Journal of Artificial Intelligence Practice (2021) Vol. 4: 30-38
Clausius Scientific Press, Canada

DOI: 10.23977/jaip.2020.040104
ISSN 2371-8412

Progressive Sampling-Based Joint Automatic Model Selection of Machine Learning and Feature Selection

Sufen Chen[1], Xueqiang Zeng[2,*]

[1]*School of Information Engineering, Nanchang Institute of Technology, Nanchang, Jiangxi Province 330099, P.R. China*

[2]*School of Computer & Information Engineering, Jiangxi Normal University, Nanchang, Jiangxi Province 330022, P.R. China*

*Corresponding author

Keywords: automatic selection, feature selection, Bayesian optimization, progressive sampling.

Abstract: In most machine learning applications, selecting an appropriate machine learning model requires advanced knowledge and many labor-intensive manual iterations. As a result, automatic machine learning is particularly important in order to lower the threshold for machine learning. In addition, feature selection is a very important data preprocessing process. Selecting important features can alleviate the dimension disaster problem, and removing irrelevant features can reduce the difficulty of learning tasks. The existing automatic selection methods cannot perform the automatic selection of machine learning model and feature selection model simultaneously on large-scale data. Therefore, in order to adapt to the rapid development of the era of big data, this paper proposes to establish a unified hyperparameter space for machine learning and feature selection, and adopt Bayesian optimization model based on progressive sampling for automatic model selection. By extensive experiments, we show that our approach can significantly reduce search time and classification error rates compared to the most advanced automated model selection methods.

1. Introduction

In the era of big data, as the speed of data generation continues to accelerate, the volume of data has an unprecedented growth, and new types of data to be analyzed are also emerging, making big data machine learning and feature selection play an extremely important role in the intelligent analysis and processing application of big data.

To lower the barriers to machine learning, computer science researchers have built a variety of open source software tools, such as WEKA [1], MLBox, Scikit-Learn [2], PyBrain [3] and Knime [4]. These software tools incorporate a number of machine learning algorithms and allow for hyper-parametric search and setting. However, the existing methods can only carry out automatic selection of machine learning model, but cannot carry out automatic selection of machine learning model and feature selection model simultaneously.

To further improve the efficiency of data processing, a unified hyperparameter space was

established for the machine learning model and feature selection model, and a Bayesian optimization [5] model based on progressive sampling [6] was used for joint automatic model selection. Therefore, we call our approach machine learning and feature selection combined with automatic model selection based on progressive sampling. We show that our approach can significantly reduce search time, classification error rates, and the number of unnecessary or false feature selections compared with the most advanced automated model selection methods.

Our contributions are summarized as follows:

- A unified hyperparameter space for machine learning and feature selection was constructed, and a combined automatic model selection based on progressive sampling was proposed.
- We conduct extensive experiments on multiple data sets, and the results demonstrate that our model produces promising results compared with state-of-the-art methods.

2. Related works

As our work in this paper aims to achieve efficient automatic selection of machine learning models and feature selection models using Bayesian optimization methods of progressive sampling, we will briefly review the following three topics:

2.1 Bayesian optimization

Our goal is to automatically select an effective combination of a machine learning algorithm, a feature selection technique (if it is desirable to consider feature selection), and hyper-parameter values. The current way of dealing with this problem [7] is to regard the choice of algorithm and the choice of feature selection technique as new hyper-parameters at the top level and convert this problem to selecting an effective hyper-parameter value combination. Bayesian optimization [8], also known as sequential model-based optimization, is an iterative method for solving such black box optimization problems.

In Bayesian optimization [9], a regression model R is first established to predict the error rate of machine learning model according to the value of hyper-parameters. R is usually a random forest, which can process and classify hyper-parameters. Compared with the traditional Bayesian optimization based on Gaussian process, this method can solve the situation that the parameter types in Gaussian regression process cannot be discrete. Bayesian optimization then iterates the following steps: (1) Use R to find multiple promising combinations of hyper-parameter values for further evaluation; (2) For each such hyper-parameter value combination λ, the machine learning model is trained and its error rate e is evaluated on the data set at λ; And (3) update R with a new data point (λ, e). Iteration stops when a pre-selected stop criterion is met.

