INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
ISSN 1841-9836, e-ISSN 1841-9844, 14(4), 557-573, August 2019.

Hyperparameter Importance Analysis based on N-RReliefF Algorithm

Y. Sun, H. Gong, Y. Li, D. Zhang

Yunlei Sun*
College of Computer & Communication Engineering
China University of Petroleum(East China), China
No.66, West Changjiang Road, Huangdao District, Qingdao 266580, China
*Corresponding author: sunyunlei@upc.edu.cn

Huiquan Gong
Faculty of Information Technology
Beijing University of Technology, China
No.100, Pingleyuan, Chaoyang District, Beijing, 100124, China
xinel_ghq@126.com

Yucong Li
College of Computer & Communication Engineering
China University of Petroleum(East China), China
No.66, West Changjiang Road, Huangdao District, Qingdao 266580, China
291731931@qq.com

Dalin Zhang
National Research Center of Railway Safety Assessment
Beijing Jiaotong University, China
No.3, Shangyuancun, Haidian District, Beijing, 100044, China
dalin@bjtu.edu.cn

Abstract: Hyperparameter selection has always been the key to machine learning. The Bayesian optimization algorithm has recently achieved great success, but it has certain constraints and limitations in selecting hyperparameters. In response to these constraints and limitations, this paper proposed the N-RReliefF algorithm, which can evaluate the importance of hyperparameters and the importance weights between hyperparameters. The N-RReliefF algorithm estimates the contribution of a single hyperparameter to the performance according to the influence degree of each hyperparameter on the performance and calculates the weight of importance between the hyperparameters according to the improved normalization formula. The N-RReliefF algorithm analyses the hyperparameter configuration and performance set generated by Bayesian optimization, and obtains the important hyperparameters in random forest algorithm and SVM algorithm. The experimental results verify the effectiveness of the N-RReliefF algorithm.
Keywords: Hyperparameter optimization, Bayesian optimization, RReliefF Algorithm.

1 Introduction

In the process of machine learning, the performance of the algorithm highly depends on the selection of hyperparameters, which has always been a crucial step in the process of machine learning. Automated machine learning, represented by Bayesian optimization algorithm, has recently achieved great success in hyperparameter optimization, which exceeds the performance of human experts in some cases.

However, there are some constraints and limitations in the selection of hyperparameters for Bayesian optimization algorithm. Researchers and users can only get the hyperparameter configuration after the operation of Bayesian optimization algorithm and cannot get the importance analysis of the hyperparameter configuration. Therefore, it is necessary to study the algorithm of hyperparameter importance analysis based on a wide range of data sets, so that researchers and users can understand which hyperparameter adjustment will significantly improve the performance of the algorithm.

This paper introduces the basic principle of Bayesian optimization algorithm and optimized random forest and SVM based on OpenML100 data set. By comparing with grid search and random search algorithm, it could be seen that the hyperparameter performance of Bayesian optimization algorithm is high and time-consuming. Therefore, we used the hyperparameter configuration and performance data generated by Bayesian optimization algorithm to analyze hyperparameter importance. Hyperparameter configuration based on the experimental data, we used N-RReliefF algorithm to evaluate the importance of interaction between hyperparameter, so as to determine the important hyperparameter in random forest algorithm and SVM algorithm.

2 Related works

Hyperparameter selection is a key step in machine learning process [12]. From the initial manual selection to automatic optimization using algorithms later, the evolving optimization algorithms have made great contributions to improving performance. Hyperparameter selection algorithms can be roughly divided into four categories: traditional algorithms (e.g., grid search algorithm [26], random search algorithm [2]), heuristic optimization algorithm [4, 18, 30], meta-learning algorithm [13, 22], Bayesian optimization algorithm [14, 28], etc.

The disadvantage of these hyperparameter selection algorithms is that they cannot provide researchers and users with information about the importance analysis of the selected hyperparameters and cannot understand the impact of different hyperparameters and their interactions on performance. Scientists have proposed a method for evaluating the importance of hyperparameter machine learning algorithms. Sequential parameter optimization (SPO) [1] is a model-based parameter optimization approach. SPO starts by running the target algorithm with parameter configurations. It then builds a response surface model based on Gaussian process regression and uses the models predictions and predictive uncertainties to determine the next parameter cofiguration to evaluate. In 2007, Nannen et al. [20] proposed an evolutionary algorithm for parameter correlation estimation. In 2009, Chiarandini et al. [6] used a linear mixed effect model to design and analyze the optimization algorithm. With a mixed-effects multi-linear regression they [8] assessed the individual and joint effect of problem features on the performance of both algorithms, within and across the instance classes defined by benchmark parameters. In [21] Probst formalized the problem of tuning from a statistical point of view, define data-based defaults and suggest general measures quantifying the tunability of hyperparameters of algorithms. Falkner proposed a new hyperparameter optimization method-BOHB [10], which combines the benefits of both Bayesian optimization and bandit-based methods, consistently outperforms both Bayesian optimization and Hyperband on a wide range of problem types. Breiman [29] uses random forests to assess the importance of attributes: If attributes are deleted from the data set, performance will be degraded, indicating that this attribute is important. Based on this principle, Forward Selection [15] predicts the performance of machine learning algorithms using a subset of hyperparameters, which is initialized to be empty and greedily chooses the next most important hyperparameter. Ablation Analysis [15] requires default settings and optimization settings, and calculates the ablation trajectory, which reflects the contribution of hyperparameters to the performance difference between the two settings.

3 N-RReliefF algorithm design

This paper introduces the basic principle of hyperparameter optimization of Bayesian optimization algorithm based on Gauss process modeling. To hyperparameter configuration and performance data generated by optimization process, we used N-RReliefF algorithm to evaluate the importance of hyperparameter, so that users can understand the important hyperparameter. As shown in Fig. 1, the input of hyperparameter optimization includes: algorithm A with configuration space, instance set and cost matrix C. The optimal hyperparameter configuration can be obtained by modeling and optimization of Bayesian optimization algorithm. At the same time, the trajectory of searching for the optimal hyperparameter configuration, as well as the hyperparameter configuration and its performance data can be obtained. Based on the output data, N-RReliefF algorithm is used to evaluate the importance of hyper-parameters and the effect of interaction between them on performance. Next, the two parts are explained in detail.

