Cross-Project Software Defect Prediction Based on Feature Selection and Transfer Learning

Tianwei Lei[1](✉), Jingfeng Xue[1], and Weijie Han[1,2]

[1] School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China
absherry123@163.com

[2] School of Space Information, Space Engineering University, Beijing 101416, China

Abstract. Cross-project software defect prediction solves the problem that traditional defect prediction can't get enough data, but how to apply the model learned from the data of different mechanisms to the target data set is a new problem. At the same time, there is the problem that information redundancy in the training process leads to low accuracy. Based on the difference of projects, this paper uses MIC to filter features to solve the problem of information redundancy. At the same time, combined with the TrAdaboost algorithm, which is based on the idea of aggravating multiple classification error samples, this paper proposes a cross-project software prediction method based on feature selection and migration learning. Experimental results show that the algorithm proposed in this paper has better experimental results on AUC and F1.

Keywords: Transfer learning · TrAdaboost · MIC · Cross-project software defect prediction

1 Introduction

With the rapid development of information technology in recent decades, the scale of software is becoming larger and larger, and the software vulnerabilities are becoming hidden, which makes the assurance of the software's quality more difficult to achieve [1].

Software defect prediction can discover the defects in software before the application is put into production, to reduce the cost of subsequent manual testing and the development cycle [2]. It is difficult for many organizations to obtain enough historical versions to build data sets for defect prediction in practical, and it will lead the problem of cold start is encountered in the initial prediction. A possible solution to this problem is to use the data of other projects to build a model to predict the software defects of the target project, that is, cross-project software defect prediction [3]. But there are also some problems will also lead the low training accuracy in the cross-project software defect prediction, such as information redundancy, class imbalance [4].

This paper propose a cross-project software defect prediction method based on feature selection and transfer learning. The main contributions of this paper are as follows:

1. This paper proposes a method of cross-project software defect prediction based on the combination of feature selection and transfer learning and the experiments show that the defect prediction is effective.
2. MICs method is used to solve the problems of data redundancy and feature dimension explosion;
3. After feature selection, the MuTrAdaboost algorithm is proposed to increase the TrAdaboost algorithm. MuTrAdaboost enhances the weight of instances which is like the target project data, and the final model is formed by multiple training.

The experimental results show that the MuTrAdaboost algorithm is better than the TrAdaboost algorithm in AUC and F1.

2 Related Works

Cross-project software defect prediction [5], as mentioned above, is mainly to build a defect prediction model for the target project by using the data sets already collected by other projects, so as to predict and analyze the data of the target project. And it is mainly based on traditional software defect prediction technology, aiming at finding similarities between different factors and using potential links to propose a reasonable and efficient prediction model.

In recent years, the transfer learning method in the field of machine learning provides a good idea for the prediction of cross project [6]. Transfer learning [7] is a machine learning method which uses the existing knowledge to solve the problems in the related fields.

Project variation is the first problem to be solved in the direction of cross project defect prediction [8]. He [9] have carried out the corresponding research from the perspective of the disaster problem of the feature dimension of data sets. Amasaki [10] try to use unsupervised learning to remove the irrelevant features and instances from the target project to improve the accuracy.

In the field of transfer learning, Dai et al. [11] proposed the TrAdaboost method, which applies the idea of AdaBoost to transfer learning to improve the instance weight of target classification task and reduce the instance weight of target classification task. TraAdaboost method is one of the classical research of transfer learning, and there are also some research works in the field of software defect prediction based on this algorithm. Chen [12] uses data preprocessing and TrAdaboost to predict defects from the perspective of reducing the weight of negative instances in the source project. Shen [13] considers the way of multiple projects migrating one target project, proposes two improved prediction algorithms based on TrAdaboost, and constructs the final prediction model by inheritance learning.

3 Framework of the Approach

The approach of this paper is illustrated as Fig. 1.

