Transactions of the Korean Nuclear Society Autumn Meeting
PyeongChang, Korea, October 30-31, 2008

Estimation of Risky Modules in Safety-Critical Software

Y. M. KIM and C. H. Jeong
Korea Institute of Nuclear Safety
Ymkim@kins.re.kr

1. Introduction

With the rapid development of digital computer and information processing technologies, nuclear I&C (Instrument & Control) system which needs safety-critical function has adopted digital technologies. Software used in safety-critical system must have high dependability. Software testing and V&V activities are very important for assuring high software quality. If we can predict the risky module of safety-critical software, we can focus testing activities and regulation activities more efficiently.

In this paper, we present an estimating model of risky module during early software lifecycle using support vector machines (SVM). SVM is very attractive model for solving both classification and regression problems. SVM has been applied in many applications successfully. We use software matrices and risk level of each module for input variables (feature spaces). We can get these information during requirement phase, design phase and development phase.

This paper is organized as follows. Section 2 describes related research and Section 3 describes architecture for estimating risky modules. Section 4 shows the experiment which includes environment and results. Section 5 concludes the paper.

2. Related Research and Background

2.1 Software Matrices

From now, there have been many researches for searching relationships between software matrices and software faults. There are two categories for estimating software faults using software matrices. One is using statistical techniques and the other is using machine learning. For example, the models which used statistical classification techniques are Discriminant Analysis and Factor Analysis and which used machine learning classification techniques are decision trees, artificial neural networks, support vector machines and etc. Many parts of these researches had treated finding the subset of the software matrices that are most likely to predict the existence of faults. But, relationships between software metrics and fault-proneness are often complex.

2.2 Support Vector Machines (SVM)

Support Vector Machines (SVM) are kernel based learning machines that introduced by Vapnik in 1995[1, 4]. SVMs used structural risk minimization principle (SRM) for minimizing the generalization error and they can generalize the high dimensional feature spaces using small training sample data.

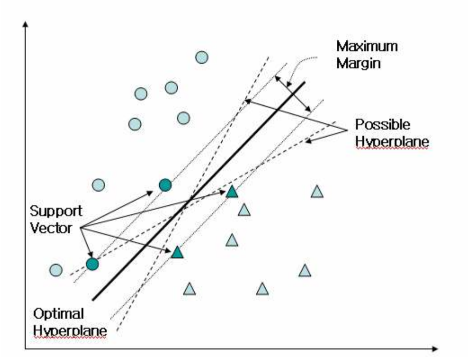

Figure 1 Basic concept of a SVM

Figure 1 shows the linear separation of two classes by SVM in two-dimensional spaces. The hyperplane corresponding to $w \cdot x + b = 0$ is optimal hyperplane. Optimal hyperplane corresponds to the one that minimizes the training error and has the maximal margin. In order to generalize to the case where the input spaces can not separate the two classes properly, a hyperplane is established in high dimensional feature space and the nonlinear classification is replaced by a linear classification problem. If the dimensionality of the new feature space is sufficiently high, the data will always be linearly separable. For supporting nonlinear mapping into feature space, the kernel function is used. The most common kernel functions are linear, polynomial, gaussian, and sigmoid.

3. Architecture

Figure 2 shows the overall architecture for estimating risky module using SVM. First, we must select training datasets according to characteristics of target application.

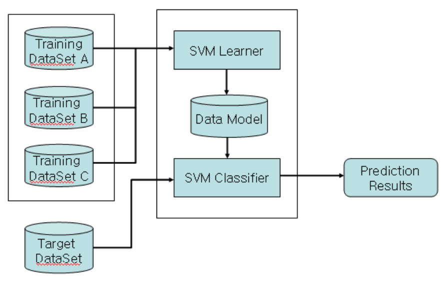

Figure 2 Architecture for estimating risky module using SVM

Characteristics of applications are methodologies, languages, program size, operating environments, and etc. Next, we execute SVM Learner and SVM Classifier using selected training dataset and generated Data Model using target datasets in order. Prediction results are estimated risk classes of each module.

