Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

A feature selection approach based on a similarity measure for software defect prediction*

Qiao YU[†1], Shu-juan JIANG[†‡1,2], Rong-cun WANG[1], Hong-yang WANG[1]

([1]*School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China*)

([2]*Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China*)

[†]E-mail: yuqiao@cumt.edu.cn; shjjiang@cumt.edu.cn

Received June 11, 2016; Revision accepted Sept. 14, 2016; Crosschecked Nov. 26, 2017

Abstract: Software defect prediction is aimed to find potential defects based on historical data and software features. Software features can reflect the characteristics of software modules. However, some of these features may be more relevant to the class (defective or non-defective), but others may be redundant or irrelevant. To fully measure the correlation between different features and the class, we present a feature selection approach based on a similarity measure (SM) for software defect prediction. First, the feature weights are updated according to the similarity of samples in different classes. Second, a feature ranking list is generated by sorting the feature weights in descending order, and all feature subsets are selected from the feature ranking list in sequence. Finally, all feature subsets are evaluated on a k-nearest neighbor (KNN) model and measured by an area under curve (AUC) metric for classification performance. The experiments are conducted on 11 National Aeronautics and Space Administration (NASA) datasets, and the results show that our approach performs better than or is comparable to the compared feature selection approaches in terms of classification performance.

Key words: Software defect prediction; Feature selection; Similarity measure; Feature weights; Feature ranking list
https://doi.org/10.1631/FITEE.1601322 **CLC number:** TP311

1 Introduction

Software defect prediction plays an important role in software testing. It is aimed to find potential defects before releasing a new software product, and is of great significance in improving software quality and software reliability.

Software defect prediction can be regarded as a binary classification problem. Software modules can be divided into defective modules or non-defective modules based on historical data and software features. Software features can reflect the characteristics of software modules, such as complexity, the number of operators, and operands. However, some of these features may be more relevant to the class (defective or non-defective), but others may be redundant or irrelevant. By feature selection the high correlation features can be selected from high-dimensional features. In other words, those features that are more relevant to the class can be selected from all features. Therefore, introducing feature selection into software defect prediction can not only improve its efficiency, but also improve its accuracy (Miao *et al.*, 2012; Khoshgoftaar *et al.*, 2014; Liu *et al.*, 2014).

Generally, feature selection approaches can be divided into feature ranking and feature subset selection with different outputs. Feature ranking is designed to evaluate the worth of features and generate

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (Nos. 61673384 and 61502497), the Guangxi Key Laboratory of Trusted Software (No. kx201530), the China Postdoctoral Science Foundation (No. 2015M581887), and the Scientific Research Innovation Project for Graduate Students of Jiangsu Province, China (No. KYLX15_1443)

ORCID: Shu-juan JIANG, http://orcid.org/0000-0003-0643-0565

a feature ranking list. There are many feature ranking approaches, such as OneR (Holte, 1993), ReliefF (Kononenko, 1994), Correlation (Guyon and Elisseeff, 2003), and Gain Ratio (Karegowda *et al.*, 2010). OneR is designed to evaluate the worth of features by the OneR classifier. ReliefF is a feature weighted approach based on a distance measure. Correlation is designed to measure the importance of features by calculating Pearson's correlation coefficient between different features and the class, and Gain Ratio uses the information gain ratio to evaluate the worth of features. These four approaches use different measures to evaluate the worth of features. In contrast, feature subset selection is aimed to select the optimal feature subset combined with search algorithms. Correlation-based feature selection (Hall, 1999) is one of the most commonly used feature subset selection approaches. It is used to evaluate the individual predictive ability of each feature and the redundancy between different features. Moreover, it can obtain the optimal feature subset using search algorithms, such as best-first search and particle swarm optimization (PSO).

By feature selection those features that are more relevant to the class can be selected. However, many approaches fail to fully consider the distribution differences of samples in different classes, which may reduce the efficiency of feature selection and the performance of software defect prediction.

This paper presents a feature selection approach based on a similarity measure (SM) for software defect prediction. In our approach, we design a feature ranking algorithm to update the feature weights according to the similarity of samples in different classes, and we can obtain a feature ranking list by sorting the feature weights in descending order. Then, we select all feature subsets from the feature ranking list in sequence and evaluate them on a k-nearest neighbor (KNN) (Aha *et al.*, 1991) model. The area under the receiver operating characteristic curve (AUC) metric (Huang and Ling, 2005) is used to measure the classification performance. Finally, we conduct experiments on 11 National Aeronautics and Space Administration (NASA) datasets and make comparisons with other feature selection approaches to show the validity of our approach.

The main contributions of this paper are as follows:

1. A feature ranking algorithm is designed to update the feature weights and generate the feature ranking list.
2. A feature selection approach based on a similarity measure is proposed for software defect prediction.

2 Related work

In recent years, feature selection has become the focus of machine learning and data mining. Based on the association with learning algorithms, feature selection approaches can be divided into three categories: embedded, filter, and wrapper (Liu *et al.*, 2010). Embedded approaches take feature selection as a part of the learning algorithm, and it can select the optimal features during the training process. Filter approaches focus on analyzing the characteristics of data to obtain the feature subset, and do not use any learning algorithms. Wrapper approaches use learning algorithms as the evaluation criteria to identify those relevant features. Filter approaches are the most commonly used in empirical studies.

