Software Defect Prediction Using Dynamic Support Vector Machine

Bo Shuai
School of Electronic Science and Engineering
National University of Defense Technology
Changsha, Hunan, P. R. China, 410073
shuaibo85@163.com

Haifeng Li, Mengjun Li, Quan Zhang, Chaojing Tang
School of Electronic Science and Engineering
National University of Defense Technology
Changsha Hunan, China
quanzhang@nudt.edu.cn

Abstract—**In order to solve the problems of traditional SVM classifier for software defect prediction, this paper proposes a novel dynamic SVM method based on improved cost-sensitive SVM (CSSVM) which is optimized by the Genetic Algorithm (GA). Through selecting the geometric classification accuracy as the fitness function, the GA method could improve the performance of CSSVM by enhancing the accuracy of defective modules and reducing the total cost in the whole decision. Experimental results show that the GA-CSSVM method could achieve higher AUC value which denotes better prediction accuracy both for minority and majority samples in the imbalanced software defect data set.**

Keywords-software defect; CSSVM; GA; AUC

I. INTRODUCTION

Software testing is a critical step to ensure software quality. Software testing is aimed at find hidden flaws in the software module as much as possible before running into production. Now, with the size and complexity of software increasing, the traditional software testing methods such as preparing test cases and tracking failures artificially become less adaptive. Machine learning methods such as SVM could automatically predict whether software modules contain defects. Thus, it could help eliminating the testing time for non-defective module and limiting human resources to focus on high-risk defective software modules, thus improving software development efficiency.

Statistical, machine learning, and mixed techniques are widely used in the literature to predict software defects. Khoshgoftaar [1] used zero-inflated Poisson regression to predict the fault-proneness of software systems with a large number of zero response variables. Munson and Khoshgoftaar [2] also investigated the application of multivariate analysis to regression and showed that reducing the number of "independent" factors (attribute set) does not significantly affect the accuracy of software quality prediction. Lesley, Barbara, and Susan [3] found that multivariate regression analysis performed better if the data has only minor skewness, and residual analysis performed the best for data with severe heteroscedasticity. Yong Wang [4] developed PACE regression, and showed that it performs the best compared with other regression models for high dimensional data.

In this paper, we introduce the GA theory into the construction of the CSSVM classifier in order to adapt the imbalanced data set dynamically, which improves the software defect prediction performance.

II. FRAMEWORK

The specific process of the proposed method is shown in Fig. 1. Firstly, a CSSVM classifier is constructed based on the features of software defect data. Secondly, m chromosomes are generated randomly according to the rules of GA and Applied to the CSSVM model. Thirdly, considering the misclassification cost for imbalanced data, the fitness function in GA method is confirmed. Then, the selection, crossover and mutation operations are executed in turn and iteratively until the termination condition is satisfied. In the end, the optimized CSSVM classifier is achieved.

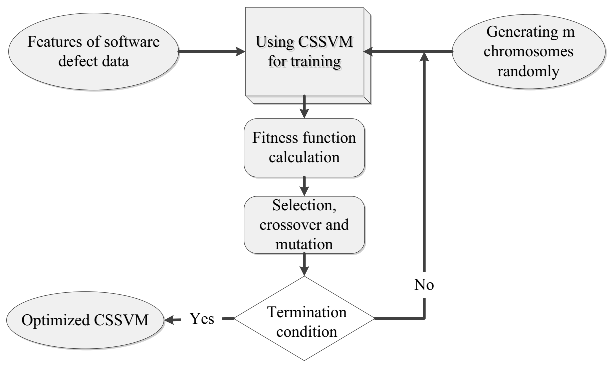
Features of software defect data
Using CSSVM for training
Generating m chromosomes randomly
Fitness function calculation
Selection, crossover and mutation
No
Optimized CSSVM
Yes
Termination condition