4. Experiment

4.1 Datasets and Experiment Environment

In this study, we used two safety-critical NASA software projects because they can be publicly accessible. We used two datasets that are CM1 and PC1. CM1 is NASA spacecraft instrument software consisting of 20KLOC. PC1 is flight software for an earth orbiting satellite consisting of 40KLOC. Both are written in C language. Each dataset includes the information about the risk level of requirements which is represented from 1 to 3 and also contains the relationship between the requirements and modules. For the experiments, we constructed three test cases which contained different input variables. Input variables in Case 1 are from Elish's research [2], input variables in Case 2 are from Gondra's research [3], and input variables in Case 3 are all software matrices which described in NASA datasets. We added the risk level to the input variables for estimating risk class. We defined the estimated risk class R_m as follows :

$$R_i = Risk\_Level_r + Risk\_Level_r * Error\_Count_m \quad (1)$$

$$R_m = \begin{cases} Risk\_Level_r + 1 & \alpha < R_i \le \beta \\ Risk\_Level_r + 2 & R_i > \beta \\ Risk\_Level_r & otherwise \end{cases} \quad (2)$$

Risk_Level$_r$ is risk level of each module and *Error_Count$_m$* is the number of error which the module has. In this experiment, α is 4 and β is 10. We used that tradeoff value C between the maximization of the margin and minimization of the training error was 5000 and kernel for the SVM was linear. We used *MATLAB* and *SVM^{multiclass}* Version 2.12 [5] to conduct this study.

4.2 Result Analysis

After training each dataset using training data, we classified test dataset. In case CM1 dataset, after 253 training cycles, the classification error rate (zero/one-error) is 22.13%, 21.34% and 35.18% for each test case. In case PC1 dataset, after 554 training cycles, the classification error rate is 15.52%, 16.06% and 26.90% for each test case. The classification error rate of the Case1 and Case 2 are similar, but the Case 3's is a little bit big. Table 1 and Table 2 show the results of the classification. We can compare the distribution of estimated risk classes of test cases and original risk class of the test dataset. The Case 3 has big classification error rate but we can use the classification results more effectively. We can consider that the modules which belong to Risk Class 4 and Risk Class 5 are relatively more risky.

Table 1 Distribution of estimated risk classes (CM1 dataset)

Risk Class	Case 1	Case 2	case 3	Test Dataset (original)
1	245 (78.35%)	252 (99.60%)	200 (79.05%)	201 (79.45%)
2	5 (10.24%)	0	35 (13.83%)	31 (12.25%)
3	1 (0.40%)	0	5 (1.97%)	11 (4.35%)
4	1 (0.40%)	1 (0.40%)	13 (5.14%)	6 (2.37%)
5	1 (0.40%)	0	0	3 (1.19%)

Table 2 Distribution of estimated risk classes (PC1 dataset)

Risk Class	Case 1	Case 2	case 3	Test Dataset (original)
1	552 (99.64%)	547 (98.74%)	420 (75.81%)	470 (84.99)
2	0 (0 %)	0 (0 %)	97 (17.51%)	41 (7.41)
3	2 (0.36%)	3 (0.54%)	26 (4.69%)	33 (5.97)
4	0 (0 %)	4 (0.72%)	11 (0.99%)	8 (1.45)
5	0 (0 %)	0 (0 %)	0 (0 %)	2 (0.36)

5. Summary

In this paper, we classified the estimated risk class which can be used for deep testing and V&V. We predicted the risk class for each module using support vector machines. We can consider that the modules classified to risk class 5 and 4 are more risky than others relatively. For all classification error rates, we expect that the results can be useful and practical for software testing, V&V, and activities for regulatory reviews. Future, for improving the practicality, we will have to investigate other machine learning algorithms and datasets.

REFERENCES

[1] Burges, C., A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998
[2] Karim O. Elish, Mahmoud O. Elish, Predicting defect-prone software modules using support vector machines, The Journal of Systems and Software 81(2008) 649-660
[3] I. Gondra, Applying machine learning to software fault-proness prediction, The Journal of Systems and Software 81 (2008) 186-195
[4] Steve R, Gunn, Support Vector Machines for Classification and Regression, Technical Report, University of Southampton, 10 May 1998.
[5] http://svmlight.joachims.org/svm_multiclass.html