The evaluation criteria of filter approaches include the distance, information, dependency, and consistency measures (Liu and Yu, 2005). For example, the Relief algorithm (Kira and Rendell, 1992) and its improved ReliefF algorithm use distance measures to update the feature weights. Yang and Gu (2004) proposed a feature subset selection method based on mutual information, and they also defined a redundancy-synergy coefficient to measure the redundancy and synergy of features. They conducted experiments on UCI datasets to show the validity of this method. Liu *et al.* (2009) proposed a feature selection approach based on dynamic mutual information, and Wang Z *et al.* (2015) proposed a feature selection approach based on mutual information to select features with maximal relevance and minimal redundancy.

At present, feature selection is widely used in various classification problems. For example, Uysal and Gunal (2012) proposed a probabilistic feature selection method for text classification. The results indicated that this method performs better than the compared methods in terms of classification accuracy, reduction rate, and processing time. Han *et al.* (2013) proposed a feature selection framework based on the topological similarity of subgraphs for graph classification. It was designed to remove the redundant subgraphs to select an optimal feature subset.

The results showed that this framework outperforms the compared methods in terms of classification accuracy and running time. Xue *et al.* (2013) regarded feature selection as a multi-objective problem for maximizing the performance with fewer features, and they applied PSO for multi-objective feature selection. Ghareb *et al.* (2016) combined filter feature selection methods with an enhanced genetic algorithm, indicating that this approach is efficient for dimensionality reduction in text classification.

Feature selection has also been found to be efficient in software defect prediction. Gao *et al.* (2011) presented a hybrid feature selection approach, which combines feature ranking with feature subset selection. The results indicated that the proposed hybrid approach outperforms other feature subset selection approaches for software defect prediction. Khoshgoftaar *et al.* (2014) proposed an iterative feature selection approach for the class imbalance problem. The experimental results showed that the proposed iterative approach outperforms non-iterative approaches for software defect prediction. Liu *et al.* (2014) combined feature clustering with feature ranking for software defect prediction, and they investigated the validity of this approach on Eclipse and projects from NASA. Laradji *et al.* (2015) combined feature selection with ensemble learning for dealing with class imbalance and feature redundancy problems. Wang H *et al.* (2015) investigated the stability of feature selection approaches with changed datasets in software quality prediction.

3 Our approach

In the process of software defect prediction, the features that are more relevant to the class should make samples in the same class more compact in distribution, but make samples in different classes more discrete. Based on this idea, we present a feature selection approach based on a similarity measure. The details are described as follows.

3.1 Similarity measure

There are many similarity measures in machine learning algorithms, such as distances known as Euclidean, Manhattan, and Hamming. Euclidean distance is the most widely used method. It measures the distance between two points in space. Similarly, it can measure the distance between two vectors.

Generally, a dataset is composed of a large number of samples or instances, and each sample contains many features to describe the characteristics of software modules. Suppose that a standard dataset is expressed as $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, and $\boldsymbol{x}_i \in \mathbb{R}^d$ $(i = 1, 2, \ldots, n)$. It indicates that there are n samples in dataset D and d features in each sample. These features are expressed as $F = \{f_1, f_2, \ldots, f_d\}$, and each sample can be regarded as a d-dimensional vector. The Euclidean distance between two samples can be expressed as

$$D_{ij} = \sqrt{\sum_{m=1}^{d} (x_{im} - x_{jm})^2}. \tag{1}$$

However, when the features are on different scales, the Euclidean distance cannot measure the actual distance of two samples. In this case, we attempt to standardize all features to eliminate the constraint of different scales. The standardization is shown in Eq. (2):

$$x^*_{im} = \frac{x_{im} - \mu_{f_m}}{\sigma_{f_m}},\ m = 1, 2, \ldots, d, \tag{2}$$

where x_{im} and x^*_{im} represent the values before and after standardization respectively, μ_{f_m} represents the average of feature f_m, and σ_{f_m} represents the standard deviation of feature f_m. After standardization, the values of each feature fit to the standard normal distribution with an average of 0 and standard deviation of 1. Then the standardized Euclidean distance between two samples can be expressed as

$$D^*_{ij} = \sqrt{\sum_{m=1}^{d} \left(\frac{x_{im} - x_{jm}}{\sigma_{f_m}} \right)^2} = \sqrt{\sum_{m=1}^{d} (x^*_{im} - x^*_{jm})^2}. \tag{3}$$

The smaller the value of D^*_{ij} is, the higher the similarity of two samples is. If the similarity of two samples in different classes is higher, the features with larger feature differences between them may be more relevant to the class. Based on this, we present the following feature selection approach.

3.2 Feature selection based on a similarity measure

The overview of our approach is shown in Fig. 1. First, we count the number of defective samples

and the number of non-defective samples in dataset D, marked as n_1 and n_2. The imbalance ratio of dataset D is defined as the number of non-defective samples divided by the number of defective samples (Galar *et al.*, 2012) as n_2/n_1. Based on this definition, we redefine the imbalance ratio as the floor of n_2/n_1, $\lfloor n_2/n_1 \rfloor$. Then, we standardize all features in dataset D and calculate the standardized Euclidean distances between each defective sample and all non-defective samples. After that, we select k nearest non-defective samples from each defective sample according to the standardized Euclidean distances, and calculate the feature differences between each defective sample and its k nearest non-defective samples. Note that k is equal to the imbalance ratio of dataset D. Finally, we update the feature weights according to the feature differences and obtain a feature ranking list.

The proposed SM approach is also a feature weighted approach based on a distance measure, which is similar to ReliefF. However, their feature weighted approaches are different. SM updates the feature weights according to the feature differences of each defective sample and its nearest non-defective samples, and ReliefF updates the feature weights according to the feature differences of a random sample's nearest neighbors in the same class and the opposite class. As a result, their feature ranking lists are also different.