Figure 1. Framework of the GA-CSSVM method

III. COST-SENSITIVE SUPPORT VECTOR MACHINE

A. Standard Support Vector Machines

Support Vector Machine (SVM) [5] is a supervised classification algorithm which was first presented by Cortes and Vapnik. SVM classifications may be more accurate than the widely used alternatives such as classification by maximum likelihood, decision tree, and neural network-based approaches. An SVM aims to fit an optimal separating hyper-plane (OSH) between classes by focusing on the training samples that lie at the edge of the class distributions, the support vectors. The OSH is oriented such that it is placed at the maximum distance between the sets of support vectors. It is because of this orientation that SVM is expected to generalize more accurately on unseen cases relative to classifiers that aim to minimize the training error such as neural networks. Thus, with SVM classification only some of the training samples that lie at the edge of the class distributions in feature space called support vectors are needed in the establishment of the decision surface unlike statistical classifiers such as the widely used maximum-likelihood classifiers in which all training cases are used to characterize the classes. This potential for accurate classification based on small training sets means that the adoption of SVM classification can provide the analyst with considerable savings in training data acquisition.

B. Construction of Cost-Sensitive Support Vector Machines

Standard SVM has shown a good classification performance on the balanced data set. However, SVMs are not cost-sensitive, like other traditional classifiers perform poorer in the imbalanced situation. The reason is that the misclassification cost for positive and negative samples in SVM model is equal which leads to a bias prediction in the imbalanced data set.

Therefore, the reference [6] proposed a cost-sensitive support vector machine (CSSVM), given different misclassification cost according to different category samples. As shown in (1), the former k samples are labeled positive class, the latter l-k samples are labeled negative class, k is far less than l-k. In order to predict more correctly for positive class samples, the misclassification cost for positive samples C^+ should be much larger than C^- which denotes the misclassification cost for negative samples.

$$\min\left\{\frac{1}{2}\|W\|^2 + C^+\left(\sum_{i=1}^{k}\xi_i\right) + C^-\left(\sum_{i=\mathrm{k}+1}^{l}\xi_i\right)\right\} \tag{1}$$

$$s.\mathrm{t}.$$

$$y_i(w\cdot x_i + b) \ge 1-\xi_i, \mathrm{i}=1,2,\dots,\mathrm{l}$$

Where, ξ_i denotes relaxation coefficient.

Equation (1) is transformed to the QP problem with the Lagrange multiplier method. Then the final decision function is obtained as (2).

$$\mathrm{f}(x) = \mathrm{sign}\,\{\sum_{i=1}^{l} y_i\alpha_i + b\} \tag{2}$$

By introducing the RBF kernel function, CSSVM model is expressed as (3).

$$max\sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j y_i y_j\, K(x_i,x_j)$$

$$s.\mathrm{t}.$$

$$\sum_{i=1}^{l} y_i\alpha_i = 0 \tag{3}$$

$$0\le \alpha_i \le C^+, \mathrm{i}=1,2,\dots,\mathrm{k}$$

$$0\le \alpha_i \le C^-, \mathrm{i}=\mathrm{k}+1,\mathrm{k}+2,\dots,\mathrm{l}$$

Then, the corresponding final decision function is obtained as (4).

$$\mathrm{f}(x) = \mathrm{sign}\,\{\sum_{i=1}^{l} y_i\alpha_i K(x,x_i) + b\} \tag{4}$$

The CSSVM model is represented by (2) and (4), which have taken the consideration of misclassification costs for different category samples, where, , α_i is called cost-sensitive support vector.

IV. Optimization of the CSSVM Classifier Using Genetic Algorithm

A. Genetic Algorithm

Genetic Algorithm (GA) [7] is an adaptive search method for finding optimal or near optimal solutions, premised on the evolutionary ideas of natural selection. The basic concept of GA is designed to simulate processes in the natural system necessary for evolution, specifically those that follow the principles first laid down by Charles Darwin in terms of the survival of the fittest. As such, they represent an intelligent exploitation of a random search within a defined search space to solve a problem. In general, the process of GA is as follows.

