Isolation Forest Wrapper Approach for Feature Selection in Software Defect Prediction

Zhiguo Ding

College of Mathematics and Computer Science, Zhejiang Normal University, Jinhua, Zhejiang, 321004, P.R. China

E-mail: dingzhiguo@zjnu.cn

Abstract. Software defect prediction is one of the hot research topics in the software engineering application. The performance of predictor largely depends on the quality of dataset used for learning the predictor. High dimensionality is a noteworthy characteristic of software defect dataset, which has some side-effect on the predictor building using data mining or machine learning algorithm. Feature selection, being an effective measure of dimensionality reduction, uses the optimal feature subset to represent the entire feature space and alleviate the dimensionality curse problem. In this paper, a wrapper feature selection approach applying genetic algorithm as a search strategy to find the optimal feature subset is firstly introduced. Secondly, an improved isolation forest based defect prediction method is proposed. The exploring experiments on 5 real NASA software defect datasets demonstrate the proposed method can improve the defect prediction performance to some extent and proves the positive effect of feature selection in SDP application.

1. Introduction

Software defect prediction (SDP) is one of hot research topics in the advanced software engineering research community, which usually builds a predictor using the historical defect database and applies it to judge whether a new software module is defect-prone or not [1, 2, 3]. The prediction results can optimize the testing plan and allocate the limited testing resources intentionally to those critical modules for enhancing software quality [4, 5].

There are many SDP methods emerged in literature [6]. A noticeable fact is that software defect could be introduced in software development different stage and caused by kinds of reasons. Consequently, there are massive metrics (such as Halstead, McCabe, CK, MOOD and others) to depict the software module characteristic, it is well known that applying these methods on a dataset with high dimensionality is time consuming and has some side-effect on prediction performance, and redundant or irrelevant features usually are unnecessary for obtaining predictor with high performance [7].

Feature selection (FS) has been regarded as an effective way to alleviate the dimension curse and has been proved to improve the prediction performance in many different applications [8]. In this paper, a novel feature selection method based on the genetic algorithm (GA) [9] is firstly proposed. Further, isolation forest (IForest) [10] based defect predictor is built based on the optimal feature subset, which not only alleviates the random characteristic caused by isolation tree construction but also improves the predictor performance.
The rest of this paper is organized as follows. Section 2 introduces the related work of our research. Section 3 describes the detail framework of proposed method. Section 4 presents the experiment setup and results analysis. Finally, section 5 concludes the paper and presents the future work.

2. Related Work
In this section, two related works of this paper are described below.

2.1. SDP and IForest
SDP is a key activity in the modern software development to allocate the limited testing resources reasonably and improve the software quality. Because the quantity of defective modules is usually significantly less than the non-defective modules in software system, then these defective modules can be regarded as anomalies. To detect these minorities as early as possible, anomaly detection method is consequently employed and used in this application. Many machine learning or data mining method has been used for SDP, such as Naïve Bayesian network, associating rules mining, artificial neural network, support vector machine, semi-supervised learning, multiple kernel learning, genetic programing, and so on. [11]

IForest, being an anomaly detection method, has been applied in the SDP fields [12]. Compared to above methods, the most advantage is that it has low computational complexity and high performance, which does not rely on any distance or density measure, and it has no need to obtain the dataset distribution. But the most disadvantage of IForest is selecting feature and its feature values are all random during the procedure of predictor building. In SDP applications, there are too many features available, selecting the feature random means the high relevant and low relevant features are chosen with the same low probability, which is not only hard to obtain the optimal predictor but also consumes much more computational resource. It is necessary to select the critical features to optimize predictor when applying the IForest. In this paper, an improved IForest based on FS is proposed in section 3.

2.2. FS in SDP
Dimensionality curse is one of the biggest hinder in the community of machine learning and data mining [13]. FS, being an effective decreasing dimensionality method, aims to identify and select these relevant and informative features and remove these irrelevant and redundant ones from the original dataset, so that only useful features are left for predictor building. It uses the feature subset to re-construct dataset for obtaining the similar or better performance and decreasing the requirement of computational resources. FS methods existed can roughly be divided into three groups, i.e., filter-based, wrapper-based and embedded-based method. Although each of them present strengthen and weakness, the wrapper approach for feature selection outperforms the other approaches even if it is more time consuming and requires more computational resources[8].