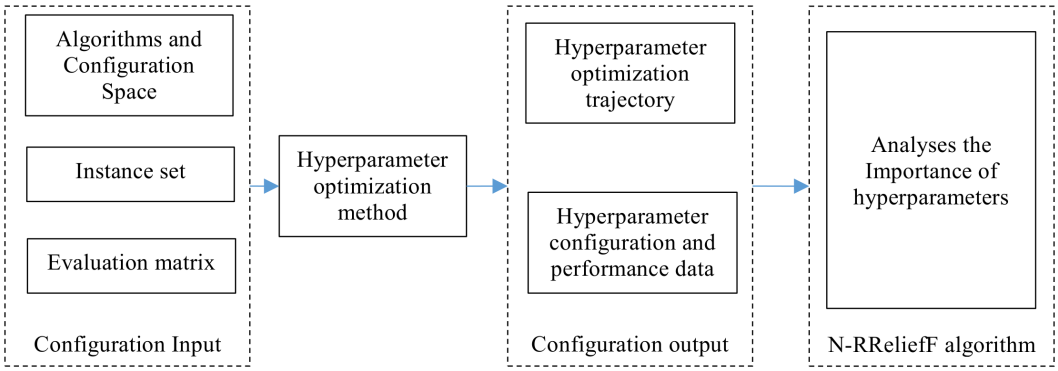

Figure 1: Algorithm configuration and analysis workflows

3.1 Bayesian optimization algorithm

Question definition

The hyperparameter selection problem of machine learning model is regarded as an unknown black box function optimization problem reflecting generalization performance. Let $\theta_1, ..., \theta_n$ represent n hyperparameters of machine learning algorithm, whose domain space is expressed as $\Theta_1, ..., \Theta_n$. The configuration space of the algorithm is defined as $\Theta = \Theta_1 \times ... \times \Theta_n$. The hyperparameters $\theta \in \Theta$ are trained on the training data set D_{train}, and the loss function $l(\theta, D_{train}, D_{valid})$ of machine learning algorithm is obtained on the verification set.

The objective function of hyperparametric combinations in optimization problems is defined as follows [11]:

$$f(\theta) = \frac{1}{k} \sum_{i=1}^{k} l(\theta, D_{train}^{(i)}, D_{valid}^{(i)}) \tag{1}$$

$$\theta^* = arg_{\theta \in \Theta} min f(\theta)$$

For an unknown objective function $f(\theta)$, Bayesian optimization algorithm searches for a hyperparameter configuration that minimizes the function $f(\theta)$ on a bounded set Θ. The basic idea of Bayesian optimization algorithm is to construct a probability model for function $f(\theta)$, establish evaluation criteria based on this model, determine the next hyperparameter for evaluation in configuration space Θ, and at the same time, all the information available in previous evaluation

can be reused for learning the shape of objective function [9]. The use of historical data enables Bayesian optimization to find the minimum value of complex non-convex functions through fewer evaluations, but the corresponding cost is to perform more calculations to determine the next sampling point.

Therefore, the key of Bayesian optimization algorithm can be summarized as the following two parts [7]:

- Establishing a probabilistic model to evaluate the objective function instead of the original complex objective function, which is expensive to evaluate.
- The acquisition function is constructed by using the posterior information of the probability model to determine the next sampling point.

Question definition

There are many models to model the objective function, among which the Gauss process has been proved to be a convenient and powerful model optimization algorithm. Gauss process is a set of random variables. If these random variables obey Gauss distribution, then these random variables are Gauss process. A Gauss process consists of a mean function $m: \Theta \to R(m(\theta) = 0)$ and a covariance function $(kernelfunction)m: \Theta \to R(m(\theta) = 0)$.

Its concrete form is [25]:

$$f(\theta) \sim GP(m(\theta), k(\theta, \theta')) \tag{2}$$

Mean function $m(\theta) = E[f(\theta)]$, covariance function $k(\theta, \theta') = E[(f(\theta) - m(\theta)(f(\theta') - m(\theta')))]$, for simplicity, usually set mean function $m(\theta) = 0$. Covariance function is a function of calculating the similarity between two data points in Gauss process, which specifies the smoothness and amplitude of the unknown objective function. The selection of covariance function is very important, which affects the matching degree between Gauss process and data properties. This paper chooses Matérn $\frac{5}{2}$ Kernel Function [15]. Compared with other popular Gauss Kernel Functions, it has fewer constrained smoothness assumptions and is very helpful to the optimization settings.

The formulas are as follows:

$$k_{\frac{2}{5}}(\theta, \theta') = \theta_0(1k_{\frac{2}{5}}(\theta, \theta') = \theta_0(1+\sqrt{5}d_\lambda(\theta, \theta')+\frac{3}{5}d_\lambda^2(\theta, \theta'))e^{-\sqrt{5}d_\lambda(\theta,\theta')}+\sqrt{5}d_\lambda(\theta, \theta')+\frac{3}{5}d_\lambda^2(\theta, \theta'))e^{-\sqrt{5}d_\lambda(\theta,\theta')} \tag{3}$$

Among them, θ_0 and λ denote the covariance amplitude and length dimensions respectively, and $d_\lambda(\theta, \theta') = (\theta, \theta')^T diag(\lambda)(\theta - \theta')$ denotes the Mahalanobis distance.

Given the input set $G = \theta_1, .., \theta_t$ and the output $y = f(\theta_1), f(\theta_2), ..., f(\theta_t)$ of the observation set, the Gauss process $GP(m, k)$ is adjusted. Due to the mixing of noise, the observed value is likely to be affected, and there is a certain deviation from the actual output value. In order to approach the actual situation, noise should be added to the probability distribution in the experiment.

The formula is as follows:

$$y = f(\theta) + \varepsilon \tag{4}$$

The noise ε satisfies the independent and identically distributed Gauss distribution: $p(\varepsilon) \sim N(0, \sigma^2)$. The prior distribution of y is $y \sim N(0, S + \sigma^2 I)$, I is n-dimensional unit matrix, and S represents the covariance matrix $k(\theta, \theta')$.