2.2 Progressive sampling for automatic machine learning model selection

In the process of training the model, we often conduct data sampling, so that our model can better learn the characteristics of the data, so as to achieve better results. More fundamentally, data sampling is the simulation of a random phenomenon, simulating a random event according to a given probability distribution. Sampling has previously been used to select hyper-parameter values or combinations of hyper-parameter values for a particular machine learning algorithm [10], but not so that both hyper-parameter values and algorithms can be selected without limiting the hyper-parameter value combinations of algorithms and feature selection techniques to a fixed set. Our goal is to make our approach have better search result quality and search efficiency. Like typical Bayesian optimization, our Bayesian optimization method based on asymptotic sampling attempts to avoid searching in regions of hyper-parametric space containing low-quality combinations.

2.3 Feature selection

In high-dimensional data [11], irrelevant features can interfere with the true features, which in turn introduces heterogeneity in the data and generate dependence across the features [12]. So, we have to select features that play a vital role in estimation and which are independent. feature selection is a very important data preprocessing process. Selecting important features can alleviate the disaster problem of dimension, and removing irrelevant features can reduce the difficulty of the learning task. Automated feature selection is used to determine the role of each feature and to select the most useful feature from the raw data features. The search strategy for feature selection involves three types of algorithms [13]: complete search, heuristic search, and random search.

3. Method

In this section, we will elaborate on our proposed Bayesian optimization method based on progressive sampling.

3.1 Problem statement

For the model automatic selection problem[14], given a data set D, a set of machine learning algorithms $\mathcal{A}$, a set of feature selection techniques $\mathcal{B}$, and a hyper-parameter space Λ, the goal of model selection is to identify the algorithm $A^* \in \mathcal{A}$, feature selection technique $B^* \in \mathcal{B}$ and hyper-parameter value combination $\lambda^* \in \Lambda$ having the lowest error rate for generalization among all algorithms in $\mathcal{A}$, feature selection techniques in $\mathcal{B}$ and hyper-parameter value combinations in Λ. The generalization performance of an algorithm $A \in \mathcal{A}$, a feature selection technique $B \in \mathcal{B}$ and their hyper-parameter value combination $\lambda \in \Lambda$ is A_λ's error rate and B_λ's error rate for new data instances that are not in D, where A_λ denotes A using λ, B_λ denotes B using λ. The generalization performance is estimated by $M(A_\lambda, B_\lambda D)$, the error rates attained by A_λ and B_λ when trained and tested on D, e.g., via stratified multi-fold cross validation to decrease the possibility of overfitting. Using this estimate, the objective of machine learning model selection is to find:

$$A^*_{\lambda^*}, B^*_{\lambda^*} \in \operatorname{argmin}_{A \in \mathcal{A}, B \in \mathcal{B}, \lambda \in \Lambda} M(A_\lambda, B_\lambda, D).$$

3.2 Our Main Techniques

3.2.1 Consider the Hamming distance between combinations of hyper-parameter values

The few combinations of hyper-parameter values selected for the test need to reasonably cover the hyper-parameter space, so we try to select test combinations that are at least a certain distance from each other and have the lowest estimate of error rate in the previous round. In the implementation, the Hamming distance [15] is used and the default number of combinations of super parameter values selected for the test is $n_c=10$ and the default distance threshold is $t_d=2$. If the Hamming distance between two combinations is greater than 2, they can be considered sufficiently far from each other.

Let λ_i ($1 \le i \le g$) denote the hyper-parameter value combinations selected for testing, g denote the number of them, and r_i denote the error rate ratio of λ_i. For each non-selected combination with an error rate estimate of 100% from the previous round, we keep its error rate estimate at 100% for the current round. For each other non-selected combination λ_u, we multiply its error rate estimate from the previous round by a factor r_u to obtain its rough error rate estimate for the current round. If the rough error rate estimate is >100%, it is set to 100%. Using inverse distance weighting, r_u is computed as a weighted average of the selected combinations' error rate ratios:

$$r_u = \sum_{i=1}^{g} w_{i,u} \times r_i / \sum_{i=1}^{g} w_{i,u}.$$

Where the weight $w_{i,u}$ for λ_i is set to $1/distance(\lambda_u, \lambda_i)$, reflecting the intuition that the further away λ_i is from λ_u, the less weight λ_i should have. In case distance(λ_u, λ_j)=0 for a specific j ($1 \leq j \leq g$), we set $r_u=r_j$. To prevent the factor r_u from being overly influenced by an extreme r_i, we force each r_i to fall into the range [0.25, 2.5]. If an r_i is <0.25, we set it to 0.25. If an r_i is >2.5, we set it to 2.5.