The proposed feature ranking algorithm is described in Algorithm 1.

First, we standardize the dataset DataSet (line 1) and obtain the nearest neighbors k (lines 2–5). Then, we calculate the standardized Euclidean distance (line 8). We select k nearest non-defective samples from defective sample $\boldsymbol{x}_i$ and save them into kNeighborList (line 10). Next, we calculate the feature differences and update the feature weights (lines 11–14). The larger difference of one feature indicates that this feature is more relevant to the class, and we would give a larger weight for such a feature. We can obtain the final weight of each feature after updating kn_1 times (lines 6–15). Finally, we can obtain the feature ranking list and return WeightList (lines 16 and 17). The time complexity of Algorithm 1 can be expressed as

$$T = n_1(n_2 + n_2 \log_2 n_2 + kd + kd \log_2 d) + d \log_2 d. \tag{4}$$

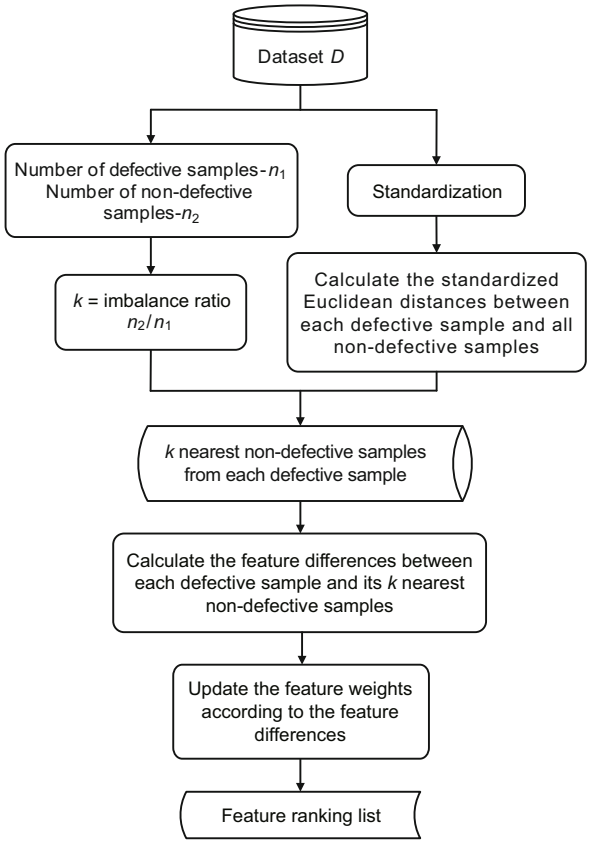

Fig. 1 Overview of our approach

In the brackets, n_2 represents the complexity of calculating the Euclidean distances between one defective sample and all non-defective samples, and $n_2 \log_2 n_2$ represents the complexity of Euclidean distance ranking. Then, d represents the number of features, and kd represents the complexity of calculating the feature differences for k nearest neighbors. $kd \log_2 d$ represents the complexity of the feature difference ranking for k nearest neighbors. For most datasets, $n_2 \log_2 n_2$ is larger than others in the brackets. Outside the brackets, $d \log_2 d$ represents the complexity of the final feature weight ranking. Generally, $n_1 n_2 \log_2 n_2$ is larger than $d \log_2 d$, so the time complexity of Algorithm 1 is $O(n_1 n_2 \log_2 n_2)$.

To better understand the process of Algorithm 1, we give an example in Table 1.

As shown in Table 1, there are nine samples in dataset D and three features in each sample of dataset D. $\boldsymbol{x}_1$ – $\boldsymbol{x}_3$ are defective samples, and $\boldsymbol{x}_4$ – $\boldsymbol{x}_9$ are non-defective samples. The number of nearest neighbors k is equal to 2. Following steps 1–5, we can obtain the final weight of each feature and the feature ranking list $\{f_3, f_2, f_1\}$. It indicates that f_3 is the

Algorithm 1 Feature ranking

Input: DataSet, the dataset
Output: WeightList, the feature ranking list
1: StandSet = Standardize(DataSet);
2: Divide StandSet into defective dataset DefectSet and non-defective dataset NonDefectSet;
3: n_1 = DefectSet.size();
4: n_2 = NonDefectSet.size();
5: Number of nearest neighbors k = Math.floor(n_2/n_1);
6: **for each** DefectSet **do**
7: **for each** NonDefectSet **do**
8: Calculate the standardized Euclidean distance between defective sample $\boldsymbol{x}_i$ in DefectSet and non-defective sample $\boldsymbol{x}_j$ in NonDefectSet;
9: **end for**
10: Select k nearest non-defective samples from $\boldsymbol{x}_i$ and save them into kNeighborList;
11: **for each** kNeighborList **do**
12: Calculate the feature differences between $\boldsymbol{x}_i$ and each sample in kNeighborList;
13: Update the feature weights according to the feature differences;
14: **end for**
15: **end for**
16: WeightList = the feature ranking list by sorting the final feature weights in descending order;
17: **return** WeightList

most relevant to the class, and that f_1 is the least relevant to the class.