At first, GA generates the initial population randomly. In GA, population means a set of solutions, and each solution is called a chromosome. A chromosome has a form of binary strings in usual and all the parameters to be found are encoded on it. After generating the initial population, GA computes the fitness function of each chromosome. The fitness function is a user-defined function which returns the evaluation results of each chromosome, thus a higher fitness value means its chromosome is a dominant gene. According to the fitness values, offspring are generated by applying genetic operators. In general, three operators are frequently used—reproduction, crossover, and mutation. Applying these genetic operators and generating new generations of the population are repeated over and over until the stopping criteria are satisfied. In most cases, the stopping criterion is set to the maximum number of generations

B. Goal of the Optimization

Even after considering the effect of different misclassification costs, the performance of CSSVM is not good enough for imbalanced data set. The reason is the CSSVM model adopts a fixed misclassification cost value, other than combing the application domain background knowledge. In order to make a dynamic misclassification cost mechanism, the GA method is introduced into the CSSVM. The purpose is to find an ideal choice for each sample data set, which means to improve the prediction accuracy for minority class samples and meanwhile as little as possible at the expense of misclassification for the majority class samples.

C. Major Operations

1) Coding

Typically, the GA first need to be binary coded. Let's set x domain of the aim function to [m2, m1], solving accuracy to ε, then the code length L can be calculated as (5).

$$\frac{\mathrm{m2}-\mathrm{m1}}{2^L} \le \varepsilon \text{ 或 } \mathrm{L} \ge log_2 \frac{m2-m1}{\varepsilon} \qquad (5)$$

2) Fitness Function

Construction of the fitness function is the core issue of the GA method. When using GA to solve the problem, a appropriate fitness function should be selected for individuals. In the issue of software defect prediction

In order to achieve a better software defect prediction for the imbalance class samples, the fitness function should take both the minority and majority samples into consideration. The author proposed a novel method as (6).

$$\mathrm{fitness}(x) = \sqrt{F1_r(x) \cdot F1_m(x)}$$

where,

$$F1 = \frac{2 \times \Pr(x) \times Re(x)}{\Pr(x) + Re(x)} \qquad (6)$$

Where, P_r represents Precision (accuracy), R_e represents Recall (response rate), r represents the minority class, m represents the majority class, F1 [8] is the combination of P_r and R_e.

P_r estimates the probability that a software defect really belongs to the category C_i which the classifier assigns to it. However, R_e indicates the probability that a software defect is correctly assigned to the category C_i which it actually belongs to. Moreover, P_r and R_e are generally combined into one single metric called F1. F1 is defined to be a harmonic mean of P_r and R_e. The higher F1 of classifier indicates better performances. The fitness function value equals to the geometric mean of the F1 value of minority class and that of majority class.

3) Reproduction, Crossover and Mutation

By the reproduction operator, solutions with higher fitness values are reproduced with a higher probability. Here, we choose the roulette wheel selection method and the selection probability is calculated as (7).

$$P_i = \frac{fitness(x_i)}{\sum_{j=1}^{m} fitness(x_j)} \qquad \mathrm{i} = 1,2,\ldots\ldots,\mathrm{m} \qquad (7)$$

Crossover means exchanging substrings from pairs of chromosomes to form new pairs of chromosomes. The single point crossover, which separates chromosomes into two substrings, and the double point crossover, which separates them into three substrings, are the most popular crossover methods. Mutation involves generating mutations of the chromosomes. Mutation prevents the search process from falling into local maxima, but a mutation rate that is too high may cause great fluctuation. So, the mutation rate is generally set to a low value. Here, the crossover probability Pc is set to 0.9, while the chromosome mutation probability Pm is 0.1.