The joint prior distribution of the observed value y and predicted value $f(\theta_*)$ is as follows:

$$\begin{bmatrix} y \\ f(\theta_*) \end{bmatrix} \ N\left(0, \ \begin{bmatrix} S+\sigma^2 I & K_* \\ K_*^T & K_{**} \end{bmatrix}\right) \tag{5}$$

In the formula, θ_* represents the predictive input, $K_*^T = k(\theta_1, \theta_*), k(\theta_2, \theta_*), ..., k(\theta_t, \theta_*), K_{**} = k(\theta_*, \theta_*)$.

According to the estimation of posterior probability of input value by Gauss distribution, the predicted distribution at a given test point θ_* is expressed as [7]:

$$p(f(\theta_*)|G, y, \theta_*) = N(\overline{f(\theta_*)}, cov(f(\theta_*))) \tag{6}$$

Among,

$$\overline{f(\theta_*)} = K_*^T[S+\sigma^2 I] - 1y, cov(f(\theta_*)) = K** - K*T[S+\sigma^2 I] - 1K* \tag{7}$$

GP evaluates $f(\theta_*)$ with all historical observation points as conditions, and then uses the posterior mean and variance of prediction to select the next set of superparameters on the basis of balanced development and exploration acquisition functions.

Acquisition function

This section introduces the active strategy of selecting the next evaluation point in Bayesian optimization: acquisition function, which is a function$\alpha : \chi \times \Theta \to R$ mapped from input space χ, observation space $\mathbb{R}$ and hyperparameter space Θ to real space.

The function is constructed from a posterior distribution obtained from known observation data sets $D_{1:t}$, and the next evaluation point θ_{t+1} is selected by maximizing its guidance:

$$\theta_{t+1} \in max_{x \in \chi} \alpha_t(\theta; D_{1:t}) \tag{8}$$

This paper uses the promotion-based (Expected Improvement, EI) strategy [19], which performs better. EI strategy has been proved to be effective in the evaluation of global optimization of many black box functions.

The closed form of EI strategy in Gauss process is as follows:

$$a_{EI}(\theta; D_{1:t}) = E[max(f_{min} - f(\theta), 0)] \tag{9}$$

f_{min} is the optimal solution based on observation set so far. Formula below describes the balance between the development and exploration of new sampling points. If the standard deviation of the prediction point is large, it means that the understanding of the point is small, and it is worth exploring; if the mean value is large, it means that the point may be the maximum point, which is worth developing. Because the initial sampling data is very few, the algorithm will sample the points with large standard deviation; when the sampling points increase, the standard deviation decreases, and the algorithm tends to the points with large sampling mean, and eventually converges to the global optimal value.

3.2 Importance assessment of hyperparameter

Bayesian optimization algorithm obtains the optimal hyperparameter configuration of machine learning algorithm through two important steps: GP process and iteration of acquisition function. However, due to the abstraction and black-box nature of its internal process, it is impossible to analyze the importance of hyperparameters. In order to increase the interpretability of the hyperparameters selected by Bayesian optimization algorithm and to understand the importance ranking of the hyper-parameters of the algorithm itself, an N-RReliefF algorithm is proposed to evaluate the importance of the hyperparameters.

Relief algorithm

The Relief algorithm [17] was originally used in the field of feature selection. The main idea of the Relief algorithm is to estimate the ability of this feature to distinguish adjacent samples according to the degree of discrimination of each attribute to an instance. Relief's process is to randomly select an instance I in the training set, search for k instances I_j which are similar to instance I, and the samples which belong to the same category with instance I in I_j are called H, and the samples of different categories are called M [9].

The weight $W[A]$ of attributes A are estimated according to the values of example I and H and M in I_j, and the approximate values of the probabilistic difference shown in formula below are obtained:

$$W[A] = P(diff.valueofA|nearestinst.fromdiff.class) - P(diff.valueofA|nearestinst.fromsameclass) \tag{10}$$

If instances I and H have different attribute values A, then attribute a separates two instances from the same kind of instances, and the formula is expressed as reducing the weight estimation $W[A]$. If instances I and M have different attribute values A, then attribute values a separates two instances from different instances in a formula that correspondingly increases the weight estimation $W[A]$. The bigger the weight of the feature is, the stronger the classification ability of the feature is; on the contrary, the weaker the classification ability of the feature is [27].

Among them, the difference of attribute value A between different instances I_1 and I_2 in $W[A]$ is defined as [27]:

$$dif(A, I_1, I_2) = 0, value(A, I1) = value(A, I2)1, value(A, I1) \neq value(A, I2) \tag{11}$$

The above formulas can be further calculated as follows:

$$dif(A, I_1, I_2) = \frac{|value(A, I_1) - value(A, I_2)}{max(A) - min(A)} \tag{12}$$

N-RReliefF algorithm

The importance of hyperparameters of machine learning algorithm is evaluated. The input data used are the hyperparameter configuration and performance data of machine learning algorithm. On such data sets, performance data are continuous values, and can not be calculated using the latest samples of the same or different types presented in Relief. In order to solve this problem, RReliefF [23] introduces the probability of two different instances to determine whether two instances belong to the same class. The probability definition can simulate and predict the relative distance between two instances. RReliefF is currently mainly used in the field of feature selection. This section improves and fuses the RReliefF algorithm and proposes an N-RReliefF algorithm to evaluate the importance of the interaction between hyperparameters.