Based on the feature ranking list, we can select all feature subsets in sequence. Suppose that the feature ranking list is expressed as $\{f_d, f_{d-1}, f_{d-2}, \ldots, f_2, f_1\}$, and d represents the number of features. The top features indicate that they are more relevant to the class. We can obtain d feature subsets in sequence according to this ranking list, as $\{f_d\}$, $\{f_d, f_{d-1}\}$, $\{f_d, f_{d-1}, f_{d-2}\}, \cdots$, $\{f_d, f_{d-1}, f_{d-2}, \ldots, f_2\}$, $\{f_d, f_{d-1}, f_{d-2}, \ldots, f_2, f_1\}$. We will evaluate the classification performance of these feature subsets in the following experiments.

Table 1 Example of our algorithm

$D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_9\}$, $F = \{f_1, f_2, f_3\}$; Defective: $\boldsymbol{x}_1 - \boldsymbol{x}_3$; non-defective: $\boldsymbol{x}_4 - \boldsymbol{x}_9$; Nearest neighbors: $k = 2$.
Step 1: calculate the standardized Euclidean distances, e.g., between $\boldsymbol{x}_1$ and $\boldsymbol{x}_4 - \boldsymbol{x}_9$: $D^*_{14} = 0.3321$, $D^*_{15} = 0.8532$, $D^*_{16} = 0.2200$, $D^*_{17} = 1.5860$, $D^*_{18} = 1.9931$, $D^*_{19} = 0.9840$. $\boldsymbol{x}_4$ and $\boldsymbol{x}_6$ are the nearest neighbors.
Step 2: calculate the feature differences, e.g., between $\boldsymbol{x}_1$ and $\boldsymbol{x}_4$: $d(f_1) = \lvert x_{11} - x_{41}\rvert = 0.7320$, $d(f_2) = \lvert x_{12} - x_{42}\rvert = 0.4572$, $d(f_3) = \lvert x_{13} - x_{43}\rvert = 1.5330$. $d(f_3) > d(f_1) > d(f_2)$.
Step 3: obtain the feature weights: $W_{14}(f_1) = 2$, $W_{14}(f_2) = 1$, $W_{14}(f_3) = 3$. Repeat step 2 to calculate the feature differences between $\boldsymbol{x}_1$ and $\boldsymbol{x}_6$. Obtain new feature weights, e.g., $W_{16}(f_1) = 1$, $W_{16}(f_2) = 2$, $W_{16}(f_3) = 3$.
Step 4: update the feature weights: $W(f_1) = W_{14}(f_1) + W_{16}(f_1) = 3$, $W(f_2) = W_{14}(f_2) + W_{16}(f_2) = 3$, $W(f_3) = W_{14}(f_3) + W_{16}(f_3) = 6$.
Step 5: obtain the feature ranking list: Replace $\boldsymbol{x}_1$ with $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$, and repeat steps 1–4 to update $W(f_1)$, $W(f_2)$, and $W(f_3)$. Obtain the final weight of each feature, e.g., $W(f_1) = 8$, $W(f_2) = 10$, $W(f_3) = 18$. Obtain the feature ranking list $\{f_3, f_2, f_1\}$.

4 Empirical study

To investigate the validity of our approach, we design experiments on datasets in the Tera-PROMISE repository (http://openscience.us/repo/). All experiments are conducted on Open JDK 1.7 and Weka 3.7 (http://www.cs.waikato.ac.nz/ml/weka/).

4.1 Experimental subjects

We select 11 NASA datasets as the subjects. These datasets are commonly used in software defect prediction. Gray *et al.* (2011) indicated that the quality of datasets is important for software defect prediction, and there are constant, repeated, and inconsistent features in the original NASA datasets. Moreover, Shepperd *et al.* (2013) indicated that there are many repeated and inconsistent data in the original NASA datasets, and they provided the cleaned datasets. Therefore, we use these cleaned NASA datasets, which can be obtained from the Tera-PROMISE repository.

The details of NASA datasets are described in Table 2. It shows the names of these datasets (column 1) and the number of features (except for the class feature) in each dataset (column 2). Next, it shows the number of all samples, defective samples, and non-defective samples, respectively (columns 3–5). Finally, it shows the defect rate and imbalance ratio of each dataset (columns 6 and 7). The higher the value of the imbalance ratio is, the more imbalanced the dataset is. For example, the imbalance ratio of dataset MC1 is equal to 42, and that of PC2 is equal to 45. It indicates that MC1 and PC2 are more imbalanced than other datasets. Note that we

Table 2 NASA datasets

Dataset	Number of features	Number of all samples	Number of defective samples	Number of non-defective samples	Defect rate	Imbalance ratio
CM1	37	327	42	285	12.8%	6
JM1	21	7782	1672	6110	21.5%	3
KC3	39	194	36	158	18.6%	4
MC1	38	1988	46	1942	2.3%	42
MC2	39	125	44	81	35.2%	1
MW1	37	253	27	226	10.7%	8
PC1	37	705	61	644	8.7%	10
PC2	36	745	16	729	2.1%	45
PC3	37	1077	134	943	12.4%	7
PC4	37	1287	177	1110	13.8%	6
PC5	38	1711	471	1240	27.5%	2

take the imbalance ratio as the number of nearest neighbors k in our approach.

All features in the above datasets are numeric, and they are designed at the function level, including McCabe metrics (McCabe, 1976) and Halstead metrics (Halstead, 1977). McCabe metrics use a flow graph to measure the complexity of software modules, and Halstead metrics use the number of operators and operands to measure the complexity of software modules. They consider that the more complicated modules tend to be defective. Based on these features, our approach aims to fully measure the correlation between different features and the class. What is more, we expect to select those features that are more relevant to the class.

