowitz, and U. Weiser, "Smt-sa: Simultaneous multithreading in systolic arrays," *IEEE Computer Architecture Letters*, vol. 18, no. 2, pp. 99–102, 2019.

[9] K. Asanovic, R. Bodik, J. Demmel, T. Keaveny, K. Keutzer, J. Kubiatowicz, N. Morgan, D. Patterson, K. Sen, J. Wawrzynek *et al.*, "A view of the parallel computing landscape," *Communications of the ACM*, vol. 52, no. 10, pp. 56–67, 2009.

[10] B. Dunkel and N. Soparkar, "Data organization and access for efficient data mining," in *Proceedings 15th International Conference on Data Engineering (Cat. No. 99CB36337)*. IEEE, 1999, pp. 522–529.

[11] Y. Liu, L. Bian, Y. Meng, H. Wang, S. Zhang, Y. Yang, X. Shao, and B. Wang, "Discrepancy measures for selecting optimal combination of parameter values in object-based image analysis," *ISPRS journal of photogrammetry and remote sensing*, vol. 68, pp. 144–156, 2012.

[12] C. Staelin, "Parameter selection for support vector machines," *Hewlett-Packard Company, Tech. Rep. HPL-2002-354R1*, vol. 1, 2003.

[13] C.-W. Hsu, C.-C. Chang, C.-J. Lin *et al.*, "A practical guide to support vector classification," 2003.

[14] Y. Bao and Z. Liu, "A fast grid search method in support vector regression forecasting time series," in *International Conference on Intelligent Data Engineering and Automated Learning*. Springer, 2006, pp. 504–511.

[15] Y.-Y. Ou, C.-Y. Chen, S.-C. Hwang, and Y.-J. Oyang, "Expediting model selection for support vector machines based on data reduction," in *SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme-System Security and Assurance (Cat. No. 03CH37483)*, vol. 1. IEEE, 2003, pp. 786–791.

[16] R. W. Emerson, "Convenience sampling, random sampling, and snowball sampling: How does sampling affect the validity of research?" *Journal of Visual Impairment & Blindness*, vol. 109, no. 2, pp. 164–168, 2015.

[17] Q. Zhang, J. Sun, E. Tsang, and J. Ford, "Hybrid estimation of distribution algorithm for global optimization," *Engineering computations*, 2004.

[18] M. Pelikan, K. Sastry, and D. E. Goldberg, "Scalability of the bayesian optimization algorithm," *International Journal of Approximate Reasoning*, vol. 31, no. 3, pp. 221–258, 2002.

[19] R. E. Wright, "Logistic regression." 1995.

[20] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: a big comparison for nas," *arXiv preprint arXiv:1912.06059*, 2019.

[21] C. S. W. Ng, A. J. Ghahfarokhi, M. N. Amar, and O. Torsæter, "Smart proxy modeling of a fractured reservoir model for production optimization: implementation of metaheuristic algorithm and probabilistic application," *Natural Resources Research*, vol. 30, no. 3, pp. 2431–2462, 2021.

[22] X. Shi, Y. Li, H. Li, R. Guan, L. Wang, and Y. Liang, "An integrated algorithm based on artificial bee colony and particle swarm optimization," in *2010 Sixth international conference on natural computation*, vol. 5. IEEE, 2010, pp. 2586–2590.

[23] E. Grossmann, "Adatree: boosting a weak classifier into a decision tree," in *2004 Conference on Computer Vision and Pattern Recognition Workshop*. IEEE, 2004, pp. 105–105.

[24] R. J. Lewis, "An introduction to classification and regression tree (cart) analysis," in *Annual meeting of the society for academic emergency medicine in San Francisco, California*, vol. 14. Citeseer, 2000.

[25] M. V. Johns, "Importance sampling for bootstrap confidence intervals," *Journal of the American Statistical Association*, vol. 83, no. 403, pp. 709–714, 1988.

[26] S. Weinberg, "A priori probability distribution of the cosmological constant," *Physical Review D*, vol. 61, no. 10, p. 103505, 2000.

[27] R. K. Vinayak, W. Kong, G. Valiant, and S. Kakade, "Maximum likelihood estimation for learning populations of parameters," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6448–6457.

[28] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein, "Deep neural networks as gaussian processes," *arXiv preprint arXiv:1711.00165*, 2017.

[29] N. Frosst and G. Hinton, "Distilling a neural network into a soft decision tree," *arXiv preprint arXiv:1711.09784*, 2017.

[30] T. Pan, J. Zhao, W. Wu, and J. Yang, "Learning imbalanced datasets based on smote and gaussian distribution," *Information Sciences*, vol. 512, pp. 1214–1233, 2020.

[31] J. Castro, D. Gómez, E. Molina, and J. Tejada, "Improving polynomial estimation of the shapley value by stratified random sampling with optimum allocation," *Computers & Operations Research*, vol. 82, pp. 180–188, 2017.