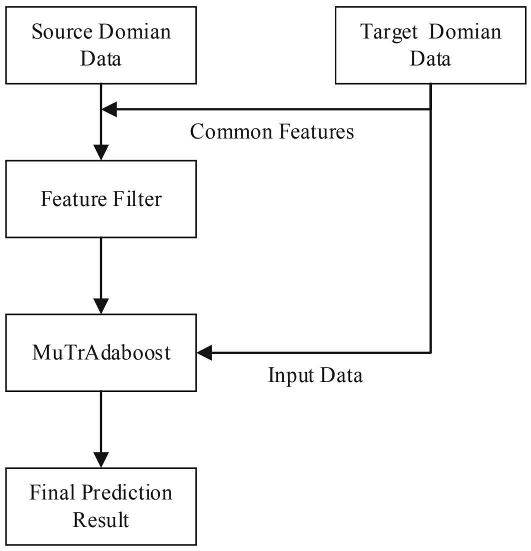

Fig. 1. The framework of the proposed approach

1. Firstly, the data from different project are divided into two domains, source domains and target domain. Our goal is to use the data in source domain to build prediction model to predict the fault in source data. The common features in both source data domain and target data domain are reserved for the next step.
2. The method MIC is used to filter the features. We do the MIC calculation between features of target dataset and source dataset and choose the features which have higher MIC results.
3. We build the prediction model with MuTradaboost which is an improved TrAdaboost algorithm. After the building process, the part of target data is inputted into model as test data to get final prediction result.

4 Key Technologies

4.1 Feature Selection Based on Maximum-Information-Coefficient MIC

Before the formal establishment of the model, it is necessary to filter the features in the training project, exclude the feature vectors that are too different from the features of the target project, and select the feature vectors that are highly correlated with the features in the training project as the basis for model construction.

In our experiment, we use the maximum information coefficient (MIC) to calculate the correlation between two vectors. It is a method proposed by David et al. [14] to express the dependence between two groups of variables.

The MIC calculation is done between the same features in source dataset and target dataset, and the features is sorted by their MIC result by descending order. We select the required number of features to build the model in training.

4.2 MuTrAdaboost, Cross-Project Defect Prediction Based on Improved TrAdaboost

After filtering the appropriate features of the source project, we proposed a improved TrAdaboost algorithm MuTrAdaboost to build the data model and predict the defects. In our method, we still adopt the basic idea of TrAdaboost: to strengthen the weight of the samples that are misclassified in a certain training, and hope that they can be correctly classified in the next iteration.

The core of our thinking is: while strengthening the weight of the samples that have been wrongly divided, we give those samples that have been wrongly divided many times higher weight to quickly adjust the model and update the data. Therefore, the defect prediction algorithm based on the improved TrAdaboost is as follows:

```
***MuTrAdaboost Algorithm***
*Input*
Dataset from source data field $T_s = \{T_{d_1}, T_{d_2}, \ldots, T_{d_{N_S}}\}$
Sample target dataset $T_t$
The unlabeled data set $S$
Base learning algorithm Learner
Maximum number of iterations $N$
Error limits $\varepsilon$

***Algorithm***
1: Initialize the weight factor $\beta = 1$,err= $zeros(1 \ldots n + m)$ and the weight vector $w^t = (\omega_1^1, \ldots, \omega_{n+m}^1)$
   $$\omega_i^1 = \begin{cases} \frac{1}{n}, & i = 1 \ldots n \\ \frac{1}{m}, & i = n + 1 \ldots n + m \end{cases}$$
2: For each instance $t = 1 \ldots N$ do
3:    Normalize the weight vector $p^t = \frac{w^t}{\sum_{i=1}^{n+m} \omega_i^t}$ ;
4:    Use the $p^t$ as the weight data to train the base learning algorithm learner , return the hypothesis $h_t$
5:    Calculate the error rate $\varepsilon_t$ of $h_t$ on $T_t$,
      $\varepsilon_t = \sum_{i=n+1}^{m} \frac{\omega_i^t}{\sum_{j=1}^{n+m} \omega_j^t} |h_t(x_i) - c(x_i)|$
```

```
6:  Update the weight factor $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$, and the error rate $\varepsilon_t$ must be less than
0.5, otherwise set $\varepsilon_t = 0.5$
7:  For each instance j = 1…n+m do
8:     Calculate the err[j]:
$$err[j] = \begin{cases} err[j] + 1, & |h_t(x_i) - c(x_i)| \geq \varepsilon \\ err[j], & |h_t(x_i) - c(x_i)| < \varepsilon \end{cases}$$
9:  End for
10: Update the weight vector:
$$\omega_i^{t+1} = \begin{cases} \omega_i^t \beta_t^{|h_t(x_i)-c(x_i)|} & i = 1 \ldots n \& |h_t(x_i) - c(x_i)| < \varepsilon \\ \omega_i^t \beta_t^{|h_t(x_i)-c(x_i)| \cdot (-ln(err[i])+1)} & i = 1 \ldots n \& |h_t(x_i) - c(x_i)| \geq \varepsilon \\ \omega_i^t \beta_t^{-|h_t(x_i)-c(x_i)|} & i = n+1 \ldots n+m \& |h_t(x_i) - c(x_i)| < \varepsilon \\ \omega_i^t \beta_t^{-|h_t(x_i)-c(x_i)| \cdot errs[i]} & i = n+1 \ldots n+m \& |h_t(x_i) - c(x_i)| \geq \varepsilon \end{cases}$$
11: End for
```