4.2 Experimental design

We design experiments to make comparisons with four feature selection approaches, OneR, ReliefF, Correlation, and Gain Ratio, which have been mentioned in Section 1. We mark them as OR, RF, CL, and GR, respectively. They are all feature ranking approaches that can be implemented in Weka. We take the default parameters of these approaches from Weka. In addition, the Wilcoxon signed rank test (Wilcoxon, 1945), a non-parametric test for two related samples, is applied to determine the statistical significance of different approaches.

We use the KNN model as the classifier, which can also be implemented in Weka. In our experiments, the parameter K (number of nearest neighbors) of KNN is set to 10, and the parameter 'distanceWeighting' is set to 'Weight by 1/distance'. All experiments are conducted over 10 times 10-fold cross-validation.

Next, we should select a reasonable metric to measure the classification performance. Generally, precision (P), recall (R), F-measure, and AUC metrics (part or all) are commonly used in software defect prediction (Jing *et al.*, 2014a; 2014b; 2015; Nam and Kim, 2015a). Jiang *et al.* (2009) proved that AUC is more accurate and reliable than other metrics. Moreover, AUC has been widely used in empirical studies (Catal and Diri, 2009; Khoshgoftaar *et al.*, 2014; Laradji *et al.*, 2015; Tantithamthavorn *et al.*, 2016). Therefore, we also use AUC to measure the classification performance in our experiments.

4.3 Experimental results and analysis

To fully compare the performance of different feature selection approaches, we evaluate all feature subsets obtained from the feature ranking list in sequence. Considering that SM and RF are feature weighted approaches based on distance measures, we first compare them as follows:

We apply SM and RF to perform feature selection on each dataset in Table 2, and obtain two different feature ranking lists. After that, we use AUC to measure the classification performance of all feature subsets obtained from the above ranking lists in sequence. The results are displayed with the line charts in Fig. 2, where the x-axis represents the number of features, which also represents different feature subsets, and the y-axis represents the value of AUC. The value of AUC is between 0 and 1. The larger the value of AUC is, the higher the classification performance is.

As shown in Fig. 2, we find that SM performs better than RF on most datasets. We also find that there are indeed variations of the performance (AUC) with different numbers of features. To be specific, the performance can be significantly improved

