The cross-project defect prediction based on PSO and Feature Dependent Naive Bayes

To cite this article: Zhexi Yao et al 2019 J. Phys.: Conf. Ser. 1237 022126

View the article online for updates and enhancements.
The cross-project defect prediction based on PSO and Feature Dependent Naive Bayes

Zhexi Yao\(^1\), Li Sun\(^1\), Tao Zhang\(^2\) and Jinbo Wang\(^2\)

\(^1\)University of Chinese Academy of Sciences, Beijing
\(^2\)Technology and Engineering Center for Space Utilization, Chinese Academy of Sciences, Beijing, China
\(^*\)yaozheqian14@csu.ac.cn

Abstract. The software defect prediction often assumes that the software under test has rich historical information, which is a harsh condition in reality. Cross-project defect prediction can predict projects with rare historical information by using historically informative projects. The differences in software architecture and code environment pose challenges to cross-project software defect prediction. Based on the characteristics of software defect prediction sets, this paper introduces a two-stage algorithm based on particle swarm optimization algorithm and Feature Dependent Naive Bayesian classifier.

1. Introduction
Software defect prediction is an important part of the software testing field. The software defect prediction can predict the overall defect distribution of the module to be tested and guide the allocation of test resource. Within-project defect prediction often assumed that the target project has rich historical data, which can be used to mine the historical data of the software and predict new versions. However, the historical information is often rare, and there may not even be historical version information, such as new projects. At this point, cross-project defect prediction will show its advantage. The basic goal is to mine historically informative projects to predict another project that lacks historical information. The features of two projects in cross-project defect prediction are the same. But the differences in software architecture and code environment pose challenges to cross-project software defect prediction.

Classification imbalance is common in software defect prediction, and it is one important factor affects the effect of defect prediction. Classification imbalance means that the number of samples in a data set is significantly larger than the number of other samples. Classification imbalance problems exist in various research fields, such as text classification\(^1\), image processing\(^2\), medical diagnosis\(^3\) and so on. Especially in software defect prediction, the number of defective samples is larger than the samples without defects. It is more important to identify the defective samples than the non-defective ones. However, traditional classifiers often maximize the overall classification accuracy, which causes the prediction model to bias toward a certain type of sample. Existing methods for classification imbalance include sampling method\(^4\), cost sensitive learning\(^5\), ensemble learning\(^6\) and so on. Considering the characteristics of software defect prediction data set, this paper adopts the ensemble learning method to reduce the impact of classification imbalance.

A two-stage ensemble cross-project defect prediction method is proposed. In the first stage of ensemble learning, the Feature Dependent Naive Bayes (FDNB) classifier is used to train the model.
In the second stage of ensemble learning, multiple source projects are used to predict the target project, and the Particle swarm optimization (PSO) algorithm is used to search and combine different classifiers. Experiments are carried out on the commonly used cross-project defect prediction data set. The results show excellent prediction ability of the proposed algorithm.

2. Related works

2.1 Cross-project defect prediction

In cross-project defect prediction, the samples and features in the source and target projects are key factors related to their predictive performance. Cross-project defect prediction, which can be divided as (1) Supervise defect prediction, build defect prediction model according to the source project, and predict target project. (2) Unsupervised defect prediction, which is based on unmarked samples and predicts the target project using clustering analysis mostly. (3) Semi-supervised defect prediction, which is to build a model based on the source project together with a few labelled modules of the target project, and predict the remaining unlabelled modules in the target project. This paper belongs to semi-supervised defect prediction, and its structure is shown in Figure 1. The shaded part of the target project data set represents the tagged sample and will be used as the training set to train the model.

![Figure 1. Semi-supervised defect prediction](image)

The relationship between features has always been the challenge in defect prediction. The existing research mainly focuses on the study of the relationship between features and categories, while ignoring the relationship among features. The classification effect of one feature on another feature may have both positive and negative effects. Therefore, research on the relationship between features in the data set will have far-reaching significance for the prediction results.