The main idea of the N-RReliefF algorithm for evaluating the importance of hyperparameters is to estimate the contribution of each hyperparameter to performance according to the degree of influence of each hyperparameter on performance. The N-RReliefF algorithm consists of two parts. The first part is to evaluate the importance of a single hyperparameter. In the training set, we randomly select a hyperparameter configuration instance I and select k instances I_j which are similar to the instance I. In order to judge whether the instance I_j and I belong to the same class, we introduce probability simulation and prediction of the relative distance between the two instances [24], as shown in formula below. Among them, θ denotes the probability of different hyperparameter values, P_{difA} denotes the probability of different categories in similar instances, P_{difC} denotes the probability of different categories in similar instances, and $P_{difC|difA}$ denotes

the probability of different categories in similar instances with different hyperparameter values.

$$P_{difA} = P\left(difvalue\left(\theta\right)|similarinstance\right)\ P_{difC} = P(difprediction|similar\ instance) \quad (13)$$

According to conditional probability:

$$P_{difC|difA} = P(difprediction|difvalue(\theta)\ similar\ instance) \quad (14)$$

Combined formula (11) is available:

$$W[\theta] = PdifC|difA \times PdifAPdifC - (1 - PdifC|difA) \times PdifA1 - PdifC \quad (15)$$

Repeat the above process k times to get $W[\theta]$, and evaluate the importance of a single hyperparameter according to the weight $W[\theta]$.

The second part is to measure the importance of hyper-parameters and understand the influence of the interaction between hyperparameters on the performance of machine learning algorithm. The N-RReliefF algorithm divide the contribution of the hyperparameters by the sum of all the contributions of the hyperparameters to normalization to calculates the importance of the hyperparameters.

The formula is defined as (17):

$$W[\theta_m \& \theta_n] = \frac{e^{W[\theta_m]+W[\theta_n]}}{e^{\sum W[\theta]}} \quad (16)$$

θ_m and θ_n denote two different hyperparameters, and $\sum W[\theta]$ denotes the sum of the importance weights of all hyperparameters.

This formula is a distortion of the normalization formula, which can more stably evaluate the influence of the interaction between hyperparameters on the performance. The whole process of the N-RReliefF algorithm is as follows: firstly, the contribution (weight) vector of each hyperparameter to the performance is calculated, the elements in the vector are accumulated, the importance of each hyperparameter is sorted by the accumulated value, and t significant hyperparameters are selected to enter the candidate subset of the hyperparameter, thus the iteration process begins. In the iteration process, the importance weights between the significant hyperparameters are calculated, and the importance weights between all the hyperparameters and the hyperparameters are finally output.

The flow chart of the algorithm is shown in algorithm 1.

In the algorithm, N_{dc}, $N_{dA}[\theta]$ and $N_{dC\&dA}[\theta]$ represent weight vectors of different predicted values (line 8), weight vectors of different attributes (line 10), and weight vectors of different predicted values and attributes (line 11). The algorithm calculates the importance weight $W[\theta]$ of each hyperparameter in line 16. H_{list} denotes the most important first t hyperparameters.

Variables $d(i, j)$ (lines 8, 10, 11) are used to measure the distance between two instances R_i and I_j. The basic principle is that closer instances should have greater impact:

$$d(i,j) = \frac{d_1(i,j)}{\sum_{l=1}^{k} d_1(i,l)}\ d_1(i,j) = e^{-(\frac{rank(R_i,I_j)}{\sigma})^2} \quad (17)$$

$rank(R_i, I_j)$ is the ranking of distance between instance I_j and instance R_i. σ is used to control distance, which is customized by users. Because the expected results can be interpreted by probability, divide the contribution of each instance in k-nearest neighbor instance by the sum of all K contributions to normalization. The reason for using rankings instead of actual distances is that actual distances are related to specific issues, and by using rankings, we ensure that recent instances always have the same impact on weights.

Algorithm 1 N-RRelief algorithm

Input: A training set D_{train} consisting of a hyperparameters set $G(\theta_1, .., \theta_n)$ and performance set $C(c_1, .., c_n)$, $D_{train} = ((\theta_1, c_1), (\theta_2, c_2), ..., (\theta_n, c_n))$;
Output: Weight evaluation vector W of Interaction between hyperparameters.
Initialize N_{dc}, $N_{dA}[\theta]$, $N_{dC\&dA}[\theta]$, $w[\theta]$, H_{list} to 0.
for $i = 1$ to m **do**
    Random selection example R_i;
    Selecting k Neighbors I_j;
    for $j = 1$ to k **do**
        $N_{dC} = N_{dC} + diff(C, R_i, I_j) \cdot d(i, j)$;
        for $\theta = 1$ to n **do**
            $N_{dA}[\theta] = NdA[\theta] + diff(\theta, Ri, Ij) \cdot d(i, j)$;
            $N_{dC\&dA}[\theta] = NdC\&dA[\theta] + diff(C, Ri, Ij) \cdot diff(\theta, Ri, Ij) \cdot d(i, j)$;
        end for
    end for
end for
for $\theta = 1$ to t **do**
    $W[\theta] = N_{dC\&dA}[\theta]/N_{dC} - (N_{dA}[\theta] - N_{dC\&dA}[\theta])/(m - N_{dC})$;
    According to the importance weights of hyperparameters from high to low, and taking the first t into the set of hyperparameters, calculate the interactive importance of hyperparameters in H_{list}:
    for $a = 1$ to t **do**
        for $b = a + 1$ to t **do**
            $W[\theta_m \& \theta_n] = \frac{e^{W[\theta_m] + W[\theta_n]}}{e^{\sum W[\theta]}}$;
        end for
    end for
end for
Return $W[\theta]$ and $W[\theta_m \& \theta_n]$

The N-RReliefF algorithm evaluates the importance measure of the interaction between hyperparameters, assigns a weight value to each hyperparameter, and evaluates how the weight is affected by the hyperparameters, so as to determine a series of the most important algorithm hyperparameters. Hyperparameters with large weight and hyperparameters combination indicate that the adjustment of these hyperparameters is very important to the performance of machine learning algorithm, while other hyperparameters with small weight mean that even if the hyperparameters are adjusted repeatedly, the influence on the performance of machine learning algorithm is not great. When the computational resources are limited, we can focus on adjusting the hyperparameters with large weights. For the hyperparameters with small weights, we can use their default values in machine learning algorithm. When sufficient computing resources are available, it is still recommended to adjust all hyperparameters. N-RReliefF algorithm can identify important hyperparameters in machine learning algorithm. The results can guide Bayesian optimization algorithm to optimize the hyperparameters and improve the performance and efficiency of the algorithm.