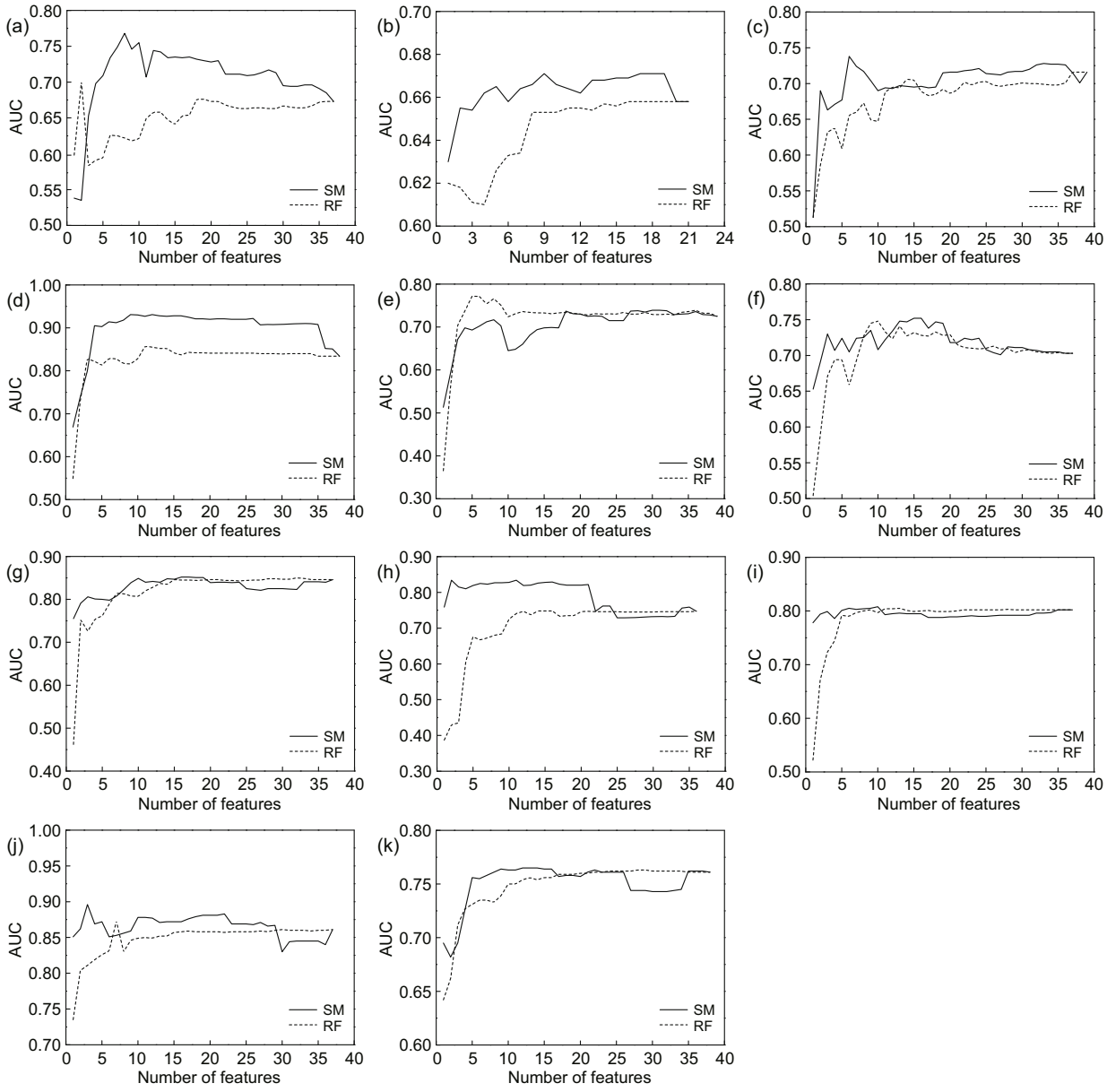

Fig. 2 Comparisons of similarity measure (SM) and ReliefF (RF) with different numbers of features on NASA datasets: (a) CM1; (b) JM1; (c) KC3; (d) MC1; (e) MC2; (f) MW1; (g) PC1; (h) PC2; (i) PC3; (j) PC4; (k) PC5

at the beginning of the line charts, because the top features are more relevant to the class. As the number of features increases, the performance would be stable or may drop slightly due to some redundant or irrelevant features.

We also design three experiments to make comparisons with OR, CL, and GR approaches. At the same time, we apply the Wilcoxon signed rank test to evaluate the statistical significance of different approaches. The significance level is $\alpha = 0.05$. If $P < 0.05$, it indicates that there is a significant difference between two approaches. If $P > 0.05$, it indicates that there is no significant difference between two approaches.

All test results are listed in Table 3. It shows the average performance of all feature subsets with our approach and four state-of-the-art approaches, marked as SM/A, OR/A, RF/A, CL/A, and GR/A (columns 2–6), respectively. Then, it shows the P values of SM against OR, RF, CL, and GR, marked

Table 3 Wilcoxon signed rank test results of SM against OR, RF, CL, and GR ($\alpha = 0.05$)

Dataset	SM/A	OR/A	RF/A	CL/A	GR/A	P(SM-OR)	P(SM-RF)	P(SM-CL)	P(SM-GR)
CM1	0.7069	0.6813	0.6514	0.7006	0.6900	0.0002	1.45e−5	0.0059	0.0396
JM1	0.6629	0.6569	0.6446	0.6567	0.6450	0.0064	0.0001	0.0083	0.0001
KC3	0.7033	0.6543	0.6799	0.7275	0.7168	1.06e−6	1.09e−6	**1.08e−5**	0.1254
MC1	0.8977	0.7922	0.8269	0.8848	0.9011	2.56e−7	1.55e−7	0.0471	0.7399
MC2	0.7046	0.7044	0.7212	0.6952	0.6953	0.9499	**0.0016**	0.0119	0.0553
MW1	0.7183	0.6955	0.7041	0.7377	0.7222	0.0001	0.0015	**0.0001**	0.2101
PC1	0.8295	0.7926	0.8179	0.8512	0.8514	2.06e−5	0.4601	**2.68e−7**	**6.58e−5**
PC2	0.7872	0.7444	0.7032	0.8188	0.8134	4.33e−6	8.12e−5	**0.0010**	**2.85e−5**
PC3	0.7946	0.7776	0.7860	0.7905	0.8026	0.0025	0.1072	0.0100	**0.0005**
PC4	0.8650	0.8161	0.8481	0.8695	0.8836	3.16e−6	0.0016	**0.0278**	**1.09e−6**
PC5	0.7509	0.7383	0.7484	0.7441	0.7354	0.0001	0.4262	0.0067	6.78e−5

$\underline{P}$: our approach (SM) performs better; $\boldsymbol{P}$: the compared approach performs better; P: no significant difference

as P(SM-OR), P(SM-RF), P(SM-CL), and P(SM-GR) (columns 7–10), respectively. As noted at the bottom of Table 3, underlined P values represent that SM outperforms the compared approach, and bold P values represent that the compared approach outperforms SM. Others represent that there is no significant difference between two approaches.

From the P(SM-OR) values in Table 3, we find that SM outperforms OR on most datasets, but there is no significant difference on dataset MC2. From the P(SM-RF) values in Table 3, we also find that SM outperforms RF on most datasets, which is consistent with Fig. 2. However, SM is comparable to RF on datasets PC1 and PC5. Unfortunately, RF outperforms SM on dataset MC2, because the imbalance ratio of dataset MC2 equals to 1. In this case, the value of k is equal to 1. As a result, the accuracy of our approach may be reduced with fewer neighbors.

Similarly, from the P(SM-CL) and P(SM-GR) values in Table 3, we find that SM outperforms CL or GR on part of datasets. However, CL outperforms SM on datasets KC3, MW1, PC1, PC2, and PC4. GR outperforms SM on datasets PC1, PC2, PC3, and PC4. There is no significant difference between SM and GR on datasets KC3, MC1, MC2, and MW1.

To clearly illustrate the efficiency of our approach, we count the test results in Table 3. We use 'Win', 'Tie', and 'Loss' to show the comparative results in Table 4.

'Win' indicates that SM outperforms the compared approach, and 'Tie' indicates that there is no significant difference between two approaches. 'Loss' indicates that the compared approach outperforms SM. Columns 2–5 represent the numbers of datasets matched with Win/Tie/Loss results. As shown in Table 4, we can conclude that SM outperforms OR and RF significantly, and SM is comparable to CL and GR.

