Research of Software Failure Prediction Based on Support Vector Regression

Zheng Qiuhong
College of Computer Science and Information Technology
Zhejiang Wanli University
Ningbo, China

Abstract—**Software failure prediction is currently a hot subject of research all over the world. The support vector regressions (SVRs) are very efficiency for solving regression problems. The parameters just as C 、ε 、γ performs very important roles in the generalization of SVR, and it's hard for beginner to choose them. But in formar models, they diden't care about this problem.A SVR-based generic model adaptive to the characteristic of the given data set is used for software failure time prediction. We also compare the prediction accuracy of software reliability prediction models based on 1-norm SVM, 2-norm SVM, v- SVM and artificial neural network (ANN). Experimental results by four data sets show that the new software reliability prediction model could achieve higher prediction accuracy than that of the ANN-based or SVM-based models.**

Keywords- Software Reliability Prediction, Support Vector Regression, Artificial Neural Network

I. INTRODUCTION

In modern society, computers are used for many different applications, such as nuclear reactors, aircraft, banking systems, and hospital patient monitoring systems. As the demand of the application quality becomes higher and higher, the research of the computer software reliability becomes more and more essential. The software reliability is defined as the probability that the software will operate without a failure under a given environmental condition during a specified period of time [1]. To date, the software reliability model is one of the most important tools in software reliability assessment.

Most of the existing software reliability models [2-5] depend on a priori assumptions about the nature of software faults and the stochastic behavior of software failure process. As a result, each model has a different predictive performance across various projects. To overcome this problem, several alternative solutions are introduced. One possible solution is to employ the ANNs [6- 11]. ANNs have many advantages that account for its popularity in data mining and analysis. For example, ANNs can learn nonlinear mapping between the input and output of a system/process, and it has been theoretically proven that ANNs can approximate nonlinear functions to arbitrary accuracy. In this case, the focus of the training process is model fitting and tends to cause over fitting. The error on the training data set is driven to a very small value for known data, but when out-of-sample data is presented to the network, the error is unpredictably large, which yields limited generalization capability.

As an alternative, a novel type of learning machine, SVM, has been receiving increasingly attention in areas ranging from its original application in pattern recognition to the extended application of regression estimation. This was brought about by the remarkable characteristics of SVM such as good generalization performance, absence of local minima, and sparse representation of solution. SVM was developed by Vapnik [12] and it is based on the SRM principle which seeks to minimize an upper bound of the generalization error consisting of the sum of the training error and a confidence interval. This induction principle is different from the commonly used ERM principle which only minimizes the training error. Established on the unique principle, SVM usually achieves higher generalization performance than traditional neural networks that implement the ERM principle in solving many machine learning problems. Another key characteristic of SVM is that training SVM is equivalent to solving a linearly constrained quadratic programming problem so that the solution of SVM is always unique and globally optimal, unlike other networks' training which requires nonlinear optimization with the danger of getting stuck into local minima. In software reliability prediction domain, there have been some studies on building SVM-based SRPMs as well. Tian and Noore [13] proposed an SVM-based model for software reliability prediction. Pai and Hong [14] also made their contributions.

However, despite its success, we can identify a number of significant disadvantages of the SVM reliability model. The parameters just as C 、ε 、γ performs very important roles in the generalization of SVM, and it's hard for beginner to choose them. In Tian and Yang's models, they diden't care about this problem. Scholkopf[15] proposed a new type of SVM which can choose the parameters automatically. In this paper, we use 1-norm SVR, 2-norm SVR，v- SVR to model software reliability and gets better performance than previous work.

II. SUPPORT VECTOR REGRESSION [12]

We have already discussed the problem of learning a real-valued function ，the 1-dimensional output of a real-valued function can be seen as a special case. The term regression is generally used to refer to such real-valued learning. We will begin this section by giving a fuller description of support vector regression methods.

