Research and Appalication of Software Defect Predictionn based on BP- Migration learning

Jie Zhang[1], Gang Wang[2], Haobo Jiang[1], Fangzheng Zhao[1] and Guilin Tian[1]

[1]Graduate School, Air Force Engineering University, 710054 Xi'an Shaanxi, China
[2]Anti-air Defense Guide College, Air Force Engineering University, 710054 Xi'an Shaanxi, China

Abstract. Software Defect Prediction has been an important part of Software engineering research since the 1970s. This technique is used to calculate and analyze the measurement and defect information of the historical software module to complete the defect prediction of the new software module. Currently, most software defect prediction model is established on the basis of the same software project data set。The training date sets used to construct the model and the test data sets used to validate the model are from the same software projects. But in practice, for those has less historical data of a software project or new projects, the defect of traditional prediction method shows lower forecast performance. For the traditional method, when the historical data is insufficient, the software defect prediction model cannot be fully studied. It is difficult to achieve high prediction accuracy. In the process of cross-project prediction, the problem that we will faced is data distribution differences. For the above problems, this paper presents a software defect prediction model based on migration learning and traditional software defect prediction model. This model uses the existing project data sets to predict software defects across projects. The main work of this article includes: 1) Data preprocessing. This section includes data feature correlation analysis, noise reduction and so on, which effectively avoids the interference of over-fitting problem and noise data on prediction results. 2) Migrate learning. This section analyzes two different but related project data sets and reduces the impact of data distribution differences. 3) Artificial neural networks. According to class imbalance problems of the data set, using artificial neural network and dynamic selection training samples reduce the influence of prediction results because of the positive and negative samples data. The data set of the Relink project and AEEEM is studied to evaluate the performance of the f-measure and the ROC curve and AUC calculation. Experiments show that the model has high predictive performance.

1 Introduction

In the process of software development, the introduction of some defects is unavoidable and predictable despite careful plan, good document, and proper process control. These software defects may lead to debase the quality of software. And it may be essential to cause the failure of software. Nowadays, in frontier competitions, it is necessary to consciously control and minimize defects in software engineering. We can predict the occurrence of defects in other modules of the software or other software based on the historical records of the software project in the development process, including whether the defects are included and the number of defects. As a rule of thumb, the more complex a software is, the more likely it is to have defects. Before the defect prediction, the most important step is to get the training set, which includes the software measurement information of the training set and the measurement information [4] for the defect record and the project to be measured. In the case of no training set, the defect of the target project can be predicted due to the connection between the training set and the test set. Since training set date cannot be guaranteed to be valid, the data need to eliminate noise before the defect prediction model is established; in addition, although software metrics is used to describe the complexity of the software project and the software defects, the correlation between different software metrics and defect information may be different. Similarly, supplemented by correlation analysis can improve the accuracy of software defect prediction. For the same software project set, the above method can effectively predict the occurrence of defects in other modules in the same project set. But for projects with only a small number of data items or new projects, the accuracy of the prediction is low and cannot be persuasive due to less training set data. As a result, the concept of migration learning is proposed.

2 Software defects

Software defects often referred to as bugs. In general, software defects refer to errors or problems in software

development projects that affect the normal function of the program, as well as potential defects that are not discovered [1]. The occurrence of software defects will seriously affect the progress of software development, greatly increasing the cost of software development. That is to say, the later the software defects are discovered, the higher cost is needed to fix the defects and maintain the normal operation of the software. For example, it is assumed that producer fixes up an error in the requirement phase. It takes the cost as 1. When it comes to the design phase, it takes at the 3 to 6 times of the cost. At the programming development stage, it takes 10 times of the cost. To modify and maintain the cost of a defect is up to 20 to 40when the internal test is performing. And the time of external test increases from 30 to 70 times. Surprisingly, after the product is released, the cost of repairing software defects is up to 40 to 1000 times. The cost of defect repair does not vary linearly with time but almost exponentially increase. The figure shows the repair costs for defects in different stages of software project development.

