Neuro – Genetic Approach for Predicting Maintainability Using Chidamber and Kemerer Software Metrics Suite

Lov Kumar and Santanu Ku. Rath

Dept. CS&E
NIT Rourkela, India
lovkumar505@gmail.com,
skrath@nitrkl.ac.in

Abstract. Accurate estimation of attributes such as effort, quality and risk is of major concern in software life cycle. Majority of the approaches available in literature for estimation are based on regression analysis and neural network techniques. In this study, Chidamber and Kemerer software metrics suite has been considered to provide requisite input data to train the artificial intelligence models. Two artificial intelligence (AI) techniques have been used for predicting maintainability viz., neural network and neuro-genetic algorithm (a hybrid approach of neural network and genetic algorithm). These techniques are applied for predicting maintainability on a case study i.e., Quality Evaluation System (QUES) and User Interface System (UIMS). The performance was evaluated based on the different performance parameters available in literature such as: Mean Absolute Relative Error (MARE), Mean absolute error (MAE), Root Mean Square Error (RMSE), and Standard Error of the Mean (SEM) etc. It is observed that the hybrid approach utilizing Neuro-GA achieved better result for predicting maintainability when compared with that of neural network.

Keywords: Artificial neural network, CK metrics suite, Maintainability, Genetic algorithm.

1 Introduction

Software quality is considered as one of the most important parameter of software development process. Software quality attributes that have been identified by ISO/IEC 9126 [7] are efficiency, functionality, maintainability, portability, reliability and usability. In recent years, maintainability plays a high priority role for achieving considerable success in software system and it is considered as essential quality parameter. The ISO/IEC 9126 [7] standard defines maintainability as the capability of the software product to be modified, including adaptation or improvements, corrections of the software to changes in environment and in requirements and functional specifications. In this paper, Maintainability is considered as the number of source of lines changed per class. A line change can be an 'addition' or 'deletion' of lines of code in a class [9].

H. Unger et al. (eds.), *Recent Advances in Information and Communication Technology 2015*, Advances in Intelligent Systems and Computing 361, DOI: 10.1007/978-3-319-19024-2_4

Some of the Object-Oriented software metrics available in literature are as follows: Abreu MOOD metrics suite [1], Briand *et al.* [8], Halstead [6], Li and Henry [9], McCabe [11], Lorenz and Kidd [10] and CK metrics [4] suite etc. In this paper, CK metrics suite has been considered to provide requisite input data to design the models for predicting maintainability using ANN with Gradient descent learning method [2] and hybrid approach of ANN and genetic algorithm i.e., Neuro-genetic (Neuro-GA) [3]. CK metrics suite is a six metrics set. The important aspect of CK metrics suite is that it covers most of the feature of the Object-Oriented software i.e, size, cohesion, coupling and inheritance. WMC show size or complexity of class, DIT and NOC show the class hierarchy, CBO and RFC show class coupling and LCOM show cohesion.

The remainder of the paper is organized as follows: Section 2 shows the related work in the field of software maintainability and Object-Oriented metrics. Section 3 briefs about the methodologies used to predicting maintainability. Section 4 emphasizes on mining of CK metrics values from data repository. Section 5 highlights on the results for effort estimation, achieved by applying ANN and Neuro-GA techniques. Section 6 gives a note (comparison) on the performance of the designed models based on the performance parameters. Section 8 concludes the paper with scope for future work.

2 Related Work

It is observed in literature that software metrics are used in design of prediction models which serve the purpose of computing the prediction rate in terms of accuracy such as fault, effort, re-work and maintainability. In this paper, emphasis is given on work done on the use of software metrics for maintainability prediction. Table 1 shows the summary of literature review done on Maintainability, where it describes the applicability of numerous software metrics used by various researchers and practitioners in designing their respective prediction models. From table 1, it can be interpreted that many of the authors have used statistical methods such as regression based analysis and their forms in predicting the maintainability. But keen observation reveals that very less work has been carried out on using neural network models for designing their respective prediction models. Neural network models act as efficient predictors of dependent and independent variables due to sophisticated modeling technique where in they posses the ability to model complex functions. In this paper, two artificial intelligence techniques are applied to design the model to estimate the maintainability of the software product using CK metrics suite.