Output

$$h_f(x) = \begin{cases} 1, & \prod_{t=\lceil N/2 \rceil}^{N} \beta_t^{-h_t(x)} \geq \prod_{t=\lceil N/2 \rceil}^{N} \beta_t^{-\varepsilon} \\ 0, & otherwise \end{cases}$$

5 Experiments

5.1 Dataset

To show the performance of the software defect prediction of cross-project, we select the open NASA dataset and SOFTLAB dataset as the experimental set and compare the experimental results. We use NASA dataset as the source dataset and SOFTLAB data set as the target dataset for prediction. These datasets are available on the PRIMISE website.

We selected 10 items of NASA dataset, 3 items of SOFTLAB dataset, and their common features as the initial data feature set. Table 1 shows their details.

5.2 Experimental Indicators

In this paper, AUC (area under ROC) and F-measure are used as evaluation criteria. AUC refers to the area surrounded by ROC curve and axis A good prediction model should have a high precision at the same time of high recall. However, the high recall is often achieved at the cost of low precision, so we introduce the indicator F-measure which is used to measure the harmonic average of the two.

Table 1. Source data and target data

Source data			
Project	Examples	%Defective	Description
CM1	327	12.84	Space craft instrument
KC3	194	9.38	Storage management
MC1	1988	2.31	Video guidance system
MC2	161	32.30	Video guidance system
MW1	403	7.69	A zero gravity experiment
PC1	1109	6.94	Flight software
PC2	745	2.15	Flight software
PC3	1077	12.44	Flight software
PC4	1287	13.75	Flight software
PC5	1711	27.53	Flight software
Target data			
ar3	63	12.70	Embedded controller
ar4	107	18.69	Embedded controller
ar5	36	22.22	Embedded controller

5.3 Experiment Comparison

In order to test the experimental effect of this paper, we designed three comparative experiment: separate TrAdaboost experiment; separate MuTrAdaboost experiment for the data; and use mic for feature selection first, and MuTrAdaboost experiment for the selected samples.

We select all NASA data sets as the data of the source data, and select one SOFTLAB data set for the target data at a time, and randomly select 10% of the target data as the test data for the experiment. Table 2 shows the comparison of experimental indicators of three algorithms on multi-source data sources. In TrAdaboost and MuTrAdaboost experiments, we used all the common features of each data set and the target data set in the source data domain. In feature filtering, we selected some features according to the order of MIC value. We can see that the use of MuTrAdaboost algorithm can significantly improve the AUC index of the experiment compared with TrAdaboost algorithm. In addition to dataset ar5, the F-measure index can also be improved to some extent. The third column of Table 2 shows that the index selected by adjusting the proportion of using features is the best after using the MIC algorithm for feature filtering and MuTrAdaboost algorithm. We can see that the use of MIC algorithm can improve the AUC index AUC and F-measure after the method have been improved to some extent. The defect rate of ar3 project is only 12.3%, which leads to the poor performance of the three algorithms.