4 Experiment

The experiment in this section is divided into two parts. The first part is to analyze the performance of several hyperparameter optimization algorithms in machine learning hyperparameter optimization process through experiments, and select the best performance hyperparameter optimization algorithm, using its hyperparameter configuration history data to evaluate the importance of hyperparameter experiments; the second part is to use the N-RReliefF algorithm based on the hyperparameter configuration history data obtained from experiments. Evaluate the importance of super parameters and combinations of machine learning algorithms and obtain a series of super parameters which have a significant impact on the performance of the algorithms. Bayesian optimization algorithm is used to validate the effectiveness of the results obtained by N-RReliefF algorithm, which further ensures that the importance ranking of the machine learning algorithm is accurate.

4.1 Hyperparameter tuning

Experimental data and methods

In order to fully verify the superiority of Bayesian optimization algorithm, all experiments are carried out on the data set from OpenML100 [3] this section. OpenML100 is a benchmark suite that contains 100 data sets from different domains and 500 to 1000 data points with balanced distribution.

Two classification methods were analyzed on data sets from OpenML100: SVMs [5] and Random Forest [15]. For SVMs [5], two types of kernels are analyzed: radial basis function and sigmoid kernels. All algorithms use the same data pretreatment steps, including interpolation of missing data and coding of discrete features by One-Hot-Encoding. Support Vector Machine (SVM) is sensitive to the proportion of input variables, so it is necessary to standardize the input variables.

After data pretreatment, each classification method is optimized by using grid search, random search and Bayesian optimization algorithm. In order to ensure that the hyperparameter optimization method does not produce any deviation because of the configuration space of the hyperparameter, this section uses the same type and range of hyperparameter for the three hyperparameter optimization methods. The hyperparameter types and ranges of the two algorithms are shown in Tables 1 and 2.

Table 1: Hyperparameter configuration space in SVM algorithm

SVM hyperparameters	Types	Configuration space	Default value
complexity	float	[0.001, 1000.0]	[1.0]
coef0	integer	[0.0, 10.0]	[0.0]
gamma	float	$[2^{-15}, 2^{3}]$	$[2^{-15}]$
shrinking	categorical	true, false	[true]
tolerance	float	$[10^{-5}, 10^{-1}]$	$[10^{-2}]$
imputation	categorical	mean, median, mode	[mean]

Table 2: Hyperparameter configuration space in random forest algorithm

Random forest hyperparameter	Types	Configuration space	Default value
split criterion	categorical	entropy, gini	[entropy]
bootstrap	categorical	true, false	[true]
max.features	float	[0.1, 0.9]	[0.1]
min.samples leaf	integer	[1, 20]	[1]
min.samples split	integer	[2, 20]	[2]
imputation	categorical	mean, median, mode	[mean]

Experimental results

Table 3 and 4 show the SVMs performance results of different hyperparameter tuning methods under the RBF kernel function and sigmoid kernel function, respectively. Table 5 shows the performance results of random forest algorithm under different hyperparameter optimization methods. In order to increase the credibility of the results, this section uses 100 data sets from different domains to verify the average results of each index of SVMs and random forest algorithm using default parameters, grid search parameters, random search parameters and Bayesian optimization algorithm parameters. The experimental results show that Bayesian optimization algorithm can obtain the optimal performance and running time in the process of SVMs and random forest algorithm optimization.

Table 3: Average performance results of the SVM (RBF) algorithms under different hyperparameter tuning algorithms

Parameter adjustment method	Precision	$F_1 - score$	Recall	Runtime
No (using the default value)	0.22	0.30	0.47	1.8s
Grid search algorithm	0.78	0.80	0.78	40.3s
Random search algorithm	0.82	0.82	0.82	28s
Bayesian optimization algorithm	0.94	0.92	0.94	18.2s

Table 4: Average Performance Results of the SVM (sigmoid) Algorithms under Different hyperparameter Tuning Algorithms

Parameter adjustment method	Precision	$F_1 - score$	Recall	Runtime
No (using the default value)	0.54	0.37	0.49	2s
Grid search algorithm	0.80	0.82	0.81	42.9s
Random search algorithm	0.85	0.83	0.81	30s
Bayesian optimization algorithm	0.95	0.94	0.94	20.1s

Table 5: Average performance results of the random forest algorithms under different hyperparameter tuning algorithms

Parameter adjustment method	Precision	$F_1 - score$	Recall	Runtime
No (using the default value)	0.87	0.89	0.87	2.5s
Grid search algorithm	0.93	0.93	0.90	45.2s
Random search algorithm	0.94	0.93	0.93	35.3s
Bayesian optimization algorithm	0.98	0.97	0.97	25.5s

4.2 Importance assessment of hyperparameter

In Section 4.1, two classifiers use hyperparameter optimization N-RReliefF algorithm to generate a large number of hyperparameter configuration and performance data (including the optimal hyperparameter configuration and performance) during the operation process. These data are used to evaluate the importance of hyperparameter in the two classifiers.

Each classifier is shown by a figure and a table. Fig. 2 shows the average hyperparameter importance of hyperparameter and hyperparameter combination by bar graph. The X axis represents the hyperparameter and hyperparameter combination name, and the Y axis represents the hyperparameter importance weight. The higher the weight, the greater the impact of hyperparameters or combinations on performance. If it can not be adjusted to the appropriate value, the accuracy of the algorithm will be reduced.