3 Proposed Work for Predicting Maintainability

This section highlight on the use of artificial intelligence techniques (AI) such as Artificial neural network (ANN) with Gradient descent learning method [2], hybrid approach of Artificial neural network (ANN) and genetic algorithm i.e., Neuro-genetic (Neuro-GA) [3] for predicting maintainability.

Table 1. Summary of Empirical Literature on Maintainability

Author	Prediction technique
Li and Henry (1993) [9]	Regression based models
Paul Oman (1994) *et al.* [13]	Regression based models
Don Coleman (1994) *et al.* [5]	Aggregate complexity measure, Factor analysis, Hierarchical multidimensional assessment model, polynomial regression models , principal components analysis.
Scott L. Schneberger (1997) [14]	Regression based models
Van Koten *et al.* (2006) [15]	Bayesian Network, Backward Elimination and Stepwise Selection, Regression Tree.
Yuming Zhou and Hareton Leung (2007) [16]	Artificial neural network, Multivariate linear regression, Multivariate adaptive regression splines, Regression tree and Support vector regression

3.1 Artificial Neural Network (ANN) Model

ANN is used for solving problems such as classification and estimation [12]. In this study, ANN is being used for predicting maintainability using CK metrics.

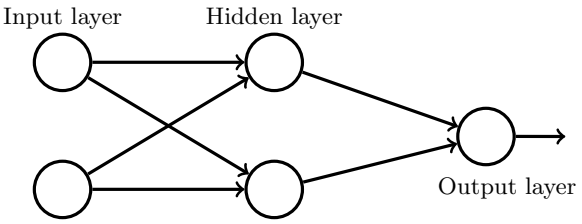

Fig. 1. Artificial neural network

Figure 1 shows the architecture of ANN, which contains three layers i.e., input layer, hidden layer and output layer. Here, for input layer, linear activation function is used and for hidden layer and output layer, sigmoidal function or squashed-S function is used.

A neural network can be represented as:

$$Y' = f(W, X) \tag{1}$$

where X is the input vector, $Y^{'}$ is the output vector, and W is the weight vector.

Gradient Descent Learning Method. Gradient descent learning method is one of the methods for updating the weights during learning phase [2]. It uses first-order derivative of total error to find the minima in error space. Normally

Gradient vector G is defined as the 1st order derivative of error function E_k, and the error function is represented as:

$$G = \frac{\mathrm{d}}{\mathrm{d}W}(E_k) = \frac{\mathrm{d}}{\mathrm{d}W}(\frac{1}{2}(y_k' - y_k)^2) \tag{2}$$

After obtaining the value of gradient vector G in each iteration, weighted vector W is updated as:

$$W_{k+1} = W_k - \alpha G_k \tag{3}$$

where W_{k+1} is the updated weight, W_k is the current weights, G_k is gradient vector, α is the learning constant, y and y$'$ are the actual and expected output respectively.

3.2 Neuro-Genetic (Neuro-GA) Approach

Neuro-genetic (Neuro-GA) is a hybrid approach of ANN and genetic algorithm. In Neuro-GA, genetic algorithm is used for updating the weight during learning phase. A neural network with a configuration of 'l-m-n' is considered for estimation. The number of weights N required for this network can be computed using the following equation:

$$N = (l + n) * m \tag{4}$$

with each weight (gene) being a real number and assuming the number of digits (gene length) in weights to be d. The length of the chromosome L is computed using the following equation:

$$L = N * d = (l + n) * m * d \tag{5}$$

For determining the fitness value of each chromosome, weights are extracted from each chromosome using the following equation:

$$W_k = \begin{cases} -\frac{x_{kd+2}*10^{d-2}+x_{kd+3}*10^{d-3}+....+x_{(k+1)d}}{10^{d-2}} & \text{if } 0 <= x_{kd+1} < 5 \\ +\frac{x_{kd+2}*10^{d-2}+x_{kd+3}*10^{d-3}+....+x_{(k+1)d}}{10^{d-2}} & \text{if } 5 <= x_{kd+1} <= 9 \end{cases} \tag{6}$$