3.2.2 Use multiple folds to estimate each selected combination's error rate

We use a technique similar to multiple cross validation [16] on a small data set, using multiple folds to increase the robustness of the estimate obtained. We use *k* times progressive sampling, which defaults to *k=3*. M data instances are randomly divided into K parts of roughly the same size. In any contraction that is not the last round, we use k folds. In the *i*-th ($1 \leq i \leq k$) collapse, the *i*-th part of M data instances forms the verification sample. The combination of all other parts of M data instances forms the largest training set. For each combination of the machine learning algorithm and hyper-parameter values selected for the test, we train the model using training samples and combinations and estimate the error rate of the model on the verification samples. The average of all *k* times the estimated model error rate is used as the estimated error rate of the wheel combination.

3.2.3 In each round of the search process, limit the time that can be spent on testing a combination

In each round of searching, for each test of the combination of the machine learning algorithm, feature selection techniques and the hyper-parameter values on the folding, we impose a limit of L_f on the feature selection time and L_t on the model training time. Like Auto-WAKE [7], once the time spent on feature selection exceeds L_f, the test will be terminated and the estimated error rate of the model will be set to 100%. We store the feature selection technique and its hyper-parameter values in a cache shared by all algorithms. If feature selection is completed in L_f, once the time spent on model training exceeds the time spent on L_t, the model training is stopped and the error rate of part of the training model is estimated on the validation sample. For a small data set, we start with L_f = 10 s and L_t = 10 s in the first round of searching. For large data sets, we start with L_f = 20 s and L_t = 20 s in the first round. Then increase L_f and L_t by 50% each round.

3.2.4 Processing of feature selection

The results should be checked after feature selection is completed. Some combination of feature selection techniques and their hyper-parameter values will result in all or no features being selected. Using all features equals no feature selection[18] . If the combination selects all or none of the features, we set the estimated generalization error rate for λ to 100%, skip the model training, and update the regression model with (λ, 100%) as new data points. The data point (λ, 100%) prompts to avoid future testing of the same feature selection technique and combinations of hyper-parameter values for that technique. This helps save time. For the same purpose, we store the feature selection technique and its hyper-parameter values in a cache shared by all algorithms. This cache is shared by all algorithms. At the end of the first round, for each pair of combination λ_1 in the cache, we select the algorithm's random hyper-parameter value combination λ_2 and update the algorithm's regression model with ((λ_1, λ_2), 100%) as new data points. This helps the regression model guide subsequent searches away from areas of the hyper-parameter space where all or none of the features are selected.

Appropriate penalties for the use of feature selection. To prevent overfitting, we impose penalties for using feature selection as a form of regularization. For a combination of machine learning algorithms, feature selection techniques, and hyper-parameter values, if feature selection is used in the combination, we update the regression model by multiplying its estimated error rate by a predetermined constant > 1. We chose the default value of the constant to be 1.1 in order to punish

the use of feature selection adequately, but not excessively.

3.3 The Complete, Progressive Sampling-Based Bayesian Optimization Method

In this part, we will introduce the progressive sampling-based joint automatic model selection of machine learning and feature selection in detail. The whole method will be carried out in 5 rounds, and the complete process is shown in Algorithm framework 1.

Algorithm framework 1: Progressive sampling-based joint automatic model selection

1: Form the validation and initial training samples;
// the first round
2: For each machine learning algorithm and feature selection technique, test its default and 20 random hyper-parameter value combinations;
3: Identify and remove some unpromising algorithms and techniques;
// the intermediate rounds
4: For round 2 to 4 {
5: Expand the training sample;
6: For each remaining algorithm {
7: For hyper-parameter value combinations used in the previous round, consider the Hamming distance between the combinations to update their error rate estimates;
8: Conduct Bayesian optimization;
9: }
10: Identify and remove some unpromising algorithms;
11: }
// the last round
12: Use the cross-validation method to select the final combination of an algorithm and hyper-parameter values, use this combination to build a model on the whole data set, and return this model;

3.3.1 The first round

In the first round, we tested every applicable machine learning algorithm and feature selection technique. For each algorithm and technique, we test the default combination of hyper-parameter values and a predetermined number (the default is 20) of random hyper-parameter values, if any. Firstly, the algorithms with error rate $\geq \tau$ are eliminated, in which $\tau = 0.5$ is the threshold value of error rate. Then if the number of remaining algorithms and techniques exceeds 40%, it will continue to delete, keeping only the top 40% with the lowest error rate. The 0.5 and 40% numbers were selected in order to strike a balance between removing enough algorithms in the first round to improve search efficiency and the goal of not removing high-quality algorithms too soon.

