ISSN:2249-5789
Samridhi Bhutani et al , International Journal of Computer Science & Communication Networks,Vol 5(2),92-95

Prediction of Software Maintainability using Neural Networks

Samridhi Bhutani
University School of Information and Communication Technology
GGS Indraprastha University, Dwarka
New Delhi, India
samridhibhutani@gmail.com

Anuradha Chug
University School of Information and Communication Technology
GGS Indraprastha University, Dwarka
New Delhi, India
a_chug@yahoo.co.in

***Abstract*--Software maintenance is an important aspect of software quality as it is the most expensive activity in the development lifecycle of a software. Hence, in the recent past many researchers have shown an increasing interest in predicting the most accurate model for maintainability of a software. In this paper, the three models have been proposed which are quantitatively compared to each other. The proposed models are Group Method of Data Handling (GMDH), General Regression Neural Network (GRNN) and Probabilistic Neural Network (PNN). In this study, an open source software with two versions was used in order to calculate the CHANGE which is defined as the number of lines of code which are modified, added or deleted from version 1 to version 2 of the software. Based on this study, it was concluded that General Regression Neural Network (GRNN) is the best neural model for prediction of software maintainability. Hence, it can be used an alternative to the other existing models by the companies to maintain their software as it predicts software maintainability more accurately than other alternative neural networks.**

Keywords--Software Maintainability Prediction; General Regression Neural Network (GRNN); Group Method of Data Handling (GMDH); Probabilistic Neural Network (PNN)

I. INTRODUCTION

Software maintainability means the ease with which a software system or component can be modified to correct faults, improve performance or other attributes or adapt to the changed environment[1]. Software maintenance is of three types:

- Perfective maintenance: In this type of maintenance the performance of the software is improved.
- Adaptive maintenance: In this type of maintenance the software is changed so that it is able to adapt to its new environment.
- Corrective maintenance: In this type of maintenance various faults in a software are corrected.

Software maintenance is one of the most important activities of the software lifecycle as software maintenance takes up 75 percent of the project's overall budget making it the most expensive activity of the software lifecycle. Hence, there are continuous studies going on to determine how this cost can be reduced to minimal.

This study has proposed and evaluated three models based on neural networks which establish a relationship between the various software metrics and software maintainability. Neural networks are based on the concept on which a human brain works. In a human brain neurons transfer information from one neuron to another that is it solves problems based on past experiences. Similarly, in neural networks neurons transfer information from one layer to another hence making decisions based on previous knowledge.

The models used in this study are:

- Group Method of Data Handling (GMDH): This is a machine learning algorithm which builds a polynomial function. In this algorithm iterative layers are formed by the genetic component in the data [7].
- Probabilistic Neural Network (PNN): This is also a machine learning algorithm which is a multi-layered feed forward network. This algorithm is used for classification [8].
- General Regression Neural Network (GRNN): This is a memory based network. This algorithm has a parallel structure and is used for solving regression based problems [9].

The metrics used in this study to evaluate the above discussed models are: Weighted Methods per Class (WMC), Number of Children (NOC), Coupling between Classes (CBO) and Depth in Inheritance (DIT) were taken from Chidambar and Kemerer suite of metrics [11]. Whereas, Lines of Code (LOC), McCabe's Cyclomatic Complexity (CC) are the traditional metrics and CHANGE is a metric which was calculated in this study by calculating the change in number of lines of code from version 1 to version 2 of the software.

II. RELATED WORK

Various methods have been proposed in the past to reduce the cost of software maintenance using object oriented metrics. Here, few of those studies have been discussed. In 1998, Chidamber, Darcy and Kemerer [2] determined the various uses of object oriented metrics. In

1998, Li and Henry [3] used multiple linear regression model to predict software maintenance effort. In 1999, Stavrinoudis, Xenos and Christodoulakis [4] determined which metrics should be used to predict software maintainability. In 2005, Thwin and Quah [5] used neural networks to predict software quality using object oriented metrics. In 2005, Misra [6] used linear regression to predict software maintenance effort. In 2010, A. Kaur, K. Kaur and Malhotra [15] used artificial neural network, fuzzy inference system (FIS), and adaptive neuro fuzzy inference system (ANFIS) approaches to predict software maintainability. In 2010, [16] MO Elish and KO Elish used the TreeNet technique to predict software maintainability.