2.2 The Particle swarm optimization algorithm

The PSO algorithm is an excellent heuristic algorithm and is widely used in various fields. Inspired by the behavior of birds, Kennedy and Eberhart proposed the population evolution algorithm in 1995\cite{7}. The particle swarm optimization algorithm uses a velocity-position model. The algorithm initializes a group of random particles and sets the velocity value for each particle to clarify its flight direction and distance information. In the iteration, the particle evolves by recording its two maximum values: the best place of each particle $p_{best}$ and the best place of global $G_{best}$. All particles are guided by a fitness function to achieve an overall guiding effect on the population. The update formula is listed in formula (1)-(2).

\begin{equation}
\begin{align}
    v_{jm}^{k+1} &= wv_{jm}^k + c_1r_1^k(p_{best_{jm}^k} - x_{jm}^k) + c_2r_2^k(G_{best_m}^k - x_{jm}^k) \\
    x_{jm}^{k+1} &= x_{jm}^k + v_{jm}^{k+1}
\end{align}
\end{equation}
Where w is the weight, c1 and c2 are the acceleration factors, r1 and r2 are the real number between (0, 1), \( x_{m+1}^k \) and \( x_{m+1}^k \) are the velocity and position of the \( j \)th particle in \( m \)th space at the \((k+1)\)th iteration, respectively. \( p_{\text{best}}^j \) is the individual optimal value that the \( j \)th particle reaches in the \( m \)th space at the \( k \)th iteration. It is the global optimal value achieved by the \( k \)th dimension of the population in the \( k \)th evolution. \( G_{\text{best}}^k \) is the global optimal value achieved by the \( k \)th evolution of the \( m \)th space of the population.

The PSO algorithm is a random and parallel optimization algorithm. The convergence speed is fast, and it is easy to program. In this paper, the PSO algorithm is applied in software defect prediction. Based on the idea of ensemble learning, the PSO is used to search the optimal combination of each classifier, which can effectively improve the efficiency of the final ensemble algorithm.

2.3 Feature Dependent Naive Bayes

In the Naive Bayes classification method, all features are assumed to be independent and have equal weight. However, these features are commonly interrelated in practice. The Naive Bayes classifier is widely used in various fields because of its clear principle, less sensitive to missing data, and fast running speed. And in the small-scale data set such as the software defect data set, the Naive Bayes classifier can achieve better results. The original Naive Bayes classification formula is shown in formula (3) and (4).

\[
P(C_i | X) = \frac{P(C_i)P(X | C_i)}{P(X)}
\]

\[
P(C_i | X) = \frac{P(C_i) \prod_{i=1}^{n} P(x_i | C_i)}{P(X)}
\]

Where \( C_i \) represents a category, \( k=0, 1 \) (defect, defect-free). \( X=(x_1, x_2, ..., x_n) \) represents the features. It can be seen from equation (3) that \( P(X) \) is determined, and the main calculations are concentrated on \( P(X|C_i) \). The assumption that the features are independent is shown in equation (4). This assumption reduces the amount of computation in the prediction; it also has a negative impact on the effect of the model.

However, the Naive Bayesian classification principle is premised on the mutual independence of features. It limits the effect of Naive Bayes classifiers on classification problems. In order to improve its performance, the researchers made great efforts. The literature [8] assumes that the features are independent of each other and proposes FDNB. The results of the experiment on the software defect prediction data set show the method can effectively improve the classification effect compared to Naive Bayes classifier. The core calculation formula of FDNB is shown in equation (5)-(6). A joint feature dictionary is established for the defective module and the non-defective module respectively. The detail information is listed in [8].

\[
P(x_i^j | C_{dp}) = \frac{dpN(x_i^j)}{dpN(x_i^j) + ndpN(x_i^j)}
\]

\[
P(x_i^j | C_{ndp}) = \frac{ndpN(x_i^j)}{dpN(x_i^j) + ndpN(x_i^j)}
\]

Where \( x_i^j \) represents the \( j \)th discrete quantity of the \( i \)th feature, \( dpN(x_i^j) \) represents the number of defective samples in the samples in the training set containing \( x_i^j \); The corresponding \( ndpN(x_i^j) \) indicates the number of defect-free samples in the training set containing \( x_i^j \).
In this paper, the FDNB classifier is applied to cross-project defect prediction problems. The FDNB models between different source datasets and the target dataset are established as the base classifier. And it provides better cross-project defect prediction in the second phase of ensemble learning.

3. The classifier based on PSO and FDNB

The algorithm in this paper is based on FDNB and PSO. The process framework is shown in Figure 2.

Firstly, the source projects in the cross-project defect dataset are combined with parts of the labeled samples of the target project to form a new source project. There are N+1 source projects: S1… SN, and TN. The algorithm combines N source items with TN respectively and trains the base classifiers. Meanwhile, the (N+1)th base classifier is established on TN. Then the PSO algorithm is used to find the optimal combination of each base classifier. The prediction process is shown in equation (7)-(8):

\[
\text{Label}(j) = \begin{cases} 
1 \text{(i.e., defective)}, & \text{if } \text{Comp}(j) \geq \text{threshold} \\
0 \text{(i.e., defectfree)}, & \text{otherwise} 
\end{cases} 
\]

\[
\text{Comp}(j) = \sum_{i=1}^{N+1} \alpha_i \cdot \text{Score}(j) / \text{LOC}(j) 
\]

Where \( \text{Label}(j) \) is the predict result of j instance; \( \text{threshold} \) determine the final result; \( \text{Score}(j) \) is the final score of j instance determined by i classifier, and it is the F1 in this paper; \( \alpha_i \) is the weight of classifier i.