In SDP applications, the dataset used to training the predictor are usually characterized by high dimensionality and some metrics are irrelevant or redundant to the module label (defective or non-defective), it is necessary to proceed with a dimensionality reduction measure for obtaining optimal features subset used for constructing predictor. Many related works in SDP had applied a FS to select the optimal feature subset instead of using the full features. For example, considering the distribution gap between software features extracted from the source and the target projects too large to make the mixed data useful for training, a cross-project SDP method using cluster-based FS method is proposed [14]. Due to each FS method behaves differently, such as different FS strategy, empirical studies on FS methods for SDP produce contradictory and inconsistent quality outcomes, comprehensive experiments and results analysis show that application of FS can improve the predictive performances of predictor and the FS method can vary across datasets and predictors [8]. Besides, to select the optimal feature automatically, a two-stage SDP method is proposed, the first stage is using deep learning to obtain the feature subset adaptively [15]. In this paper, GA and
IForest are adopted as the FS method and feature evaluation method to do some exploring work for SDP.

3. IForest Wrapper Approach for FS in SDP

The existed SDP methods in literature have presented good performance and some of them have been adopted in the practical applications. Nevertheless, research of obtaining the optimal predictor to continuously improve the software quality is never stopped. In this paper, we focus on the high dimensionality of dataset and propose a new predictor using FS to obtain the optimal feature subset. Figure 1 demonstrates an overview of the whole architecture of the proposed method, its implementation integrates three procedures, i.e., dataset preprocessing, IForest wrapper approach for FS and SDP activity.

![Diagram](image)

**Figure 1.** Overview of Whole Architecture of IForest Wrapper Approach for FS in SDP

3.1. Dataset Preprocessing

The first activity of SDP is data collection and preprocessing, which is one of key procedures. It includes of collecting the history software repositories, separating modules from original software, such as file, object, function, and so on, which is guided by the specified granularity, features/metrics extraction from the modules, labeling samples extracted defective or not and building SDP dataset for training defect predictor.

The dataset used in the context of SDP usually consists of different software metrics, which would have some nosing data (such as feature value missing, wrong feature value and so on) and with varying resolution and ranges. Most of data mining and machine learning algorithm can’t compatible with such dataset and can’t be used directly to train the predictor. Therefore, it is necessary to proceed with these dataset ahead of time and let them fit for the selected algorithm, the detail data processing strategies and procedure is presented in section 4.1. Besides, the SDP dataset obtained in this stage is further divided into two sub-datasets, i.e., training dataset and validating dataset, which are used in FS stage and SDP stage, respectively.

3.2. IForest Wrapper Approach for FS

The right part of Figure 1 demonstrates the complete procedure of proposed FS strategy based on GA searching and IForest evaluating. The main object of this stage is to find the subset of features having the higher prediction performance and using the less number of features.
Comparison to filter-based and embedded-based feature selected methods, although the wrapper-based would consume more computational resources, it usually outperforms better than two of them. As number of features and features selected are all important factors effecting on the prediction performance and efficiency, heuristics can be adopted to deal with this problem. GA is adopted in this paper as a FS strategy. Further, IForest is adopted as the fitness function to evaluate the selected subsets. This procedure consists of applying GA to select the candidate features and using IForest to judge whether the current feature subset is much more relevant.

The initial population with M chromosomes is generated randomly regardless of any constraints, each chromosome presents a possible solution and the length of chromosome is same to the length of feature vector. Each value “1” or “0” in chromosome represents the corresponding feature selected or not. each feature subset is used to the whole training data set to train the predictor, and those have the higher fitness values predictor are kept, and corresponding feature subset is regarded as the candidate ones. Crossover and mutation operation are conducted following to create new population. This procedure is repeated until a predefined number of generation iteration is reached or all chromosomes become identical.