III. MODELING TECHNIQUES USED

This section describes the three modelling techniques Group method of data handling (GMDH), Probabilistic neural network (PNN) and General regression neural network (GRNN) which were compared during this study. These modelling techniques are as described below:

A. Group Method of Data Handling (GMDH)

GMDH model was proposed by A.G. Ivahnenko [7]. It is based on neural networks hence is also known as Polynomial neural network. It is called a polynomial network as two inputs give an output which is a pure polynomial function. A GMDH network is a simple feed forward network which means no directed cycle is formed between the connected units. Iterative layers are formed in this network just like layers are formed in a neural network. Each new layer is created by using two or more neurons taken from the previous layers. This model has three layers which are:

- Input layer: This layer is formed by the inputs which are fed in the network.
- Hidden layer: This layer contains pre-specified neurons.
- Output layer: This layer contains the final output neurons of the network which are determined by the apiori algorithm.

B. Probabilistic Neural Network (PNN)

This model was proposed by Specht [8]. This model originated from neural networks [8]. This is also a feed forward neural network. This network is called a forward network as the information in this network moves only in one direction which is forward direction. It is a supervised learning algorithm. This algorithm uses a smoothing parameter known as sigma. It contains four layers which are:

- Input layer: In this layer each neuron contains a predictor variable. Each neuron of this layer represents an independent variable.
- Pattern layer: This layer contains one case from each training set.
- Summation layer: This layer contains all the target categories.
- Output layer: This layer contains the largest target category.

C. General Regression Neural Network (GRNN)

This model was proposed by Specht [9]. This is a memory based network. This network has a parallel structure hence the speed of training in this network is faster than all other networks. This network works on the principle of back-propagation. The advantage of this network is that it works well with sparse data or small datasets. The smoothing factor is used as a parameter in this algorithm whose value should range from 0.01 to 1 for obtaining good results. There are three layers in this network which are:

- Input layer: This layer acts as slab1 in the network used for training of the dataset. This contains the number of neurons inputted in the network.
- Hidden layer: This layer acts as slab 2 and contains the number of patterns in the training set.
- Output layer: This layer acts as slab 3 in the network. This contains the number of neurons outputted by the network.

IV. DESCRIPTION OF DATA AND METRICS USED

This section discusses the metrics and the data used for conducting this study.

A. Metrics Used

The metrics used in this study are described below.

- Depth of Inheritance (DIT): DIT is defined as the longest path from root to leaf node. The deeper the hierarchy the higher is the complexity of the predecessor.
- Weighted Method per Class (WMC): WMC is the number of local methods in a class.
- Number of Children (NOC): NOC is the number of subclasses a class contains.
- Coupling between Classes (CBO): CBO is the number of classes to which a class is interdependent on.
- Lines of Code (LOC): LOC is the number of lines a program contains.
- Cyclomatic Complexity (CC): CC was defined by McCabe [10]. It determines the complexity of a program by determining the number of independent paths in a program.
- CHANGE: Change is used as the dependent variable in this study and is defined as the number of lines modified, added or deleted from version 1 to version 2 of the software.

TABLE I STATISTICS OF VERSION 1

Metrics	Statistics			
	Min	Max	Mean	Standard Deviation
LOC	4	115	26.65	22.95
WMC	0	15	3	2.77
DIT	0	1	0.73	0.44
NOC	0	15	0.65	2.64
CBO	0	17	3	2.84
CC	0	32	1.63	5.15
CHANGE	0	20	4.39	5.12

TABLE II STATISTICS OF VERSION 2

Metrics	Statistics			
	Min	Max	Mean	Standard Deviation
LOC	4	130	29.15	25.36
WMC	0	15	3.02	2.80
DIT	0	2	0.76	0.48
NOC	0	15	0.76	2.68
CBO	0	17	3.26	2.85
CC	0	34	1.68	5.49
CHANGE	0	20	4.39	5.12

