Empirical Evaluation of an SVM-based Software Reliability Model

Yasutaka Kamei
Nara Institute of Science and Technology
Nara, Japan
yasuta-k@is.naist.jp

Akito Monden
Nara Institute of Science and Technology
Nara, Japan
akito-m@is.naist.jp

Ken-ichi Matsumoto
Nara Institute of Science and Technology
Nara, Japan
matumoto@is.naist.jp

ABSTRACT
Support Vector Machines (SVMs) are known as some of the best learning models for pattern recognition, and an SVM can be used as a software reliability model to predict fault-prone modules from complexity metrics. We experimentally evaluated the prediction performance of an SVM model, comparing it with commonly-used conventional models including linear discriminant analysis, logistic regression, a classification tree, and a neural network. The results revealed that the SVM model exhibited showed the best performance among all the models tested.

Categories and Subject Descriptors
D.2.4 [**Software/Program Verification**]: Reliability; D.2.8 [**Metrics**]: Complexity; I.5.1 [**Models**]: Neural nets, Statistical;

General Terms
Reliability, Experimentation

Keywords
Support vector machine, multivariate analysis, discriminant analysis, prediction stability

1. INTRODUCTION
Identification of fault-prone modules, which may need reworking and/or comprehensive testing, is an important issue in software quality assurance. Various multivariate modeling techniques applicable to fault-prone module prediction have been proposed, including among others linear discriminant analysis [9], logistic regression analysis [8], artificial neural network models [5], and classification trees [7].

To achieve high prediction accuracy, in this paper we employ a Support Vector Machine (SVM) as an alternative modeling technique. An SVM is non-linear learning machine used for two-group classification problems [4]. SVMs have been theoretically proven not to fall into a local minimum in contrast to conventional non-linear models, such as neural networks, which harbor the chance of finding a local minimum in learning. Also, SVMs are robust against noisy samples. As shown in Fig. 1, input vectors are non-linearly transformed and mapped onto a very high-dimension feature space. Then, a separator for two groups is determined in this feature space. SVMs are now used in a wide variety of research areas such as optical character recognition [10].

This paper follows a recent research by Xing, Guo and Lyu, which showed that it is feasible to an SVM as a fault-proneness model, and its prediction performance was better than linear discriminant analysis and classification trees [11]. This paper extends their research by comparing the SVM model used above with other commonly-used models, i.e. logistic regression and neural networks, as well as linear discriminant analysis and classification trees.

In what follows, Section 2 describes the design of our experiment to evaluate the prediction accuracy of the models, and Section 3 provides the results and discussion. Finally, Section 4 concludes the paper with a summary and some future topics.

2. EVALUATION SETTINGS
The goal of the experiment is to evaluate the prediction performance of an SVM-based model, comparing it with linear discriminant analysis, logistic regression analysis, a neural network, and a classification tree.

2.1 Dataset
The target software is an application software, written in COBOL, that has been developed and used in a Japanese software company. Its size is about 300,000 source lines of code (SLOC) and it contains 514 modules, 277 of which are not-fault-prone and 237 that

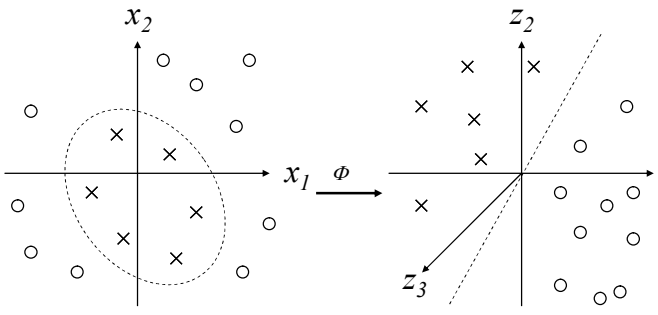

Figure 1. Basic concept of SVM

Table 1. Collected metrics

METRICS	Mean	Median	Std. Dev.
Source lines of code	547.88	405.50	531.15
Commented lines	211.40	165.00	166.51
Commented lines per SLOC	0.59	0.58	0.08
Newly developed SLOC	304.99	218.50	300.78
Reused SLOC	29.69	0.00	247.69
Number of test items	36.018	20.00	60.96
Cyclomatic number	32.31	21.00	38.88
Cyclomatic number per SLOC	3.28	2.49	2.70
The number of Loops	2.98	2.00	3.87
Maximum of nest level	5.80	6.00	1.47
Sum of nest level	3.69	3.63	0.43
Number of jump nodes	4.09	1.00	9.40
Number of declared variables	68.20	33.00	98.47
Number of referred variables	109.01	69.00	153.56
Number of assigned variables	84.98	54.00	123.79
Number of inner calls	12.15	9.00	11.89
Number of external calls	4.24	1.00	7.96
Number of global assigned variables	43.31	25.00	55.99
Number of global referred variables	56.66	36.00	68.37
Number of macros	5.11	4.00	3.26
Number of parameters	4.49	1.00	7.14

are (i.e. contain one or more faults). The value of the objective variable is 0 if the modules are not fault-prone, and is 1 if the modules are fault-prone. These faults were recorded during testing phases (unit, integration and system testing).