Fitness evaluating is a key procedure to decide whether the current chromosome is kept or not, or being a candidate to create the next generation individual of population. Given a chromosome $S_i$, its fitness value can be computed by formula (1).

$$
Fitness(S_i) = \alpha * F_{measure}(S_i) + (1 - \alpha) / \sum_{j=1}^{n} S_{i,j}
$$

Because the prediction performance and computation efficiency are both important in the software testing application, the design of fitness function needs to take these into consideration. For prediction, F-measure (defined in section 4.2), is a part of fitness function, the other part of fitness function aims to reflect feature quantity. More features would induce more computational resource and slow down the application, therefore, the higher the F-measure and the smaller the feature quantity are, the higher fitness value and better feature subset are. In formula (1), $S_i$ is a binary vector and denotes a chromosome, $\alpha$ ($\alpha \in [0,1]$) is a predefined weight, which is used to indicate tradeoff between the prediction performance and the quantity of selected feature ($\sum_{j=1}^{n} S_{i,j} , S_{i,j} \in \{0,1\}$ ). The pseudo-code implementing the IForest wrapper approach for FS is given by Algorithm GA_IForest_Wrapper_FS ()

**Algorithm: GA_IForest_Wrapper_FS ()**

- **Input**: Training dataset $X_T$, Testing dataset $X_V$.
- **Output**: Optimal Feature Subset
- **1** $P$=Generate_Initila_Population ($M$) // $P$: initial population; $M$: the size of initial population
- **2** Computing_Fitness($P$) //Computing the fitness of each chromosome in $P$.
- **3** while (! All_Cromosomes_Idential && iteration $\leq$ Maximum_Generation_Iteration)
- **4** $P_{new}$=Null; // $P_{new}$ : the next generation population
- **5** Selection ($P$, $p_c$) //Selection operation, $p_c$: selection probability
- **6** $P_{new}+=Crossover( S_i, S_j)$ // $S_i, S_j$: two selected chromosomes being crossover operation
- **7** $P_{new}+=Mutation( S_i, p_m)$ // $p_m$: mutation probability.
- **8** Computing_Fitness($P_{new}$) //Computing the fitness of new created chromosomes.
- **9** Next_M_Generation ($P$, $P_{new}$) //creating new population with the M highest fitness value.
- **10** End while
- **11** End Algorithm;

The Next_M_Generation ($P$, $P_{new}$) function in Algorithm GA_IForest_Wrapper_FS is used for choosing the next generation individuals. For the $P$ and $P_{new}$, sorting all chromosomes based on
their fitness value, these ones with M highest fitness values are regarded as offspring individuals and will attend the next evolution until the stop criterion is met.

3.3. Predictor Building

IForest continues to be used in defect prediction stage (the lower left part of Figure 1.) to build defect predictor. Based on FS results, the new dataset are created, which is only consisted of the selected key features. Following its initial assumption, the isolation feature and feature value are selected randomly during the procedure of predictor building. Further, multiple isolation trees are built to construct an isolation forest[10].

4. Experimental Setup and Result Analysis

In this section, the experimental study is conducted, including the summaries of the datasets, performance evaluation metrics, experiment result analysis and the comparison with some baseline algorithms. All experiments were conducted on a PC running the Windows 10 OS with Intel i7 processor@3.4G Hz and 8GB memory. The algorithm mentioned in Section 3 is implemented by Microsoft Visual C++ platform. Some baseline experiments (in section 4.2) are conducted on the Waikato environment for knowledge analysis software with default parameter setting[16].