Table 2. Comparison of multi-source data experiments

Metric	TraAdaboost	MuTraAdaboost	Mic + MuTraAdaboost
ar3			
AUC	0.571	0.667	0.733
F-measure	0.275	0.333	0.375
ar4			
AUC	0.545	0.745	0.812
F-measure	0.325	0.364	0.453
ar5			
AUC	0.577	0.793	0.833
F-measure	0.375	0.372	0.449

Table 3 shows the data of AUC and F1 indexes with different proportion of characteristic number filtered by MIC value in each data set. It can be seen that ar3, ar4 and ar5 data sets have the best data results when selecting 70%, 80% and 70% of the number of features, respectively. Figures 2, 3 and 4 show the change of AUC and F1 indexes of target data sets ar3, ar4 and ar5 with the increase of feature proportion of filtering, where abscissa is the proportion of feature quantity selected after feature filtering to total feature quantity, while ordinate is the value range of AUC and F1.

From the experimental comparison of multi-source data, it can be seen that the introduction of MuTrAdaBoost can improve the accuracy of the experiment, while the use of MIC to filter features rather than select all features for the experiment can also improve the effect of the experiment to a certain extent, and at the same time, the speed

Table 3. Experimental results of feature selection scale of target domain dataset

	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
ar3										
AUC	0.21	0.28	0.34	0.40	0.50	0.53	0.57	0.56	0.57	0.53
F1	0.07	0.10	0.15	0.18	0.24	0.26	0.28	0.27	0.27	0.27
ar4										
AUC	0.21	0.36	0.44	0.54	0.63	0.69	0.73	0.75	0.74	0.74
F1	0.06	0.10	0.18	0.25	0.28	0.31	0.34	0.36	0.36	0.36
ar5										
AUC	0.27	0.30	0.42	0.52	0.66	0.77	0.83	0.82	0.83	0.81
F1	0.05	0.13	0.20	0.27	0.34	0.38	0.41	0.41	0.41	0.41