Predictor variables are complexity metrics of modules. We measured 21 metrics for each module, with Table 1 showing the statistics of each metric.

2.2 Models

There are the several types of SVM, which depend on the kernel function being used, such as the radial basis function (RBF), the polynomial function, and the sigmoid function. In this paper, the RBF kernel was used since it has an advantage in model selection (variable selection) [2], and its performance was excellent in fault-prone module prediction [11].

Regarding neural network models, five types of three-layer neural network were constructed, having one of the following number of learnings: 10,000; 20,000; 30,000 50,000; 100,000. Also, as a classification tree model, we used the classification and regression trees (CART) algorithm [7] for model construction.

The bootstrap method was applied in this experiment. The dataset (of 514 modules) was randomly divided into two datasets (fit dataset and test dataset). The fit dataset was used for model construction and the test dataset was for model evaluation. This division was repeated 25 times, and stepwise variable selection was applied to all models at each repetition.

2.3 Evaluation Criteria

We used three commonly-used criteria (recall, precision, and the F1-value) to evaluate the prediction performance of constructed models. Recall is the ratio of correctly predicted fault-prone modules to actual fault-prone modules, and precision is the ratio of actual fault-prone modules to the modules predicted as fault-prone. F1-value is a combined value of recall and precision, formally defined in Eq. (1):

$$F_1 = \frac{2 \times Recall \times Precision}{Recall + Precision}. \quad \textbf{(1)}$$

3. RESULTS AND DISCUSSION

Figure 2 shows boxplots of F1-values of five models: linear discriminant analysis (LDA); logistic regression analysis (LRA); a neural network (NN, # of learnings = 30,000, which was the best among the neural network models); a classification tree (CT); and the SVM. Obviously, the SVM performed the best in terms of median value as well as best-case and worst-case values. The F1-values of the SVM ranged from 0.52 to 0.72, while the linear discriminant analysis ranged from 0.46 to 0.64, the logistic regression analysis from 0.45 to 0.61, the neural network from 0.51 to 0.69, and the classification tree from 0.48 to 0.70.

Figure 3 shows recall and Fig. 4 illustrates the precision of the five models. While the SVM showed the best performance in recall (median value was 0.06 to 0.23 higher than in other models), precision was slightly worse (median value was 0.03 lower than the best model, LDA). This indicates that the SVM model has a great advantage to avoid overlooking actual fault-prone modules, with just a small loss in precision.

Interestingly, LRA (logistic regression) showed the worst performance in this experiment; it was even worse, in fact, than LDA (linear model).

NN (neural network) and CT (classification tree) performed better than LRA and LDA in terms of F1-value and recall. However, the precision of CT was the worst among all models.

We consider that the accuracy and the stability of the SVM model derived from using the soft margin [4] since it allows noisy samples in the training dataset, and this contributed to avoiding the over-fitting problem.

