RFC: A feature selection algorithm for software defect prediction

Journal of Systems Engineering and Electronics
Vol. 32, No. 2, April 2021, pp.389 – 398

RFC: a feature selection algorithm for software defect prediction

XU Xiaolong[1], CHEN Wen[2], and WANG Xinheng[3,*]

1. Jiangsu Key Laboratory of Big Data Security & Intelligent Processing, Nanjing University of Posts and Telecommunications, Nanjing 210023, China; 2. Institute of Big Data Research at Yancheng, Nanjing University of Posts and Telecommunications, Yancheng 224000, China; 3. School of Computing and Engineering, University of West London, London W5 5RF, UK

Abstract: Software defect prediction (SDP) is used to perform the statistical analysis of historical defect data to find out the distribution rule of historical defects, so as to effectively predict defects in the new software. However, there are redundant and irrelevant features in the software defect datasets affecting the performance of defect predictors. In order to identify and remove the redundant and irrelevant features in software defect datasets, we propose ReliefF-based clustering (RFC), a cluster-based feature selection algorithm. Then, the correlation between features is calculated based on the symmetric uncertainty. According to the correlation degree, RFC partitions features into *k* clusters based on the *k*-medoids algorithm, and finally selects the representative features from each cluster to form the final feature subset. In the experiments, we compare the proposed RFC with classical feature selection algorithms on nine National Aeronautics and Space Administration (NASA) software defect prediction datasets in terms of area under curve (AUC) and F-value. The experimental results show that RFC can effectively improve the performance of SDP.

Keywords: software defect prediction (SDP), feature selection, cluster.

DOI: 10.23919/JSEE.2021.000032

1. Introduction

Software with defects can bring unexpected results or behaviors at run time, which can cause huge damage or even disasters in serious cases [1].

Software defect prediction (SDP) [2] is an effective method to identify defects in system modules in advance. First, the software code or the development process is analyzed, the metrics related to software defects are designed, and then the defect dataset is created by mining the software historical repositories. Finally, based on the defect dataset, the SDP model is constructed.

In order to better predict software defects, many metrics that have strong correlation with software defects are proposed to measure modules [3−6]. Attributes of software quality, such as defect density and failure rate, are external measures of the software product and its development process. Generally, a software quality prediction model is built with software metrics and defect data collected from the previously developed systems or similar software projects. The selection of the specific set of metrics becomes an integral component of the model-building process. However, the redundant and irrelevant features in the software defect dataset increase the time complexity and affect the performance of defect predictors [7].

Therefore, the defects of the existing algorithms can be summarized as follows:

(i) The feature selection algorithms based on filter can remove irrelevant features, while cannot remove redundant features.

(ii) The feature selection algorithms based on the embedded method combining the advantages of filter and encapsulation have the problem of high complexity.

In order to solve the above problems, we propose ReliefF-based clustering (RFC), a cluster-based feature selection algorithm. First, the ReliefF algorithm [8] is used to calculate the relevance between each feature and target class, features are sorted to remove irrelevant features, then the features are clustered according to the correlation between the remaining features, and finally the representative features of each cluster are selected. We implement experiments based on the software defect prediction datasets released by the National Aeronautics and Space Administration (NASA) to test and verify the proposed algorithm. RFC considers the correlation between

Manuscript received January 20, 2020.
*Corresponding author.
This work was supported by the National Key Research and Development Program of China (2018YFB1003702) and the National Natural Science Foundation of China (62072255).

features and the relevance between features and the target class, which can effectively remove redundant features and irrelevant features to solve the problem of dimension disaster and improve the performance of SDP.

This paper is organized as follows. Section 2 introduces the related work. Section 3 describes RFC in detail. Section 4 presents the experimental results and the performance analysis. Section 5 concludes this paper and presents our possible future work.

2. Related work

The methods of SDP can be divided into two categories. On the one hand, some researchers focus on utilizing software metrics [9−11], such as the Hastead scientific metrics [3], the McCabe loop complexity [4], quality model for object oriented design (QMOOD) [5], the Chidamber and Kenerer (CK) metrics [6]. On the other hand, some researchers focus on the quality of SDP datasets. There are many problems to be solved in SDP, such as the data imbalance [12−15], and the dimension disaster [16]. The feature selection is an effective method to solve the problem of dimension disaster.

Appropriate feature selection results can improve the learning accuracy, reduce the learning time, and simplify the learning results. Kira et al. [17] proposed a feature selection method, Relief, which randomly selects m instances from the training set. For each selected instance i, Relief computes the nearest neighbor from the same class of i and the nearest neighbor from the opposite class. The quality of each feature is estimated with respect to whether the feature differentiates two instances from the same class and from different classes. A feature has an undesired property if it differentiates two near instances that belong to the same class. The original Relief algorithm is limited to binary classification problems. ReliefF [8] is extended from Relief to solve the multiclass problem. Its main idea is to take the Euclidean distance as the correlation index and then weights features according to how well they differentiate instances of different classes. Meanwhile, the embedded methods select feature in the training process of the learning model, and the feature selection result outputs automatically while the training process is finished. For example, a hybrid feature selection method proposed by Shivkumar et al. [18] first uses the filter or the classifier to score each feature, then removes features with lower scores, uses the classifier to evaluate the prediction performance of the remaining features, and finally chooses the best prediction performance as the final feature set. When the characteristics are reduced to a certain extent, predictive performance begins to decline due to the lack of important information. These algorithms combine the advantages of the filter method and the wrapper method, while it has a high time complexity.