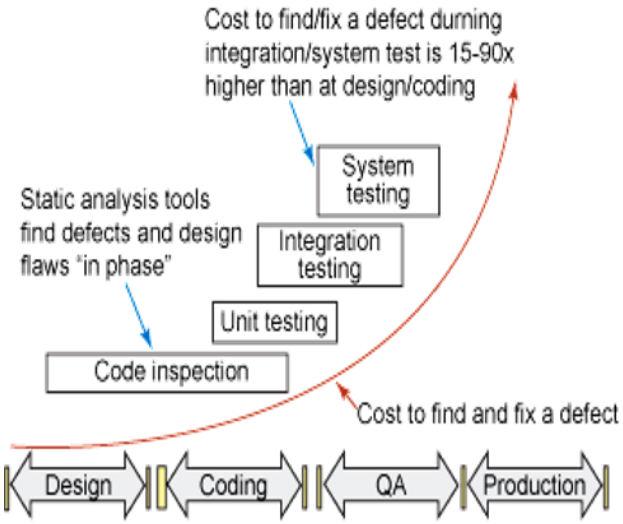

Figure 1. Software defect repair costs

In addition, the time and cost increase rapidly. The most essential impact of software defects is that software products or development projects cannot achieve the expected goals and even lead to unpredictable and bad consequences. Potential defects can even seriously threaten our safety in production, life and technology. For software defects, IEEE 729-198 has the standard and specific definitions: internally, software defects are errors in the entire project development process and the entire maintenance process. But in externally, defects are failures and incompleteness of system functions.

2.1 BP neural network

BP neural network is a multi-layer artificial neural network, which is frequently used at present. The learning of neural networks adopts the rules of rapid descent, and it does not need to describe the equation that indicates the mapping first. It directly learns to train and saves the input-output mappings of high order. In order to minimize the neural network error (to prevent the positive and negative error offset by the sum of the squares of the errors), the BP neural network uses the back-propagation algorithm again and again to determine the threshold and weight of the network model. The topology of the BP neural network includes: an input layer, an implicit layer and an output layer. The BP neural network belongs to a supervised neural Network. The learning and training process of the network is basically the input of the data sample, which is used to train the whole model, and then continuously adjusts by using the back-propagation algorithm. By adjusting the network weight value and the deviation generated by the intermediate learning process, the difference between the output result and the expected result is as small as possible. Once the error obtained by the output layer is less than the error value given before training, the current weight value and deviation of the BP neural network model are saved, and it is determined that the entire training process has been completed.

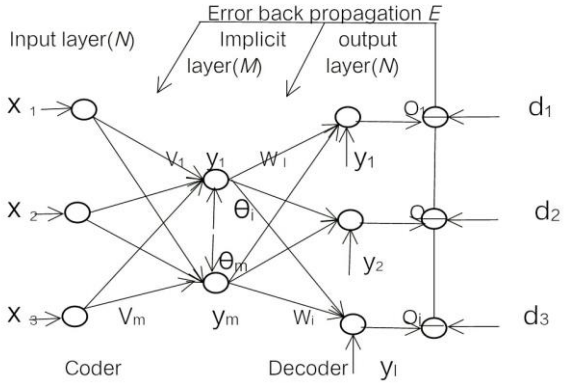

Figure 2. BP neural network schematic

2.2 Dynamic sample selection

In the process of migrating learning, the difference in the distribution of the source and target domain data may cause the deviation of the test result, which leads to the result that can be greatly reduced [13]. In dealing with the problem of class imbalances, the combination of dynamic sample selection and artificial neural network (ANN) can be effectively solved by the class imbalance problem. The process of solving class imbalance with the reference dynamic sample selection method includes [8]: (1)firstly, the training data set is divided into a series of different balanced subsets according to the agreed division method; (2)secondly, the different sets of balanced subsets obtained are allocated to their respective base classifiers to serve as the initial training data set; (3) according to the dynamic samples selection method, cyclic training is carried out on each base classifier; (4) for the test set, the decision results can be obtained by using the decision function that is obtained from each training. (5) according to the decision result, to select the data item that can be used to adjust the neuron parameters instead of using all the data to adjust the neuron parameters, which can avoid the over-fitting problem of the neural network; (6) Each decided result is integrated to get the final recognition result.