3.3.2 The intermediate rounds

In each subsequent round, which is not the final round, we reduce the error rate differential threshold τ by multiplying by 0.8, expand the training sample, increase the time limit L_f and L_t, and test and tweak promising combinations on an extended training set and further reduce the search space.

For each machine learning algorithm and feature selection technique remaining from the last round, we do it in three steps. In the first step, considering the Hamming distance between the combinations of hyper-parameter values (Chapter 3.2), the combination of hyper-parameter values used in the last round of tests is selected and its error rate in this round is obtained. The second step is to obtain rough error rate estimates for the combinations used in the previous round but not selected in the first round. As a final step, build the regression model using all the combinations tested against the algorithm so far. Then the Bayesian optimization was carried out in C cycles. In the second round, $C = 3$, and in each subsequent round, C decreased by 1. In each cycle of Bayesian optimization, like Auto-WEKA [7], 10 new combinations are selected for testing and used to modify the regression model. To better cover the new region of the hyper-parameter space, each combination is randomly selected. Finally, we used a method similar to the first round to identify and remove invalid algorithm combinations, except that we increased the target percentage of the retention algorithm from 40% to 70%.

3.3.3 The last round

In the last round, we use the cross-validation method to select the final combination, and use the final combination to build a model on the whole data set as the final model returned by our automatic selection method.

4. Results

In this section, we have carried out extensive experiments on a variety of datasets and compared the results with those of Auto-WEKA [7], which shows the effectiveness of our algorithm.

4.1 Experimental setting

In the experiment, we compare the progressive sampling-based joint automatic model selection of machine learning and feature selection with the Auto-WEKA automatic selection method. As a preliminary study, our aim is to demonstrate the effectiveness and feasibility of using progressive sampling methods for joint selection machine learning and feature selection models and hyperparameter values. Considering all 39 classification algorithms and feature selection techniques in the standard Weka package [19], 21 well-known benchmark datasets are used. Each dataset is divided into a training set and a test set. Training data is used in the search process and test data is used to assess the error rate of the automatic selection method returning to the final model. Other settings in the experiment are basically the same as Auto-WEKA. For the Auto-WEKA method, the total time budget for each run on the dataset is 30 hours.

4.2 Overall results of the search process

For each data set, we run the Auto-WEKA automatic selection method and automatic model selection based on machine learning and feature selection combined with progressive sampling for five times using different random seeds. Each parameter used in our method is set to the default value. For each automatic selection method and dataset, three indicators are given: the total time spent in the search process, the number of different combinations of hyper-parameter values tested, and the error rate of the final model in the test data, as shown in Table 1. For each indicator, the mean and standard deviation of the five runs are displayed as average value ± standard deviation. Small datasets are shown in the first 11 rows and large datasets in the last 10 rows.

The Auto-WEKA automatic selection method runs for 30 hours each time. By contrast, our method takes much less time, on average, it is about 15 times faster on small datasets and 4 times faster on

large datasets. During the search process, our method achieves accelerations that are different than the Auto-WEKA method from 3 in the MNIST Basic dataset to 31 in the German Credit dataset.

Compared with the Auto-WEKA method, our method tests more combinations during the search process, especially for large datasets. On small datasets, our method tests an average of three times more different combinations than the Auto-WEKA method. Of these, on four datasets, such as Car, the Auto-WEKA method tests more combinations because it tests more distinct combinations than our method by spending more time in the search process. On large datasets, our method averaged 40 times more combinations than the Auto-WEKA method.

On nearly all datasets, our automatic model selection based on machine learning and feature selection combined with progressive sampling yields a lower error rate than the Auto-WEKA method on test data. On Car and KDD09-Appentency datasets, the error rates are the same. Our method error rate is slightly lower on three small datasets, such as Yeast, but the difference is very small (< 1%). In small datasets, on average, our Bayesian optimization method based on progressive sampling has a 4% lower error rate than the Auto-WEKA automatic selection method. On average, our method has a 21% lower error rate than the Auto-WEKA method on large datasets. Overall, our approach has a more significant advantage over small datasets in terms of large datasets.