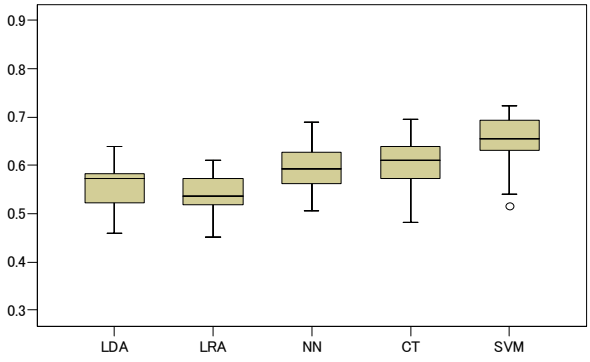

Figure 2. F1 Value of Each Method

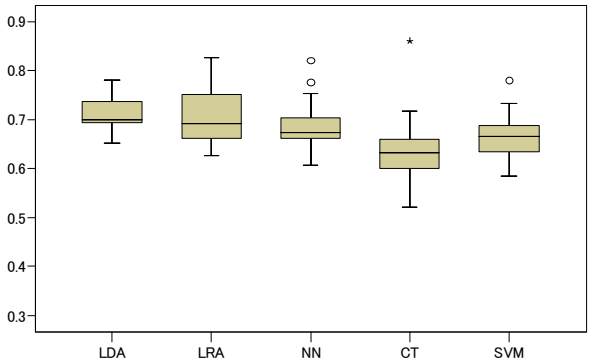

Figure 4. Precision of Each Method

4. SUMMARY

In this paper, we experimentally evaluated the prediction performance of an SVM model and five conventional models by using an actual project dataset. The results indicated that the SVM model exhibited the best performance among all the models tested.

One of our future works will be to evaluate these models with an imbalanced dataset that contains a very low proportion of fault-prone modules. For such an "imbalanced" dataset, we may need to apply the re-sampling method [3] to a fit dataset before constructing SVM models.

5. ACKNOWLEDGMENTS

This work was partially supported by the EASE (Empirical Approach to Software Engineering) project, which is part of the Comprehensive Development of e-Society Foundation Software program of the Ministry of Education, Culture, Sports, Science and Technology of Japan.

6. REFERENCES

[1] Burges, C. J. C. A tutorial on Support Vector Machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2, 2(Jun. 1998), 121–167.

[2] Cai, J., and Li, Y. Classification of Nuclear Receptor Subfamilies with RBF Kernel in Support Vector Machine. *Proc. International Symposium on Neural Networks (ISNN'05)*(Chongqing, China, May 30- Jun. 1, 2005), Springer, New York, NY, 2005, 680-685.

[3] Chawla, V. N, Bowyer, W. K. Hall, O. L., and Kegelmeyer, P.W. SMOTE: Synthetic Minority Over-sampling TEchnique. *Journal of Artificial Intelligence Research*, 16(Jun. 2002), 321-357.

[4] Cortes, C., and Vapnik, V. N. Support Vector Networks. *Machine Learning*, 20, 3(Sep. 1995), 273–297.

[5] Gray, A. R., and MacDonell, S. G. Software metrics data analysis - Exploring the relative performance of some commonly used modeling techniques. *Empirical Software Engineering*, 4, 4(Dec. 1999), 297-316.

[6] Herlocker, J., Konstan, J., Terveen, L., and Riedl, J. Evaluating collaborative filtering recommender systems. *ACM Trans. on Information Systems*, 22, 1(Jan. 2004), 5-53.

[7] Khoshgoftaar, T. M., and Allen, E. B. Modeling software quality with classification trees. *Recent Advances in Reliability and Quality Engineering*, Hoang Pham Editor. World Scientific, Singapore, 1999, 247-270.

[8] Munson, J., and Khoshgoftaar, T. The detection of fault-prone programs. *IEEE Trans. on Software Engineering*, 18, 5(May 1992), 423-433.

[9] Ohlsson, N., and Alberg, H. Predicting fault-prone software modules in Telephone Switches. *IEEE Trans. on Software Engineering*, 22, 12(Dec. 1996), 886-894.

[10] Schölkopf, B., Sung, K., Burges, C.J.C., Girosi, F., Niyogi, P., Poggio, T., and Vapnik, V. N. Comparing Support Vector Machines with Gaussian Kernels to Radial Basis Function Classifiers. *IEEE Trans. on Signal Processing*, 45, 11(Nov. 1997), 2758-2765.

[11] Xing, F., Guo, P., and Lyu, M. R. A novel method for early software quality prediction based on support vector machines. *Proc. Int'l Symposium on Software Reliability Engineering (ISSRE'05)*(Chicago, Illinois, Nov. 8-11, 2005), 2005, 213-222.

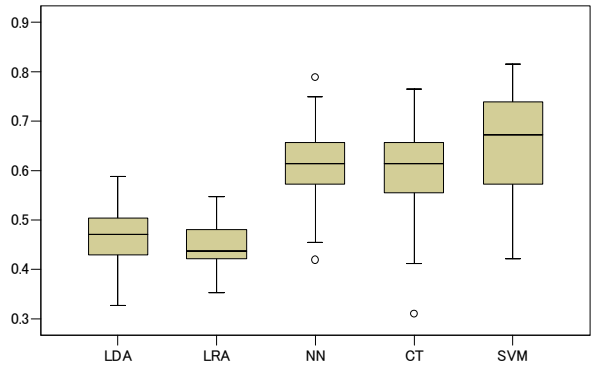

Figure 3. Recall of Each Method