Optimization of Object-Oriented Metrics using Hopfield Neural Network

Vijay Pal Dhaka, Swati Agrawal

Abstract-This paper examined the application of Artificial neural network for software quality prediction using object-oriented metrics. Quality estimation include estimating maintainability of software. In this study maintenance effort was chosen as the dependent variable and principal components of object-oriented metrics as the dependent variables. We are prediction the number of lines per changed per class. Two neural network models are used, they are ward neural network and Hopfield neural network. The Artificial neural network prossesses the advantages of predicting software quality accurately and identifies the defects by efficient discovery mechanisms.

Keywords- Software quality metrics, maintainability, object-oriented, neural network, principal component analysis

I. INTRODUCTION

Currently software quality is a major factor of concern. Maintainability is an important quality attribute of software quality and a difficult concept as it involves a number of measurements. Prediction models using object-oriented design metrics can be used for obtaining assurances about software quality. However quality estimation means estimating maintainability or reliability of software [1].

As the object-oriented system use a larger number of small methods, a unique maintenance problem is associated with it. The relationship between object-oriented metrics and software maintenance effort is complex and non linear. Object-oriented metrics is a good platform to access maintenance effort. Maintainability is usually measured as the change in effort i.e. the number of lines changed per class. Artificial neural network is one such technique which involves a series of steps for the computation of maintainability effort.

Artificial neural network is used as a predictive model because of it is minimal computation and complex modeling ability. Our neural network aims to predict object-oriented software quality by estimating the number of lines changed per class. We used object-oriented metrics for quality estimation. The relationship between object-oriented metrics and software maintenance effort is complex and non linear[2].

We also introduce ward neural network and Hopfield neural network to improve prediction result for estimating software quality. Ward neural network is a back propagation network with different activation functions. We are applied to hidden layer slabs to detect different features in a pattern processed through a network to lead to better prediction. We use a Gaussian function in one hidden slab to detect features in the mid-range of the data and a Gaussian complement in another hidden slab to detect features for the upper and lower extremes of the data.

Thus, the output layer will get different "views of the data". Combining the two feature sets in the output layer leads to a better prediction [3].

Hopfield model consists of a single layer of processing elements where each unit is connected to every other unit in the network other than itself. Hopfield model computes its output recursively in time until the system becomes stable. The Network capacity of the Hopfield network model is determined by neuron amounts and connections within a given network. Therefore, the number of memories that are able to be stored are dependent on neurons and connections. The Network capacity of the Hopfield network model is determined by neuron amounts and connections within a given network. Therefore, the number of memories that are able to be stored are dependent on neurons and connections. Hopfield was able to show that with the nonlinear activation function, the dynamical rule will always modify the values of the state vector in the direction of one of the stored patterns [4].

The Principal components analysis (PCA), a data reduction technique is used with the intention of reducing the dimensionality of a multivariate data. This method is more economical in the sense of storing the data and saving the memory space. The main goal of PCA is to reduce that raw data by removing the correlations among the software metrics, thereby making the prediction model more stable[5].

II. RELATED WORK

Many researches described the impact of object-oriented metrics on software maintainability. With the increasing use of Object-oriented methords in new software development there is a growing need to both document and improve current practies in Object-oriented design and development. A variety of statistical techniques are used in software quality modeling. Models are based on statistical relationship between the measure of quality and measure of software attributes. These relationship are often complex and non linear but Artificial neural network work on non-linear modeling.

Khoshgoftaar presented a case study of realtime avionics software to predict the testability of each module from static measurements of source code [6]. They found that ANN is a promising technique for building predictive models, because they are able to model nonlinear relationships.

One such set of object-oriented metrics is the set proposed by Chidamber and Kemerer (1994). Chidamber and Kemerer also reported empirical data from two commercial organizations and suggested ways in which the metrics could be used to manage OO design efforts.

Manuscript Received July, 2013.

Dr.Vijay Pal Dhaka, Head of Computer & Engg.Deptt. Jaipur National University, India

Swati Agrawal, Research scholar, Computer & Engg.Deptt. Jaipur National University, India.

Thwin and Quah applied neural networks for software quality speculation. OO metrics was considered as independent variable and prediction was done by two neural network models namely ward neural network and general regression neural network (GRNN). They have shown that GRNN network model can predict more accurately than Ward network model [7].
Our neural network models aims to predict object oriented software quality by estimating the number of lines changed per class. We also introduce using Ward neural network and Hopfiled neural network to improve prediction results for estimating software quality.

III. RESEARCH BACKGROUND

- Ward neural network

We have selected Ward neural Network. Backpropagation network is the most popular network for practical applications.