The fitness value of each chromosome is computed using following equation:

$$\boxed{F_i = \frac{1}{E_i} = \frac{1}{\sqrt{\frac{\sum_{j=1}^{j=N} E_j}{N}}} = \frac{1}{\sqrt{\frac{\sum_{j=1}^{j=N} T_{ji} - O_{ji}}{N}}}} \tag{7}$$

where N is the total number of training data set. T_{ji} and O_{ji} are the estimated and actual output of input instance j for chromosome i.

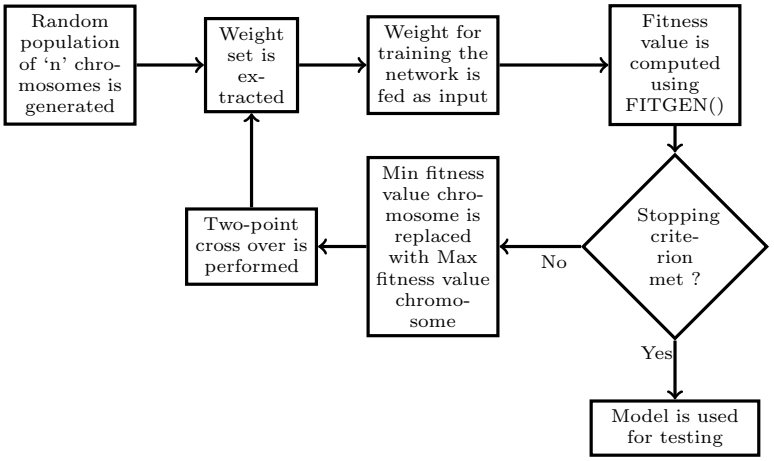

Fig. 2. Flow chart representing Neuro-GA execution

Figure 2 shows the block diagram for Neuro-GA approach. This block diagram represents the steps followed to design the model using Neuro-GA approach.

4 Metrics Set and Empirical Data Extraction

Metrics suites are defined for different goals such as effort estimation, fault prediction, re-usability and Maintainability. In this paper, CK metrics suite has been considered for Maintainability estimation i.e., the number of lines changed per class is considered as a criterion is determining the Maintainability. A line change can be addition or deletion of code in a class. The CK metrics values were extracted using Chidamber and Kemerer Java Metrics (CKJM) tool. This tool extracts Object-Oriented metrics by processing the byte code of compiled Java classes. Hence in this approach, Maintainability for a class is considered as a dependent variable and each of the CK metrics is an independent variable. In this analysis, we disregarded the CBO metric of the CK metrics suite for computing maintainability as it measures non-inheritance related coupling" [9]. Maintainability is assumed to be a function of WMC, DIT, NOC, RFC and LCOM. It is represented as:

$$Maintainability = Change = f(WMC, DIT, NOC, RFC, LCOM) \tag{8}$$

4.1 Case Study

In this paper, to analyze the effectiveness of the proposed approach, two commercial software products were used as case studies. Softwares such as Quality

Table 2. Descriptive statistics of classes for UIMS and QUES

	UIMS						QUES					
	WMC	NOC	DIT	RFC	LCOM	CHANGE	WMC	NOC	DIT	RFC	LCOM	CHANGE
Max.	69	8	4	101	31	289	83	0	4	156	33	217
Min.	0	0	0	2	1	2	1	0	0	17	3	6
Median	5	0	2	17	6	18	9	NA	2	40	5	52
Mean	2.15	11.38	0.94	23.20	7.48	46.82	14.95	0	1.91	54.38	9.18	62.18
Std Dev.	15.89	2.01	0.90	20.18	6.10	71.89	17.05	0	0.52	32.67	7.30	42.09

