International Journal of Latest Trends in Engineering and Technology
Vol.(9)Issue(3), pp.197-199
DOI: http://dx.doi.org/10.21172/1.93.34
e-ISSN:2278-621X

OBJECT ORIENTED METRICS IN DEFECT PREDICTION

Gursimrat Singh[1] & Shelly Garg[2]

Abstract:-Testing of large systems is an example of a resource- and time-consuming activity. Applying equal testing and verification effort to all parts of a software system has become cost-prohibitive. Therefore, one needs to be able to identify fault-prone modules so that testing/verification effort can be concentrated on these classes. Fault prediction is a mechanism used in software development life cycle to reduce the software failure and is carried out mostly during initial planning to identify fault-prone modules. Fault prediction not only increases the quality of monitoring during software development but also gives suggestions for suitable verification and validation approaches that eventually lead to improvement of efficiency and effectiveness of fault prediction. In this paper we used ANN model for predicting defects in the system. Object-Oriented metrics are used as input for ANN model. The results show that Object-Oriented metrics are performed well for predicting fault-prone modules in a system.
Keywords: - Object-oriented metrics, ANN model

1. LITERATURE SURVEY

Aggarwal et al. [1] used OO metrics with an ANN model for quality estimation. They used maintenance effort as dependent variable in this experiment along with eight OO metrics as independent variables those are LCOM, NOC, DIT, WMC, RFC, DAC, MPC, NOM. software quality is predicted by estimating the number of lines changed per class. They used data from UIMS (User Interface System) which contains 39 classes and QUES (Quality Evaluation System) which contains 71 classes. They used Multilayer Feed Forward network. Mean Absolute Relative Error (MARE) is used as performance evaluation parameter. The ANN model demonstrated that they were able to estimate maintenance effort within 30 percent of the actual maintenance effort in more than 72 percent of the classes in the validate set, and with a MARE of 0.265.

Victor R. Basili et al. [2] perform experiment using OO metrics at the University of Maryland to check whether these metrics can be used as defect predictors or not. They used data from eight MIS projects which are based on same requirements.They used logistic regression to analyze the relationship between metrics and the fault-proneness of classes. They used both univariate and multivariate logistic regression. After this experiment they concluded that five out of six CK's OO metrics are useful in defect prediction.

Qinbao Song et al. [3] focuses on the classification of software components according to their defect-proneness i.e. to classify them as fault-prone or non-defect prone. They argued on how the attributes are used to build predictors than which particular attributes are used for defect prediction. They focused on which i.e. the selection of attribute data set used for learning or training. They designed a framework that includes a scheme evaluation and defect prediction. In scheme evaluation various learning or training strategies are used. In defect prediction stage, the best suited learning scheme is selected to build the prediction model. They used ROC (Receiver Operating Characteristic) as performance measurement parameter to calculate AUC (Area under Curve). They have used 12 different learning schemes to predict defect-prone modules. They come up with the result that different learning schemes are selected for different data sets.

Zimmermann et al. [4] calculated the performance of defect prediction for cross projects by using data from 12 projects (622 combinations). Among of these combinations, only 21 pairs resulted in efficient prediction performance. Data distributions of the initial and final projects are different which results in low prediction performance. It is expected that training and test data have the same distribution of data. This assumption is good for within-project prediction and may be not suited for cross-project prediction. Cross-project prediction can be indicted in two dimensions: the domain dimension, and the company dimension. Zimmermann et al. noticed that in many software companies may or may not provide local data for defect prediction as they are small or they do not have any past data. Zimmermann et al. observed the data sets from Firefox and IE. They experimented on these web browsers and found that Firefox data could predict for defects in IE very well, but vice versa was not true. They come up with the result that "building a model from a small population to predict a larger one is likely more difficult than the reverse direction".

Atchara Mahaweerawat et al. [5] introduced a new approach for predicting faults in object-oriented software systems. This model is based on supervised learning using neural network. MLP neural network with back-propagation learning algorithm has been used to identify faulty classes, while RBF neural network to categorize the faults according to fault types. Their experiment results show 90% accuracy for predicting faultiness of a module/class.