Assuming that a total of n pairs of training patterns are given during SVR learning process,

$$(x_1,t_1),(x_2,t_2),\cdots,(x_i,t_i),\cdots,(x_N,t_N)$$

Where the inputs are n-dimensional vectors $x_i \in R^n$ and the target outputs is continuous value $t_i \in R$. The RVM model used for function approximation is:

$$t = y(x;w) = \sum_{i=1}^{M} w_i k(x,x_i) + w_0$$

These $\{w_i\}$ are the parameters of the model, generally called weights, and $K(,)$ is the kernel function which is the inner product of two vectors in feature space $\Phi(x)$ and $\Phi(x_i)$. By introducing the kernel function, we can deal with the feature spaces of arbitrary dimensionality without computing the mapping relationship $\Phi(x)$ explicitly. Some commonly used kernel functions are polynomial kernel function and Gaussian kernel function. In this paper, we make the choice to utilize Gaussian data-centre basis functions:

$$K(T,T_m) = \exp\{-\frac{\|T-T_m\|^2}{\delta^2}\}$$

Where $\delta > 0$ is a constant that defines the kernel width.

The "empirical risk" $R_{emp}[f]$ is defined to be just the measured mean error rate on the training set.

$$R = \frac{1}{2}\|w\|^2 + R_{emp}[f] = \frac{1}{2}\|w\|^2 + C\frac{1}{l}\sum_{i=1}^{l}|T_i - f(t_i)|_{\varepsilon}$$

其中， $R_{emp}[f]$ 为 where $C(\cdot)$ is lose function，

$$|T_i - f(t_i)|_{\varepsilon} = \begin{cases} 0 & if\ |T_i - f(t_i)| \le \varepsilon, \\ |T_i - f(t_i)| - \varepsilon & otherwise \end{cases}。$$

It's the same to solve the following optimization problem:

$$\min J = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i^* + \xi_i)$$

$$s.t.\begin{cases} (\omega,\phi(T_i)) + b - t_i \le \varepsilon + \xi_i) \\ t_i - (\omega,\phi(T_i)) - b \le \varepsilon + \xi_i^*) \\ \xi_i, \xi_i^* \ge 0 \end{cases}$$

A. Norm SVR [15]

The linear ε-insensitive loss for support vector regression raises the question of what stability analysis is appropriate. A straightforward rewriting of the optimization problem that minimizes the linear loss is as follows:

Process find α^* to the optimization problem:

$$W(\alpha) = \sum_{i=1}^{l} y_i\alpha_i - \varepsilon\sum_{i=1}^{l}|\alpha_i|$$

max α

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

subject to $\sum_{i=1}^{l}\alpha_i = 0, -C \le \alpha_i \le C, i = 1,2,\cdots,l$

$$w = \sum_{i=1}^{l}\alpha_j^*\phi(x_j)$$

$$b^* = -\varepsilon - (\alpha_j^*/C) + y_i - \sum_{i=1}^{l}\alpha_j^* k(x_i,x_j)$$

for i with $0 < \alpha_i^*$

$$f(x) = \sum_{i=1}^{l}\alpha_j^* k(x_j,x) + b^*$$

B. Norm SVR [15]

We can optimize the sum of the quadratic ε-insensitive losses again subject to the constraint that the norm is bounded. This can be cast as an optimization problem by introducing separate slack variables for the case where the output is too small and the output is too large. Rather than have a separate constraint for the norm of the weight vector we introduce the norm into the objective function together with a parameter C to measure the trade-off between the norm and losses. This leads to the following computation. The weight vector w and threshold b for the quadratic ε-insensitive support vector regression are chosen to optimize the following problem:

Process find α^* to the optimization problem:

$$W(\alpha) = \sum_{i=1}^{l} y_i\alpha_i - \varepsilon\sum_{i=1}^{l}|\alpha_i|$$

max α

$$-\frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j\ (k(x_i,x_j) + \frac{1}{C}\delta_{ij})$$

subject to

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

$$w = \sum_{i=1}^{l} \alpha_j^* \phi(x_j)$$

$$b^* = -\varepsilon + y_i - \sum_{i=1}^{l} \alpha_j^* k(x_i, x_j)$$

for i with $0 < \alpha_i^* < C$

$$f(x) = \sum_{i=1}^{l} \alpha_j^* k(x_j, x) + b^*$$

ν-SVR [15]

One of the attractive features of the 1-norm support vector machine was the ability to reformulate the problem so that the regularization parameter specifies the fraction of support vectors in the so-called *ν*-support vector machine. The same approach can be adopted here in what is known as *ν*-support vector regression. The reformulation involves the automatic adaptation of the size ε of the tube.