2.3 Migration learning

Migration learning, the purpose of which is to extend the knowledge and develop skills in a certain field to complete a similar learning work in another new environment or field. That also means to use the knowledge already exist or have learned to finish the pair that is not exactly the same but a machine learning method that solves certain problems or situations [14]. The human has an ability to transfer the knowledge what they learned. The method that traditional machine learned is difficult to carry out and study in the absence of a large number of tagged data set. However, in order to make sure the relative application to keep machine learning, a large amount of manpower and a large number of material resources need to be consumed to calibrate those in various fields. For migration learning, it is possible to migrate existing data to complete the learning of future tasks. The goal is to migrate the knowledge of learning tasks to achieve the migration of different environmental applications. This method can effectively solve the problems about unsufficient learning data and new projects. Cross-project is the biggest advantage of migration learning. Therefore, in recent years, migration learning has become more and more widely used in multi-source heterogeneous data, including image classification processing, artificial intelligence decision, text processing and emotional analysis, etc.

For image classification, Zhu et al. use Tag tags information as a bridge for knowledge transfer between texts and images, and propose a heterogeneous migration learning method. In terms of text processing, Dai et al. make use of the word features of documents to transfer knowledge from different fields to share the same word features. Pan et al. migrate the feature values from different fields to a cluster and apply spectral feature alignment to improve the accuracy of classification in the target domain.

In the case of Cross-Project defect prediction, migration learning can also be used to solve the problem of data distribution feature differences, for example:

The migration component analysis is to map the data of two domains to the same hidden space. In Cain space, the distribution distance of data in the two domains can be minimized, which can reduce the difference caused by different data distribution. Migration component analysis was proposed by Nam et al. Because different preprocessing methods of data sets lead to differences in software defect prediction performance, Nam et al. define data preprocessing method selection rules before performing migration analysis. Experiments show that the migration component analysis has better performance for cross-project prediction.

Ma et al. proposes a cross-project software defect prediction method for Migrating Bayes. The Bayesian model is used to convert the weight of the source project data items to the best between the source domain and the destination domain. Experiments show that the Migration Bayes has a better performance on cross-project defect prediction.

3 Software defect prediction model framework

According to the purpose of high accuracy of predicting software defects, the software defect prediction model based on transfer learning can be simplified to the following input and output steps:

Input: Pre-processed source project defect data includes the metric vector of the source project and the corresponding software defect information; and the metric vector set of the target project, in where the software metrics of the source domain and the destination domain are in correlation analysis and association analysis.

Output: The target domain data set is tag set. The prediction result is Boolean, and the predicted result is respectively recorded as a module defects and the other without defects, namely, Buggy and Clean.

According to the input and output steps, the software defect prediction process evolves as follows:

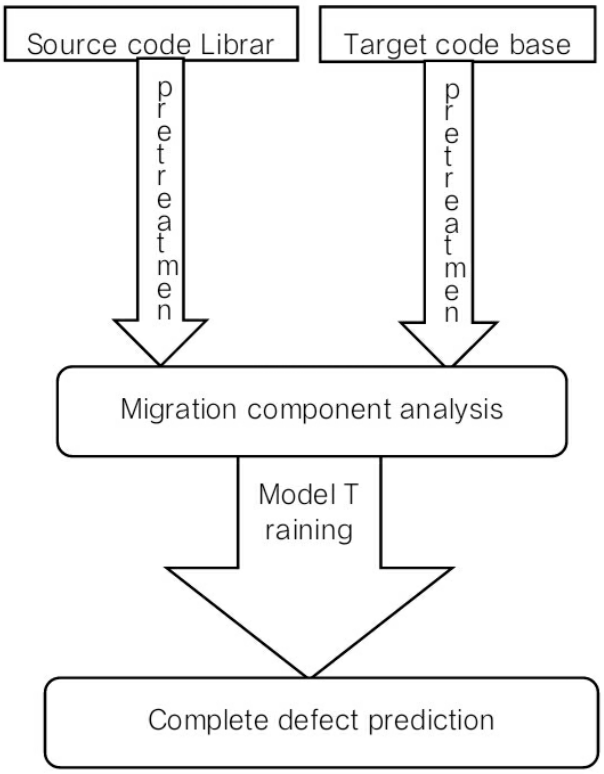

Figure 3. Software defect prediction process

The basic principle of software defect prediction based on migration learning can be roughly divided into three parts: data preprocessing for denoising data and feature selection, software defect prediction model based on migration process analysis and prediction based on model result. The basic principle is shown in Figure 4.

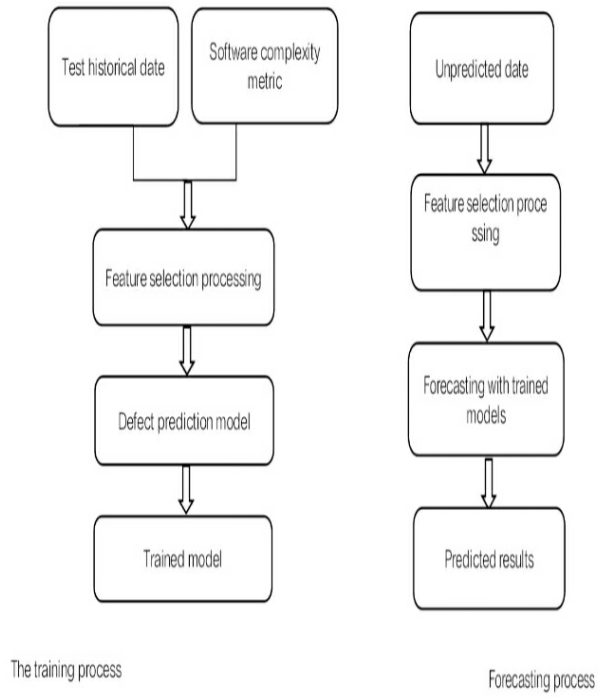

Figure 4. Basic principles of software defect prediction

Data preprocessing includes data normalization, correlation analysis and correlation analysis of software defects and software metrics. It is the basic and necessary steps to establish the model. Model establishment includes migration learning, dynamic sample selection and artificial neural network and other parts. The defect attribute of the sample data used to train the model is a Boolean data type: Buggy (including defects) and Clean (without defects). Defects is represented by "1" and no defects is represented by "0". Thus, the output of the prediction result is also a Boolean type data: 1 indicates that the module has a higher defect tendency and 0 indicates that the module has a low tendency to be defective.

4 Data sets

In this study, two data sets, Relink and AEEEM, are selected for research. Both data sets are from software projects already recorded in the data warehouse. Software metric information is as Table 1 and Table 2

Table 1. AEEEM part of the metric description.

Software project name	Metric feature	Metric feature description
AEEEM	ck_oo_cbo	Inter-class coupling
	ck_oo_numberOf LineOfCode	Total number of lines of code
	numberOfBugsF oundUntil	Number of BUG found
	numberOfCritical BugsFoundUntil	Number of Important BUG found
	CvsEntropy	Code change entropy
	CvsLogEntropy	Code change Logarithmic decay entropy
	LDHH_cbo	Coupling linear attenuation entropy among different classes
	LDHH_numberO fLineOfCode	Linear attenuation Entropy of the total number of lines of code
	WCHU_cbo	Coupling weight transformation of different classes
	WCHU_number OfLineOfCode	Code line weight transformation

The data metrics selected in the same data from the two data set are different. And the software metrics selected by the ReLink data set and the AEEEM data set are also different. These metric features include code line count, class coupling, loop complexity and more.

