International Journal of Computer Science
and Engineering (IJCSE)
ISSN(P): 2278-9960; ISSN(E): 2278-9979
Vol. 3, Issue 3, May 2014, 83-90
© IASET

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

PREDICTION OF SOFTWARE MAINTENANCE EFFORT ON THE BASIS OF UNIVARIATE APPROACH WITH SUPPORT VECTOR MACHINE

DIMPLE CHANDRA & PARTIBHA YADAV

Department of Computer Science and Engineering, PDM College of Engineering for Women,
Bahadurgarh, Haryana, India

ABSTRACT

The connection among object oriented metrics and software maintenance effort is complex and non-linear. Therefore, there is wide research area in development and application of sophisticated techniques which can be used to construct models for predicting software maintenance effort. The aim of this paper is to evaluate Support Vector Machine for Regression in software maintainability prediction using object-oriented metrics to construct models for prediction of Software Maintenance Effort. Support Vector Machine has already proved its importance in Banking Sector and in other areas also. We are using SVM with Radial kernel function. It is observed that Support Vector Machine can be used for constructing accurate models for prediction of software maintenance effort which gives most accurate models for prediction. We are using maintenance effort data of software product QUES (Quality Evaluation System) in this study. The dependent variable in our study is maintenance effort. The independent variables are eight Object Oriented metrics. We will verify the dataset by Univariate performance basis. The results show that the MARE of MPC in QUES Dataset is 0.644, while other metrics have larger MARE value. Thus we found that Univariate approach of evaluating the OO Metrics is useful in constructing software quality model.

KEYWORDS: Kernels Function, Object, Oriented Metric, Regression, Software quality, Support Vector Machine and Univariate

INTRODUCTION

Currently software quality is a major factor of concern. The growing research activity in software quality leads to innovation of novel practice, to predict its attributes. There are several empirical studies which show that there is a strong relationship between Object Oriented (OO) metrics and OO software quality attributes such fault proneness (V. Basili, L. Briand, W. Melo), maintenance effort (W. Li and S. Henry) and testability (S. R. Chidamber and C. F. Kemerer). Maintainability is an important quality attribute and a difficult concept as it involves a number of measurements. OO metrics are used in quality estimation. However quality estimation means estimating maintainability or reliability of software. Software reliability is a valuable ingredient to make the system work properly without a fail (M. R. Lyu). As the OO system uses a huge amount of small methods, it is time consuming, error prone and has a distinctive maintenance problem (R. E. Johnson and B. Foote). In our knowledge there are no significant research studies showing application Support Vector Machine with Kernel functions. As the traditional computers are not excellent to interact because of the noised data, immense parallelism, fault tolerant, and failure to adapt to certain circumstance, so Support Vector Machine provides a better option for handling software quality. The application of Support Vector Machine for software quality prediction using object-oriented metrics is focused in this paper.

Support Vector Machine

SVM (Support Vector Machines) are a useful technique for data classification and regression. SVM is a learning system using a high dimensional feature space. It yields prediction functions that are expanded on a subset of support vectors.

SVM can generalize complicated gray level structures with only a very few support vectors. SVM is a large margin linear classifier.

One interesting property of support vector machines and other kernel-based systems is that, once a valid kernel function has been selected, one can practically work in spaces of any dimension without any significant additional computational cost, since feature mapping is never effectively performed. In fact, one does not even need to know which features are being used.

Another advantage of SVMs and kernel methods is that one can design and use a kernel for a particular problem that could be applied directly to the data without the need for a feature extraction process

Given a Set of Data Points

$$\begin{aligned}&\{(\mathbf{x}_i, y_i)\},\ i = 1, 2, \cdots, n\\ &\text{For } y_i = +1,\ \ \mathbf{w}^T\mathbf{x}_i + b \geq 1\\ &\text{For } y_i = -1,\ \ \mathbf{w}^T\mathbf{x}_i + b \leq -1\end{aligned}$$

With a scale transformation on both *w* and *b*, the above is equivalent to

$$\begin{aligned}&\mathbf{w}^T\mathbf{x}^+ + b = 1\\ &\mathbf{w}^T\mathbf{x}^- + b = -1\end{aligned}$$

The margin width is:

$$\begin{aligned}M &= (\mathbf{x}^+ - \mathbf{x}^-)\cdot\mathbf{n}\\ &= (\mathbf{x}^+ - \mathbf{x}^-)\cdot\frac{\mathbf{w}}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}\end{aligned}$$