Process find α^* to the optimization problem:

max α
$$W(\alpha) = \sum_{i=1}^{l} y_i \alpha_i - \varepsilon \sum_{i=1}^{l} |\alpha_i| - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j \, k(x_i, x_j)$$

subject to
$$\sum_{i=1}^{l} \alpha_i = 0, \sum_{i=1}^{l} |\alpha_i|$$

$$-C/l \le \alpha_i \le C/l, i = 1, 2, \cdots, l$$

$$w = \sum_{i=1}^{l} \alpha_j^* \phi(x_j)$$

$$b^* = -\varepsilon + y_i - \sum_{i=1}^{l} \alpha_j^* k(x_i, x_j)$$

for i with $0 < \alpha_i^* < C/l$

$$f(x) = \sum_{i=1}^{l} \alpha_j^* k(x_j, x) + b^*$$

III. Formulation of the SVM-predictor

Suppose that we have observed a total number of *n* failures, the constructed model will first be trained with collected data, and then it can be used for prediction purpose. The SVM learning scheme is applied to the failure time data, forcing the network to learn and recognize the inherent internal temporal property of software failure sequence, thus it can be used for prediction purpose. Use the new vector as model input, and the predicted value of can be obtained. Our approach for software reliability prediction can be illustrated as Figure 1.

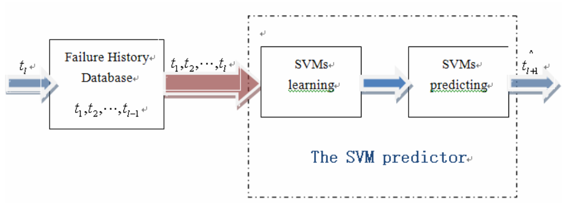

Figure 1. Software reliability model based on SVR

IV. Experimental Results

The performance of our proposed approach is tested using the same real-time control application and flight dynamic application data sets as cited in Park et and Karunanithi et. We choose a common baseline to compare our results with related work cited in the literature. All four data sets used in the experiments are summarized as follows [5]:

- DATA-1: Real-time command and control application consisting of 21,700 assembly instructions and 136
- DATA-2: Flight dynamic application consisting of 10,000 lines of code and 118 failures.
- DATA-3: Flight dynamic application consisting of 22,500 lines of code and 180 failures.
- DATA-4: Flight dynamic application consisting of 38,500 lines of code and 213 failures.

When testing a proposed model, it is necessary to quantify its prediction accuracy in terms of some meaningful measures. The following statistical metrics are used for comparing prediction performance, represented by RE, and AE [5]:

$$RE = \left| \frac{\hat{t}_i - t_i}{t_i} \right|, \quad AE = \frac{1}{n-m} \sum_{i=m+1}^{n} \left| \frac{\hat{t}_i - t_i}{t_i} \right| \times 100$$

Where $\hat{t}_i$ denotes the predicted value of failure time, and t_i the actual value of failure time.

Table I summarizes the results of modeling the temporal inter-relationship among software failure time sequence using our proposed SVM approach. We use the same data sets as cited in Tian Liang et al. [13], Park et al. [7] and Karunanithi et al. [5] in order to establish a common baseline for comparison purposes. Park et al. applied failure sequence number as input and cumulative failure time as desired output in feed-forward neural network (FFNN). Based on the input-output learning pair of cumulative execution time and the corresponding accumulated number of defects disclosed, Karunanithi et al. employed both feed-forward neural network (FFNN) and recurrent neural network (RNN) structures to model the failure process. These results are also summarized in Table 2. For example, using our proposed approach with data set DATA-1, the average relative prediction error (AE) is 0.82, 0.89and 0.46.

