SOFTWARE DEFECT PREDICTION MODEL BASED ON LLE AND SVM

***Chun Shan*[1], *Boyang Chen*[1], *Changzhen Hu*[1], *Jingfeng Xue*[1], *Ning Li*[2]**

[1]*School of software, Beijing Institute of Technology, Beijing100081, China*
[2]*15th Research Institute of China Electronics Technology Group Corporation, Beijing 100083, China*
sherryshan@bit.edu.cn, chenboyang0709@163.com, chzhoo@bit.edu.cn , xuejf@bit.edu.cn, ning.li@263.net

Keywords: Software Security, Software Defect Prediction, Support Vector Machine, Locally Linear Embedding

Abstract

Software defect prediction strives to improve software security by helping testers locate the software defects accurately. The data redundancy caused by the overmuch attributes in defects data set will make the prediction accuracy decrease. A model based on locally linear embedding and support vector machine (LLE-SVM) is proposed to solve this problem in this paper. The SVM is used as the basic classifier in the model. And the LLE algorithm is used to solve data redundancy due to its ability of maintaining local geometry. The parameters in SVM are optimized by the method of ten-fold cross validation and grid search. The comparison between LLE-SVM model and SVM model was experimentally verified on the same NASA defect data set. The results indicate that the proposal LLE-SVM model performs better than SVM model, and it is available to avoid the accuracy decrease caused by the data redundancy.

1. Introduction

With the high-speed progress of computer science and software technology, the process of informationization has been rapidly developed. Software systems are playing an increasingly important role in the socio-economic and political life. Software defects are increasing with software function variety and software structure complexity. Software defects may cause software failure, which do serious harm to software security [1]. So it is necessary to study software defect prediction technology in this background. About 80% of the defects come from 20% of the modules, and more than half of the modules are defects free [2]. It has very important significance that software testers find out those modules which contain defects and this is the goal of defect prediction. Software defect prediction technology can be classified into two kinds. They are static defect prediction technology and dynamic defect prediction technology. The static technology is based on the metric data of defects, which is used to predict the quantity or distribution of defects; and the dynamic technology is based on the generate time of defects or failures, which is used to predict the defects over time [1].

There are many outstanding techniques appeared in the dynamic defect prediction technology. The software defect prediction model based on grey prediction theory (GPT) was mentioned in [3], the idea of the method was: the number of software defects was a grey value changed within a certain range, and the random process was a time related gray process changed in a certain range. The method using GPT predicted software defects which might appear in the iteration according to the historical statistics of software defect number [3]. An improved model of software defect prediction based on Rayleigh model was mentioned in [4]. The model corrected the unreasonable hypothesis of Rayleigh model considering the effect of removing failure on defect prediction result [4].

There are a lot of static defect prediction technologies such as support vector machine (SVM) [5], neural network (NN) [6], and Bayesian network (BN) [7]. The advanced software defect prediction model based on SVM was mentioned in [8]. The defect prediction was regarded as a binary classification problem, and SVM model was used to predict the current version of the project. New defect information found was added to the history data and then new incremental data sets would be produced. A kind of improved prediction model would be constructed based on the incremental data sets in the new project life cycle [8]. The technology involved in this paper is static prediction techniques.

Software defect prediction is a kind of technology which can reveal the possibility whether a software system contains defects by analyzing the metric data of software. More and more attributes are being introduced to metric software with the developing of technology. Facing the growing number of attributes, one of the problems which must be solved in software defect prediction area is how to deal with the data redundancy. This problem may lead to higher cost and lower prediction accuracy. For this reason a software defect prediction model based on locally linear embedding and support vector machine is proposed in this paper. Locally linear embedding algorithm is used to reduce dimensionalities and maintain the local geometry of the metric data set, and support vector machine is proposed to classify the data set of software defects.

2. Related Research

Support vector machine is one of the most common methods of software defect prediction, and it is a machine learning method based on statistical learning theory and structural risk minimization [9]. SVM method is used to minimize the structural risk and improve the learning ability of generalization.