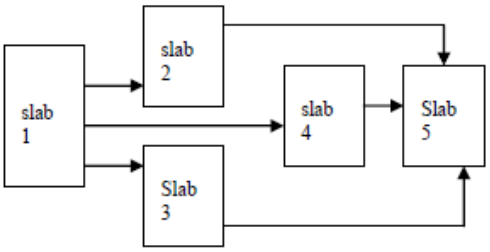

Figure 1: Ward neural network [8]

We choose three layer Backpropagation Ward neural network because it is an effective network for most applications and trains much more quickly than 4 or 5 layer networks. We use three different activation functions. It has three slabs (slab2, slab3 and slab4) in the hidden layer. Hidden layers in a neural network are known as feature detectors. A slab is a group of neurons. Each slab in the hidden layer has a different activation function; this offers three ways of viewing the data. We use a linear function for the output slab (slab5). The hyperbolic tangent (tanh) function is used in one slab of hidden layer (slab3) because it is better for continuous valued outputs especially if the linear function is used on the output layer. The Gaussian function is used in another slab of the hidden layer (slab2). This function is unique, because unlike the others, it is not an increasing function [9]. It is the classic bell shaped curve, which maps high values into low ones, and maps mid-range values into high ones. The Gaussian Complement is used in the third slab of the hidden layer (slab4) to bring out meaningful characteristics in the extremes of the data. We use a smaller learning rate 0.1 and momentum 0.1 as our network is a predictive network where outputs are continuous values rather than categories.

- Hopfield neural network

A **Hopfield network** is a form of recurrent artificial neural network. Hopfield nets serve as content-addressable memory systems with binary threshold units. They are guaranteed to converge to a local minimum. The Hopfield model computes its output recursively in time until the system becomes stable. Below is a Hopfield model with six units, where each node is connected to every other node in the network.

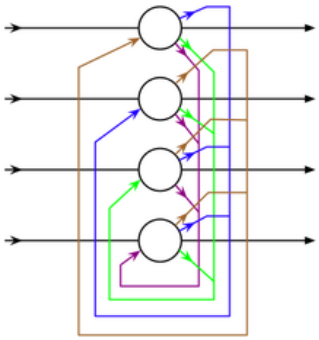

Figure 2: A Hopfield net with four nodes [10].

The Hopfield model consists of a single layer of processing elements where each unit is connected to every other unit in the network other than itself. The memory capacity of the Hopfield model can be increased. The units in Hopfield nets are binary threshold units, i.e. the units only take on two different values for their states and the value is determined by whether or not the units' input exceeds their threshold. Hopfield nets can either have units that take on values of 1 or -1, or units that take on values of 1 or 0. So, the two possible definitions for unit *i*'s activation, a_i, are:

$$a_i \leftarrow \begin{cases} 1 & \text{if } \sum_j w_{ij} s_j > \theta_i, \\ -1 & \text{otherwise.} \end{cases} \quad (1)$$

$$a_i \leftarrow \begin{cases} 1 & \text{if } \sum_j w_{ij} s_j > \theta_i, \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

Where:

- w_{ij} is the strength of the connection weight from unit j to unit i (the weight of the connection).
- s_j is the state of unit j.
- θ_i is the threshold of unit i.

The requirement that weights be symmetric is typically used, as it will guarantee that the energy function decreases monotonically while following the activation rules, and the network may exhibit some periodic or chaotic behaviour if non-symmetric weights are used. However, Hopfield found that this chaotic behaviour is confined to relatively small parts of the phase space, and does not impair the network's ability to act as a content-addressable associative memory system. The Network capacity of the Hopfield network model is determined by neuron amounts and connections within a given network [11]. Therefore, the number of memories that are able to be stored are dependent on neurons and connections.
Hopfield would use a nonlinear activation function. Hopfield was able to show that with the nonlinear activation function, the dynamical rule will always modify the values of the state vector in the direction of one of the stored patterns.

- **Principal component analysis**

Principal components (PCs) are linear combinations of the standardized independent variables.

In case of the original data are object oriented metrics, the new PC variables are termed as domain metrics. PC analysis is used to maximize the sum of squared loadings of each factor extracted in turn [12]. In PC analysis a new variable (Pi) is constructed, called Principal Component (PC) out of a given set of variables Xj' s(j = 1,2,....,k).

P1 =b11 X1 + b12 X2+ …… + b1KXK
P1 =b21 X1 + b22 X2+ …… + b2KXK
.. ..
P1 =bk1 X1 + bk2 X2+ …… + bkKXK

The factor bijs' are called loadings, and are worked out in such a way that the extracted PCs satisfy the following two conditions. 1) PCs are uncorrelated (orthogonal) and 2) The first PC (PC1) has the highest variance; the second PC (PC2) has the next highest variance and so on. The variables with high loadings facilitate to make out the dimension PC, but this usually requires some degree of interpretation. As the PCs are independent, orthogonal rotation is used. Varimax rotation is the most frequently used in the literature. The sum of squared values of loadings relating to dimension is referred to as eigenvalue. Eigenvalue or latent root is an integral part of PC [13]. The PCs with eigenvalue greater than 1 is usually taken for interpretation.