Table 1: Overall results of the search process

Data set	Time spent (hours)		# of distinct combinations tested		Error rate on the test data (%)	
	Auto-WEKA	Our method	Auto-WEKA	Our method	Auto-WEKA	Our method
Car	30	2.0±0.4	1,954±1,050	1,524±35	0.00±0.00	0.00±0.00
Yeast	30	1.7±0.4	6,675±2,721	1,602±23	38.29±1.51	38.88±1.20
German Credit	30	1.0±0.3	3,364±952	1,602±29	28.40±1.26	27.00±1.11
Abalone	30	4.3±0.5	675±181	1,501±20	72.99±0.30	73.25±0.47
Wine Quality	30	4.2±0.8	1,429±823	1,633±33	33.97±0.75	34.12±0.70
KR-vs-KP	30	1.5±0.6	2,192±1,522	1,574±23	0.42±0.24	0.40±0.11
Waveform	30	3.5±0.6	801±248	1,596±21	14.33±0.15	14.29±0.12
Semeion	30	4.7±0.7	408±103	1,594±9	5.58±0.82	5.03±0.21
Shuttle	30	1.7±0.5	275±92	1,610±22	0.010±0.004	0.008±0.003
Secom	30	1.1±0.1	119±90	1,699±47	8.13±0.57	7.83±0.09
Madelon	30	2.6±0.3	529±261	1,763±49	22.08±2.52	18.31±1.47
Convex	30	5.5±0.4	72±49	1,766±73	26.24±4.93	23.37±0.87
KDD09-Appentency	30	6.0±0.5	155±127	1,783±72	1.74±0.00	1.74±0.00
Dexter	30	4.4±0.6	98±14	1,569±28	8.33±1.24	5.11±1.20
MNIST Basic	30	11.1±0.7	53±16	1,674±23	10.14±4.40	3.70±0.46
ROT. MNIST+BI	30	7.0±0.3	13±13	1,637±17	63.31±5.76	55.81±0.16
Amazon	30	8.9±0.8	53±19	1,561±41	38.31±14.73	24.31±1.91
Gisette	30	9.4±0.3	43±25	1,586±31	2.31±0.55	1.89±0.29
CIFAR-10-Small	30	10.0±0.5	46±21	1,546±34	67.87±5.36	57.42±0.52
Dorothea	30	7.3±0.9	52±45	1,625±25	6.29±2.09	5.74±0.48
CIFAR-10	30	10.0±0.9	29±24	1,529±72	62.02±7.68	52.83±0.78

4.3 Feature selection results of the final combination

Table 2 gives the results of feature selection. The Auto-WEKA automatic selection method makes feature selection 75 times in 105 runs, but mostly makes unnecessary feature selection (all or none). By contrast, our approach has made six feature selections and has never made unnecessary selections. And it never causes all features to be selected or no features to be selected. There are two possible reasons why feature selection is rarely used in the final combination of hyper-parameter values

produced by Bayesian optimization methods based on progressive sampling: first, to help prevent overfitting, we penalize the use of feature selection as a form of regularization; Second, some top-level algorithms, such as support vector machines and random forests, have an internal feature selection mechanism embedded in them. If our method chooses any of them as the final algorithm, the separate feature selection section in the data analysis pipeline often becomes unnecessary.

In summary, from various experimental results, our automatic model selection based on machine learning and feature selection combined with progressive sampling reduces the search time by an average of 10 times, reduces the error rate by 13%, and makes feature selection more accurate than Auto-WEKA automatic selection method. This is a significant advance in achieving fast turnover in identifying high-quality solutions required for many machine learning-based data analysis tasks.

Table 2: Feature selection (FS) results

Data set	Auto-WEKA	Our method
	# of times FS	# of times FS
Car	4	0
Yeast	2	0
German Credit	3	0
Abalone	4	0
Wine Quality	4	0
KR-vs-KP	2	0
Waveform	4	0
Semeion	5	0
Shuttle	3	0
Secom	4	0
Madelon	4	4
Convex	4	0
KDD09-Appentency	5	0
Dexter	4	0
MNIST Basic	3	0
ROT. MNIST+BI	2	0
Amazon	4	1
Gisette	4	0
CIFAR-10-Small	3	0
Dorothea	4	1
CIFAR-10	3	0
Total	75	6

5. Conclusion

In this paper, a unified hyperparameter space is established for machine learning and feature selection, and an automatic model selection method combined with machine learning and feature selection based on progressive sampling is optimized. Experiments show that compared with the existing automatic selection methods, our method greatly improves the search efficiency, the quality of search results and the accuracy of feature selection. Future research will investigate other techniques to further improve search efficiency without degrading the quality of search results, and research will develop the theoretical basis to improve our approach.