Besides support vector machine, there are some other models can be used in defect prediction research like classification and regression tree (CART) [10] and artificial neural network (ANN) [11]. However the classification and regression tree has a poor ability on generalization; and there is no unified and complete guidance on the selection of the network structure of artificial neural network [12]. Support vector machine is an outstanding method which is good at dealing with small samples and nonlinear problems [13]. And the data sets of software defects are just conformed to support vector machine. Considering the issues above, support vector machine was chosen to be the basic model for software defect prediction in this paper.

Just like what was described in the section of introduction, there are more and more attributes being introduced to software metric, and the metric attributes maybe come from different metric systems or different metric methods. Those certainly lead to data redundancy, and data redundancy can cause the prediction accuracy decrease. It is strongly necessary to solve the problem of data redundancy and improve the prediction accuracy of support vector machine.

Locally linear embedding algorithm is suitable for solving the problem of data redundancy. The algorithm can maintain the local geometry of data. Some of the data contents will be lost during attributes reduction. This problem will affect the accuracy of SVM. The characteristic of LLE can be used to avoid the loss [14]. The locally linear embedding algorithm provides an outstanding solution on dealing with software defect data set by its ability to maintain the data local geometry. The LLE algorithm is used to reduce the dimensionalities of software defect data set in this paper. The comparison experiments have been done on the same NASA defect data set between proposed LLE-SVM model and SVM model.

3. Model Construction of LLE-SVM

3.1 Summary of Model Process

LLE-SVM software defect prediction model contains 5 steps:

1. Get the software defect data set;
2. Select the train set in the defect data set, and reduce the dimensionalities using LLE. The new data set will be the input of the third step.
3. Support vector machine with the Gaussian kernel (RBF) is popular for practical use [15]. RBF is chosen in this step. Optimize the parameters with ten-fold cross validation and grid search in the defined-size of interval and step, and then output the best couple of parameters.
4. Test the model with the best parameters.
5. Predict software defects.

The LLE-SVM software defect prediction model is shown in the following Figure 1.

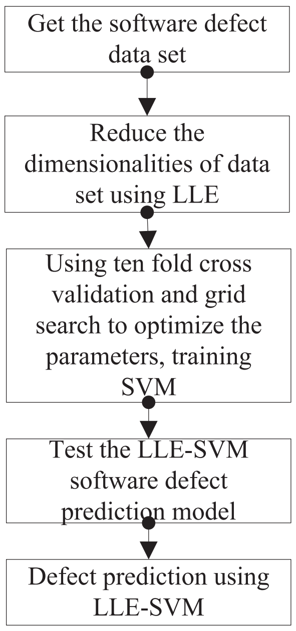
Get the software defect data set
Reduce the dimensionalities of data set using LLE
Using ten fold cross validation and grid search to optimize the parameters, training SVM
Test the LLE-SVM software defect prediction model
Defect prediction using LLE-SVM

Figure 1. Process of LLE-SVM Software Defect Prediction Model

LLE and SVM are the two main parts of the model. The exhaustive principle of the two parts will be introduced in the following Section 3.2

3.2 Principle of LLE and SVM

It has been described in Section 2 that the proposed model reduces the dimensionalities using LLE. The locally linear embedding algorithm belongs to a group of manifold learning methods that not only reduce data dimensionalities, but also attempt to discover a true low dimensional structure of the data [16]. The input of LLE algorithm are *N* points X_i, where $X_i \in R^D, i \in [1, N]$. And as an output, it gives *N* points Y_i, where $Y_i \in R^d, i \in [1, N]$, and $d<D$. The specific steps of LLE algorithm on the dimension reduction is:

1. Find the *k* nearest neighbors for each point. The formula is :

$$d_{ij} = \| \boldsymbol{X}_i - \boldsymbol{X}_j \| \tag{1}$$

2. Measure reconstruction error resulting from the approximation of each point by its nearest neighbors and compute reconstruction weights minimizing this error. That is to solve the optimization problem in formula 2.

$$\begin{cases} \min \varepsilon(\mathrm{w}) = \sum_{i=1}^{N} \| \boldsymbol{X}_i - \sum_{j=1}^{K} w_{ij} \boldsymbol{X}_{ij} \|^2 \\ s.t \sum_{i=1}^{N} w_{ij} = 1 \end{cases} \quad (2)$$

N is the number of points, w_{ij} is one of the coefficients of point I when using point j for representation, w_{ij} =0 if point j is not a nearest neighbor, all the coefficients constitute the partial reconstruction weigh matrix.