IV. PREDICTION OF MAINTENANCE EFFORT

Description of data

To estimate the maintenance effort of commercial software product UIMS data used in this study. A UIMS (**U**ser **I**nterface **M**anagement **S**ystem) should not be thought of as a *system* but rather a software architecture (a UIMS is also called a User Interface Architecture). The objective of such a separation is to increase the ease of maintainability of the software. Also, by abstracting the code generating the user interface from the rest of the application's logic or semantics, customization of the interface is better supported. The maintenance effort is measured by using the number of lines changed per class. A change in a line could be an addition or a deletion. This dimension is used in the study to evaluate the maintainability of the object-oriented systems [14].

Selection of metrics

We introducing the research on software defect prediction into the object oriented metrics using neural network. We have selected the software metrics that have a strong relationship with software maintainability. Following metrics are used:

Weighted Methods per Class (WMC) is defined as a function of the number of all member functions and operators in each class. We have selected WMC to measure the complexity of an individual class.

Depth of Inheritance Tree (DIT) of a class is the length of the longest path from the class to the root in the inheritance hierarchy. This determines the complexity of a class based on its ancestors, since a class with many ancestors is likely to inherit much of the complexity of its ancestors. The deeper a class is in the hierarchy, the greater the number of methods it is likely to inherit making it more complex to predict its behavior [15]. Therefore, the more likely it is to contain a fault.

Response For a Class (RFC) is the number of methods that can potentially be executed in response to a message received by an object of that class. The response set of a class consists of the set of M methods of the class, and the set of methods directly or indirectly invoked by methods in M. The number of methods that could potentially respond to a message indicates the complexity of that class. Therefore RFC can be used as a predictor variable for the number of faults [16].

Number of Children (NOC) measures the number of immediate descendants of a particular class. This measures an amount of potential reuse of the class. The more reuse a class might have, the more complex it may be, and the more classes are directly affected by changes in its implementation. This increases the magnitude of ripple effects. Therefore we selected the NOC metric to predict the number of faults.number of methods per class (NOM) it count number of methods defined in a class.

Message passing coupling(MPC) it count the number of send statements defined in the class [17].

We performed the preliminary analysis using multiple regression. We got an R square value of 0.756. About 75% of the variation in the criterion variable maintenance effort can be explained by the Hopfield neural network with all object oriented metrics. W e can conclude the prediction of maintenance effort form the object oriented metrics is possible [18].

V. EXPERIMENTS

In our experiment, each data pattern was examined for erroneous entries, outliers, blank entries and redundancy. After standardizing the metric data, we performed the principal component analysis.

For the UIMS (usre interface management system) system principal component analysis identifies Two pc, which capture 75% of the data set variance. For each pc, we also provide it is eigenvalues and variance percent. The interpretations of PCs are given as follows:

• P1: WMC , RFC and NOM are cohesion, coupling and size metrics. We have size, coupling and cohesion metrics in this dimension. This shows that there are classes with high internal methods (methods defined in the class) and external methods (methods called by the class). This means cohesion and coupling is related to number of methods and attributes in the class.

• P2: MPC is coupling metric that counts number of send statements defined in a class.

• P3: NOC and DIT are inheritance metrics that count number of children and depth of inheritance tree in a class.

Table 1 presents the relationship between the original object oriented metrics and the domain metrics for UIMS system. The rotated component matrix is given in Table 1.

Table 1. Rotated principal components

Metrics	P1	p2
WMC	0.815	0.221
DIT	0.03	0.085
NOC	-0.07	-0.22
RFC	0.85	0.34
NOM	0.655	-0.079
MPC	-0.17	0.829
Eigenvalue	4.84	17.64
Variance%	46.75	13.6

VI. RESULT AND DISCUSSION

The goodness of fit of the model is measured by using the coefficient of correlation(r) and mean absolute error. These statistical measures are shown in Table II. The correlation between the predicted change and the observed change is represented by the coefficient of correlation (r).

Table 2. Experimental result for UIMS system

Tests	Ward NN	Hopfield NN
R(Correlation Coefficient)	0.778	0.846
P value	0.000	<0.001
R-square	0.565	0.685
Mean absolute error	12.82	17.96

The (correlation coefficient) r value is 0.778 in used in ward neural network and 0.846 in Hopfield neural network. It shows that Hopfield model is represent high correlation. The significance level of a cross validation is indicated by a p value. It means p value in both cross-validation shows high degree for the successful validation. We conclude that the impact of our model prediction is valid and useful.