Guo et al. [19] proposed a software quality prediction method based on the random forest, where prediction accuracy of the proposed methodology is higher as compared to logistic regression and discriminant analysis. The random forest is more robust to noise and outliers than other methods. However, irrelevant features have a great impact on the predictive effect of random forest, the effect of using the first five most relevant features to train the random forest is comparable to that of using all the features to train the random forest. Menzies et al. [20] used the information gain (IG) to rank features. The prediction effects of the first three features are comparable to the use of all features. Rodriguez et al. [21] made use of feature selection methods in different datasets, and tested different data mining algorithms for classification to defect faulty modules. The results showed that in general, smaller datasets with fewer attributes maintained or improved the prediction capability with fewer attributes than the original datasets. Catal et al. [22] verified the influence of feature selection on model performance, and showed that feature selection is beneficial to improve the performance of the software defect prediction model. Gao et al. [23] compared seven different feature ranking methods and four different feature subset selection approaches based on software metrics and defect data collected from multiple releases of a large real-world software system. The results showed that the automatic hybrid search algorithm performed the best among the feature subset selection methods. Moreover, performances of the defect prediction models either improved or remained unchanged if over 85% of the software metrics are eliminated. Wang et al. [24] presented a comprehensive empirical study by examining 17 different ensembles of feature ranking methods (rankers) including six commonly used feature ranking methods, the signal-to-noise filter method, and 11 threshold-based feature ranking methods. This study utilized 16 real-world software measurement datasets of different sizes and built 13 600 classification models. Experimental results indicated that ensembles of very few rankers were very effective and even better than ensembles of many or all rankers. Bennin et al. [25] proposed a synthetic oversampling approach called MAHAKIL based on the chromosomal theory of inheritance. Experiments showed that MAHAKIL improved the prediction performance for software defect prediction. Miholca et al. [26] developed a supervised classification method called HyGRAR, which combined gradual relational association rule mining and artificial neural networks to discriminate between defective and non-defective software entities. Experiments demon-

strated the excellent performance of the HyGRAR classifier. Cao et al. [27] proposed an SDP model based on the twin support vector machines (TSVM) and a multi-objective cuckoo search (MOCS), which achieved a better performance than other SDP models.

3. Cluster-based feature selection algorithm

3.1 Workflow of RFC

Different from the above works, in order to effectively identify the redundant and relevant features in the software defect dataset, in this paper, we propose a cluster-based feature selection algorithm, RFC. RFC removes irrelevant features, then partitions features into k clusters based on symmetric uncertainty (SU), and finally selects representative features from each cluster. The specific workflow of RFC is shown in Fig. 1.

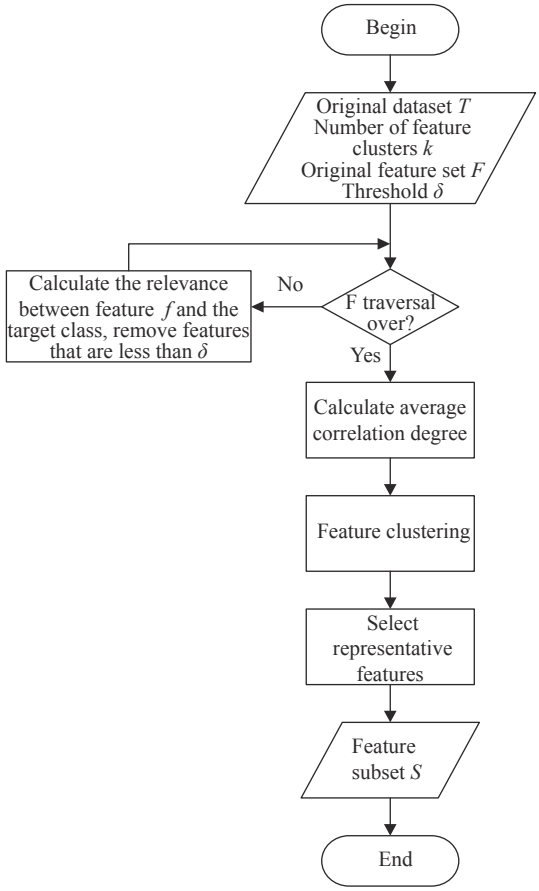

Fig. 1 Workflow of RFC

(i) Calculate the relevance between features and the target class based on ReliefF, and then remove irrelevant features according to the threshold.

The key idea of ReliefF is to estimate the quality of attributes according to how well their values distinguish between the instances that are near to each other. Given a randomly selected instance R, ReliefF searches for k nearest neighbors of R from the same class, called nearHits, and k nearest neighbors from each of the different classes, called nearMisses. If the difference between R and nearHits is less than that between R and nearMisses on a feature, which indicates that the feature is beneficial to classification, the corresponding weight of the feature will be increased. Conversely, the weight of the feature is reduced. This process is repeated until the end condition is met, and then the weight of each feature is returned, which is the relevance between the feature and the target class. The features are sorted by weight, and then the threshold δ is set to determine whether each feature is valid or not.