3. Compute low-dimensional embedding which can best preserve the local geometry represented by the reconstruction weighs.

$$\begin{cases} \min \varepsilon(\mathrm{w}) = \sum_{i=1}^{N} \| \boldsymbol{Y}_j - \sum_{j=1}^{K} w_{ij} \boldsymbol{Y}_{ij} \|^2 = \min\left(\boldsymbol{YMY}^T\right) \\ s.t : \boldsymbol{YY}^T = \boldsymbol{I} \end{cases} \quad (3)$$

$$\boldsymbol{M} = \left(\boldsymbol{I} - \boldsymbol{W}\right)^T \left(\boldsymbol{I} - \boldsymbol{W}\right) \quad (4)$$

The second to the (d+1)th feature vectors of M are the final result.

LLE-SVM model is based on SVM. The data set after dimensionalities reduction will be input to SVM for training. The process of SVM is to find a hyper plane which can classify the data points in the point space. In the area of software defect prediction, each piece of data represents a software module, and each piece of data has a label at the end of them. $y_i \in \{1, -1\}$, 1 means there are some defects in this module and -1 means no defect is in the module. The SVM process is to solve the following optimization problem like formula 5.

$$\begin{cases} \min\{\frac{1}{2} \| \boldsymbol{\omega} \|^2 + C \sum_{i=1}^{n} \xi_i \} \\ s.t.\, \mathrm{y}_i (\boldsymbol{\omega}^T \phi(\boldsymbol{x}_i) + \mathrm{b}) \geq 1 - \xi_i, i = 1, ..., \mathrm{n} \end{cases} , \xi_i \geq 0 \quad (5)$$

Constant C is the penalty factor, it represents the emphasis of the loss of outliers in training process. And ξ_i is a slack variable, only outliers have slack variables. Notation $\phi(\mathrm{x})$ represents the kernel of SVM, and we use RBF as the kernel in this paper. The RBF is:

$$K\left(\boldsymbol{x}, \boldsymbol{x}_i\right) = \exp\left\{ -\frac{|\boldsymbol{x} - \boldsymbol{x}_i|^2}{\sigma^2} \right\} \quad (6)$$

We may encounter nonlinear separable situations. The kernel can transform the data into a high dimensional space so that it can achieve linear separable. The final decision function is:

$$f(\mathrm{x}) = \mathrm{sign}(\sum_{i=1}^{n} \lambda_i \boldsymbol{y}_i K(\boldsymbol{x}_i, \boldsymbol{x}) + \mathrm{b}) \quad (7)$$

SVM has two parameters: penalty factor C and the parameter of RBF σ (or g). There are also two parameters in LLE algorithm: the size of neighborhood k, and the embedding dimension d. The method of parameters selection will be introduced in the following Section 3.3.

3.3 Parameters Selection

The methods of ten-fold cross validation and grid search are used to optimize parameters which are the most popular methods for optimizing parameters. The size of interval and step for C and g are set, and each couple of C and g is a grid. Get the average predict accuracy for each grid using K-fold cross validation (K=10 in this paper) then choose the C and g which can reach the highest accuracy as the best parameters.

K-fold cross validation is to divide the data set into K parts, each part will be the validation set and other K-1 parts will be the training set, then we will get K models, calculate the average accuracy as the index of current parameter values.

There are two parameters in LLE algorithm, the size of neighborhood k. And the embedding dimension d. No guidance was given for choosing k, an empirical value that k=12 was chosen in this paper. The intrinsic dimension was estimated using the method of maximum likelihood estimate and the result showed that d=4. An experiment to find the best value was designed in this paper to compare d=4 with other d values in the same experiment conditions.

4. Experiments

4.1 Data Set and Experimental Environment

The data set used in the experiment is MDP data set provided by NASA. It is a representative data set since it has been widely used in the area of software defect prediction. The data set contains 13 packages, each package represents a software system, each piece of data represents a module in the software system, and there is a label at the end of each piece of data (Y or N), Y means the module has defects in it and N means there is no defect in the module. The package of CM1 was used in our experiments. The attributes in this package are from LOC, Halstead and McCabe metric system.

The experimental environment in this paper is Matlab r2010a and libsvm-mat-3.01.