[1] Department of Computer Science Engineering, Student, Desh Bhagat University, Fatehgarh Sahib, Punjab,India
[2] Department of Electronics and Communication Engineering, Chandigarh group of college, Mohali, Punjab, India

Stefan Lessmann et al. [6] use metric-based classification for defect prediction. They proposed a defect prediction model which is based on the results of 22 different classification models applied over 10 public-domain data sets from the NASA Metrics Data (MDP) repository and the PROMISE repository. These comparisons are based on the area under the receiver operating characteristics curve (AUC). The selected classifiers are grouped into the categories such as statistical approaches, nearest-neighbor methods, neural networks, support vector machines, tree-based methods, and ensembles. They divide the data randomly as 2/3 for training and 1/3 for performance evaluation. Results show more than 0.7 AUC for most of the classifiers. They observed that sophisticated classifiers like RndFor, LS-SVMs, MLPs, and Bayesian networks are the best performing algorithms. Simple classifiers suffice to model the relation-ship between static code attributes and software defect.

Ying Ma et al. [8] proposed a novel transfer learning algorithm called Transfer Naive Bayes (TNB) for cross-company defect prediction. The advantage of transfer learning is that it allows that training and testing data to be heterogeneous. They have used instance-transfer approach in their research which assigns weights to source instances according to their contribution in the prediction model. They use four performance metrics, PD (Probability of Detection), PF (Probability of False Alarms), F-measure, and AUC to measure the performance of defect predictor. They show that the TNB gives good performance.

Barbara A. Kitchenham and Emilia Mendes [9] analyzed a cross-company data set of Web projects. The web projects may be Web Hypermedia or Web software application. They developed a cost estimation model based on these data sets. They used STATA^{TM} tool for the experiment, which uses regression. Their experiment results show that a cross-company effort estimation model can be constructed.

2. RESEARCH METHODOLOGY

There are many techniques which can be used for defect prediction such as Naïve Byes, LogReg, Random Forest, Nearest-neighbor, SVM, Machine Learning etc. In this research Feed Forward Neural Network (FFNN) is used for predicting the defects.

2.1 Neural Networks:-

Feed forward neural networks, trained with a back-propagation learning algorithm, are the most popular neural networks. They are applied to a wide variety of problems. A FFNN consists of neurons, which are ordered into layers.

The first layer is called the input layer, the last layer is called the output layer, and the layers between are hidden layers. The used FFNN model is shown in Figure-1.1. Each neuron in a particular layer is connected with all neurons in the next layer.

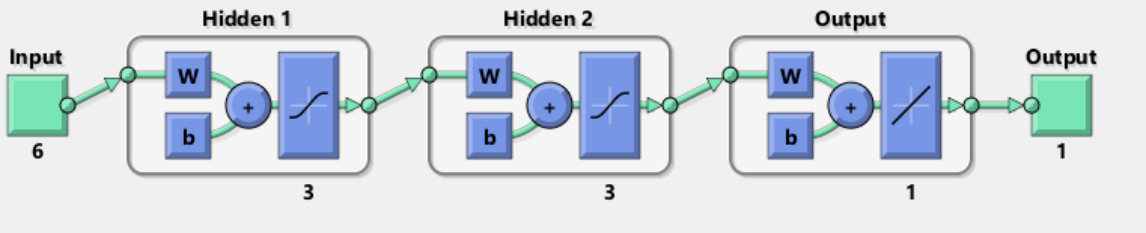

Fig. 1 A Multi-Layer Feed Forward Neural Network

Six neurons are used at input layer and 3 neurons at both the hidden layer. The 6 inputs are object-oriented metrics which are:-NOC, RFC, DIT, WMC, CBO, LCOM. [7]

The connection between the i^{th} and j^{th} neuron is characterized by the weight coefficient w_{ij}. The weight coefficient reflects the degree of importance of the given connection in the neural network. The output of a layer can be determined by equations

$$a = x_1w_1+x_2w_2+x_3w_3... +x_nw_n \quad (1)$$

Multi-layer perceptron method is used in ANN experiments. Multilayer perceptrons are feedforward neural networks trained with the standard back propagation algorithm. Feedforward neural networks provide a general framework for representing non-linear functional mappings between a set of input variables and a set of output variables. This is achieved by representing the nonlinear function of many variables in terms of compositions of nonlinear functions of a single variable, which are called activation functions.

