IJCSC International Journal of Computer Science and Communication Vol. 3, No. 1, January-June 2012, pp. 33-35

DEFECT PREDICTION BASED ON QUANTITATIVE AND QUALITATIVE FACTORS USING PSO OPTIMIZED NEURAL NETWORK

Kanu Sharma[1], Navpreet Kaur[2], Sunil Khullar[3] and Harish Kundra[4]

[1]Student RIEIT Ropar.

[2]Lect.RIEIT Ropar.

[3]Lect.RIEIT Ropar, *E-mail: sunilkhullar222@yahoo.co.in*

[4]A.P. RIEIT Ropar.

ABSTRACT

An Artificial Neural Network (ANN) is an information-processing paradigm that is inspired by the way a biological nervous system in human brain works. Large number of neurons present in the human brain forms the key element of the neural network paradigm and act as elementary processing elements. Particle Swarm Optimization (PSO) is arguably one of the most popular nature-inspired algorithms for real parameter optimization at present. The existing theoretical research on PSO focuses on the issues like stability, convergence, and explosion of the swarm. In this paper, we have done the training of the neural network with the help of PSO technique for Defect Prediction Based on Quantitative and Qualitative Factors. Data is taken from NASA named PC1 fault dataset. Norman Fenton, et al in developed a causal model that includes such process factors, both quantitative and qualitative factors [8]. A measurable and precise definition of what faults are makes it possible to accurately identify and count them, which in turn allows the formulation of models relating fault counts and types to other measurable attributes of a software system. The incomplete and ambiguous nature of current fault definitions adds a noise component to the inputs used in modeling fault content, If this noise component is sufficiently large, any attempt to develop a fault model will produce invalid results. With modern configuration management tools, the identification and counting of software faults can be automated [11].

Keywords: Particle swarm optimization, artificial neural network, NASA, fault prediction.

1. INTRODUCTION

Bellini et al (2005) compared Fault-Proneness Estimation Models and concluded that over the last years, software quality has become one of the most important requirements in the development of systems and fault-proneness estimation could play a key role in quality control of software products. Their main objective was to find a compromise between the fault-proneness estimation rate and the size of the estimation model in terms of number of metrics used in the model itself [10].

The main aim of the work is Defect Prediction Based on Quantitative and Qualitative Factors Using PSO Optimized Neural Network. The objectives of the work are as follows:

(a) Study and Collection of the dataset of Software Defect Prediction Based on Quantitative and Qualitative factors.

(b) Study the Performance of PSO optimized Neural Network for the Software Defect Prediction.

(c) The performance criterion for the above algorithm is Accuracy, Mean absolute error (MAE), Root mean squared error (RMSE) Values.

2. METHODOLOGY

The methodology consists of the following steps:

1. Find the Qualitative and Quantitative attributes of software systems:

   The first step is to find the find the Qualitative and Quantitative attributes of software systems i.e. software metrics.

2. Select the suitable metric values as representation of statement:

3. Explore the PSO based Neural network for modeling of the software fault prediction in software systems.

2.1. PSO Trained Neural Network System

The following are the steps for the hybrid PSO-Neural Network based Modeling system:

- Designing of Neural Network and Perform Training: In this step the following three sub steps are there:
  - Calculate the minimum and maximum values in the attribute of input and setting the various parameters of feed-forward back-propagation network.
    - Size of the feed-forward back-propagation Neural Network.

- Type of Transfer function of each layer to be used.
- Type of Back-propagation network training function.
- Back-propagation weight/bias learning function.

– Generate the Neural Network.

– Perform the training of the Neural Network with PSO technique discussed after the testing phase using the training dataset.

– ***Testing phase:*** In this step the PSO trained Neural Network is evaluated against the testing data on the different criteria is discussed in the next step.

3. COMPARISON OF ALGORITHMS

The comparisons are made on the basis of the more accuracy and least value of *MAE* and *RMSE* error values. *Accuracy value* of the prediction model is the major criteria used for comparison.

- ***Mean Absolute Error:*** Mean absolute error, MAE is the average of the difference between predicted and actual value in all test cases; it is the average prediction error [9]. The formula for calculating MAE is given in equation 1.

$$\frac{|a_1 - c_1| + |a_2 - c_2| + \ldots + |a_n - c_n|}{n} \quad (1)$$

Assuming that the actual output is *a*, expected output is *c*.

- ***Root mean-squared error:*** RMSEis frequently used measure of differences between values predicted by a model or estimator and the values actually observed from the thing being modeled or estimated [9]. It is just the square root of the mean square error as shown in equation 2.

$$\sqrt{\frac{(a_1 - c_1)^2 + (a_2 - c_2)^2 + \ldots + (a_n - c_n)^2}{n}} \quad (2)$$

The mean absolute error and root mean squared error is calculated for each machine learning algorithm i.e. various algorithms for Neural Networks.