D. Optimization Process

The optimization process of CSSVM using GA is shown below.

a) Generating a random initial population with m chromosomes generates POP (1), so that t = 1, x = 0 (where, m is the population size, t is the evolution of the current generation).

b) Applying each individual to the data set D, then the CSSVM classifier is constructed to get Re and Pr value after 10-fold cross-validation. The fitness value could be calculated using (6).

c) Calculating the selection probability using (7) for each chromosome and constituting a new population NewPOP (t +1) through roulette wheel selection method.

d) Using the crossover and mutation operations to form a new population POP (t) where ordering t = t +1

e) If the the stopping criterion is reached (to meet a predefined accuracy parameter ε or breeding more than 200 generations), the optimized result is the largest individual fitness value in the POP (t).

f) Applying the optimized result to the CSSVM model, thus an optimal classifier M is constructed for the software defect prediction.

V. Experiments and Results

A. Environment and Data set

The experiment condition contains a PC with Intel(R) Core(TM) i5-3450 3.10 GHZ cpu and 4.0GB memory, using Windows XP as OS. All experiments have been implemented based on open source software LIBSVM [9].

The data sets used in the experiments come from NASA's MDP project [10], which contains a series of real software defect data from NASA spacecraft software. Parts of the data are shown in Table I.

TABLE I. Data Sets of Software Defects in MDP of NASA

Name	Number of Features	Number of Samples	Positive /Negative	Deflection rate
CM1	37	505	48/457	0.095
KC1	31	2107	265/1842	0.126
KC3	39	458	43/415	0.094
KC4	14	125	61/64	0.488
MW1	37	403	31/372	0.077
JM1	21	10878	2102/8776	0.193
PC1	37	1107	76/1031	0.069
PC2	37	5589	23/5566	0.004
PC3	37	1563	160/1403	0.102
PC4	37	1458	178/1380	0.122
PC5	38	17186	516/16670	0.030
MC1	38	9466	68/9398	0.007
MC2	39	161	53/109	0.329

B. Evaluation Criteria

As the traditional evaluation criteria such as P_r, R_e and F1 values are not suitable for the imbalanced data set, we choose the ROC and AUC value to evaluate the experiment results.

Receiver operating characteristic curve (ROC) and Area Under the ROC Curve (AUC) indicators can meet the requirement applies to the imbalanced prediction and classification. ROC curve [11] uses FP_{rate} (False positive rate) as the abscissa, TP_{rate} (True positive rate) for the vertical axis. Point D located in the upper left of ROC is obviously the best classification result, where FP_{rate} is 0 and TP_{rate} is 1. That means all the samples are classified correctly. Moving through the threshold, an ROC curve could be achieved from different points.

AUC is the area under the ROC curve to measure the size classification performance of the merits of the indicators, with a better intuitive and comprehensible, which is generally larger than 0.5. Obviously, the ROC curve is more close to the upper left, the AUC value is larger, which means the more positive class samples are classified correctly and the less negative class samples are classified wrongly. Therefore, the AUC value is selected as the evaluation criteria for software defect prediction in this paper.

C. *Results and Comparisons*

The GA-CSSVM method can effectively achieve the best performance for imbalanced software defect prediction, while for different data sets the improvement is not the same either. In order to verify the classification improvement of GA-CSSVM , the standard SVM and CSSVM methods has been realized to compare with it. The results and comparisons are shown in Table II and Fig. 2.

TABLE II. AUC VALUES USING GA-CSSVM IN MDP DATA SETS

	GA-CSSVM	**CSSVM**	**SVM**
KC1	0.721	0.688	0.613
JM1	0.906	0.85	0.825
PC1	0.832	0.809	0.725
PC2	0.802	0.752	0.653
PC3	0.842	0.83	0.782
PC4	0.855	0.82	0.78
PC5	0.93	0.89	0.82
MC1	0.897	0.854	0.74

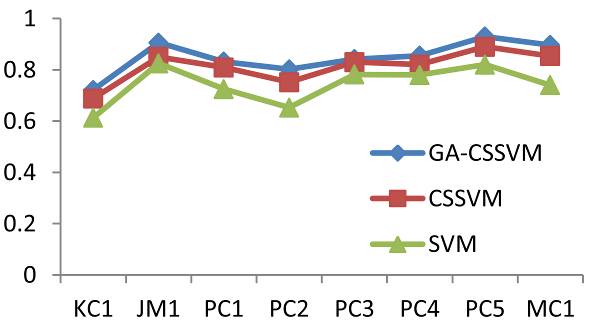

Figure 2. Value of AUC using different prediction classifiers

From the above figure we can draw two conclusions:

a) The proposed GA-CSSVM method shows better performance than the CSSVM and SVM methods on the imbalanced data sets.