(ii) Cluster feature subsets.

i) Calculate the degree of correlation between features.

We use the nonlinear SU [28] to measure correlation between features. SU is derived from mutual information by normalizing it to the entropies of feature values. It can be computed with

$$SU(X,Y)=\frac{2IG(X,Y)}{H(X)+H(Y)} \tag{1}$$

where $H(X)$ is the entropy of a discrete random variable X. Suppose $p(x)$ is the prior probabilities for all values of X, $p(x|y)$ is the posterior probabilities of X given the values of Y, $H(X)$ can be computed with

$$H(X)=-\sum_{x\in X}p(x)\log_2 p(x). \tag{2}$$

The information gain $IG(X|Y)$ can measure the amount by which the entropy of X decreases given the values of Y. It can reflect the additional information about X provided by Y. $IG(X|Y)$ can be computed with

$$IG(X\mid Y)=H(X)-H(X\mid Y)=H(Y)-H(Y\mid X) \tag{3}$$

where $H(X|Y)$ is the conditional entropy, quantifying the remaining entropy, i.e., uncertainty of a random variable X given that the value of another random variable Y is known. $H(X|Y)$ is defined with

$$H(X\mid Y)=-\sum_{y\in Y}p(y)\sum_{x\in X}p(x\mid y)\log_2 p(x\mid y). \tag{4}$$

SU can compensate for information gain's bias toward features with more values and restrict its values in [0, 1]. The value 1 of $SU(X,Y)$ indicates that knowledge of X can completely predict knowledge of Y and vice versa, while the value 0 of $SU(X,Y)$ reveals that X and Y are independent. Although the entropy-based measure can handle nominal or discrete variables, they can deal with continuous features as well, if the values are able to be discretized properly in advance.

ii) Calculate the average correlation degree.

Before feature clustering, several features should be selected as the initial representative characteristics. To further accelerate the convergence of our proposed algorithm, we consider features with the most information content as initial features. The information content of a feature is measured by the average correlation degree of the feature. The average value of the symmetry uncertainty of f_i and other features in the feature set can be computed with

$$AvgRel(f_i) = \frac{1}{m}\sum_{j=1}^{m} SU(f_i, f_j) \tag{5}$$

where m represents the number of remaining features, $SU(f_i, f_j)$ represents the correlation between feature f_i and feature f_j.

iii) Cluster the features based on k-medoids.

Step 1 Sort the features according to $AvgRel$ in the descending order, and select top k features as initial medoids.

Step 2 Assign each feature to the cluster associated with the most similar medoid.

Step 3 Update cluster's medoids. For each feature cluster, the features with the highest degree of correlation with other features are selected as new cluster medoids.

Step 4 Step 2 and Step 3 are iterated until that medoids no longer change.

(iii) Select representative features.

After features partitioned into k clusters, relevant features need to be selected from each cluster. Since the scale of different clusters cannot be uniform, the scale of a cluster is taken into account when relevant features are selected from each cluster. The larger a cluster is, the more relevant features are.

For feature cluster C_i, select the cluster's medoid first, and then sort the remaining features according to their relevance with the target class in the descending order. After that, select top $\left\lceil \frac{|C| \times r}{m} \right\rceil - 1$ features from each cluster, where $|C|$ is the scale of the cluster, m is the number of the feature subset, and r is the scale of the final feature subset that we want to select.

3.2 SDP based on RFC

The metrics of SDP datasets are composed of lines of codes (LOC) counts, Halstead complexity as well as McCabe complexity, as shown in Table 1.

Table 1 LOC counts and Halstead complexity

Number	Metric	Number	Metric
0	LOC_BLANK	20	HALSTEAD_DIFFICULTY
1	BRANCH_COUNT	21	HALSTEAD_EFFORT
2	CALL_PAIRS	22	HALSTEAD_ERROR_EST
3	LOC_CODE_AND_COMMENT	23	HALSTEAD_LENGTH
4	LOC_COMMENTS	24	HALSTEAD_LEVEL
5	CONDITION_COUNT	25	HALSTEAD_PROG_TIME
6	CYCLOMATIC_COMPLEXITY	26	HALSTEAD_VOLUME
7	CYCLOMATIC_DENSITY	27	MAINTENANCE_SEVERITY
8	DECISION_COUNT	28	MODIFIED_CONDITION_COUNT
9	DECISION_DENSITY	29	MULTIPLE_CONDITION_COUNT
10	DESIGN_COMPLEXITY	30	NODE_COUNT
11	DESIGN_DENSITY	31	NORMALIZED_CYLOMATIC_COMPLEXITY
12	EDGE_COUNT	32	NUM_OPERANDS
13	ESSENTIAL_COMPLEXITY	33	NUM_OPERATORS
14	ESSENTIAL_DENSITY	34	NUM_UNIQUE_OPERANDS
15	LOC_EXECUTABLE	35	NUM_UNIQUE_OPERATORS
16	PARAMETER_COUNT	36	NUMBER_OF_LINES
17	GLOBAL_DATA_COMPLEXITY	37	PERCENT_COMMENTS
18	GLOBAL_DATA_DENSITY	38	LOC_TOTAL
19	HALSTEAD_CONTENT		

In particular, LOC counts measure the number of code lines, comments lines, and so on. Halstead complexity estimates the program complexity by counting operators and operands in a module. McCabe complexity measures the complexity of a module's inner structure. These metrics, i.e., features, can be widely used in SDP. However, the effectiveness of current methods is influenced by irrelevant and redundant features select features.