Table 2. ReLink part metric description

Software project name	Metric features	Metric Feature Description
ReLink	AvgCyclomatic	Average cyclomatic complexity of all nested functions in a method
	MaxCyclomatic	Maximum cyclomatic complexity in all nested functions or methods
	SumCyclomatic	Total cyclomatic complexity in all nested functions or methods
	CountLine	The total number of lines of code that the nested function contains
	AvgLine	The average number of lines of code for all nested functions or methods
	CountStmt	Number of declared statements in code
	RatioCommentTo Code	Note code lines are proportional to all lines of code

Table 3. Data Information

Project name	Subproject name	The total number of file	Defect file	The total number of futures	Project description
Relink	ZXing	399	118(29.57%)	26	Bar-code reader library
	Apache http server	194	98(50.52%)	26	Web server

	OpenIntents Safe	56	22(39.29%)	26	Security
AEEEM	Equinox	325	129(39.60%)	61	OSGI framework
	Eclipse JDT	997	206(20.66%)	61	Develop-ment
	Apache Lucence	399	39(9.26%)	61	Text search engine library
	Mylyn	1862	245(13.16%)	61	Task Manage-ment
	Eclipse PDE	1492	209(14.01%)	61	Develop ment

The AEEEM data set is collected by D'Ambros [24] and others, and contained 61 software metric features and 5368 data items. ReLink is collected by Wu et al. All defect information has been manually verified and modified, including 26 software features and 658 data items. The specific information is shown in Table 3.

5 Experimental set up

The first step of the experimental process in this paper is to perform data pre-processing on the data used in the experiment, followed by model establishment and finally analysis of the results. For the sake of data items can be sorted in dimensions and volume, the data preprocessing part is divided into 3 part: data normalization, correlation analysis and so on. This paper sets up two trials of Within-project software defect prediction and Cross-project software defect. In order to verify the validity and accuracy of the TCA migration learning model, this paper first uses the Within-project defect prediction with the same data feature distribution. Namely, the training data set and the test data set used for the training model are all from the same software project. Then the Cross -Project defect prediction, which is used in the training data set and test dataset from different projects, has the characteristics of different data distribution. The Within-project data prediction results are used as a comparison and reference for Cross-Project prediction results.

5.1 Within-project

There were 8 different sub-projects in the two project data sets of Within-Project ReLink and AEEEM, so we carried out 8 sets of Within-Project prediction experiments. For each sub item, this paper selects 50% data items as training data sets, and the remaining 50% data items are used as test data sets to predict. Since 50% of the training data sets are randomly selected, the experiment will carry out 100 repetitions and random selection of the training set data and test set data in order to reduce the instability of the random selected data, and finally get the average prediction results.

In order to verify the effective effect of migration learning on the prediction of cross project defects, data denoising and the artificial neural network are carried out in the process of Within-Project, excluding the migration learning part.

5.2 Cross-Project

There are 3 and 5 sub - item data sets of the two data sets selected in this experiment, so two groups of 26 crossover prediction combinations are carried out when the Cross-Project defect prediction model is verified by the migration learning. Including ZXing→ Apache，ZXing→ Safe，Apache → ZXing，Safe → Apache，Safe → Apache，Apache → Safe, and 20 cross prediction data combinations within the ReLink data set. The sub items in these two data sets cannot be cross predicted because ReLink and AEEEM two project data sets have different types of metric characteristics,.

In every cross prediction combinations, the prediction model is trained by a subset of the sub item data set as the source item training set, while the other is the test set of the target item. For example, in ZXing Apache, ZXing data sets are used to establish defect prediction models to predict the defect results of Apache datasets.