b) When the degree of the imbalanced software defect data sets increases, the GA-CSSVM method shows the better improvements. For examples, the result of the GA-CSSVM method on PC2, MC1, PC5 and PC1 is better than that of CSSVM and SVM.

VI. CONCLUSION

For the comprehensive consideration of the imbalanced class, a novel method called GA-CSSVM is proposed using GA theory to improve the cost-sensitive support vector machine. GA-CSSVM belongs to the dynamic SVM method, which using the fitness value to adapt the misclassification cost for both the positive and negative class samples, could deal with the software defect prediction problem well. Experimental results show that the algorithm is feasible and effective. Software defect prediction is generally be treated as a two class classification problem. While in fact, according to the different degree of importance of defects, software can be divided into the high-risk, middle-risk, low-risk and safe modules. Thus how to deal with this multi-classification problem maybe our future works.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under Grant No. 61101073.

REFERENCES

[1] Khoshgoftaar, T.M., Gao, K.and Szabo, R.M., “An Application of Zero-Inflated Poisson Regression for Software Fault Prediction,” Software Reliability Engineering, 2001. ISSRE 2001. Proceedings. 12th International Symposium on, 27-30 Nov. 2001, pp. 66 -73.

[2] Munson, J. and Khoshgoftaar, T., “Regression Modeling of Software Quality: An Empirical Investigation,” Information and Software Technology, vol. 32, 1990, pp. 106 - 114.

[3] Pickard, L., Kitchenham, B. and Linkman, S., “An Investigation of Analysis Techniques for Software Datasets,” Software Metrics Symposium, 1999. Proceedings. Sixth International, 4-6 Nov. 1999, pp. 130 -142.

[4] Wang, Y, “A New Approach for fitting Linear Models in high-dimensional Spaces,” PhD Thesis, 2000, Department of Computer Science, University of Waikato, New Zealand.

[5] C Cortes, V Vapnik, “Support-Vector Networks,” Machine learning, no. 20, 1995, pp. 273-297.

[6] Veropoulos K, Campbell C,Cristianini N. “Controlling the sensitivity of support vector machines,” In Proceedings of International Joint Conference on Artificial intelligence, Sweden, 1999, pp. 55-60.

[7] David E. Goldberg, John H. Holland, “Genetic Algorithms and Machine Learning,” Machine Learning, vol 3, pp. 95-99.

[8] K.Van Rijsbergen, “Information Retrieval,” Butterworths, London, 1979.

[9] Chih-Chung Chang, Chih-Jen Lin, “LIBSVM: A library for support vector machines,” ACM Transactions on Intelligent Systems and Technology, vol. 2, no. 3, April 2011.

[10] M. Chapman, P. Callis, Metrics data program, NASA IV and V Facility, http://mdp.ivv.nasa.gov/, 2004.

[11] K. Fukunaga. “Introduction to statistical pattern recognition,” 2nd Edition. California:Academic Press, 1990.

[12] Liu QJ, Jing LH, Wang MF, Lin QZ, “Hyperspectral remote sensing image classification based on SVM optimized by clonal selection,” Chinese Academy of Sciences, 2013, vol. 33, pp. 746-751.

[13] Yuxin Wang, Wei Liu; He Guo, “An Extensible Characteristic Scenarios Simulator for Virtual Machine,” Chinagrid Conference (ChinaGrid), 2011 Sixth Annual.

[14] Peng Cao, Dazhe Zhao, Osmar Zaiane, “An Optimized Cost-Sensitive SVM for Imbalanced Data Learning,” Lecture Notes in Computer Science vol. 7819, 2013, pp. 280-292.