The pseudo code of the algorithm is as follows:

Algorithm SDP based on RFC

Input T, k, F, δ, r;

Output S;

1. $S = \varnothing$
2. Calculate the relevance between features and the target class;
3. Construct the vector $\boldsymbol{W} = \{w_i, i = 1, 2, \cdots, M\}$;
4. for i=1 to M do
5. if $(w_i < \delta)$ then
6. $F = F - f_i$;
7. for i=1 to m do
8. for j=1 to m do
9. $SU(f_i, f_j) = \dfrac{2IG(f_i, f_j)}{H(f_i) + H(f_j)}$;
10. Construct matrix $\boldsymbol{A}$;
11. $AvgRel(f_i) = \dfrac{1}{m}\sum_{j=1}^{m} SU(f_i, f_j)$;
12. Sort the features according to $AvgRel$ in the descending order;
13. Select the top k features as initial medoids;
14. Partition features into k clusters;
15. for i=1 to k do
16. For feature cluster C_i, select cluster's medoid;
17. Sort the remaining features;
18. Select top $\left\lceil \dfrac{|C| \times r}{m} \right\rceil - 1$ from each cluster;
19. Return S

In the pseudo code, T is the dataset; k is the number of feature clusters for software defect prediction; F is the original software defect feature set; δ is the threshold; r is the number of selected software defect features; S is the output software defect feature subset; w_i denotes the relevance between feature f_i and the target class; C_i represents the ith feature cluster; $|C|$ is the size of the cluster; m represents the number of feature subset; and r represents the number of the final feature subset.

For example, to the kc3 dataset, the index of the final selected feature subset is (31 35 16 24 27 7). These features are highly correlated with the target class, lowly correlated with each other.

4. Experiments and performance analysis

4.1 Datasets and experimental platform

In order to verify the effectiveness of our proposed method, nine software defect datasets are selected from the NASA Metrics Data Program (MDP) repository [29], which mainly design the measurement from LOC counts, Halstead complexity and McCabe complexity, stored in attribute-relation file format (ARFF), were processed with Weka. Table 2 describes these data in terms of the number of instances, the number of features, the number of defective modules, and the number of defect modules.

Table 2 Descriptions of the datasets

Dataset	Attribute	Module	Defective module
pc1	37	1 107	76(6.87%)
pc3	37	1 563	160(10.24%)
pc4	37	1 458	178(12.21%)
kc1	22	2 109	326(15.45%)
kc3	39	458	43(9.39%)
kc4	39	125	61(48.80%)
mc2	39	161	52(32.30%)
cm1	37	505	48(9.50%)
mw1	37	403	31(7.69%)

The hardware for experiments is as follows: 8 G RAM, Intel core i5 processor at 1.8 GHz, and 500 GB hard disk.

4.2 Metrics

In order to evaluate the prediction results, it is necessary to select reasonable performance evaluation metrics. SDP can be simplified to a two-class problem. In Table 3, TP and TN denote the number of positive and negative samples that are classified correctly, while FP and FN denote the number of misclassified positive and negative samples respectively.

Table 3 Confusion matrix for a two-class problem

Class	Predicted positive	Predicted negative
Positive	TP	FN
Negative	FP	TN

Precision is defined as

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{6}$$

Recall is defined as

$$\text{TP rate} = \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{7}$$

which indicates the ratio of positive samples that are classified correctly to actual positive samples.

F-value is defined as

$$\text{F-value} = \frac{(1+\beta^2)\cdot \text{Precision}\cdot \text{Recall}}{\beta^2\cdot \text{Precision}+\text{Recall}}. \tag{8}$$

F-value is a combination of Recall and Precision which are effective metrics for information retrieval community. F-value is high, if both Recall and Precision are high. It can be adjusted through changing the value of β. β corresponds to relative importance of Precision vs. Recall, and is usually set to 1.

The true positive rate (TPR) is the same as Recall.

The false positive rate (FPR) is defined as

$$\text{FPR} = \frac{\text{FP}}{\text{FP}+\text{TN}} \tag{9}$$

The receiver operating characteristic (ROC) curve describes the relationship between TPR and FPR.

On an ROC graph, the TPR is plotted on the Y axis and FPR is plotted on the X axis. In the ROC space, each classifier with a given class distribution and cost matrix is represented by a point on the ROC curve. The area under curve (AUC) is a single-value measurement that originated from the field of signal detection. The value of the AUC ranges from 0 to 1. On the ROC curve, points closing to (0, 1) are preferable, which means low false positive error rate and high recall value. A perfect classifier provides an AUC that equals 1.

AUC is used for evaluating the predictive capability of classifiers. Jiang et al. [30] selected different proportions of training sets on the NASA dataset for experiments to compare the performance differences of different evaluation metrics. The experimental results indicate that AUC is a more stable measurement than F-value.