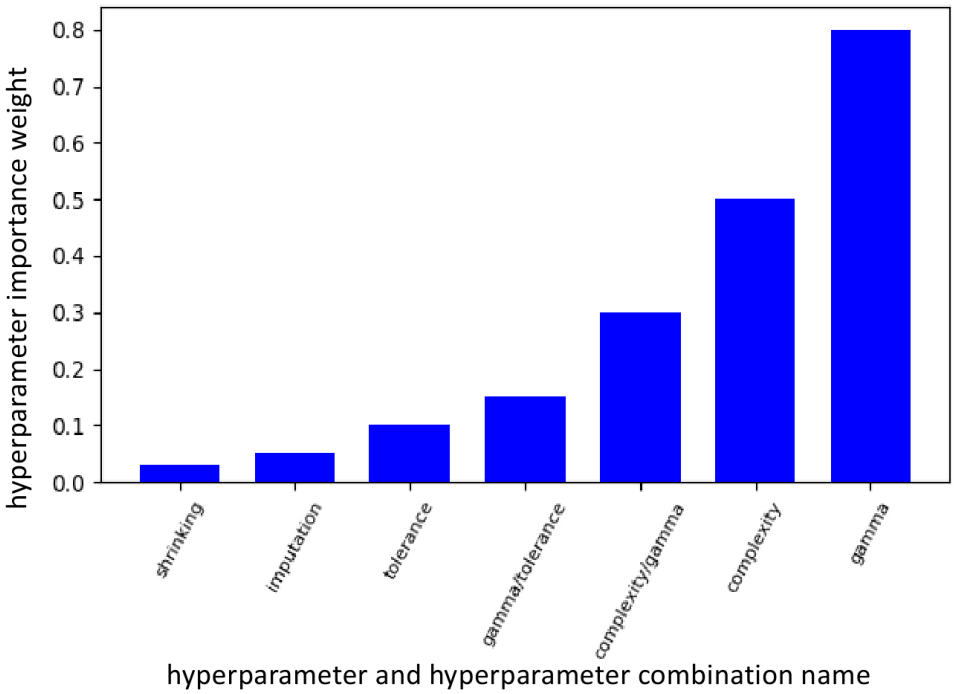

Figure 2: N-RReliefF algorithm evaluate hyperparameter and combination importance-SVM(RBF kernel)

Table 6 uses the Bayesian optimization algorithm to fix the two most important hyperparameters (using their default values) and adjust the other hyperparameters according to the most important hyperparameters selected by the N-RReliefF algorithm. The performance and running time of hyperparameter optimization are compared when all hyperparameters are adjusted

and only the first three hyperparameters with great importance are adjusted according to the algorithm.

Table 6: Bayesian optimization algorithm to adjust the performance of different hyperparameters-SVM(RBF kernel)

Hyperparameters	**Precision**	$F_1 - score$	**Recall**	**Runtime**
Fixed Gamma	0.35	0.32	0.35	16.3s
Fixed Complexity	0.75	0.70	0.72	15.8s
Adjust all hyperparameters	0.94	0.92	0.94	18.2s
Adjust N-RReliefF and select top three hyperparameters	0.92	0.92	0.90	9.4s

SVM results: Fig. 2 and Fig. 3 analyze two kinds of kernel functions of SVMs, RBF kernel function and sigmoid kernel function, respectively. The experimental results clearly show that the most important hyperparameter in both cases are gamma, and the second important is complexity. This conclusion is validated by the experiments of different hyperparameter adjustment by Bayesian optimization algorithm: the hyperparameter gamma without optimizing, even if all other hyperparameters are adjusted, the classifier will get the worst performance, so it is the most important hyperparameter, complexity is the second. In addition, the performance of Bayesian optimization algorithm adjusting the first three important hyperparameters according to N-RReliefF algorithm is not different from that of adjusting all the super-parameters, and the optimization time is greatly accelerated. Fig. 3 shows that the interaction between hyperparameters Gamma and Complexity is more important than that of Complexity itself when using the sigmoid kernel. Experience shows that Gamma and Complexity are important hyperparameters in SVM (see Table.7).This paper uses a wide range of data sets to provide systematic validation for traditional experience. The least important hyperparameter for SVM's accuracy is whether to use shrinkage heuristic algorithm. The purpose of this hyperparameter is to reduce computing resources rather than improve prediction performance. According to the criteria for calculating the impact of hyperparameters on performance, its importance weight accords with the actual results.

Table 7: Bayesian optimization algorithm to adjust the performance of different hyperparameters-SVM(sigmoid)

Hyperparameters	**Precision**	$F_1 - score$	**Recall**	**Runtime**
Fixed Gamma	0.60	0.62	0.66	15.3s
Fixed Complexity	0.75	0.75	0.73	16.2s
Adjust all hyperparameters	0.95	0.94	0.94	20.1s
Adjust N-RReliefF and select top three hyperparameters	0.93	0.92	0.93	10.2s

Results of the random forest algorithm: Fig. 4 shows the experimental results of the random forest algorithm. The performance of the random forest algorithm is contributed by a small number of hyperparameters. Min. sample leaf and Max. features are the most important hyperparameters (see Table.8). In the experimental process, bootstrap is the most important hyperparameter on only a few data sets. The split criterion is the most important parameter in the data set 'scene'. Similarly, the experimental results are consistent with the Bayesian optimization algorithm validation experiment and manual parameter adjustment experience.

The final conclusion: For all classifiers, the performance changes in most cases depend

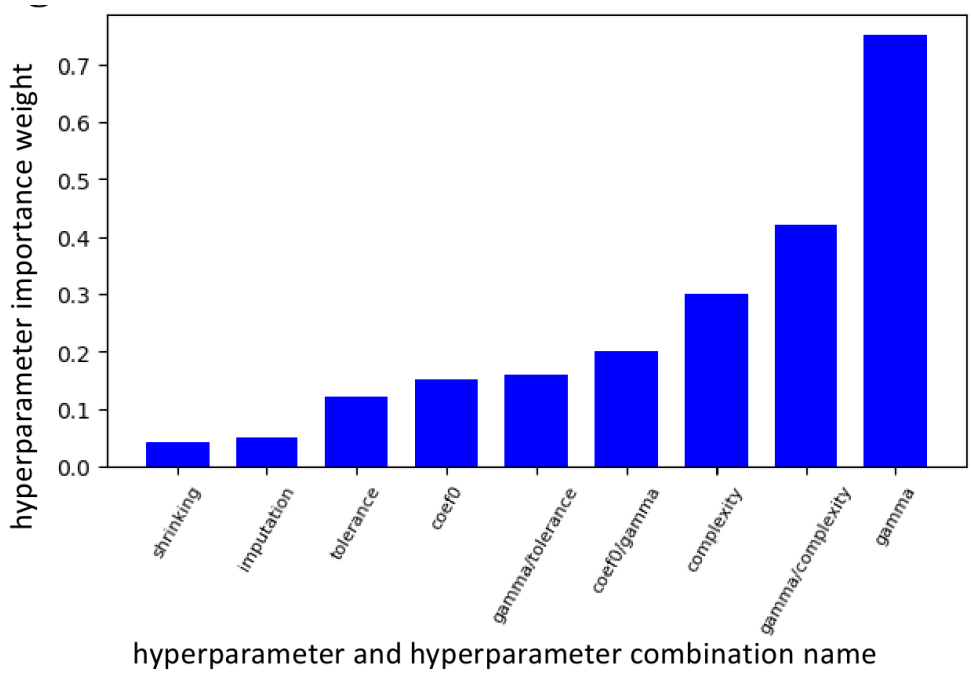

Figure 3: N-RReliefF algorithm evaluate hyperparameter and combination importance-SVM(sigmoid)