Table 4 Win/Tie/Loss results

	SM-OR	SM-RF	SM-CL	SM-GR
Win	10	8	6	3
Tie	1	2	0	4
Loss	0	1	5	4

However, the performance of feature selection approaches may be affected by many factors, such as the dataset size, defect rate, and imbalance ratio. We find that SM performs better than the compared approaches on a large-scale dataset JM1. Moreover, SM performs better than or is comparable to the compared approaches on imbalanced dataset MC1. However, SM performs better than OR and RF only on imbalanced dataset PC2, because there are only 16 defective samples in dataset PC2, which may decrease the total number of nearest neighbors in our algorithm, thereby affecting the performance of our approach.

4.4 Discussions

We find several threats to the validity of our experiments, which can be summarized into internal validity and external validity.

On one hand, if the number of defective samples is too small, or the number of nearest neighbors k is too small, the internal validity may be affected. Our approach is designed to search k nearest non-defective samples from each defective sample. As a result, the number of defective samples may affect the search times. In addition, k is equal to the imbalance ratio of datasets, which can reduce the negative effect of the class imbalance problem. On the other hand, the quality of datasets may be the threat to the

external validity of our experiments. We conduct experiments on 11 NASA datasets with different sizes, defect rates, and imbalance ratios for full comparisons. Moreover, these datasets are commonly used in software defect prediction.

Generally, when evaluating the performance of feature selection approaches, we should set a reasonable threshold to select the optimal feature subset. Gao *et al.* (2011) investigated that the performance may be improved or unchanged when 85% features were eliminated for software defect prediction. Based on this, Nam and Kim (2015b) selected the top 15% features from the feature ranking list for heterogeneous defect prediction. However, this may not apply to different datasets. In contrast, Duch *et al.* (2004) made an attempt to combine feature ranking with feature subset selection, and evaluated all feature subsets obtained from the feature ranking list in sequence. Xu *et al.* (2012) followed this method to compare the performances of different feature selection approaches. Therefore, we apply this method to evaluate all feature subsets in our experiments, and it has proved efficient for software defect prediction.

5 Conclusions and future work

This paper presents a feature selection approach based on a similarity measure for software defect prediction. We design a feature ranking algorithm to update the feature weights and generate a feature ranking list. In our approach, we evaluate all feature subsets obtained from the feature ranking list in sequence, thereby making full comparisons with other feature selection approaches.

To show the validity of our approach, we conduct experiments on 11 NASA datasets and make comparisons with four feature selection approaches. The experimental results show that our approach performs better than or is comparable to the compared approaches. Moreover, our approach performs better than the compared approaches on large-scale datasets and imbalanced datasets.

However, we have evaluated our approach only on the KNN model. More prediction models should be evaluated in following work.

References

Aha, D.W., Kibler, D., Albert, M.K., 1991. Instance-based learning algorithms. *Mach. Learn.*, **6**(1):37-66. https://doi.org/10.1007/BF00153759

Catal, C., Diri, B., 2009. Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem. *Inform. Sci.*, **179**(8):1040-1058. https://doi.org/10.1016/j.ins.2008.12.001

Duch, W., Wieczorek, T., Biesiada, J., *et al.*, 2004. Comparison of feature ranking methods based on information entropy. Int. Joint Conf. on Neural Networks, p.1415-1419. https://doi.org/10.1109/IJCNN.2004.1380157

Galar, M., Fernández, A., Barrenechea, E., *et al.*, 2012. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Trans. Syst. Man Cybern. Part C*, **42**(4):463-484. https://doi.org/10.1109/TSMCC.2011.2161285

Gao, K., Khoshgoftaar, T.M., Wang, H., *et al.*, 2011. Choosing software metrics for defect prediction: an investigation on feature selection techniques. *Softw. Pract. Exper.*, **41**(5):579-606. https://doi.org/10.1002/spe.1043

Ghareb, A.S., Bakar, A.A., Hamdan, A.R., 2016. Hybrid feature selection based on enhanced genetic algorithm for text categorization. *Expert Syst. Appl.*, **49**:31-47. https://doi.org/10.1016/j.eswa.2015.12.004

Gray, D., Bowes, D., Davey, N., *et al.*, 2011. The misuse of the NASA metrics data program data sets for automated software defect prediction. Int. Conf. on Evaluation and Assessment in Software Engineering, p.96-103. https://doi.org/10.1049/ic.2011.0012

Guyon, I., Elisseeff, A., 2003. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, **3**:1157-1182.

Hall, M.A., 1999. Correlation-Based Feature Selection for Machine Learning. University of Waikato, Hamilton, New Zealand.

Halstead, M.H., 1977. Elements of Software Science. Elsevier, New York, USA.

Han, Y., Park, K., Guan, D., *et al.*, 2013. Topological similarity-based feature selection for graph classification. *Comput. J.*, **58**(9):1884-1893. https://doi.org/10.1093/comjnl/bxt123

Holte, R.C., 1993. Very simple classification rules perform well on most commonly used datasets. *Mach. Learn.*, **11**(1):63-90. https://doi.org/10.1023/A:1022631118932

Huang, J., Ling, C.X., 2005. Using AUC and accuracy in evaluating learning algorithms. *IEEE Trans. Knowl. Data Eng.*, **17**(3):299-310. https://doi.org/10.1109/TKDE.2005.50

Jiang, Y., Lin, J., Cukic, B., *et al.*, 2009. Variance analysis in software fault prediction models. Int. Symp. on Software Reliability Engineering, p.99-108. https://doi.org/10.1109/ISSRE.2009.13