4.3 Parameter setting

The main parameters of RFC include the number of clusters k, and the threshold δ of ReliefF. We select $\lceil \log_2 M \rceil$ relevant features from the original dataset to construct the final selected feature subset, inspired by Gao et al. [23], where M represents the number of original features. By adjusting δ via experiments, the final feature number is close to $\lceil \log_2 M \rceil$. k is set to be $\lceil \log_2 (m/2) \rceil$ heuristically, and m represents the number of remaining features after removing irrelevant features with ReliefF.

In defect predictor construction phase, we employ the Naïve Bayes and the J48 classification algorithm. We implement 10×10 fold cross validation. The measurement of performance is averaged.

First, we implement experiments to compare our proposed method with the method using all the original features, denoted as NONE. Then, we compare our proposed method with four representative feature selection methods. Both IG and Chi-Square (CS) consider the relevance between feature and target class in terms of IG and card square value respectively. ReliefF measures the relevance between the features and the target class based on instances. Compared with ReliefF, we can find out the influence of feature clustering. Compared with IG and CS, we can find out the influence of different correlation measurement methods on prediction performance. Correlation-based feature selection (CFS) [31] exploits the best first search based on the evaluation of a subset that contains features highly correlated with the target class, yet uncorrelated with each other. This method can deal with both irrelevant features and redundant features.

4.4 Experimental results

The performance of the feature selection algorithm can be usually evaluated from two aspects: in the case of the same classification effect, the smaller the feature subset, the better the performance; in the case of the same effect on the feature reduction, the better the classification effect, the better the performance.

(i) For J48 classifier

Table 4 shows that the RFC algorithm has obvious advantages compared with NONE, IG, CS, ReliefF and CFS based on AUC, which greatly improves the performance of the classifier. RFC achieves the best performance on datasets pc3, kc4 and cm1. On dataset mc2, RFC performs only worse than NONE, and on dataset mw1, RFC performs only worse than CFS. AUC of RFC on average is 0.673, and AUC of CFS is 0.676. Table 5 shows that the F-value of RFC is better than other methods on most datasets.

Table 4 AUC of the J48 classifier after using different feature selection methods

Dataset	NONE	IG	CS	ReliefF	RFC	CFS
pc1	0.669	0.753	0.699	0.519	0.715	0.727
pc3	0.647	0.602	0.601	0.534	0.668	0.646
pc4	0.755	0.873	0.881	0.511	0.680	0.859
kc1	0.700	0.747	0.746	0.715	0.737	0.702
kc3	0.567	0.574	0.624	0.547	0.656	0.679
kc4	0.738	0.750	0.751	0.757	0.767	0.761
mc2	0.647	0.562	0.577	0.611	0.639	0.550
cm1	0.531	0.561	0.553	0.500	0.631	0.575
mw1	0.429	0.551	0.553	0.527	0.564	0.583
Average	0.631	0.664	0.665	0.580	0.673	0.676

Table 5 F-value of the J48 classifier after using different feature selection methods

Dataset	NONE	IG	CS	ReliefF	RFC	CFS
pc1	0.309	0.230	0.191	0.025	0.289	0.222
pc3	0.282	0.016	0.012	0.012	0.133	0.027
pc4	0.518	0.471	0.465	0.004	0.118	0.393
kc1	0.381	0.320	0.330	0.285	0.273	0.277
kc3	0.292	0.040	0.195	0.088	0.094	0.279
kc4	0.789	0.768	0.776	0.778	0.806	0.789
mc2	0.487	0.367	0.377	0.414	0.464	0.330
cm1	0.132	0.026	0.026	0.014	0.103	0.061
mw1	0.213	0.207	0.231	0.139	0.200	0.303
Average	0.378	0.272	0.289	0.195	0.276	0.298

(ii) For Naïve Bayes classifier

Table 6 shows that the RFC has advantages over the NONE, IG, CS, ReliefF, and CFS. Based on AUC, RFC achieves the best performance on datasets pc1, kc4, mc2, cm1, mc2 and mw1. On the dataset kc1, the performance of RFC is very similar to ReliefF. AUC of RFC on average is the highest, followed by NONE and CFS. As shown in Table 7, RFC can achieve the best performance to F-value on most datasets, and F-value of RFC is the highest on average, followed by CFS.

Table 6 AUC of Naïve Bayes after using different feature selection methods

Dataset	NONE	IG	CS	ReliefF	RFC	CFS
pc1	0.733	0.703	0.687	0.712	0.749	0.75
pc3	0.767	0.79	0.8	0.744	0.771	0.78
pc4	0.836	0.825	0.823	0.805	0.813	0.818
kc1	0.792	0.769	0.771	0.789	0.788	0.75
kc3	0.814	0.751	0.765	0.772	0.791	0.787
kc4	0.752	0.742	0.743	0.739	0.753	0.701
mc2	0.703	0.627	0.636	0.628	0.712	0.652
cm1	0.736	0.748	0.741	0.745	0.768	0.746
mw1	0.751	0.753	0.727	0.686	0.755	0.751
Average	0.765	0.745	0.744	0.736	0.767	0.748

Table 7 F-value of Naïve Bayes after using different feature selection methods

Dataset	NONE	IG	CS	ReliefF	RFC	CFS
pc1	0.278	0.269	0.288	0.081	0.267	0.277
pc3	0.256	0.348	0.348	0.171	0.338	0.373
pc4	0.434	0.44	0.437	0.388	0.447	0.43
kc1	0.4	0.381	0.383	0.429	0.421	0.38
kc3	0.346	0.345	0.341	0.324	0.34	0.378
kc4	0.508	0.441	0.437	0.644	0.532	0.433
mc2	0.448	0.425	0.425	0.366	0.49	0.45
cm1	0.307	0.294	0.296	0.304	0.333	0.303
mw1	0.326	0.361	0.365	0.203	0.403	0.391
Average	0.367	0.367	0.369	0.323	0.397	0.379

Figs. 2−5 show AUC and F-value of different feature selection algorithms based on the J48 and Naïve Bayes classifiers. For the J48 classifier, Fig. 2 and Fig. 4 indicate that the advantage of RFC against other algorithms is significant. Compared with the Naïve Bayes defect predictor, RFC also has obvious advantages.