B. Description of Data

This study used two versions of the open source software: Natural CLI [17]. Both these versions of the software contain 38 classes which are implemented in Java. This study compared the two versions by calculating seven metrics of the two versions using the CCCC tool [18] out of which six were independent variables and one was a dependent variable. Then these metrics of the two versions were compared to one another individually whose statistics can be seen in Table I and Table II. CHANGE metric was calculated by taking the difference between the lines of code of the two versions of the software. The variables used were:

- Independent variables which were taken from Chidambar and Kemerer [11] are: Weighted Methods per Class (WMC), Number of Children (NOC), Coupling between Classes (CBO), Depth in Inheritance (DIT)
- Independent variables which were taken from traditional metrics are: Lines of Code (LOC), McCabe's Cyclomatic Complexity (CC).
- The dependent variable used was CHANGE which is defined as the number of lines of code that were modified, added and deleted from version 1 to version 2 of the studied software.

A few observations made from table I and table II are:

- It was observed that mean and standard deviation of DIT of both the versions was lesser than 1 hence, it can be concluded that use of inheritance in both the versions is less.
- It was also observed that coupling between classes was more for version 2 than for version 1 as the value for CBO is greater for version 2.
- It can be seen that complexity of version 2 is more than version 1 as both values of CC and NOC are greater for version 2.
- It can also be observed that there is a significant change in the mean and standard deviation of LOC of the two versions hence it can be concluded that a large number of lines were changed in version 2 of the software.

V. RESULTS AND ANALYSIS

This section discusses the measures used to analyse theprediction of software maintainability, the results obtained from this study and its analysis.

A. Measures used to analyse software maintainability

Various measures have been suggested to analyse the accuracy of prediction of a model. All these measures are based on the predicted value and the actual value. The actual and predicted values of the three models proposed in this study were calculated by using the Nueroshell 2 tool [19]. The measures used in this study are described below:

- Magnitude of Relative Error (MRE): It is measured by taking the absolute value of the difference between the actual value and the predicted value as given by Kitchenham [12]. The formula for this measure is:

$$MRE = \frac{|ActualValue - PredictedValue|}{ActualValue} \quad (1)$$

- Mean Magnitude of Relative Error (MMRE): MMRE is the mean of MRE as proposed by Conte, Dunsmore and Shen [13]. The formula for this measure is:

$$MMRE = \sum_{i=1}^{N} MRE_i \quad (2)$$

- Pred: Pred is measured by the predicted values whose MRE is less than or equal to a specified value. This was proposed by Fentom. [14]. The formula for this measure is given in (3), where k is the number of predicted values which are less than or equal to the specified value, q is the specified value and N is the total number of cases.

$$Pred(q) = \frac{K}{N} \quad (3)$$

B. Results obtained

Table III shows the results which were obtained by this study. Values of MRE, MMRE, Pred (0.25) and Pred (0.75) were calculated to compare the three proposed models in this study.

TABLE III COMPARISON OF PROPOSED MODELS

Models Used	**Measures**			
	Max. MRE	**MMRE**	**Pred (0.25)**	**Pred (0.75)**
GMDH	3.42656	0.3340	0.2894	0.5263
GRNN	2.40739	**0.3094**	**0.2987**	**0.5526**
PNN	3.05611	0.3353	0.2631	**0.5526**

C. Result Analysis

From table III it can be observed that the GRNN has the least MMRE hence it can be concluded that GRNN is the best model out of the three proposed models in the study. It can also be observed that values of Pred (0.25) and Pred (0.75) are the highest in GRNN among the three proposed models hence the prediction accuracy of GRNN is the best. These observations make it evident that GRNN is the best suited model for prediction of software maintainability.