6 Results and analysis

6.1within-project prediction results analysis and discuss the F-measure value varies with the maximum number of iterations. The F-measure will remain at a higher value after a range of up and down fluctuations. The number of neurons will also affect the F-measure. As the number of neurons increases, the F-measure will be maintained first. In a higher range, it will decrease rapidly due to oversize of neurons.

The result of each prediction is slightly different. The following figure is the result of the defect prediction in one of the two data sets.

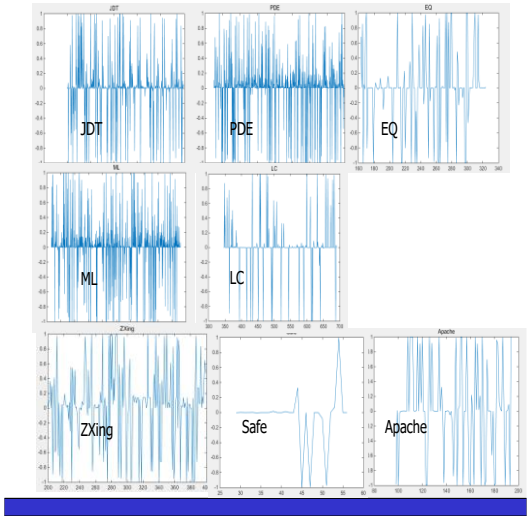

Figure 5. Project prediction results

The point between plus and minus 0.5 is the point of accurate prediction results. The ROC curve is shown in

figure 6.

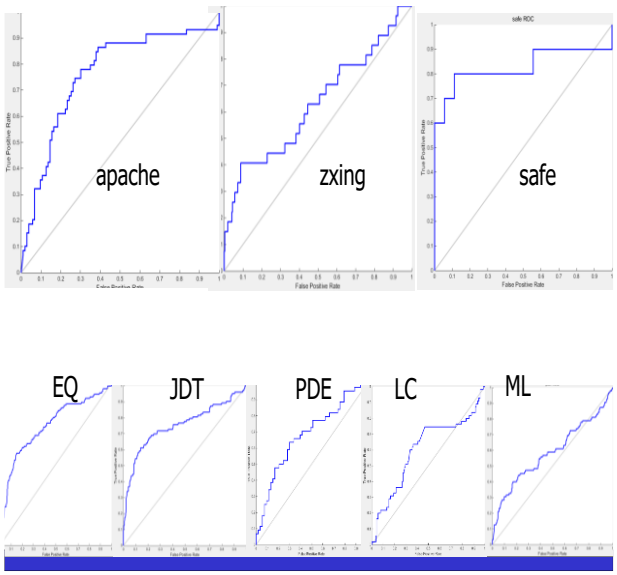

Figure 6. ROC curve of the prediction results within the project**.**

The area AUC below the ROC graph indicates the accuracy of the prediction results.

According to statistics, the average F-measure value of the inter-project forecast results is as follows:

Table 4. Project prediction results

Project name	Subproject	F-measure
ReLink	ZXing	0.34
	Apache	0.67
	Safe	0.65
AEEEM	EQ	0.52
	JDT	0.53
	LC	0.34
	ML	0.29
	PDE	0.43

6.2 Cross-project prediction results analysis and discussion

The results of the Cross-Project forecast are as follows, in which the average value of the prediction value in the last column is the average result of the prediction of the F-measure by the items that train the data set in the table as the test data set. The specific prediction results are shown in Table 5 and Table 6.