Table 8: Bayesian optimization algorithm to adjust the performance of different hyperparameters-random forest

Hyperparameters	**Precision**	$F_1 - score$	**Recall**	**Runtime**
Fixed Gamma	0.92	0.90	0.90	20.3s
Fixed Complexity	0.93	0.92	0.92	18.9s
Adjust all hyperparameters	0.98	0.97	0.97	25.5s
Adjust N-RReliefF and select top three hyperparameters	0.96	0.95	0.95	13.5s

on a small number of hyperparameters. In many cases, the same set of hyperparameters can be applied to different data sets that have the same domain and similar data characteristics. Therefore, it is necessary to understand the importance of hyperparameters in many cases, such as setting the default value of the algorithm, analyzing the automated hyperparameter optimization program and so on. In addition, understanding the importance of hyperparameters is a scientific attempt in itself, and can also provide guidance for algorithmic developers.

More interestingly, the experimental results show that the hyperparametric interpolation strategy has little effect on the performance of the classifier. In people's experience, the interpolation strategy is important, but this experiment shows that for interpolation, which strategy has little effect on the results.

It should be noted that the results provided in this section do not mean that only adjusting the most important hyperparameters and combinations is sufficient. Although Hutter [16] and others have shown that this can indeed lead to faster improvements, they also say that when there are enough computing resources, it is still recommended to adjust all hyperparameters.

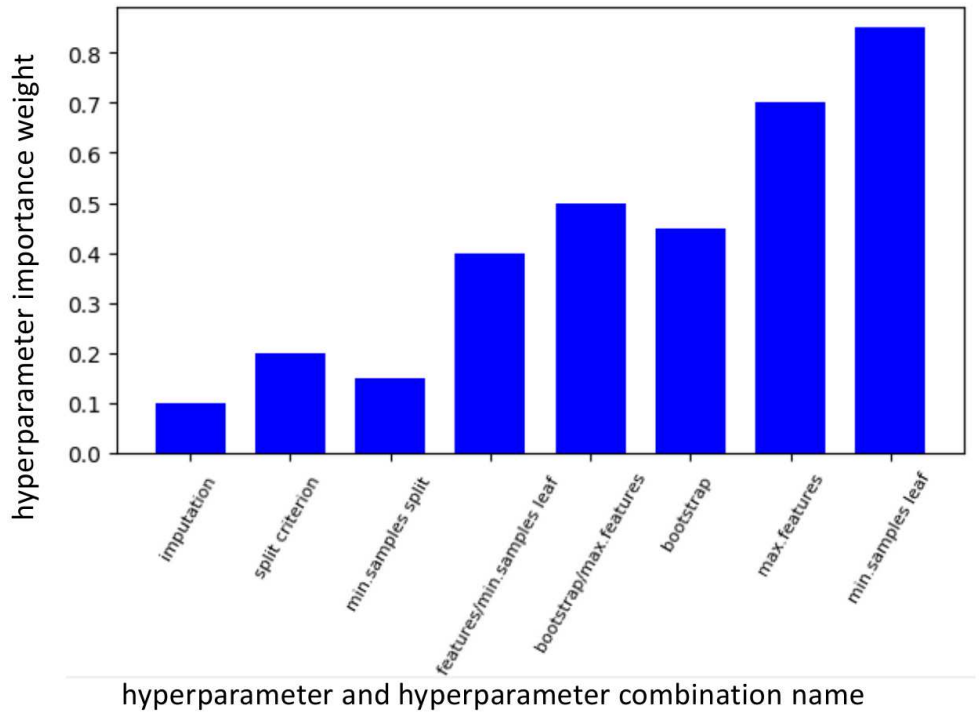

Figure 4: N-RReliefF algorithm evaluate hyperparameter and combination importance-random forest

5 Conclusion

In this paper, we compare the performance of Bayesian optimization algorithm, grid search and random search on several datasets. The results show that Bayesian optimization algorithm is superior to other two algorithms in classifier performance and running time. At the same time, N-RReliefF algorithm is used to determine the importance of the interaction between hyperparameters and hyperparameters on 100 data sets. The results show that the same hyperparameters have similar importance on different data sets. For SVMs, the hyperparameters Gamma and Complexity are the most important, and for random forest, Min. sample leaf and Max features are the most important. In order to verify the experimental results, Bayesian optimization algorithm optimizes different hyperparameters for each classifier. The results of this experiment are consistent with those of N-RReliefF, and to a large extent with popular views. A surprising result of this analysis is that data interpolation strategy has little impact on performance, which may be limited to the interpolation strategy used in this experiment. It is further proved that this conclusion needs to be studied as a whole and other interpolation strategies are added.

Future work will analyze which values of important hyperparameters are important, and provide users with the range and information of hyperparameters that can achieve better performance. In addition, it can be extended to regression and clustering algorithms to provide useful experimental support for the field of machine learning algorithm hyperparameter tuning optimization. In addition, the algorithm will be extended to the field of in-depth learning to optimize various network models (e.g. CNN, RNN models) and determine the importance of hyperparameters.

Funding

This work was funded by the Fundamental Research Funds for the Central Universities (Grant No.18CX02019A).

Author contributions. Conflict of interest

The authors contributed equally to this work. The authors declare no conflict of interest.

Bibliography

[1] Bartz-Beielstein, T. (2006). *Experimental research in evolutionary computation: The New Experimentalism*, Springer Berlin Heidelberg, 2006.

