Transactions of the Korean Nuclear Society Spring Meeting
Jeju, Korea, May 22, 2009

Using Machine Learning for Risky Module Estimation of Safety-Critical Software

Young-Mi Kim and Choong-Heui Jeong

Korea Institute of Nuclear Safety, P.O.Box 114, Yuseong-gu, Daejeon, Korea, 305-600
**Corresponding author: 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. Highly dependable software needs strict software testing and V&V activities.

These days, regulatory demands for nuclear power plants are more and more increasing. But, human resources and time for regulation are limited. So, early software risky module prediction is very useful for software testing and regulation activities.

Early estimation can be built from a collection of internal metrics during early development phase. Internal metrics are measures of a product derived from assessment of the product itself, and external metrics are measures of a product derived from assessment of the behavior of the systems [1]. Internal metrics can be collected more easily and early than external metrics. In addition, internal metrics can be useful for estimating fault-prone software modules using machine learning [1, 2, 3].

In this paper, we introduce current research status and techniques related to estimating risky software module using machine learning techniques. Section 2 describes the overview of the estimation model using machine learning and section 3 describes processes of the estimation model. Section 4 describes several estimation models using machine leanings. Section 5 concludes the paper.

2. Overview of the Estimation Model

The area of machine learning focuses on the study of models that improve their performance at some task automatically through experience [4, 5]. Using machine learning techniques, we can use the methods which are to learn the relationship between certain observable features of a metrics and the estimation from software metrics data during early software development phases. Using machine learning, a regression function is called a model, the input variables are called features, the output variable is called target, and a pair consisting of a feature vector and the corresponding target value is called a pattern [6]. Some patterns are used for training for iteratively learning the regression function; other patterns are used for testing the performance of the regression function. Fig. 1 shows the basic steps of our estimation model.

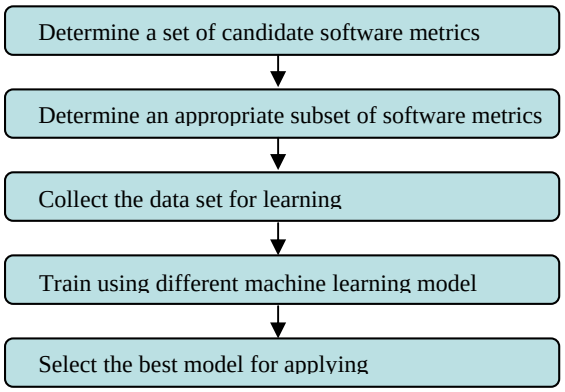

Fig. 1. Basic Steps of the Estimation Model

3. Processes of the Estimation Model

3.1 Determine a Set of Candidate Software Metrics

For the estimation of fault-prone software module, we can use software metrics. There are many kinds of software metrics. The followings are some of representative metrics used by NASA software testing project: number of blank lines, branch count, number of calls to other functions, number of conditionals, total number of lines, the number of lines of comments, the cyclomatic complexity, the cyclomatic density, number of decision points, the decision density, the design complexity, the halstead length, the halstead difficulty, and the halstead programming time.

3.2 Determine an Appropriate Subset of Software Metrics

There have been many researches for searching relationships between software metrics and software faults. There are two categories for estimating software faults using software metrics. One is using statistical techniques and the other is using machine learning. The models which using statistical classification techniques are Discriminant Analysis

and Factor Analysis and which the ones using machine learning classification techniques are decision trees, artificial neural networks, support vector machines, etc. Many parts of these researches have treated finding the subset of the software metrics that are most likely to predict the existence of faults. But, relationships between software metrics and fault-proneness are often complex [6, 7].

3.3 Collect the Data Set for Learning

We should get data sets for training. The larger training data set is available, the more information can be used as input variable for risky module estimating. But, it is very difficult to obtain the failure data set especially in the case of safety-critical software. Also, data sets of the safety critical software used in a different application or same software used in a different environment may be less meaningful for estimation. This is one choice to get the data sets from the application which has similar characteristics. Characteristics of applications are methodologies, languages, program size, operating environment, and etc.

3.4 Train using Different Machine Learning Model

We can use several different machine learning models to estimating risky software modules. Some of these are artificial neural network learning and support vector machine. Training a machine learning models is a nonlinear optimization process which intends fitting the regression function to the training data.

3.5 Select Best Model for Applying

The performance of the prediction models for separation between the two classes (e. g. risky or non-risky) can be evaluated using a confusion matrix. Accuracy, precision, recall and F-measure which can be derived from the confusion matrix also can be used prediction performance measures. In the case of multi-classes, zero/one error can be used for measure.

4. Selection of Learning Models

Representative machine learning techniques for estimating risky software modules are Artificial Neural Networks (ANN) and Support Vector Machines (SVM).

4.1 Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANNs) provide a general, practical method for learning real-valued, discrete-valued, and vector-valued function from examples. ANN learning is robust to errors in the training data and has been successfully applied to problems such as interpreting visual sense, speech recognition, and learning robot control strategies [8].

4.2 Support Vector Machines (SVMs)

Support Vector Machines (SVMs) are kernel based learning machines that introduced by Vapnik in 1995 [6, 7]. 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. The hyperplane corresponding to $w \cdot x + b = 0$ is optimal hyperplane. The hyperplanes corresponding to $w \cdot x + b = \pm 1$ are the bounding hyperplanes. 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.

5. Summary

High dependability software usually requires high cost for software testing, verification and validation. Also, regulatory activities are very expensive to be exhaustively performed. More and more the safety critical system of NPP adopted digital technologies, more and more the regulatory activities are increased. If we can estimate the risky software module in early development phases, their costs can be reduced and regulatory activities can be focused on the more risky modules relatively. Future, we will experiment using various machine learning algorithms and consider their applicability to the safety-critical system.

REFERENCES

[1] Nachiappan Nagappan, *Toward a Software Testing and Reliability Early Warning Metric Suite*, ICSE'04, 2004.

[2] P. Bellini, I. Bruno, P. Nesi, D. Rogai, *Comparing Fault-Proneness Estimation Models*, IEEE, 2005.

[3] Fei, Ping Guo, and Michael R. Lyu, *A Novel Method for Early Software Quality Prediction Based on Support Vector Machine*, ISSRE'05, 2005

[4] Bo Yang, Lan Yao, Hong-Zhong Huang, *Early Software Quality Prediction Based on a Fuzzy Neural Network Model*, ICNC 2007, 2007

[5] Frank Padberg, Thomas Ragg, and Ralf Schoknecht, *Using Machine Learning for Estimating the Defect Content After an Inspection*, IEEE Transaction on Software Engineering, 2004

[6] E.O. Elish, M.O. Elish, *Predicting defect-prone software modules using support vector machines*, The Journal of Systems and Software, 2008

[7] Y. M. Kim and C. H. Jeong, *Estimation of Risky Modules in Safety-Critical Software*, KNS, Fall, 2008

[8] Tom M. Mitchell, *Machine Learning*, McGraw-Hill, 1997