Jing, X., Ying, S., Zhang, Z., *et al.*, 2014a. Dictionary learning based software defect prediction. Int. Conf. on Software Engineering, p.414-423. https://doi.org/10.1145/2568225.2568320

Jing, X., Zhang, Z., Ying, S., *et al.*, 2014b. Software defect prediction based on collaborative representation classification. Companion of Int. Conf. on Software Engineering, p.632-633. https://doi.org/10.1145/2591062.2591151

Jing, X., Wu, F., Dong, X., *et al.*, 2015. Heterogeneous cross-company defect prediction by unified metric representation and CCA-based transfer learning. Joint Meeting

on Foundations of Software Engineering, p.496-507. https://doi.org/10.1145/2786805.2786813

Karegowda, A.G., Manjunath, A.S., Jayaram, M.A., 2010. Comparative study of attribute selection using gain ratio and correlation based feature selection. *Int. J. Inform. Technol. Knowl. Manag.*, **2**(2):271-277.

Khoshgoftaar, T.M., Gao, K., Napolitano, A., *et al.*, 2014. A comparative study of iterative and non-iterative feature selection techniques for software defect prediction. *Inform. Syst. Front.*, **16**(5):801-822. https://doi.org/10.1007/s10796-013-9430-0

Kira, K., Rendell, L.A., 1992. A practical approach to feature selection. Int. Workshop on Machine Learning, p.249-256.

Kononenko, I., 1994. Estimating attributes: analysis and extensions of RELIEF. European Conf. on Machine Learning, p.171-182. https://doi.org/10.1007/3-540-57868-4_57

Laradji, I.H., Alshayeb, M., Ghouti, L., 2015. Software defect prediction using ensemble learning on selected features. *Inform. Softw. Technol.*, **58**:388-402. https://doi.org/10.1016/j.infsof.2014.07.005

Liu, H., Yu, L., 2005. Toward integrating feature selection algorithms for classification and clustering. *IEEE Trans. Knowl. Data Eng.*, **17**(4):491-502. https://doi.org/10.1109/TKDE.2005.66

Liu, H., Sun, J., Liu, L., *et al.*, 2009. Feature selection with dynamic mutual information. *Patt. Recogn.*, **42**(7):1330-1339. https://doi.org/10.1016/j.patcog.2008.10.028

Liu, H., Motoda, H., Setiono, R., *et al.*, 2010. Feature selection: an ever evolving frontier in data mining. Int. Workshop on Feature Selection in Data Mining, p.4-13.

Liu, S., Chen, X., Liu, W., *et al.*, 2014. FECAR: a feature selection framework for software defect prediction. Annual Computer Software and Applications Conf., p.426-435. https://doi.org/10.1109/COMPSAC.2014.66

McCabe, T.J., 1976. A complexity measure. *IEEE Trans. Softw. Eng.*, **SE-2**(4):308-320. https://doi.org/10.1109/TSE.1976.233837

Miao, L., Liu, M., Zhang, D., 2012. Cost-sensitive feature selection with application in software defect prediction. Int. Conf. on Pattern Recognition, p.967-970.

Nam, J., Kim, S., 2015a. CLAMI: defect prediction on unlabeled datasets. Int. Conf. on Automated Software Engineering, p.452-463. https://doi.org/10.1109/ASE.2015.56

Nam, J., Kim, S., 2015b. Heterogeneous defect prediction. Joint Meeting on Foundations of Software Engineering, p.508-519. https://doi.org/10.1145/2786805.2786814

Shepperd, M., Song, Q., Sun, Z., *et al.*, 2013. Data quality: some comments on the NASA software defect datasets. *IEEE Trans. Softw. Eng.*, **39**(9):1208-1215. https://doi.org/10.1109/TSE.2013.11

Tantithamthavorn, C., McIntosh, S., Hassan, A.E., *et al.*, 2016. Automated parameter optimization of classification techniques for defect prediction models. Int. Conf. on Software Engineering, p.321-332. https://doi.org/10.1145/2884781.2884857

Uysal, A.K., Gunal, S., 2012. A novel probabilistic feature selection method for text classification. *Knowl. Based Syst.*, **36**:226-235. https://doi.org/10.1016/j.knosys.2012.06.005

Wang, H., Khoshgoftaar, T.M., Seliya, N., 2015. On the stability of feature selection methods in software quality prediction: an empirical investigation. *Int. J. Softw. Eng. Know. Eng.*, **25**:1467-1490. https://doi.org/10.1142/S0218194015400288

Wang, Z., Li, M., Li, J., 2015. A multi-objective evolutionary algorithm for feature selection based on mutual information with a new redundancy measure. *Inform. Sci.*, **307**:73-88. https://doi.org/10.1016/j.ins.2015.02.031

Wilcoxon, F., 1945. Individual comparisons by ranking methods. *Biometr. Bull.*, **1**(6):80-83. https://doi.org/10.2307/3001968

Xu, J., Zhou, Y., Chen, L., *et al.*, 2012. An unsupervised feature selection approach based on mutual information. *J. Comput. Res. Dev.*, **49**(2):372-382 (in Chinese).

Xue, B., Zhang, M., Browne, W.N., 2013. Particle swarm optimization for feature selection in classification: a multi-objective approach. *IEEE Trans. Cybern.*, **43**(6):1656-1671. https://doi.org/10.1109/TSMCB.2012.2227469

Yang, S., Gu, J., 2004. Feature selection based on mutual information and redundancy-synergy coefficient. *J. Zhejiang Univ.-Sci.*, **5**(11):1382-1391. https://doi.org/10.1631/jzus.2004.1382