Evaluation System (QUES) and User Interface System (UIMS) are chosen for computing the Maintainability using Neuro-GA and ANN approach. The softwares systems viz., UIMS and QUES had 39 and 71 classes respectively. Table 2 shows the statistical analysis of UIMS and QUES for CK metrics indicating Max, Min, Median and Standard deviation.

5 Results

In this section, the relationship between value of metrics and the maintainability of class (changes in class). Six CK metrics are considered as input nodes and the output is the computed maintainability. The number of hidden nodes vary from six to thirty three. In this paper, 10-fold and 5-fold cross-validation concept has been considered in QUES and UIMS for comparing the models i.e., each fold contain nearly seven number of data samples. True error and estimate of true error determine the suitable model to be chosen for predicting maintainability. The most suitable number of hidden node in each fold is chosen on the basis of minimum deviation between true error and the estimate of true error. The number of hidden node in final model chosen for predicting maintainability is based on the median values of the hidden nodes in their respective folds.

The following sub-sections give a brief implementation details of the applied neural network techniques.

5.1 Artificial Neural Network (ANN) Model

In this paper, three layers of ANN are considered, in which five nodes act as input nodes, the number of hidden nodes vary from six to thirty three and one node acts as an output node. The network is trained using Gradient descent learning method unless and until the neurons achieve the threshold value of 'MSE' or reach maximum iteration limit (of 2000 epochs). Table 3 shows the various performance parameters which were used to evaluate the best suitable model to be designed for maintainability estimation. From Table 3 it can be interpreted that the high value of 'r' is indicate the pearson's correlation between the actual maintainability and estimated maintainability. Figure 3 shows the variance of MSE vs Number of iterations.

Table 3. Performance matrix

	r	Epochs	MRE	MARE	SEM
UIMS	0.8624	578	0.1820	0. 6931	0.0391
QUES	0.8674	2000	0.1580	0.4384	0.0184

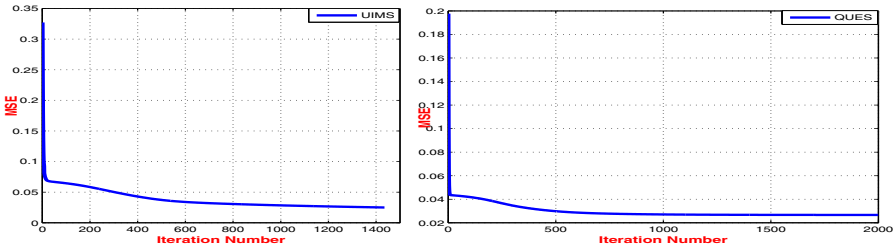

Fig. 3. MSE Vs No. of Iterations (epoch)

5.2 Neuro-Genetic (Neuro-GA) Approach

In this paper, 5-n-1 configuration of neural network is considered (5 numbers of input neurons, n numbers of hidden neurons, 1 output neuron). The total number of weights used in 5-n-1 configuration model are determined using equation 4 i.e., $(5+1)*n = 6*n$ (where 'n' represents the number of hidden nodes varying from six to thirty three), each weight is considered as a gene of length 5, so the length of one chromosome is calculated using equation 5 i.e., L=(5 + 1) * n * 5 = 30 * n.

In this study a population of size 50 is considered i.e., initially 50 chromosomes are randomly generated. The input-hidden layer and hidden-output layer weights of the network are computed using equation 6. Two-point cross-over operation is performed on the generated population. The execution of the algorithm converges when 95% of the chromosomes achieve same fitness values or reach maximum iteration limit (of 200 epochs).