Table 5. ReLink cross project prediction results

Project Data Set	Training data sets	Training data sets	F-measure	Average
ReLink	ZXing	Apache	0.67	0.28
		Safe	0.59	
	Apache	ZXing	0.46	0.59
		Safe	0.54	
	Safe	Apache	0.51	0.57
		ZXing	0.11	

By comparing the results of Within-Project and Cross-Project, we can see that the prediction results of cross project defects are generally slightly lower than those in the project, but the basic trend is the same. So it shows that the migration learning model for defect prediction proposed in this paper is feasible. However, we can see from the results that there are some data values with large deviations. The above is the ROC curve of the ML and EQ projects. As you can see from figure 4, the AUC of the EQ project is larger than the AUC of the ML project.

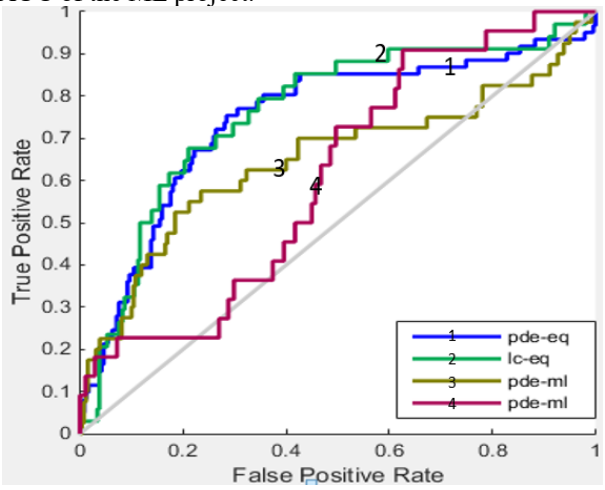

Figure 7. The ROC and AUC of the ML and EG project

In addition, by comparing the prediction results of the ML and EQ projects, it can also be seen that the F-measure of the EQ project is higher than that of the ML project.

Table 6. ASSUME cross project prediction results

Project data set	Training data set	Test data set	F-measure	Average test value
AEEEM	JDT	EQ	0.33	0.44
		LC	0.24	
		ML	0.28	
		PDE	0.27	
	EQ	JDT	0.40	0.38
		ML	0.17	
		LC	0.29	
		PDE	0.30	
	ML	EQ	0.24	0.23
		JDT	0.42	
		LC	0.10	
		PDE	0.27	
	PDE	JDT	0.47	0.29
		EQ	0.43	
		ML	0.27	
		LC	0.33	
	LC	JDT	0.48	0.24
		ML	0.20	
		EQ	0.50	
		PDE	0.32	

For projects with similar defect rate, the prediction performance is good. However, for projects with low defect rate, such as ML and LC, the prediction results are cross-sectional, which may be caused by the lack of defect data sets and insufficient learning. When the target data is small, the lack of sufficient data is more likely to lead to wrong predicted results. Compared with the AEEEM project, the general prediction results of the ReLink project are better than that of the AEEEM project. At the same time, compared with the defect data rate, it can be concluded that the ReLink project has a better unbalanced equilibrium rate. Therefore, it is further concluded that the effect of migration learning of this model is better [19][20] when the data balance rate of the project set is high. Since each prediction result is slightly different, the F-measure value is the average value after testing 100 times.

7 Summary

In this paper, the migration component analysis and the artificial neural network based on dynamic selection are used in the model construction. Among them, the migration component analysis is to map the high-dimensional data set kernel to the low-dimensional hidden space to achieve the purpose of dimensionality reduction migration. The artificial neural network based on dynamic selection can effectively reduce the data prediction bias caused by class imbalance. In the model verification, we define the validity and characteristics of the model as explained by the accuracy and F-measure. In addition, the paper also evaluates the model prediction results through ROC and AUC.

Through the verification of final experiment, the software defect prediction model based on migration learning proposed in this paper is feasible, and its prediction result is more accurate.