VII. CONCLUSION

This study presents the prediction of maintenance effort using Artificial neural network technique. This study are applicable to improve the quality of software products. This paper presents ward neural network and Hopfield neural network of the UIMS (user interface management system), predicting the number of lines changed per class. The independent variables were principal component analysis from six object oriented metrics. This results shows that these independent variable appear to be useful in predicting maintenance effort. Principal component analysis plays an important role in object oriented metrics. It reduce the multidimensional problem of the huge data by converting it to a small datasets. This method is more economical in the sense of storing the data and saving the memory space. Principal component analysis prediting the software quality using neural network models. Our future research aims to estimate software defect and minimize the issues related to software maintainability by using object oriented metrics and neural network models.

REFERENCES

1. Y. Dash, S.K. Dubey and A. Rana, "Maintainability Measurement in Object Oriented Paradigm", International Journal of Advanced Research in Computer Science (IJARCS), Vol.3, no.2, , April 2012, pp. 207-213.
2. H. D. Rombach, "A controlled experiment on the impact of software structure on maintainability", Software Engineering, IEEE Transactions on, SE-13(3):344–354, March 1987.
3. D. R Moreau and W. D. Dominick, "Object-Oriented Graphical Information Systems: Research Plan and Evaluation Metrics," Journal of Systems and Software, vol. 10, 1989, pp. 23-28.
4. Quah T. S, and M.M.T.Thewin (2003): Application of Neural Networks for Software Quality Prediction using Object-Oriented Metrics, Proceedings of the International Conference on Software Maintenance (ICSM'03), Vol 3.
5. Kanmani S., V.Sankaranarayanan and P.Thambidurai (2003): A Measurement Model for C++ Program Complexity Analysis, Proceddings of the 9th International Conference EPMESC, Macao, China, pp. 575-580.
6. Y. Dash, S.K. Dubey and A. Rana, Maintainability Measurement in Object Oriented Paradigm‖, International Journal of Advanced Research in Computer Science (IJARCS), Vol.3, no.2, pp. 207-213, April 2012.
7. J. T. S. Quah, M. M. T. Thwin, Application of Neural Networks for Software Quality Prediction Using Object-Oriented Metrics, Proceedings of the International Conference on Software Maintenance (ICSM'03), IEEE Computer Society, 2003.
8. M. M. T. Thwin,T. S. Quah, Application of neural networks for software maintainability prediction using Object-oriented metrics‖, Journal of systems and software, Vol.76, No.2, pp.147-156, 2005.
9. Tong-Seng Quah, Mie Mie Thwin, Prediction of software development fault in PL/SQL files using Neural network models.
10. K. K. Aggarwal, Y. Singh,A. Kaur and R. Malhotra, Application of Artificial Neural Network for Predicting Maintainability using Object-Oriented Metrics, World Academy of Science, pp. 140-144, 2006.
11. Y. Dash and S.K. Dubey, ―Quality Prediction in Object Oriented System by Using ANN: A Brief Survey‖, International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE), Vol. 2, no. 2, Feb. 2012.
12. F. Nielsen, Neural Networks – algorithms and applications. 2001 Available at: http://www.glyn.dk/download/Synopsis.pdf. Accessed on Dec.2011.
13. K. K. Aggarwal, Y. Singh, A. Kaur and R. Malhotra, Investigating effect of design metrics on fault proneness of object oriented systems, Journal of Object Technology,vol.6, no. 10, pp. 127-141, 2007.
14. A. Shaik, N. Satyanarayana, M Huzaifa, N. Shaik, M. Z. Naveed, S. V. A. Rao and C. R. K. Reddy. Invetsigate the result of object oriented design software metrics on fault proneness in object oriented systems: A case study,‖ Journal of emerging trends in computing and emerging sciences, Vol. 2, no. 4, pp. 201-208, 2011.
15. C. Zhong, Q. Hu, F. Yang and M. Yin. Software Quality Prediction Method with Hybrid Applying Principal Components Analysis and Wavelet Neural Network and Genetic Algorithm,' International Journal of Digital Content Technology and its Applications, Vol. 5, no. 3, pp.225-234, 2011.
16. [16] F. Lanubile, A. Lonigro, G. Visaggio, ―Comparing models for identifying fault-prone software components‖, In: Proc. of the 7th Int'l. Conf. Software Eng. and Knowledge Eng., pp. 312–319, June 1995.
17. L.C. Briand, J. Wüst, and H. Lounis, ―Replicated Case Studies for Investigating Quality Factors in Object-Oriented Designs,‖ Empirical Software Engineering. International Journal (Toronto, Ont.), 6(1), pp.11–58. 2001.
18. P.V.G.D Prasad Reddy, K.R. Sudha, S. P. Rama and S.N.S.V.S.C Ramesh, ―Software Effort Estimation using Radial Basis and Generalized Regression NeuralNetworks‖, Journal Of Computing, Volume 2, Issue 5, May 2010