Table 4 shows the various performance parameters which were used to evaluate the best suitable model to be designed for maintainability estimation. From Table 4 it can be interpreted that the high value of 'r' is indicate the pearson's correlation between the actual effort and estimated maintainability. Figure 4 shows the variance of number of chromosomes having same fitness value and generation number of UIMS and QUES.

Table 4. Performance matrix

	r	Epochs	MRE	MARE	SEM
UIMS	0.8108	76	0.1553	0.5331	0.0258
QUES	0.8227	74	0.1480	0.4180	0.0155

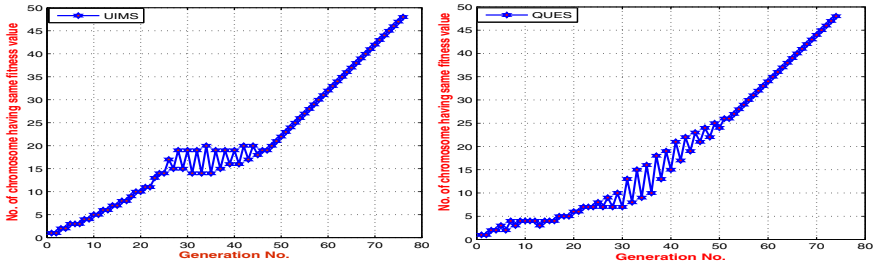

Fig. 4. MSE Vs Generation No.

6 Comparison of Models

Figure 5 shows the Pearson residual boxplots for neural network and Neuro-GA, allowing a visual comparison. The line in the middle of each box represents the median of the Pearson residual. Of two analysis, Neuro-GA in both UIMS and QUES has the narrowest box and the smallest whiskers, as well as the few number of outliers. Based on these boxplots, it is evident that Neuro-GA gave best estimation accuracy as compared to neural network with gradient descent algorithm.

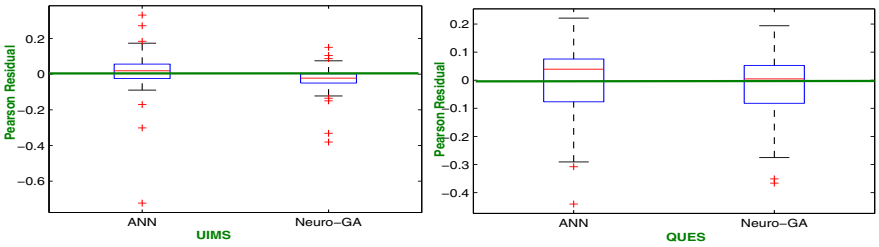

Fig. 5. Residual boxplot for UIMS and QUES

Apart from the comparative analysis done to find the suitable model which can predict the best software maintainability, this paper also makes the comparison of the proposed work with the work done by Yuming Zhou *et al.* [16] and Van Koten *et al.* [15]. Yuming Zhou *et al.* [16] and Van Koten *et al.* [15] have used same dataset i.e., UIMS and QUES for predicting maintainability based on different regression and neural network models . They have considered 'MMRE' as a performance parameter to compare the models designed for predicting maintainability of Object-Oriented software systems. Table 5 shows the MMRE value of the proposed work and the work done by Yuming Zhou *et al.* [16] and Van Koten *et al.* [15]. From Table 5, it can be observed that, in case of QUES software MMRE value is almost same but in case of UIMS software, the proposed approach obtained better prediction rate for maintainability.