4.1. Dataset and Data Preprocessing

Five NASA datasets obtained from the Promise Repository of Empirical Software Engineering Data[17] are used in this work. Each dataset consists of many software modules with quality metrics as input and each sample has a label as output to denote whether it is defective or not. The dataset used in this paper are all numerical with varying resolution and value domain. The value of label feature is Y or N, denotes defective or non-defective. It is necessary to eliminate noisy data or alleviate the side-effect caused by different data units. For those numerical features with a large integer value ranges, a logarithmic with base 10 is applied to scale and reduce the value ranges. Further, linear scaling method is adopted, which is a min-max normalization that consists of finding the minimum and maximum value of the \( i \)-th feature and then transform each feature value \( x_{i,j} \) is linearly scaled to \( x'_{i,j} \) in the range of \([0,1]\) using the formula (2). For Boolean type feature (label), it do not need to any modification, 1 for defective and 0 for non-defective.

\[
x'_{i,j} = \frac{x_{i,j} - \min(x_{i,j})}{\max(x_{i,j}) - \min(x_{i,j})}
\]

(2)

The summary of these 5 datasets preprocessed are presented in Table 1.

| Name | Modules(#) | Features(#) | Defectives(#) | Non-Defectives(#) |
|------|------------|-------------|---------------|-------------------|
| KC1  | 1203       | 21          | 314           | 889               |
| KC2  | 520        | 21          | 105           | 415               |
| CM1  | 327        | 37          | 42            | 285               |
| PC1  | 705        | 37          | 61            | 644               |
| JM1  | 7782       | 21          | 1672          | 6110              |

4.2. Performance Evaluation Metrics and Experiment Analysis

Many performance evaluation criteria are available in SDP, such as accuracy, precision, recall rate, F-measure and AUC. In this paper, to evaluate the performance of the proposed method, a standard metrics, F-measure, which is used widely in SDP application for evaluating predictor performance, is adopted. In this paper, it is used not only being as the part of GA fitness function but also the predictor performance evaluation metrics[18].
In formula (3), \( \#TP \) denotes the number of the defective module predicted correctly, i.e., the defective module is predicted as the defective by predictor. \( \#FP \) denotes the number of non-defective module predicted wrongly, i.e., the non-defective module is predicted as the defective by predictor. \( \#FN \) denotes the number of the defective module predicted wrongly, i.e., the defective module is predicted as the non-defective by predictor. All these can be defined by using the confusion matrix.

\[ F - \text{measure} = \frac{(1 + \beta^2) \cdot \text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}} = \frac{(1 + \beta^2) \cdot \#TP}{(1 + \beta^2) \cdot \#TP + \beta^2 \cdot \#FN + \#FP} \quad (3) \]

\( \beta \) is a coefficient to adjust the relative importance of between precision and recall. The value of \( \beta \) is near to the relative small value of them. Consequently, the relative big value of F-measure means that precision and recall are both with higher value. Usually, \( \beta \) is set as 0.5, 1 or 2. In SDP application, because prediction precision and recall rate are almost same important, in this paper, \( \beta = 1 \) is set.

1. Exploring Experiment for FS

Exploring experiments about FS are done firstly to research whether the optimal feature subset can improve the prediction performance. For proposed method, Weka presents a weakness which is the process time because the wrapper methods for FS. The other reason is that the fitness function of GA used in this paper is IForest, which is not embedded in Weka platform. Then, GA implementation, FS and IForest are all implemented by C++ programing language for reducing the computational time.

For the parameters in GA, population size, crossover rate and mutation rate are set as 10, 0.8 and 0.02, respectively. The maximum generation is 100. In SDP application, the prediction performance is more important than the computation resources requirement, the coefficient \( \alpha \) is set as 0.9 in the fitness function. For the IForest, its scale (ensemble size) is set as 150 based on our previous work[12]. To guarantee the reliability of the experiment results, 10-fold cross validation technique is adopted and each experiment repeated 10 times. The first experiment is to find the optimal feature subset. Then for each dataset eight tenth dataset are used to conduct the wrapper-based FS and one tenth dataset is used to compute the fitness functions and another one tenth dataset is used to performance evaluation. Table 2 presents some optimal feature numbers for each dataset when the corresponding fitness value reaches almost the same.