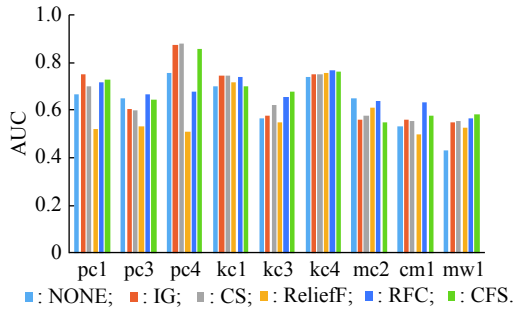

Fig. 2 AUC of J48 after using different feature selection methods

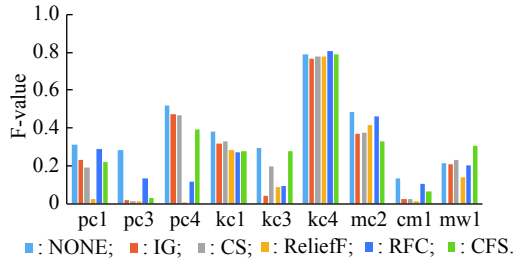

Fig. 3 F-value of J48 after using different feature selection methods

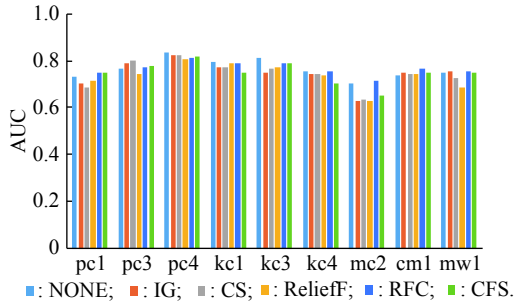

Fig. 4 AUC of Naïve Bayes after using different feature selection methods

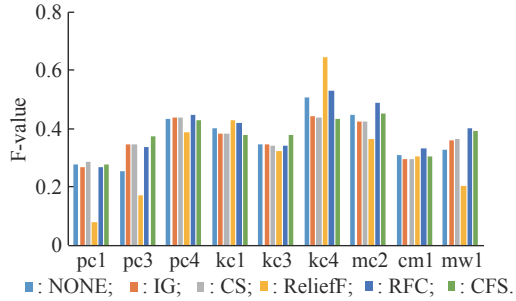

Fig. 5 F-value of Naïve Bayes after using different feature selection methods

We present the proportion of selected features by all the feature selection methods for each dataset in Table 8. The number of features retained by RFC is set to be close to or same as the number of features retained by the IG, CS, and ReliefF.

From Table 8, we can see that RFC on average can obtain 18.43% of selected features. IG, RFC, and CSF rank first with 13.8% of selected features. RFC performs similar to those methods only using feature ranking on the proportion of selected features. Table 9 lists the indexes of the features selected by different feature selection algorithms. RFC can select fewer features than CFS, and more than CS, IG and ReliefF.

Table 8 Proportion of features selected with different feature selection methods

Dataset	IG	CS	ReliefF	RFC	CFS
pc1	15	15	15	17.25	20.67
pc3	15	15	15	17.25	23.42
pc4	15	15	15	18.41	12.50
kc1	23.81	23.81	23.81	27.78	41.75
kc3	15	15	15	17.25	19.25
kc4	15	15	15	17.25	14.75
mc2	15	15	15	17.25	27.67
cm1	15	15	15	16.17	23.25
mw1	15	15	15	17.25	20.92
Average	15.98	15.98	15.98	18.43	22.69

Table 9 Indexes of features on all data sets selected with different feature selection methods

Dataset	IG	CS	ReliefF	RFC	CFS
pc1	0 40 4 34 19 36	3 39 40 4 34 36	38 40 16 27 7 11	0 38 2 27 34 11 36	0 31 3 40 4 30 7 19 36
pc3	0 3 40 19 34 36	0 3 40 34 19 36	38 40 16 27 7 11	0 31 38 16 19 11 36	0 38 3 40 4 27 19 26 34
pc4	38 0 3 40 29 5	28 38 3 40 29 5	9 38 40 16 24 14	9 38 39 35 16 24 27	38 3 40 16 8
kc1	21 16 11 15 12	17 21 16 15 12	21 19 13 9 8	20 19 7 13 9 8	0 6 21 3 7 9 1 8 18
kc3	40 2 32 22 23 26	40 2 32 22 1 6	40 16 18 24 27 7	31 35 16 24 27 7	3 40 32
kc4	12 31 40 2 30 10	12 31 40 2 30 10	31 39 40 2 27 11	12 31 27 1 30 11 36 6	12 31 40 2 1
mc2	12 21 40 2 18 30	12 40 2 18 20 30	40 16 18 27 7 11	2 16 35 18 27 20 7 11	28 21 0 3 40 18 33 2 4 22 20 30 14
cm1	39 40 35 15 4 34	39 40 35 15 4 34	38 31 40 24 27 11	31 38 35 24 27 11	38 40 4 15 20 7 19 10 36
mw1	12 40 22 30 34 36	12 0 40 30 36 10	38 40 24 27 19 11	28 38 39 5 27 8 34 11	21 0 40 4 5 30 19 34 10