The fitness function of the PSO algorithm needs to predict the category of the target project. The F1 value is set as the optimization target in PSO. The optimal combination of base classifiers for maximizing the F1 value is found and output. The definition of F1 value is listed in equation (9)-(11).

\[
\text{recall} = \frac{TP}{TP + FN} 
\]

\[
\text{precision} = \frac{TP}{TP + FP} 
\]

\[
F1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} 
\]

Where TP is the number of defective modules correctly predicted as dp; FP is the number of non-defective modules incorrectly predicted as dp; FN is the number of non-defective modules correctly predicted as ndp; TN is the number of non-defective modules incorrectly predicted as ndp. It can be seen that the F1 formula is a trade-off between accuracy and recall rate, which can better measure the prediction effect.

The algorithm calculates the predicted value of the current combination classifier for the target instance through formula (7), so as to calculate the F1 value of all samples in the target project. The \( \text{threshold} \) in formula (7) and the \( \alpha \) in formula (8) is the core elements in the process. Therefore, the
particles in the population of PSO algorithm have N+2 dimensions. The first N+1 dimension is the weight of the basic classifier, and the N+2 dimension is the threshold to be set. The pseudo code is shown in table 1:

| Table 1. Pseudo code of base classifier construction algorithm |
|-------------------------------------------------------------|
| Input: The source projects (1-N), tagged samples T_N in target project |
| Output: The best combination of bias classifier and The threshold responding |
| 1: Divide T_N into two parts T_N1 and T_N2 |
| 2: Fuse source projects with T_N1 as new source projects. N base classifiers was established through FDNB |
| 3: Establish the N+1 base classifier on the T_N1 data set through FDNB |
| 4: Initial population POP |
| 5: If maximum iterations has been reached |
| 6: Jump to step 9 |
| 7: Calculate the fitness function through equation (9)-(11). |
| 8: Updated particles by PSO algorithm in equation (7)-(8), and store the optimal position and global position |
| 9: Output the best particle and the best F1 |

As described in table 1, the TN is divided into T_N1 and T_N2. Where T_N1 contributed to getting new source projects, and T_N2 is used to validate in PSO for the total F1. After the base classifier is formed in steps 2 and 3, steps 3-8 use the global search ability of PSO to iterate the particles and obtain the final optimal position, that is, the optimal combination of the base classifier and the optimal threshold.

4. Experiments
To verify the effectiveness of the algorithm in this paper, two groups of experiments were designed: (1) experiment designed to show the effectiveness compared with other methods commonly used; (2) experiment designed to show the advantage compared to NB classifier.

4.1 The design of Experiment
This paper conducts multi-to-one experiments on the AEEEM[9] data set. In the first group, the algorithm is compared with the typical cross-project prediction method, Burak filter[10], peter[11], ensemble learning based CODEC[12], feature mapping based TCA + method[13]. The second group is a multi-to-one experiment which compares the effects between FDNB and Naive Bayes.

4.2 Dataset and measurement
The experiment is carried on the five data sets of AEEEM. The basic data is shown in table 2. In column 5, the defect rate is listed. It can be seen that class imbalance is very common in the AEEEM data set.

| Table 2. AEEEM dataset |
|------------------------|
| dataset | features | instance | defect | defect rate(%) |
| JDT   | 61       | 997      | 206    | 20.70          |
| LC    | 61       | 691      | 64     | 9.30           |
| ML    | 61       | 1862     | 245    | 13.20          |
| PDE   | 61       | 1497     | 209    | 14.00          |

In order to eliminate the difference between different evaluation indicators, this paper uses F1 value and bal value to measure the results on different data sets. The F1 value is defined in equation (9)-(11). For the bal value, it is shown in equation (12)-(13).

\[ \phi = \frac{FP}{FP + TN} \]  (12)
The definition of recall in equation (13) is listed in equation (9). It can be seen that the bal indicator can be regarded as the Euclidean distance between (recall, pf) and (1, 0), which is an intermediate balance value between the recall and pf indicators.