VI. CONCLUSION

This study quantitatively evaluates the prediction capability of three neural network based algorithms. It compares the three proposed models which were GMDH, GRNN and PNN on the basis of four measures: MRE, MMRE, Pred (0.25) and Pred (0.75) from which it was concluded that out of the three models GRNN gives the best results as GRNN has the least value for MMRE and maximum values for Pred (0.25) and Pred (0.75). This study hence concludes that GRNN is the best model for prediction of software maintainability.

Future work may contain additional quantitative studies on different datasets so as to ensure the full potential of GRNN. Another direction of study may suggest combining of GRNN model with other data mining models to develop a prediction model which would more accurately predict software maintainability.

ACKNOWLEDGMENT

The authors would like to acknowledge University School of Information and Communication Technology, Dwarka for support in development of this work.

REFERENCES

[1]. IEEE Standard. 1219-1993 IEEE Standard for Software Maintenance. INSPEC Accession Number: 4493167 DOI: 10.1109/IEEESTD.1993.11557 Journal .1993.
[2]. S. Chidamber, D. Darcy, C. Kemerer, “Managerial use of metrics for object-oriented software: An Exploratory Analysis,” IEEE Transactions on Software Engineering, vol. 24, no. 8, August 1998.
[3]. W. Li and S.Henry, “Another Metric Suite for Object-oriented Programming,” The Journal of System and Software, vol 44, no: 2, pp 155–162, December 1998.
[4]. D.Stavrinoudis, M. Xenos, D.Christodoulakis, “Relation between software metrics and maintainability,” Proceedings of the FESMA99 International Conference, Federation of European Software Measurement Associations, Amsterdam, The Netherlands, pp. 465-476, 1999.
[5]. M. Thwin and T. Quah, "Application of neural networks for software quality prediction using object oriented metrics," Journal of Systems and Software, vol. 76, no. 2, pp. 147-156, 2005.
[6]. S. Misra, "Modeling design/coding factors that drive maintainability of software systems,”Software Quality Journal, vol. 13, no. 3, pp. 297-320, 2005.
[7]. A. G. Ivakhnenko, “Group method of data handling- A Rival of the method of Stochastic Approximation,” Soviet Automatic Control,vol 13, no. 3, 43-71, 1966.
[8]. D.F. Specht , “Probabilistic neural networks”, Journal of Neural Networks, Elsevier, vol. 3, no 1, pp 109-118, DOI : .org/10.1016/0893-6080(90)90049-Q, 1990.
[9]. D.F. Specht, “A general regression neural network,” IEEE Transactions on neural networks, vol 2, no. 6, 1991.
[10]. T. J. McCabe, “A complexity measure,” IEEE Transactions on software engineering, vol. se-2, no. 4, 1976.
[11]. S. Chidamber and C. Kemerer, "A metrics suite for object oriented design," IEEE Transactions on Software Engineering, vol. 20, no. 6, pp. pp. 476-493, 1994.
[12]. B.A. Kitchenham, L.M. Pickard, S.G. MacDonell, M.J. Shepperd,” What accuracy statistics really measure,” IEEE Proceedings-Software vol. 148, no. 3, pp 81–85, 2001.
[13]. S. Conte, H. Dunsmore, and V. Shen,” Software engineering metrics and models”. Book, Menlo Park, CA: Publisher: Benjamin-Cummings publishing co., ISBN: 0-8053-2162-4, 1986.
[14]. N.E. Fentom, S.L. Pfleeger, “Software metrics: A rigorous and practical approach, second edition,” PSW publishing Company, 1997.
[15]. A Kaur, K Kaur, R Malhotra, “Soft computing approaches for prediction of software maintenance effort,” International Journal of Computer Applications, vol 1, no. 16, pp: 0975 – 8887, 2010.
[16]. MO. Elish and KO. Elish “Application of TreeNet in predicting object-oriented software maintainability: A comparative study,” European Conference on Software Maintenance and Reengineering, pp 1534-5351, DOI 10.1109/CSMR.2009.57, 2009.
[17]. http://sourceforge.net/projects/naturalcli/files/
[18]. http://sourceforge.net/projects/cccc/
[19]. http://www.wardsystems.com/