4.5 Performance analysis

From the experimental results, we can analyze and draw the following conclusions about the performances of RFC and other methods:

(i) With each above feature selection method, SDP can achieve a better performance with fewer features. All these five feature selection methods can help to select more information content and more distinctive features.

(ii) RFC performs similar to ReliefF on the proportion of selected features. From Table 4 to Table 7, we can see that RFC has a better performance than ReliefF. RFC can effectively improve the performance of SDP by feature clustering.

(iii) As shown in Table 8, RFC selects more features than IG, CS, and ReliefF. The reason is that the number of features selected in each cluster is related to the scale of the cluster, and at least one feature is selected from each cluster.

To sum up, based on clustering, RFC can effectively improve the performance of SDP.

4.6 Time complexity

Suppose that N represents the number of instances of the dataset, M represents the number of features of the original dataset and k represents the number of feature clusters. The calculation time of RFC is mainly composed of five parts:

(i) Calculate relevance between feature and target class.

ReliefF updates the weight of the sample. Its time complexity is $O(M\times N)$. Repeat multiple times, and the number of executions is close to the number of instances. Therefore, the time complexity of ReliefF is $O(M\times N^2)$.

(ii) Calculate the correlation between features.

After ReliefF removes the irrelevant features, the number of remaining features is m. The time complexity of using SU to calculate two features on dataset of N instances is $O(N)$. Therefore, the time complexity of calculating the correlation between features is $O(N\times m^2)$.

(iii) Calculate and rank the average correlation degree of features.

The time complexity of computing the average correlation degree of features is $O(m^2)$, and then quick sorting. Therefore, the time complexity of this part is $O(m^2)$.

(iv) Cluster features.

Divide the original feature set into k clusters, and then update the medoids. The time complexity of this part is $O(m^2)$.

(v) Select representative features.

Select the medoid of the cluster, and sort these features according to their relevance with the target class in the descending order. The time complexity of this part is $O(m\log_2 m)$.

The number of instances N is much larger than M and m, so that the time complexity of RFC is $O(N^2)$. Therefore, we can find that the time complexity of RFC is not higher than the typical feature selection algorithms based on filter, and lower than the feature selection algorithms based on the embedded method.

5. Conclusions

In this paper, we propose a new feature selection algorithm RFC, which first removes irrelevant features, then clusters the remaining features, and finally selects representative features from each cluster. The experimental results demonstrate that RFC achieves a better performance compared with other classical feature selection algorithms on most datasets, which proves the effectiveness of the RFC algorithm in SDP. However, RFC can be further optimized. Our future work will focus on the redundant features of high-dimensional datasets.

References

[1] CHEN X, GU Q, LIU W S, et al. Survey of static software defect prediction. Journal of Software, 2016, 27(1): 1–25. (in Chinese)

[2] WANG Q, WU S J, LI M S. Software defect prediction. Journal of Software, 2008, 19(7): 1565–1580. (in Chinese)

[3] HALSTEAD M H. Elements of software science. New York: Elsevier, 1977.

[4] MCCABE T J. A complexity measure. IEEE Trans. on Software Engineering, 1976, SE-2(4): 308–320.

[5] ABREU F B E, MELO W. Evaluating the impact of object-oriented design on software quality. Proc. of the 3rd International Conference on Software Metrics Symposium, 1996: 90–99.

[6] CHIDAMBER S R, KEMERER C F. A metrics suite for object oriented design. IEEE Trans. on Software Engineering, 1994, 20(6): 1476–1493.

[7] CHEN X, SHEN Y X, MENG S Q, et al. Multi-objective optimization based feature selection method for software defect prediction. Journal of Frontiers of Computer Science and Technology, 2018, 12(9): 1420–1433.

[8] KONONENKO I. Estimating attributes: analysis and extensions of RELIEF. Proc. of the European Conference on Machine Learning, 1994: 171–182.

[9] XIA Y, YAN G Y, SI Q R. A study on the significance of software metrics in defect prediction. Proc. of the 6th International Symposium on Computational Intelligence and Design, 2013: 343–346.

[10] DANIJEL R, MARJAN H, TORKAR R. Software fault prediction metrics: a systematic literature review. Information & Software Technology, 2013, 55(8): 1397–1418.

[11] PUNITHA K, CHITRA S. Software defect prediction using software metrics-a survey. Proc. of the Information Communication and Embedded Systems Conference, 2013: 555–558.

[12] AGRAWAL A, MENZIES T. Is "better data" better than "better data miners"? (on the benefits of tuning SMOTE for defect prediction). https://arxiv.org/pdf/1705.03697.pdf.