This error is lower than the results obtained by Tian Liang (2.4) using SVM, Park et al. (2.58) using feed-forward neural network, Karunanithi et (2.05) using recurrent neural network, Karunanithi et al. (2.50) using feed-forward neural network. In all four data sets, the AE results show that using our proposed SVRSRPM yields a lower average relative prediction error compared to the other approaches.

TABLE 1. COMPARISON OF AVERAGE RELATIVE PREDICTION ERROR

Data sets	SVM[13]	FFNN[5]	RNN[5]	FFNN[7]	1-NormSVR	2-NormSVR	ν-SVR
data-1	2.44	2.58	2.05	2.50	0.82	0.89	0.46
data-2	1.52	3.32	2.97	5.23	0.98	1.35	1.02
data-3	1.24	2.38	3.64	6.26	0.73	0.81	0.75
data-4	1.20	1.51	2.28	4.76	0. 79	0. 82	0. 82

V. SUMMARIES

In this paper, we proposed an SVM-based software reliability prediction model which has two special features. We conduct comparative studies on model performance between our proposed SRPM and existing SVM-based and some ANN-based SRPMs. Data collected from real software projects are used in the studies. Experimental results show that the proposed SVMSRPM model could achieve the best performance in terms of prediction accuracy.

REFERENCES

[1] IEEE Std 1633-2008, IEEE Recommended Practice on Software Reliability

[2] Inoue S and Yamada S. Generalized discrete software reliability modeling with effect of program size. IEEE Transactions on Systems, Man and Cybernetics, 2007, Part A, 37(2): 170–179

[3] Pham S and Pham H. Quasi-renewal time-delay fault-removal consideration in software reliability modeling. IEEE Transactions on Systems, Man and Cybernetics- Part A:Systems and Humans 2009, 39(1): 1-10

[4] Huang C Y and Huang W C. Software reliability analysis and measurement using finite and infinite server queueing models. IEEE Transactions on Software reliability, 2008, 57(1): 192-203

[5] Karunanithi N, Whitley D and Malaiya Y K. Prediction of software reliability using connectionist models. IEEE Trans. Software Engineering, 1992, 18(7): 563–574

[6] Karunanithi N, Whitley D and Malaiya Y K. Using neural networks in reliability prediction. IEEE Software, 1992, 9(4): 53-59

[7] Park J Y, Lee S U and Park J H. Neural network modeling for software reliability prediction from failure time data. J. Electrical Engineering and Information Science ,1999, 4(4) : 533–538

[8] Cai K Y, Cai L and Wang W D. On the neural network approach in software reliability modeling. Journal of Systems and Software, 2001, 58 (1): 47-62

[9] Aljahdali S H, Sheta A and Rine D. Prediction of software reliability: A comparison between regression and neural network non-parametric models, In: Proc. 16th ACM/IEEE Int. Conf. Computer Systems and Applications, Beirut, Lebanon, 2001:470-473

[10] Ho S L, Xie M and Goh T N. A study of the connectionist models for software reliability prediction. Computers and Mathematics with Applications, 2003, 46(7): 1037-1045

[11] Sitte R. Comparison of software-reliability-growth predictions: Neural networks vs parametric-recalibration. IEEE Transactions on Reliability, 1999, 48(3): 285–291

[12] Vapnik V. The nature of statistical learning theory. New York: Springer Verlag, 1995

[13] Tian L and Noore A. Dynamic software reliability prediction: An approach based on support vector machines. International Journal of Reliability, Quality and Safety Engineering, 2005, 12(4): 309–321

[14] Yang B and Li X. A study on software reliability prediction based on support vector machines. In: Proc IEEE International Conference on Industrial Engineering and Engineering Management, 2007:1176-1180

[15] Scholkopf B, Smola A. Learning with kernels: Support vector machines, regularization, and beyond. Cambridge, MA: MIT Press.2002.