4. CONCLUSION

Qualitative factors which are taken into account are as follows:

The Quantitative factors are grouped under five topics [10]:

- Specification and Documentation process.
- New Functionality.
- Design and Development process.
- Testing and Rework.
- Project Management.

TRAINPSO type of Back-propagation network training function is used as TRAINPSO is a network training function that updates weight and bias values according to particle swarm optimization.

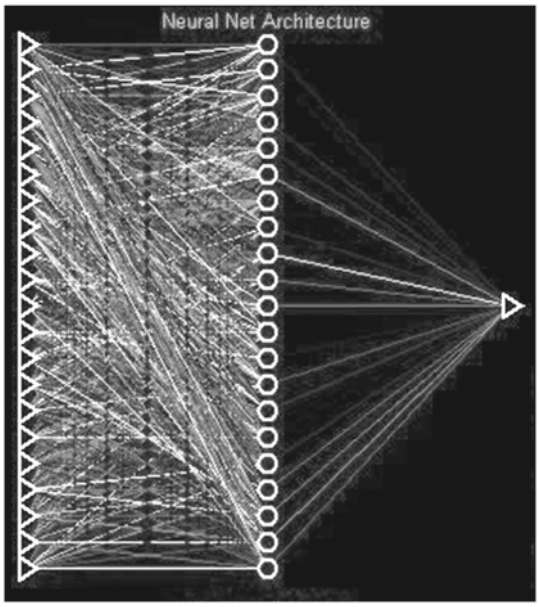

Figure 1: Architecture of Feed Forward Neural Network.

After the training, the Architecture of Feed Forward Neural Network is shown in Figure 1 where the Bright Green line shows more positive weight, bright red line shows more negative weight and dashed white line shows zero weight i.e. no connection between the neuron.

The proposed system shows high value of prediction accuracy. The value of accuracy is calculated is 85.9333%.

The system also shows very less values of MAE and RMSE calculated as error 0.1385 and 0.5819 respectively. It is therefore, concluded the PSO trained neural network model a potential algorithm for classification of the fault prone modules from the faultless modules of the software systems while considering the qualitative and quantitative factors.

REFERENCES

[1] Fenton N.E. and Neil M. (1999), "A Critique of Software Defect Prediction Models", *IEEE Transactions on Software Engineering*, **25**, Issue: 5, pp. 675-689.

[2] Runeson, Wohlin C. and Ohlsson M.C. (2001), "A Proposal for Comparison of Models for Identification of Fault-Proneness", *Journal of System and Software*, **56**, Issue: 3, pp. 301-320.

[3] Guo L., Cukic B. and Singh H. (2003)., "Predicting Fault Prone Modules by Dempster-Shafer Belief Networks", *In Proceedings of the 18th International Conference on Automated Software Engineering*, Canada, pp. 249-252.

[4] Ma Y. and Guo L. (2006), "A Statistical Framework for the Prediction of Fault-Proneness", *Product Focused Process Improvement*, Edition: First, Publisher: Springer Berlin/Heidelberg, pp. 204-214.

[5] Pigoski M. and Nelson E. (1994), "Software Maintenance Metrics: A Case Study", *Proceedings of IEEE Conference on Software Maintenance*, Canada, pp. 392-401.

[6] Jiang Y., Cukic B. and Menzies T. (2007), "Fault Prediction Using Early Lifecycle Data", ISSRE 2007, the *18th IEEE Symposium on Software Reliability Engineering, IEEE Computer Society*, Sweden, pp. 237-246.

[7] Dav'e N. and Krishnapuram R. (1997)., "Robust Clustering Methods: A Unified View", *IEEE Transactions on Fuzzy Systems*, **5**, Issue: 2, pp. 270-293.

[8] Norman Fenton, Martin Neil, William Marsh, Peter Hearty, Lukasz Radlinski, Paul Krause, "Project Data Incorporating Qualitative Factors for Improved Software Defect", *Proceedings of the PROMISE Workshop*, Year: 2007.

[9] Lanubile F., Lonigro A., and Visaggio G. (1995) "Comparing Models for Identifying Fault-Prone Software Components", *Proceedings of Seventh International Conference on Software Engineering and Knowledge Engineering*, USA, pp. 12-19.

[10] Bellini P. (2005), "Comparing Fault-Proneness Estimation Models", *10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'05)*, China, pp. 205-214.

[11] Khoshgoftaar T.M. and Munson J.C. (1990), "Predicting Software Development Errors using Complexity Metrics", *IEEE Journal on Selected Areas in Communications*, **8**, Issue: 2, pp. 253 -261.