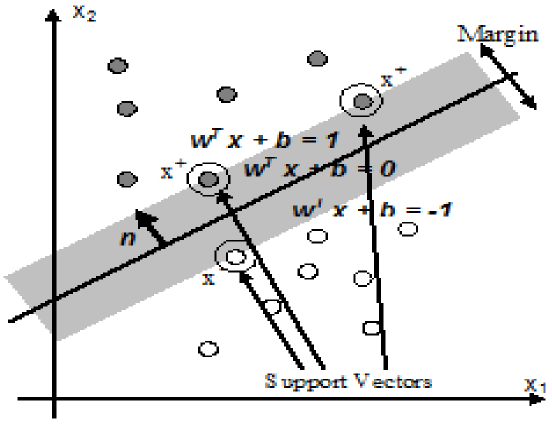

Figure 1: Large Margin Hyperplane

The kernel trick can be applied to any algorithm that solely depends on the dot product between two vectors. Wherever a dot product is used, it is replaced by a kernel function. When done, linear algorithms are transformed into a non-linear algorithm.

SVM Regression

SVMs can also be applied to regression problems by the introduction of an alternative loss function (Nello Cristianini and John Shawe-Taylor), (A. J. Smola). The loss function must be modified to include a distance measure. The regression can be linear and nonlinear. Linear models mainly consist of the following loss functions, e-intensive loss functions, quadratic and Huber loss function.

Instead of minimizing the observed training error, Support Vector Regression (SVR) attempts to minimize the generalization error bound so as to achieve generalized performance. The idea of SVR is based on the computation of a linear regression function in a high dimensional feature space where the input data are mapped via a nonlinear function (Debasish Basak, Srimanta Pal and Dipak Chandra Patranabis).

By using different loss function called the ε-insensitive loss function,

$$\|y - f(x)\|_{\varepsilon} = \max\{0, \|y - f(x)\|\varepsilon\}, \varepsilon > 0$$

SVMs can also perform regression. This loss function ignores errors that are smaller than a certain threshold ε > 0 thus creating a tube around the true output. The primal becomes:

$$t(w,\xi) = \frac{1}{2}\|w\|^2 + \frac{c}{m}\sum_{i=0}^{m}(\xi_i + \xi_i^*)$$
$$(\langle \Phi(x_i), w\rangle + b) - y_i < \varepsilon - \xi_i$$

$$y_i - (\langle \Phi(x_i), w\rangle + b) < \varepsilon - \xi_i^*$$
$$\xi_i^* \geq 0$$

We can estimate the accuracy of SVM regression by computing the scale parameter of a Laplacian distribution on the residuals ζ = y − f(x), where f(x) is the estimated decision function (Lin and Weng 2004).

The Kernel Trick

The kernel trick can be applied to any algorithm that solely depends on the dot product between two vectors. Wherever a dot product is used, it is replaced by a kernel function. When done, linear algorithms are transformed into a non-linear algorithm.

The RBF kernel is one of the most popular kernel functions. It adds a “bump” around each data point:

$$f(x) = \sum_{i=1}^{n}\alpha_i \exp(-\gamma\|x_i - x\|^2) + b$$

γ is the kernel parameter. The choice of best γ value is an important factor for the performance of the SVM.

We have chosen RBF kernel for my research in this paper because in many studies RBF kernel found to be best kernel among all kenels (Jae H. Min, Young-Chan Lee).

Empirical Data

The commercial software product QUES data are used in this research paper, which is presented in Li and Henry Paper (W. Li and S. Henry). The number of lines changed per class is termed as Maintenance Effort. Addition or a deletion could be a line change. A change of the content of a line is counted as a deletion followed by an addition. This measurement is used in this study to estimate the maintenance effort of the Object Oriented systems. QUES contains 71 classes.

Dependent and Independent Variables

The continuous dependent variable in our study is testing effort. The goal of our study is to empirically explore the relationship between OO metrics and testing effort at the class level. We use ANN to predict testing effort per class. Testing effort is defined as lines of code changed or added throughput the life cycle of the defect per class. The independent variables are OO metrics chosen for this study. The metrics selected in this study are summarized in Table 1.

Experimental Methodology

The performance is calculated on the basis of Univariate performance of each software metric. This is because, the effect of each software metric can be observed individually and which software metric is effective for calculating the maintenance effort of the software. Here in this paper we are using commercial dataset QUES. We are using SVM with Radial kernel for the regression value calculation of each metric and calculating MARE, MRE, R-Values and P-Values for each OO Metric.