Acknowledgements

The work was supported in part by the Natural Science Foundation of China under grant no. 61866017, the Natural Science Foundation of Jiangxi Province under grant no. 20192BAB207027 and the Support Program for Outstanding Youth Talents in Jiangxi Province no. 20171BCB23013.

References

[1] KOTTHOFF L, THORNTON C, HOOS H H, et al., Auto-WEKA: Automatic Model Selection and Hyperparameter Optimization in WEKA.[G]//Automated Machine Learning - Methods, Systems, Challenges. 2019: 81–95.
[2] PEDREGOSA F, VAROQUAUX G, GRAMFORT A, et al., Scikit-learn: Machine Learning in Python[J]. CoRR, 2012, abs/1201.0490.
[3] SCHAUL T, BAYER J, WIERSTRA D, et al., PyBrain.[J]. J. Mach. Learn. Res., 2010, 11: 743–746.
[4] JOVIC A, BRKIC K, BOGUNOVIC N. An overview of free software tools for general data mining.[C]//37th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2014, Opatija, Croatia, May 26-30, 2014. 2014: 1112–1117.
[5] SHAHRIARI B, SWERSKY K, WANG Z, et al., Taking the Human Out of the Loop: A Review of Bayesian Optimization.[J]. Proc. IEEE, 2016, 104(1): 148–175.
[6] PROVOST F J, JENSEN D D, OATES T. Efficient Progressive Sampling.[C]//Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 15-18, 1999. 1999: 23–32.
[7] THORNTON C, HUTTER F, HOOS H H, et al., Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms.[C]//The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013. 2013: 847–855.
[8] SNOEK J, LAROCHELLE H, ADAMS R P. Practical Bayesian Optimization of Machine Learning Algorithms.[C]//Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States. 2012: 2960–2968.
[9] KLEIN A, FALKNER S, BARTELS S, et al., Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets.[C]//Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA. 2017: 528–536.
[10] HOFFMAN M W, SHAHRIARI B, FREITAS N de. On correlation and budget constraints in model-based bandit optimization with application to automatic machine learning.[C]//Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014. 2014: 365–374.
[11] MEERA S, SUNDAR C. A hybrid metaheuristic approach for efficient feature selection methods in big data.[J]. J. Ambient Intell. Humaniz. Comput., 2021, 12(3): 3743–3751.
[12] XU Y, WANG X, ZHANG Y, et al., Summary of feature selection algorithms[J]. Control & Decision, 2012, 27(2): 161–313.
[13] ZAFFAR M, HASHMANI M A, SAVITA K S, et al., A review on feature selection methods for improving the performance of classification in educational data mining.[J]. Int. J. Inf. Technol. Manag., 2021, 20(1/2): 110–131.
[14] HE X, ZHAO K, CHU X. AutoML: A survey of the state-of-the-art.[J]. Knowl. Based Syst., 2021, 212: 106622.
[15] GRABOWSKI S, KOWALSKI T M. Algorithms for all-pairs Hamming distance based similarity.[J]. Softw. Pract. Exp., 2021, 51(7): 1580–1590.
[16] RAO R B, FUNG G. On the Dangers of Cross-Validation. An Experimental Evaluation.[C]//Proceedings of the SIAM International Conference on Data Mining, SDM 2008, April 24-26, 2008, Atlanta, Georgia, USA. 2008: 588–596.
[17] MARCOT B G, HANEA A M. What is an optimal value of k in k-fold cross-validation in discrete Bayesian network analysis?[J]. Comput. Stat., 2021, 36(3): 2009–2031.
[18] RAHMAN M A, ASYHARI A T, WEN O W, et al., Effective combining of feature selection techniques for machine learning-enabled IoT intrusion detection[J]. Multimedia Tools and Applications, 2021: 1–19.
[19] WITTEN I H, FRANK E, HALL M A. Data mining: practical machine learning tools and techniques, 3rd Edition.[M]. Morgan Kaufmann, Elsevier, 2011: 629.