References

1. Z. Yuan , L. L. Kember ,C. Liu. *Defect recognition method for fine grained software change*. J B UNIV AERONAUT ASTRONAUT.**40(9)**, 1231-1238 (2014)
2. F. Joe. *Research on software defect prediction technology*. PLA Information Engineering University (2013)
3. X. Chen, Q. Gu, W. S. Liu al et. *Research on static software defect Prediction method.* S.J. **27(1)**, 1-25 (2016)
4. H. Tian, T. Y. Pu. *Research on software defect prediction method based on migration learning*. Journal of Southwest Normal University. **39(3)**, 90-95 (Nature Science Edition, 2014)
5. Q. Wang, S. J. Wu, M. S. Lee. *Software Defect Prediction Technology*. S.J. **19(7)**, 1565-1580 (2008)
6. B. W. Lee. Cross-project software defect prediction based on migration learning. Shanghai jiaotong university (2015)
7. X. Wang, P. He, D. Chen al et. *Training data selection method in cross-project defect prediction*. Computer application. **36(11)**, 3165-3169 (2016)
8. B. Hou, J. S. Wu, J. Zhang al et. A method of biometric information recognition based on dynamic sample selection and integration. CN 101763466 B[P] (2011)

9. Watanabe S, Kaiya H, Kaijiri K. *Adapting a fault prediction model to allow inter languagereuse.* International Workshop on Predictor MODELS in Software Engineering. ACM. 19-24 (2008)
10. Turhan B, Menzies T, Bener A B, et al. On the relative value of cross-company and within-company data for defect prediction. Empir Softw Eng. Empirical Software Engineering. **14(5)**,540-578 (2009)
11. F. G. Mao, B. W. Lee, B. J. Shen. Cross-Project Software Defect Prediction Based on Instance Transfer. The only official website for computer Science and exploration. **10(1)**, 43-55 (2016)
12. Y. Ma. *Research on software defect prediction technology based on machine learning*. UESTC (2012)
13. Wu R, Zhang H, Kim S, et al. ReLink: recovering links between bugs and changes. ACM Sigsoft Symposium and the, European Conference on Foundations of Software Engineering. ACM. 15-25 (2011)
14. M. Cheng, G. Q. Wu, M. T. Yuan. *Software defect prediction based on migration learning*. CHIN J ELECTRON.**44(1)**,115-122 (2016)
15. S. Mei. *SVM ensemble based transfer learning for large-scale membrane proteins discrimination*. Journal of Theoretical Biology, **340(307)**,105-110 (2014)
16. Borgwardt K M, Gretton A, Rasch M J, et al. Integrating structured biological data by Kernel Maximum Mean Discrepancy[J]. Bioinformatics, 2006, 22(14):49-57.Nam J, Pan S J, Kim S. Transfer defect learning. International Conference on Software Engineering.382-391 (2013)
17. Y Ma, Luo G, Zeng X, et al. Transfer learning for cross-company software defect prediction. Information & Software Technology, **54(3)**, 248-256 (2012)
18. Y. K. Lee. Application of optimization algorithm in BP network. New product of new technology in China. **(18)**,34-34 (2011)
19. Watanabe S, Kaiya H, Kaijiri K. Adapting a fault prediction model to allow inter languagereuse. International Workshop on Predictor MODELS in Software Engineering. ACM. 19-24 (2008)
20. Huang J, Smola A J, Gretton A, et al. Correcting sample selection bias by unlabeled data. Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December. DBLP. 601-608 (2006)
21. Shivaji S, Whitehead E J, Akella R, et al. Reducing Features to Improve Code Change-Based Bug Prediction. IEEE Transactions on Software Engineering. **39(4)**,552-569 (2013)
22. Pan S J, Kwok J T, Yang Q. Transfer learning via dimensionality reduction. AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, Usa, July. DBLP. 677-682 (2008)
23. J. Lee, M. Z. Kim. *Research of software Process measurement technology*. Computer Engineering and Applications. **37(5)**,86-90 (2001)
24. Pan S J, Tsang I W, Kwok J T, et al. Domain Adaptation via Transfer Component Analysis. IEEE Transactions on Neural Networks. **22(2)**,199-210 (2011)