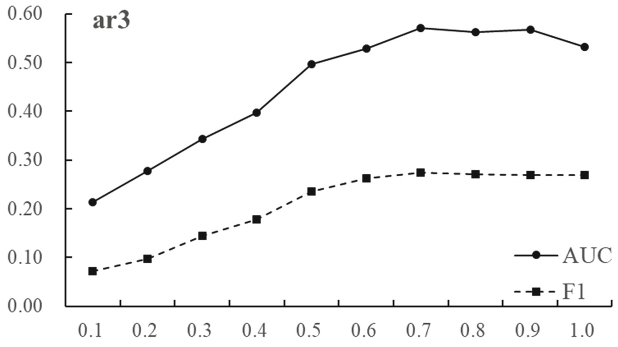

Fig. 2. AUC and F1 change chart of ar3 value with feature proportion

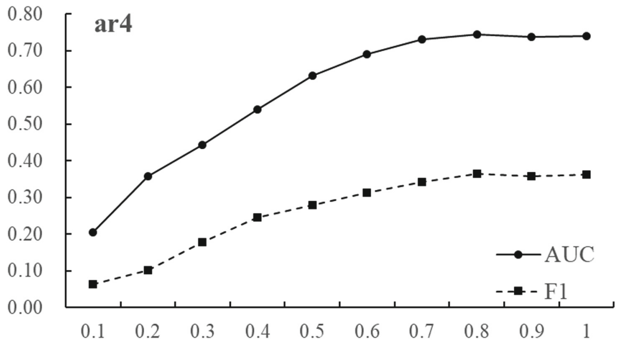

Fig. 3. AUC and F1 change chart of ar4 value with feature proportion

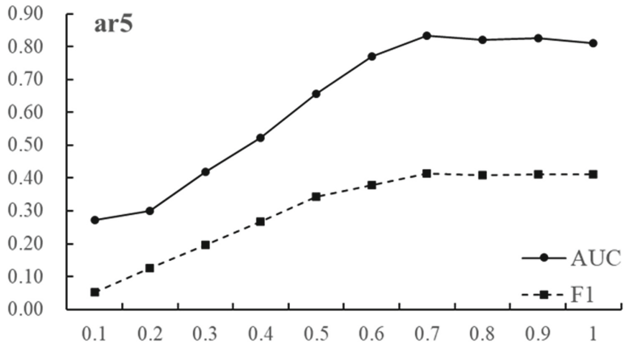

Fig. 4. AUC and F1 change chart of ar5 value with feature proportion

of the experiment has been improved due to the reduction of redundant features involved in the calculation.

In summary, from AUC and F1 experimental indicators, the experimental effect of MuTrAdaboost using all features is better than that of traditional TrAdaboost algorithm using all features, which shows that our idea of multiple weighting of wrong samples is more effective than that of TrAdaboost algorithm, which only weights the samples that are wrongly divided each time. The MuTrAdaboost algorithm filtered by MIC features is better than that using all features. This shows that the idea of feature selection is correct, and the use of feature vector MIC value as the basis of feature selection is also effective.

6 Conclusion

In this paper, we propose a cross-project software defect prediction method based on the combination of feature selection and transfer learning. And the experiments show that the method we supposed is better than that of traditional TrAdaboost algorithm.

The next research direction is to explore the experimental effect of the algorithm on larger datasets, and to carry out more comparative experiments for more indicators to explore the improvement direction of the updated algorithm.

Acknowledgement. This work was supported by the National Key Research and Development Program of China under Grant 2016QY06X1205.

References

1. Pizzi, N.J.: A fuzzy classifier approach to estimating software quality. Inf. Sci. **241**, 1–11 (2013)
2. Nam, J., et al.: Heterogeneous defect prediction. IEEE Trans. Softw. Eng. **44**, 874–896 (2017)
3. Xia, X., et al.: HYDRA: massively compositional model for cross-project defect prediction. IEEE Trans. Softw. Eng. **42**, 977–998 (2016)
4. He, Z., et al.: An investigation on the feasibility of cross-project defect prediction. Autom. Softw. Eng. **19**(2), 167–199 (2012)
5. Hall, T., et al.: A systematic literature review on fault prediction performance in software engineering. IEEE Trans. Softw. Eng. **38**, 1276–1304 (2012)
6. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. Knowl. Data Eng. **22**(10), 1345–1359 (2010)
7. Zhuang, F., et al.: Survey on transfer learning. J. Softw. **26**(1), 26–39 (2015). (in Chinese)
8. Chen, X., et al.: A survey on cross-project software defect prediction methods. Chin. J. Comput. **041**(001), 254–274 (2018). (in Chinese)
9. He, P., et al.: An empirical study on software defect prediction with a simplified metric set. Info. Softw. Technol. **59**(mar), 170–190 (2015)
10. Amasaki, S., Kawata, K., Yokogawa, T.: Improving cross-project defect prediction methods with data simplification. In: Software Engineering Advanced Applications IEEE (2015)
11. Dai, W., Yang, Q., Xue, G., et a1.: Boosting for transfer learning. In: Proceedings of the 24th International Conference on Machine Learning, Corvallis, USA, 20—24 June 2007, pp. 93–200. ACM, New York (2007)
12. Chen, L., et al.: Negative samples reduction in cross-company software defects prediction. Inf. Softw. Technol. **62**, 67–77 (2015)
13. Fagui, M., et al.: Cross-project software defect prediction based on instance transfer. J. Front. Comput. Sci. Technol. **10**, 43–55 (2016)
14. Reshef, D.N., et al.: Detecting novel associations in large data sets. Science **334**(6062), 1518–1524 (2011)