### Table 2 Results of IForest Wrapper for Feature Selectin

| Dataset | Feature (#) | FSubset (#) |
|---------|-------------|-------------|
| KC1     | 21          | \{12, 15\}  |
| KC2     | 21          | \{11, 13\}  |
| CM1     | 37          | \{22, 25, 27\} |
| PC1     | 37          | \{21, 26\}  |
| JM1     | 21          | \{12, 13, 15\} |

In Table 2, Feature (#) denotes the original feature number of defect dataset, such as there are 21 features in KC1, KC2 and JM1 and 37 features in CM1 and PC1 datasets. FSubset (#) denotes the feature number of feature subset selected by GA, the element of it denotes the number of chromosomes when the highest fitness value is computed. For KC1, 12 and 15 denotes different chromosome, which has the almost the same and the highest fitness value when the GA evolution is finished. From the result in Table 2, an obvious fact is that some key features surely existed and the predictor built by less features, which have more relevant to the classification. Because the predictor performance is more important than the consuming of computation resource in SDP, the bigger feature subsets are selected for each datasets in the later experiments. Table 3 presents the average
F1 value of 100 times independent experiments when the relative big feature subsets (bold number in Table 2) with higher prediction performance are adopted. From the results in Table 3, an obvious fact is the all predictor performance with feature selection is better than ones of without feature selection, which means the feature selection is an useful data processing in SDP application.

| Dataset | Feature (#) | F1   | FSubset (#) | F1   |
|---------|-------------|------|-------------|------|
| KC1     | 21          | 0.7524 | 15          | 0.7608 |
| KC2     | 21          | 0.7248 | 13          | 0.7413 |
| CM1     | 37          | 0.7162 | 27          | 0.7321 |
| PC1     | 37          | 0.6943 | 26          | 0.6994 |
| JM1     | 21          | 0.7709 | 15          | 0.7791 |

(2) Performance Comparison Experiment with Baseline Algorithms

To compare the proposed method with the existed methods, Weka (Waikato environment for knowledge analysis) software, an open and intelligent experiment platform in data mining and machine learning community, is adopted. Here, predictors embedded in Weka, such as Bagging, Random-Forest ensemble methods, are served as the baseline to evaluate the proposed method. For the experimental setting, the default parameters of Bagging, Random-Forest, C4.5 algorithm (named as J48 in Weka) are set and the ensemble size is set as 150 same to the parameter setting of IForest [12]. The results of performance comparison on three algorithms are depicted by Figure 2.

![Figure 2. Performance Comparison of F1 of Three Algorithms on Five Software Defect Datasets](image)

Figure 2 uses the bar plots to show the comparison of three different SDP algorithms in term of average F1 values on five real NASA datasets. Obviously, among all three methods, the FS based IForest outperforms other two methods under the given parameter setting, which shows that proposed method is effective.

5. Conclusion

Due to redundant and irrelevant features causing some side-effect on the predictor building, in this paper, a FS method based on GA is proposed to explore the optimal feature subset for training the IForest based predictor in SDP application. There are two advantages of proposed method, one is the true relevant features are chosen based on the GA and the dataset dimensionality is reduced, the other is building the better predictor using the optimal feature subset and reducing the random characteristics of IForest. Experiment demonstrates that our proposed predictor outperforms the existing baseline SDP methods. In the future, more datasets will be explored and more experiments will be done on different parameter setting to further validate proposed method effectiveness.

Acknowledgments
This work is supported by the Doctoral Research Start-up Fund of Zhejiang Normal University (ZC304016020), Project of Department of Education of Zhejiang Province (KYZ04Y19267). Besides, the authors would like to thank the editors and anonymous reviewers for their constructive comments and suggestions for the improvement of this paper.