[2] Bergstra, J.; Bengio, Y. (2012). Random search for hyper-parameter optimization, *Journal of Machine Learning Research*, 13, 281–305, 2012.

[3] Bischl, B.; Casalicchio, G.; Feurer, M. et al. (2017). OpenML benchmarking suites and the openml100, ,

[4] Carlos, A.; Sellmann, M.; Tierney, K. (2009). A gender-based genetic algorithm for the automatic configuration of algorithms, *Principles and Practice of Constraint Programming - CP 2009, International Conference Proceedings*, CP 2009, Lisbon, Portugal, 142–157, 2009.

[5] Chang, C. C. C. C.; Lin, C.C.C. (2011). LIBSVM: A library for support vector machines, *ACM Transactions on Intelligent Systems and Technology*, 27, 1–27, 2011.

[6] Chiarandini, M.; Goegebeur, Y. (2009). Mixed Models for the analysis of optimization algorithms, *Experimental Thermal & Fluid Science*, 34(7), 972–978, 2009.

[7] Cui, J.; Yang, B. (2018). Survey on Bayesian optimization methodology and applications, *Journal of Software*, 29(10), 3068–3090, 2018.

[8] Daolio, F.; Liefooghe, A.; Sebastien, V.; Aguirre, H.; Tanaka, K. (2017). Problem Features vs. Algorithm Performance on Rugged Multi-objective Combinatorial Fitness Landscapes, *Acm Sigevolution*, 9(3), 21–21, 2017.

[9] Deng, S. (2019). Hyper-parameter optimization of CNN based on improved Bayesian optimization algorithm, *Application Research of Computers*, 2019(7), 2019.

[10] Falkner, S.; Klein, A.; Hutter, F. (2018). BOHB: Robust and Efficient Hyperparameter Optimization at Scale, *Proceedings of the35thInternational Conference on MachineLearning*, Stockholm, Sweden, 2018.

[11] Feurer, M.; Springenberg, J.T.; Hutter, F. (2015). Initializing bayesian hyperparameter optimization via meta-learning, *Twenty-Ninth AAAI Conference on Artificial Intelligence. AAAI Press*, 1128–1135, 2015.

[12] Adrian-Catalin Florea, A.-C.; Andonie, R. (2019). Weighted Random Search for Hyperparameter Optimization, *International Journal of Computers Communications & Control*, 14(2), 154–169, 2019.

[13] Gomes, T.A.F.; Soares, C. (2012). Combining meta-learning and search techniques to select parameters for support vector machines, *Neurocomputing*, 75(1), 3–13, 2012.

[14] Hutter, F.; Hoos, H. H.; Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration, *Learning and Intelligent Optimization -, International Conference, Lion 5, Rome, Italy, January 17-21, 2011. Selected Papers. DBLP*, 507–523, 2011.

[15] Hutter, F.; Hoos, H. H.; Leyton-Brown, K. (2013). Identifying key algorithm parameters and instance features using forward selection, *Learning and Intelligent Optimization*, 7997, 364–381, 2013.

[16] Hutter, F.; Hoos, H. H.; Leyton-Brown, K. (2014). An efficient approach for assessing hyperparameter importance, *In Proc. of ICML 2014*, 754–762, 2014.

[17] Kira, K.; Rendell, L. A. (1992). A practical approach to feature selection, *Machine Learning: Proceedings of International Conference, 1992*, 249–256, 1992.

[18] Lin, S. W.; Ying, K. C.; Chen, S. C.; Lee, Z. J. (2008). Particle swarm optimization for parameter determination and feature selection of support vector machines, *Expert Systems with Applications*, 35(4), 1817–1824, 2008.

[19] Mockus, J.; Tiesis, V.; Zilinskas, A. (1978). The application of Bayesian methods for seeking the extremum, *Towards Global Optimisation*, 2.1978, 117–129, 1978.

[20] Nannen, V.; Eiben, A. E. (2007). Relevance estimation and value calibration of evolutionary algorithm parameters, *International Joint Conference on Artifical Intelligence Morgan Kaufmann Publishers Inc*, 103–110, 2007.

[21] Probst, P.; Bischl, B.; Boulesteix, A. L. (2018). Tunability: Importance of Hyperparameters of Machine Learning Algorithms, *arXiv:1802.09596 [stat.ML]*, 2018.

[22] Reif, M.; Shafait, F.; Dengel, A. (2012). Meta-learning for evolutionary parameter optimization of classifiers, *Machine Learning*, 87(3), 357–380, 2012.

[23] Robnik-Sikonja, M.; Kononenko, I. (1997). An adaptation of Relief for attribute estimation in regression, *Fourteenth International Conference on Machine Learning, Morgan Kaufmann Publishers*, 1997.

[24] Robnik-Sikonja, M.; Kononenko, I. (2003). Theoretical and empirical analysis of relieff and rrelieff, *Machine Learning*, 53(1-2), 23–69, 2003.

[25] Snoek, J.; Larochelle, H.; Adams, R.P. (2012). Practical bayesian optimization of machine learning algorithms, *International Conference on Neural Information Processing Systems. Curran Associates Inc*, 2951–2959, 2012.

[26] Wang, J.; Zhang, L.; Chen, G. (2012). A parameter optimization method for an SVM based on improved grid search algorithm, *Applied Science and Technology*, 39(3), 28–31, 2012.

[27] Wu, J. (2017). Complex network link classification based on RReliefF feature selection algorithm, *Computer Engineering*, 43(8), 208–214, 2017.

[28] Zhang, D. (2017). High-speed Train Control System Big Data Analysis Based on Fuzzy RDF Model and Uncertain Reasoning, *International Journal of Computers, Communications & Control*, 12(4), 577–591, 2017.

[29] Zhang, D.; Jin, D.; Gong, Y.; Chen, S.; Wang, C. (2015). Research of alarm correlations based on static defect detection, *Tehnicki vjesnik*, 22(2), 311-318, 2015.

[30] Zhang, D.; Sui, J.; Gong, Y. (2017). Large scale software test data generation based on collective constraint and weighted combination method, *Tehnicki Vjesnik*, 24(4), 1041–1050, 2017.