4.3 Results

According to the basic introduction in the previous section, the algorithm is applied to the AEEEM dataset, and the one-to-one experimental results are shown in table 3. In table 3, column 3 lists the best values of each target projects of one to one experiment listed in paper [14]. The results of the experiment in the second group are listed in table 4, in which, the bal index is used as the fitness function value.

Table 3. Comparison of FDNB and other method

| Target | FDNB(F1) | other method(F1) |
|--------|---------|-----------------|
| JDT    | 0.671   | 0.590           |
| LC     | 0.532   | 0.420           |
| ML     | 0.521   | 0.360           |
| PDE    | 0.523   | 0.400           |

Table 4. Comparison of FDNB and NB

| Target | FDNB(bal) | NB(bal) |
|--------|-----------|---------|
| JDT    | 0.5053    | 0.4863  |
| LC     | 0.4774    | 0.4457  |
| ML     | 0.5167    | 0.4439  |
| PDE    | 0.4873    | 0.4806  |

From the results in table 3, the algorithm based on FDNB is superior to the algorithm in other papers. The consideration of joint feature is proved to promote the classification ability. From the experimental results in table 4, it can be seen that the proposed algorithm can achieve better prediction results on the AEEEM dataset than the Naive Bayesian classifier. It can be seen that comprehensively considering the distribution of joint probability in cross-project defect prediction can improve the classification performance of cross-project defect prediction.

5. Conclusions

In this paper, the PSO algorithm and the FDNB algorithm are combined to form the integrated learning classifier, and the classifier is applied to cross-project defect prediction. The FDNB algorithm improves the predict ability, and the mutual relation between features in defect prediction research is considered. At the same time, the imbalance of datasets for software defect prediction is considered. A two-stage defect prediction model is constructed through PSO. By adjusting the weight of different base classifiers and synthesizing the prediction ability of the base classifier, the model with better classification ability and stronger robustness is proposed. Experiments were carried out on AEEEM data sets to verify the effectiveness of the proposed algorithm.

References

[1] Liu, Y., Loh, H.T., Sun, A. (2009) Imbalanced text classification: A term weighting approach. Expert Systems with Applications. pp. 690-701.

[2] Phua, C., Alahakoon, D., Lee V. (2004) Minority report in fraud detection: Classification of skewed data. ACM SIGKDD Explorations Newsletter. pp.50-59

[3] Mena, L., Gonzalez, J.A. (2009) Symbolic one-class learning from imbalanced datasets: Application in medical diagnosis. International Journal on Artificial Intelligence Tools. pp. 273-309.
[4] Tahir, M.A., Kittler, J., Yan, F. (2012) Inverse random under sampling for class imbalance problem and its application to multi-label classification. Pattern Recognition. pp. 3738-3750.

[5] Siers, M.J., Islam, M.Z. (2015) Software defect prediction using a cost sensitive decision forest and voting, and a potential solution to the class imbalance problem. Information Systems. pp. 62-71.

[6] Sun, Z., Song, Q., Zhu, X. (2015) A novel ensemble method for classifying imbalanced data. Pattern Recognition. pp.1623-1637.

[7] Kennedy, J., Eberhart, R.C. (1995) Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks, pp:1942-1948

[8] Arar, O, Ayan, K. (2017) A feature dependent Naive Bayes approach and its application to the software defect prediction problem, Applied Soft Computing, 58

[9] Ambros, M.D., Lanza, M., Robbes, R. (2010) An extensive comparison of bug prediction approaches,” In: Proc. of the IEEE Working Conf. on Mining Software Repositories, pp.31-41.

[10] Turhan, B., Menzies, T., Bener, A.B., Stefano, D.J. (2009) On the relative value of cross-company and within-company data for defect prediction. Empirical Software Engineering. pp.540-578.

[11] Peters, F., Menzies, T. Marcus. (2013) Better cross company defect prediction,” In: Proc. of the IEEE Working Conf. on Mining Software Repositories, pp. 409-418.

[12] Panichella, A., R. Oliveto, A. D. Lucia (2014) Cross-Project defect prediction models: L’Union fait la force. In: Proc. of the IEEE Conf. on Software Maintenance, Reengineering and Reverse Engineering, pp. 164-173.

[13] Nam, J., Pan, S.J., Kim, S. (2013) Transfer defect learning. In: Proc. of the Int’l Conf. on Software Engineering, pp.382-391.

[14] He JY, Meng ZP, Chen X, Wang Z, Fan XY. (2017) Semi-Supervised ensemble learning approach for cross-project defect prediction. Journal of Software, pp: 1455-1473.