Reference
[1] T. HALL, S. BEECHAM, D. BOWES, D. GRAY and S. COUNSELL 2012 A Systematic Literature Review on Fault Prediction Performance in Software Engineering, IEEE Transactions on Software Engineering 38 1276-304
[2] R. S. WAHONO 2015 A Systematic Literature Review of Software Defect Prediction: Research Trends, Datasets, Methods and Frameworks Journal of Software Engineering 1
[3] J. REN and F. LIU 2020 A Novel Approach for Software Defect prediction Based on the Power Law Function Applied Sciences-Basel 10
[4] N. LI, M. SHEPPERD and Y. GUO 2020 A systematic review of unsupervised learning techniques for software defect prediction Information and Software Technology 122
[5] L. GONG, S. JIANG, L. BO, L. JIANG and J. QIAN 2020 A Novel Class-Imbalance Learning Approach for Both Within-Project and Cross-Project Defect Prediction IEEE Transactions on Reliability 69 40-54
[6] M. SHEPPERD, D. BOWES and T. HALL 2014 Researcher Bias: The Use of Machine Learning in Software Defect Prediction IEEE Transactions on Software Engineering 40 603-16
[7] L. MADEYSKI and M. JURECZKO 2015 Which process metrics can significantly improve defect prediction models? An empirical study Software Quality Journal 23 393-422
[8] A. O. BALOGUN, S. BASRI, S. J. ABDULKADIR and A. S. HASHIM 2019 Performance Analysis of Feature Selection Methods in Software Defect Prediction: A Search Method Approach Applied Sciences-Basel 9 20
[9] Z. XU, J. LIU, X. P. LUO, Z. J. YANG, Y. F. ZHANG, P. P. YUAN, Y. T. TANG and T. ZHANG 2019 Software defect prediction based on kernel PCA and weighted extreme learning machine Information and Software Technology 106 182-200
[10] C. R. HOUCK, J. A. JOINES and M. G. KAY 1995 A genetic algorithm for function optimization: a Matlab implementation NCSU-IE TR 95
[11] F. T. LIU, K. M. TING and Z. H. ZHOU 2012 Isolation-Based Anomaly Detection ACM Transactions on Knowledge Discovery from Data 6 1-39
[12] Y. SHAO, B. LIU, S. WANG and G. LI 2020 Software defect prediction based on correlation weighted class association rule mining Knowledge-Based Systems 196
[13] X. CHEN, D. ZHANG, Y. Q. ZHAO, Z. Q. CUI and C. NI 2019 Software defect number prediction: Unsupervised vs supervised methods Information and Software Technology 106 161-81
[14] F. WU, X. Y. JING, Y. SUN, J. SUN, L. HUANG, F. Y. CUI and Y. F. SUN 2018 Cross Project and Within-Project Semisupervised Software Defect Prediction: A Unified Approach IEEE Transactions on Reliability 67 581-97
[15] Y. HE, X. ZHU, G. WANG, H. SUN and Y. WANG 2017 Predicting Bugs in Software Code Changes Using Isolation Forest, 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS), Prague, Czech Republic 296-305
[16] Z. DING, Y. MO and Z. PAN 2019 A Novel Software Defect Prediction Method Based on Isolation Forest, 2019 International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering, QR2MSE 2019, August 6, 2019 - August 9, 2019, Institute of Electrical and Electronics Engineers Inc., Zhangjiajie, Hunan, China 882-7
[17] C. KHAMMASSI and S. KRICHEM 2017 A GA-LR wrapper approach for feature selection in network intrusion detection Computers & Security 70 255-77
[18] C. NI, W. S. LIU, X. CHEN, Q. GU, D. X. CHEN and Q. G. HUANG 2017 A Cluster Based Feature Selection Method for Cross-Project Software Defect Prediction *Journal of Computer Science and Technology* **32** 1090-107

[19] M. LIU, L. MIAO and D. ZHANG 2014 Two-Stage Cost-Sensitive Learning for Software Defect Prediction *IEEE Transactions on Reliability* **63** 676-86

[20] *Weka*: [http://www.cs.waikato.ac.nz/ml/weka/](http://www.cs.waikato.ac.nz/ml/weka/).

[21] T. J. M. J. SAYYAD SHIRABAD 2005 *The PROMISE Software Engineering Repository* School of Information Technology and Engineering, University of Ottawa, Canada, 2005 Available from: [http://promise.site.uottawa.ca/SERepository](http://promise.site.uottawa.ca/SERepository)

[22] Z. G. DING, M. R. FEI, D. J. DU and F. YANG 2017 Streaming data anomaly detection method based on hyper-grid structure and online ensemble learning *Soft Computing* **21** 5905-17