4.2 Evaluation Index

The evaluation indexes of experimental results in this paper are calculated by the statistical values showed in the following Table 1.

Table 1. Index Matrix

Actual value	**Predict value**	
	Error-prone module	Less error-prone module
Error-prone module	True positive case(TP)	False negative case(FN)
Less error-prone module	False positive case(FP)	True negative case(TN)

Calculate the indexes by the following formulas.

Actual positive case: $P = TP + FN$ (8)

Actual negative case: $N = FP + TN$ (9)

Total actual case: $C = P + N$ (10)

Accuracy: $$Accuracy = \frac{TP + TN}{C} \quad (11)$$

Precision: $$Precision = \frac{TP}{TP + FP} \quad (12)$$

Recall: $$Recall = \frac{TP}{P} \quad (13)$$

F-measure: $$F - measure = \frac{2}{(1 / precision) + (1 / recall)} \quad (14)$$

4.3 Experiments and Results

The CM1 data set was used in our experiments. There are two experiments that have been done. One is the comparison experiment of the capability of LLE-SVM under different *d* values, and the other one is the comparison experiment of the capability between LLE-SVM and SVM. The values interval of SVM parameters in the grid search were both set to [$2^{-5}, 2^{5}$], the step was set to 0.01, the normalization interval in the experiment was set to [-1, 1].

1. The value of *d* (embedding dimension) is involved when the LLE algorithm is used to reduce the dimensionalities. We have estimated the intrinsic dimension using the method of maximum likelihood estimate, and the result is *d=4*. There is a rule in locally linear embedding algorithm that: *d<k*, we designed an experiment to compare prediction effect of LLE-SVM under different *d* values. We reduced the dimensionalities with *d*=4, *d*=6, *d*=8, *d*=10 in the experiment one, and the result showed that the best prediction capability appeared when *d*=6, then we added an experiment with the neighbor value of *d=6* (*d*=5, *d=7*). The value of *k* is the size of neighborhood, but no guidance was given how to choose it. We chose an empirical value that *k*=12, and followed the rule of *d<k*. LLE-SVM models with different *d* values were made comparison on the 4 indexes: accuracy, precision, recall and F-measure. The result of experiment one is shown in the following Table 2.

Table 2. Result of Experiment One

	LLE-SVM			
d	**Accuracy**	**Precision**	**Recall**	**F-measure**
4	74.12%	68.63%	83.33%	75.27%
5	76.47%	76.19%	76.19%	76.16%
6	81.18%	82.5%	78.57%	80.49%
7	69.41%	73.53%	60%	66.1%
8	68.24%	70.3%	62.04%	74.21%
10	67.06%	76.9%	47.62%	58.83%

2. We chose the best result of *d* value in experiment one (*d*=6) on behalf of the LLE-SVM model, and used it to compare with SVM model at the same conditions in experiment two. Also, the comparison was on the 4 indexes like experiment one. The final result of the comparison between LLE-SVM model and SVM model is shown in the following Table 3.

Table 3. Result of Experiment Two

	Accuracy	**Precision**	**Recall**	**F-measure**
LLE-SVM	81.18%	82.5%	78.57%	80.49%
SVM	69.41%	68.18%	71.43%	69.77%

The result of experiment one shows the prediction capability of LLE-SVM model with different *d* values. It can be discovered that the LLE-SVM model can reach the best capability when *d*=6, and the best parameters obtained by cross validation is : *C*=13.4543，*g*=18.3792. The conclusion is: the best d value is *d*=6 when the size of neighborhood *k*=12. Table 3 shows the comparison result between LLE-SVM model and SVM model, the conclusion is: LLE-SVM model performs better than SVM model at the same experiment conditions.

The proposed software defect prediction model also has some shortcomings. We chose an empirical value for the neighborhood size that *k*=12, however, it can be further studied whether other values of *k* could reach a better capability. We used the method of interval limited traversal to select the value of *d,* we tried to estimate the intrinsic dimension using the method of maximum likelihood estimate, but the method did not work out the best value of *d*. It can be studied how to estimate the intrinsic dimension and whether the intrinsic dimension is helpful to reach the best result. The time cost of grid search is too high, and the optimization method also needs to be improved.