Table 5. Performance based on MMRE for UIMS and QUES

MMRE			
Author	Technique	UIMS	QUES
Van Koten *et al.*[15]	Bayesian Network	0.972	0.452
	Regression Tree	1.538	0.493
	Backward Elimination	2.586	0.403
	Stepwise Selection	2.473	0.392
Zhou *et al.* [16]	Multivariate linear regression	2.70	0.42
	Artificial neural network	1.95	0.59
	Regression tree	4.95	0.58
	SVR	1.68	0.43
	MARS	1.86	0.32
	ANN	0.6931	0.4384
	Neuro-GA	0.5332	0.4180

7 Threats to Validity

Several issues affect the results of the experiment are:

- Two Object-Oriented systems, i.e., UIMS and QUES used in this study are design in ADA language. The models design in this study are likely to be valid for other Object-Oriented programing language, i.e., Java or C++. further research can extend to design a model for other development paradigms.
- In this study, only eleven set of software metrics are used to design a models. Some of the metrics which are widely used for Object-Oriented software are further considered for predicting maintainability.
- we only consider AI techniques for designing the prediction models to predict maintainability. Further, we can extend the work to reduce the feature using feature reduction techniques, i.e., PCA, RST, statistical test etc..

8 Conclusion

In this paper, an attempt has been made to use CK metrics suite in order to estimate software maintainability using gradient descent and hybrid approach of neural network and genetic algorithm. These approaches have the ability to predict the output based on historical data. The CK metrics are considered as input data to train the network and predicting software maintainability. The results reveal that the hybrid approach of Neuro-GA prediction model obtained low values of MAE, MARE, and RMSE when compared with those of gradient descent prediction model.

Further, work can be replicated by using hybrid approach of neural network and fuzzy logic. Also feature reduction techniques such as rough set and principal component analysis can be applied to minimize the computational complexity of the input data set.

References

1. Abreu, F.B.E., Carapuca, R.: Object-oriented software engineering: Measuring and controlling the development process. In: Proceedings of the 4th International Conference on Software Quality, vol. 186 (1994)
2. Battiti, R.: First and second-order methods for learning between steepest descent and newton's method. Neural Computation 4(2), 141–166 (1992)
3. Burgess, C., Lefley, M.: Can genetic programming improve software effort estimation. Information and Software Technology 43, 863–873 (2001)
4. Chidamber, S.R., Kemerer, C.F.: A metrics suite for object-oriented design. IEEE Transactions on Software Engineering 20(6), 476–493 (1994)
5. Coleman, D., Ash, D., Lowther, B., Oman, P.: Using metrics to evaluate software system maintainability. IEEE Computer 27(8), 44–49 (1994)
6. Halstead, M.: Elements of Software Science. Elsevier Science, New York (1977)
7. Jung, H.W., Kim, S.G., Chung, C.S.: Measuring software product quality: A survey of iso/iec 9126. IEEE Software 21(5), 88–92 (2004)
8. Briand, L.C., Wust, J., Daly, J.W., Porter, D.V.: Exploring the relationships between design measures and software quality in object-oriented systems. The Journal of Systems and Software 51(3), 245–273 (2000)
9. Li, W., Henry, S.: Maintenance metrics for the object-oriented paradigm. In: Proceedings of First International Software Metrics Symposium, pp. 88–92 (1993)
10. Lorenz, M., Kidd, J.: Object-Oriented Software Metrics. Prentice-Hall, Englewood (1994)
11. McCabe, T.J.: A complexity measure. IEEE Transactions on Software Engineering 2(4), 308–320 (1976)
12. McCulloch, W., Pitts, W.: A logical calculus of ideas immanent in nervous activity. Bulletin of Mathematical Biophysics 5(4), 115–133 (1943)
13. Oman, P., Hagemeister, J.: Construction and testing of polynomials predicting software maintainability. Journal of Systems and Software 24(3), 251–266 (1994)
14. Schneberger, S.L.: Distributed computing environments: effects on software maintenance difficulty. Journal of Systems and Software 37(2), 101–116 (1997)
15. Van Koten, C., Gray, A.: An application of bayesian network for predicting object-oriented software maintainability. Information and Software Technology 48(1), 59–67 (2006)
16. Zhou, Y., Leung, H.: Predicting object-oriented software maintainability using multivariate adaptive regression splines. Journal of Systems and Software 80(8), 1349–1361 (2007)