[13] JING X Y, WU F, DONG X, et al. An improved SDA based defect prediction framework for both within-project and cross-project class-imbalance problems. IEEE Trans. on Software Engineering, 2017, 43(4): 321–339.

[14] TANTITHAMTHAVORN C, HASSAN A E, MATSUMOTO K. The impact of mislabelling on the performance and interpretation of defect prediction models. Proc. of the 37th International Conference on Software Engineering, 2015: 812–823.

[15] CHEN L, FANG B, SHANG Z W. Tackling class overlap and imbalance problems in software defect prediction. Software Quality Journal, 2016, 26(9): 97–125.

[16] STUCKMAN J, WALDEN J, SCANDARIATO R. The effect of dimensionality reduction on software vulnerability prediction models. IEEE Trans. on Reliability, 2017, 66(1): 17–37.

[17] KIRA K, RENDELL L. A practical approach to feature selection. Proc. of the 9th International Workshop on Machine Learning, 1992: 249–256.

[18] SHIVKUMAR S, WHITEHEAD E J, AKELLA R, et al. Reducing features to improve code change-based bug prediction. IEEE Trans. on Software Engineering, 2013, 39(4): 552–569.

[19] GUO L, MA Y, CUKIC B, et al. Robust prediction of fault-proneness by random forests. Proc. of the 15th Software Reliability Engineering Conference, 2004: 417–428.

[20] MENZIES T, GREENWALD J, FRANK A. Data mining static code attributes to learn defect predictors. IEEE Trans. on Software Engineering, 2007, 33(1): 2–13.

[21] RODRIGUEZ D, RUIZ R, CUADRADO-GALLEGO J J, et al. Detecting fault modules applying feature selection to classifiers. Proc. of the IEEE International Conference on Information Reuse & Integration, 2007: 667–672.

[22] CATAL C, DIRI B. Unlabelled extra data do not always mean extra performance for semi-supervised fault prediction. Expert Systems, 2009, 26(5): 458–471.

[23] GAO K, KHOSHGOFTAAR T M, WANG H, et al. Choosing software metrics for defect prediction: an investigation on feature selection methods. Software: Practice and Experience, 2011, 41(5): 579–606.

[24] WANG H, KHOSHGOFTAAR T M, NAPOLITANO. A comparative study of ensemble feature selection methods for software defect prediction. Proc. of the 9th International Conference on Machine Learning & Applications, 2010: 135–140.

[25] BENNIN K E, KEUNG J, PHANNACHITTA P, et al. MAHAKIL: diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction. IEEE Trans. on Software Engineering, 2018, 44(6): 534–550.

[26] MIHOLCA D L, CZIBULA G, CZIBULA I G. A novel approach for software defect prediction through hybridizing gradual relational association rules with artificial neural networks. Information Sciences, 2018, 441: 152–170.

[27] CAO Y, DING Z M, XUE F, et al. An improved twin support vector machine based on multi-objective cuckoo search for software defect prediction. International Journal of Bio-Inspired Computation, 2018, 11(4): 282–291.

[28] YU L, LIU H. Feature selection for high-dimensional data: a fast correlation-based filter solution. Proc. of the 20th International Conference on Machine Learning, 2003: 856–863.

[29] CHAPMAN M, CALLIS P, JACKSON W. Metrics data pro-

gram. http://mdp.ivv.nasa.gov.
[30] JIANG Y, LIN J, CUKIC B, et al. Variance analysis in software fault prediction models. Proc. of the 20th International Symposium on Software Reliability Engineering, 2009: 99–108
[31] HALL M A. Correlation-based feature selection for discrete and numeric class machine learning. Proc. of the 17th International Conference on Machine Learning, 2000: 359–366.

Biographies

XU Xiaolong was born in 1977. He received his B.S. degree in computer and its applications, M.S. degree in computer software and theories and Ph.D. degree in communications and information systems at Nanjing University of Posts and Telecommunications, Nanjing, China, in 1999, 2002 and 2008, respectively. He worked as a postdoctoral researcher at Station of Electronic Science and Technology, Nanjing University of Posts and Telecommunications from 2011 to 2013. He is currently a professor in College of Computer, Nanjing University of Posts and Telecommunications. He is a senior member of China Computer Federation. His current research interests include cloud computing and big data, mobile computing, intelligent agent and information security.
E-mail: xuxl@njupt.edu.cn

CHEN Wen was born in 1994. He received his B.E. degree in computer science and technology from Anhui Engineering University, Wuhu, China, in 2016. He works as an engineer in Institute of Big Data Research at Yancheng, Nanjing University of Posts and Telecommunications, Yancheng, China. His research interest is data analysis.
E-mail: 1216043012@njupt.edu.cn

WANG Xinheng was born in 1968. He received his B.E. and M.S. degrees in electrical engineering from Xi'an Jiaotong University, Xi'an, China, in 1991 and 1994, respectively, and Ph.D. degree in computing and electronics from Brunel University, Uxbridge, UK, in 2001. He is currently a professor of networks with the School of Computing and Engineering, University of West London, London, UK. His current research interests include wireless networks, Internet of Things, converged indoor positioning, cloud computing, and applications of wireless and computing technologies for health care.
E-mail: xinheng.wang@uwl.ac.uk