5. Conclusions

In this paper we propose a software defect prediction model based on locally linear embedding and support vector machine. The idea is using LLE algorithm to reduce dimensionalities and maintain the local geometry of data set at the same time, so that the prediction accuracy will be improved. We designed experiments to prove it and the results show that the LLE-SVM model has a higher capability than SVM model on the 4 indexes in software defect prediction area. This model is an available way to predict software defects for solving the problem of data redundancy. The time cost of grid search method is relatively high yet. And also, it is necessary to find a method by which we can work out the best *d* value and choose a favorable neighborhood size *k*. These drawbacks are the major issues for further research.

Acknowledgment

This work was supported by the Key Project of National Defense Basic Research Program of China under Grant No.B1120132031.

References

[1] Wang, Q., Wu, S.J., and Li M.S.: 'Software defect prediction technology', Journal of software, 2008(19),pp. 1565-1579. (in Chinese)

[2] Shull, F., Basili, V., Boehm, B., Brown, A.W., Costa, P., Lindvall, M., Port, D., Rus, I., Tesoriero, R., and Zelkowitz M.: 'What we have learned about fighting defects'. In *Software Metrics, 2002. Proceedings. Eighth IEEE Symposium on IEEE*. Ottawa,Ont., Canada, June 2002, pp. 249-258.

[3] Li, X.K., and Jin, Y.J.: 'The study of software defect prediction model based on grey prediction theory', Computer applications and software, 2009, 26(3), pp. 101-103. (in Chinese)

[4] Shi, J.F., Yang, X., Qin, W., and Yan, H.Z.: 'Study on the improvement of a software defect prediction model', Journal of Beijing Institute of Technology, 2010, 30(9), pp. 1074-1076. (in Chinese)

[5] Elish, K. O., and Elish, M. O.: 'Predicting defect-prone software modules using support vector machines', Journal of Systems and Software, 2008, 81(5), pp. 649-660.

[6] Zheng, J.: 'Cost-sensitive boosting neural networks for software defect prediction', Expert Systems with Applications, 2010, 37(6), pp. 4537-4543.

[7] Okutan, A., and Yıldız, O.T.: 'Software defect p rediction using Bayesian networks', Empirical So ftware Engineering, 2014, 19(1), pp. 154-181.

[8] Wang, T., Li, W.H., Liu, Z., and Shi, H.B.: 'Sof tware defect prediction model based on support v ector machine', Journal of Northwestern Polytech nical University, 2011, 29(6), pp. 864-869. (in C hinese)

[9] Aydin, I., Karakose, M., and Akin, E. :'A multi-objective artificial immune algorithm for parameter optimization in support vector machine', Applied Soft Computing, 2011, 11(1), pp. 120-129.

[10] Khoshgoftaar, T.M., Allen, E.B., and Deng, J.: 'Using regression trees to classify fault-prone software modules Reliability', IEEE Transactions on, 2002, 51(4), pp. 455-462.

[11] Kanmani, S., Uthariaraj, V.R., Sankaranarayanan, V., Thambidural, P., 'Object-oriented software fault prediction using neural networks', Information and software technology, 2007, 49(5), pp. 483-492.

[12] Jiang, H.Y., Zong, M., and Liu, X.Y.: 'Research of software defect prediction model based on ACO-SVM' ,Chinese Journal of computers, 2011, 34(6), pp. 1148-1154. (in Chinese)

[13] Chen, L.J., Ni, S.H., Xie, C., and Xue, S.W.: 'Study on nonlinear dynamic strategy based on particle swarm optimization algorithm ', Computer Simulation, 2012, 29(10), pp. 122-126. (in Chinese)

[14] Liu, X.D.: 'Soft fault diagnosis of analog circuit based on LLE and SVM'. In *System Simulation Technology & Application 2010*, Changchun, Jilin, China, August 2010, pp. 514-517. (in Chinese)

[15] Keerthi, S. S., and Lin, C. J.: 'Asymptotic behaviors of support vector machines with Gaussian kernel', .Neural computation, 2003, 15(7), pp. 1667-1689.

[16] Kouropteva, O., Okun, O., and Pietikäinen, M.: 'Incremental locally linear embedding', Pattern recognition, 2005, 38(10), pp. 1764-1767.