Maintenance Effort Modeling Using SVM

For maintenance effort we are calculating MARE, MRE, R-Value and P-Value.

Mean Absolute Relative Error (MARE) (G. Finnie and G. Witting) - This is the preferred measure used by software engineering researchers and is given as

$$\text{MARE} = \frac{\sum_{i=1}^{N} \frac{abs(predicted - actual)}{actual}}{N}$$

Where *predicted* is predicted output which is calculated by using SVM, *actual* is the actual values available in the Dataset and N is the no. of observations.

Mean Relative Error (MRE) (G. Finnie and G. Witting) – This measure is used to estimate whether models are biased and tend to overestimate or underestimate and is calculated as follows:

$$\text{MRE} = \frac{\sum_{i=1}^{N} abs(predicted - actual)}{N}$$

A large positive MRE would suggest that the model over estimates the number of lines changed per class, whereas a large negative value will indicate the reverse.

R-Value

R-Value is the Correlation coefficient between the outputs and targets. It is a measure of how well the variation in the output is explained by the targets. If this number is equal to 1, then there is perfect correlation between targets and outputs.

P-Values

P-values are used for testing the hypothesis of no correlation. Each p-value is the probability of getting a correlation as large as the observed value by random chance, when the true correlation is zero. If p is small, say less than 0.05, then the correlation R is significant".

OO Software Metric

The object oriented metrics used by Li-Henry in their research paper are abbreviated as follows:

RFC

Response for Class the RFC metric measures the cardinality of the response set of a class. One may intuit that the larger the RFC metric, the harder it is to maintain the class since calling a large number of methods in response to a message makes tracing an error difficult. The calculation of RFC is number of local methods and number of methods called by local methods; ranging from 0 to N; where N is a positive integer.

NOM

Number of Methods NOM in a class, since the local methods in a class constitute the interface increment of the class, NOM serves the best as an interface metric. NOM is the number of local methods. The more methods a class has, the more complex the class' interface has incremented.

WMC

Weighted Method Complexity WMC metric measures the static complexity of all the methods. The more control flows a class' methods have, the harder it is to understand them, thus the harder it is to maintain them. The WMC is calculated as the sum of McCabe's cyclomatic complexity of each local method; ranging from 0 to N; where N is a positive integer.

DAC

Data Abstraction Coupling A class can be viewed as an implementation of an ADT (Abstract Data Type). The metric which measures the coupling complexity caused by ADTs is DAC (Data Abstraction Coupling) and is the number of ADTs defined in a class.

MPC

Message Passing Coupling MPC is used to measure the complexity of message passing among classes in the research. MPC is number of send-statements defined in a class. The number of messages sent out from a class may indicate how dependent the implementation of the local methods is upon the methods in other classes.

LCOM

Lack of Cohesion of Methods the LCOM metric measures the lack of cohesion of a class. One may intuit that the larger the metric, the harder it is to maintain the class. The calculation of LCOM is number of disjoint sets of local methods; no two sets intersect; any two methods in the same set share at least one local instance variable; ranging from 0 to N; where Nis a positive integer.

NOC

Number of Children the NOC metric measures the number of direct children a class has. One may intuit that the larger the NOC metric, the harder it is to maintain the class. The calculation of NOC is number of direct sub-classes; ranging from 0 to N; where N is a positive integer.

DIT

Depth in the Inheritance Tree the DIT metric measures the position of a class in the inheritance hierarchy. One may hypothesize that the larger the DIT metric, the harder it is to maintain the class. The calculation of the DIT metric is the level number for a class in the inheritance hierarchy. The root class DIT is zero, DIT ranges from 0 to N; where N is a positive integer.

Table 1: OO Software Metric

S. No	Metric	OO Attribute
1	Response for a Class (RFC)	Class
2	Number of Methods per Class (NOM)	Class
3	Weighted Methods per Class (WMC)	Class
4	Data Abstraction Coupling (DAC)	Coupling
5	Message Passing Coupling (MPC)	Coupling
6	Lack of Cohesion (LCOM)	Cohesion
7	15 Number of Children (NOC)	Inheritance
8	Depth of Inheritance (DIT)	Inheritance

Univariate Results

Here to observe the impact of object oriented metrics in Maintenance measurement, we have chosen each metric to calculate its parameters and see which metric is important for the Maintenance measurement of the software. Here in this paper for each metric I have calculated MARE, MRE, R-Value and P-Values. And on the basis of these results we can decide which metric is more important for Maintenance measurement.

Observations

In QUES Dataset the NOC Metric is completely zero so we are leaving this metric for the calculation.

Univariate Response of each OO Metric in Ascending Order to MARE by Using Support Vector Machine with Radial Kernel Function of QUES Dataset .

Table 2: QUES Dataset

Metric	MARE	MRE	R-Value	P-Value
MPC	0.644	31.214	0.355	0.0024
RFC	0.673	29.907	0.3129	0.0079
WMC	0.731	28.471	0.4099	0.0004
DAC	0.813	32.450	-0.1553	0.196
DIT	0.836	33.207	-0.3375	0.004
NOM	0.844	31.907	0.0349	0.7726
LCOM	0.868	32.732	-0.1584	0.1869

Graphical Representation

Graphical representation of the results is shown by a bar chart in Figure 2.

CONCLUSIONS

In this work we evaluate and compare different Object Oriented Metrics for prediction of software maintenance effort of commercial software systems. The graph in Figure 2 shows the MARE obtained with different Object Oriented Metrics to compare their accuracy for prediction of software maintenance effort. The results show that the MARE of MPC in QUES Dataset is 0.644, while other metrics have larger MARE value. It is concluded that the results of Univariate are the best followed by other approaches. The MARE of QUES are in ascending order, according to it we can use them for calculating maintenance effort measurement. Hence it is concluded that the Univariate approach can be successfully used for the prediction of software maintenance effort. However, our results need to be generalized by conducting similar studies on maintenance data of software system.

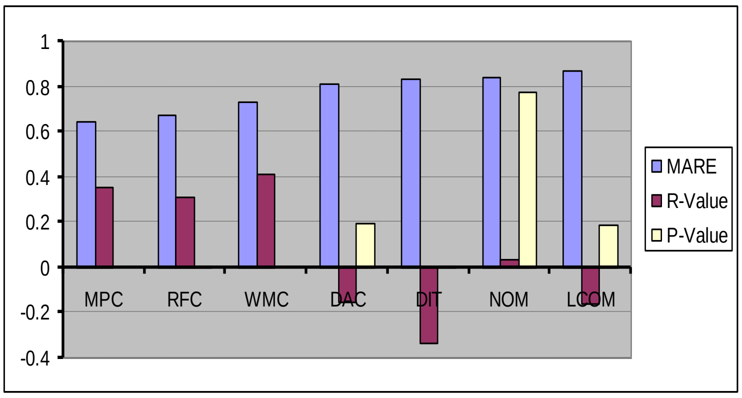

Figure 2: QUES Dataset Response on Univariate Approach

REFERENCES

1. V. Basili, L. Briand, W. Melo, "A Validation of Object- Oriented Design Metrics as Quality Indicators", IEEE Transactions on Software Engineering, vol. 22 no.10, pp. 751-761, 1996

2. W. Li and S. Henry, "Object Oriented Metrics that Predict Maintainability", Journal of Systems and Software, vol. 23 no.2, pp.111-122, 1993.

3. S. R. Chidamber and C. F. Kemerer, "A Metrics Suite for Object Oriented Design," IEEE Transactions on Software Engineering, vol. 20, pp 476- 493, 1994.

4. M. R. Lyu, "Handbook of software Reliability Engineering", IEEE Computer Society Press, McGraw Hill, 1996.

5. R. E. Johnson and B. Foote, "Designing Reusable Classes. Journal of Object-Oriented Programming", vol. 1, no. 2, pp. 22-35, 1988.

6. Debasish Basak, Srimanta Pal and Dipak Chandra Patranabis, "Support Vector Regression", Neural Information Processing – Letters and Reviews, vol. 11, No. 10, October 2007.

7. Lin CJ, Weng RC (2004). "Probabilistic Predictions for Support Vector Regression."

8. G. Finnie and G. Witting, “AI Tools for Software Development Effort Estimation”, International Conference on Software Engineering: Education and practice, 1996.

9. B. Henderson-sellers, “Object-Oriented Metrics, Measures of Complexity”. Prentice Hall, 1996.

10. www.mathworks.com.

11. Nello Cristianini and John Shawe-Taylor, “An Introduction to Support Vector Machines and Other Kernel-based Learning Methods”, Cambridge University Press, 2000.

12. A. J. Smola. “Regression estimation with support vector learning machines”. Master’s thesis, Technische Universit¨at M¨unchen, 1996.

13. Jae H. Min, Young-Chan Lee, “Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters”, Elsevier, Expert Systems with Applications 28 (2005) 603–614.