In this research, following activation functions are used:-

1. Hyperbolic Tangent Sigmoid Function (tansig)
2. Linear Transfer Function (purelin)

tansig is used as activation function for hidden layer and purelin is used as activation function for the output layer.

2.2 Hyperbolic Tangent sigmoid Function (tansig)

It takes two real-valued arguments and transforms them to a range (-1, 1).

The equation used for tansig is

$$a = \text{tansig}(n) = \frac{2}{(1+e^{-2*n})}-1 \quad (2)$$

2.3 Linear Transfer Function (purelin)
Purelin(n) takes one input and returns value n.
The equation for purelin is
a = purelin(n) = n (3)

2.4 Gradient descent method:-
Gradient descent is one of the methods for updating the weights during learning phase. Gradient descent method 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 error function is represented as:

$$E_k = \frac{1}{2}(y_k^{'} - y_k)^2 \quad (4)$$

Gradient vector G is given as:

$$G = \frac{d}{dW}(E_k) = \frac{d}{dW}(\frac{1}{2}(y_k^{'} - y_k)^2) \quad (5)$$

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

$$W_{k+1} = W_k - \alpha G_k \quad (6)$$

where W_{k+1} is the updated weights, W_k is the current weights, G_k is gradient vector and α is the learning constant [10,11].

3. RESULT AND CONCLUSION:-
In this research Accuracy and Precision are used to evaluate the performance of the model. More accuracy means the model performs better. Precision refers to the closeness of two or more measurements to each other.
The results of proposed model for 1000 epochs are shown below in table in which the model is trained on one dataset and is tested on all datasets.

Table 1: Fault Detection Confusion Matrix of results obtained

	non-faulty	faulty
non-faulty	9	142
faulty	1	77

Our model gives accuracy of 58.52% and a precision rate of 25.71%.
After this research it can be easily identified that the object-oriented metrics give better results in the field of software defect prediction. There are lots of opportunity in this area to perform better.

4. REFERENCES:-

[1] Aggarwal, K. K., et al. "Application of artificial neural network for predicting maintainability using object-oriented metrics." Transactions on Engineering, Computing and Technology 15 (2006): 285-289.
[2] Basili, Victor R., Lionel C. Briand, and Walcélio L. Melo. "A validation of object-oriented design metrics as quality indicators." IEEE Transactions on software engineering 22.10 (1996): 751-761.
[3] Qinbao Song, Zihan Jia, Martin Shepperd, Shi Ying, and Jin Liu, "A General Software Defect-Proneness Prediction Framework", IEEE Trans. Software Engineering, Vol. 37, No. 3,2011.
[4] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, "Cross-project defect prediction: a large scale experiment on data vs. domain vs. process," in Proceedings of ESEC/FSE, 2009.
[5] Atchara Mahaweerawat, Peraphon Sophatsathit, Chidchanok Lursinsap and Petr Musilek, "Fault Prediction in Object-Oriented Software Using Neural Network Techniques", In proceedings of the InTech Conference, Huston, TX, USA ,Page No. 27-34.
[6] Stefan Lessmann, Bart Baesens, Christophe Mues, and Swantje Pietsch, "Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings", IEEE Trans. On Software Engineering, Vol. 34, No. 4, 2008
[7] S.R. Chidamber and C.F. Kemerer. "A metrics suite for object oriented design". IEEE Trans. On Software Engineering, 20(6):476–493, 1994.
[8] Ying Ma., Guangchun Luo Xue Zang and Aiguo Chen "Transfer Lerning for Cross-Company Software Defect Prediction" Information and Software Technology 54, Page No. 248–256,2012.
[9] B.A. Kitchenham, E. Mendes, and G.H. Travassos, "Cross versus Within-Company Cost Estimation Studies: A Systematic Review," IEEE Trans. Software Engineering, Vol. 33, No. 5, Page No. 316-329, 2007.
[10] Mitchell, T.M., 1997. Machine Learning, McGrawHill.
[11] Martin T. Hagan and Mohammad B. Menhaj, "Training Feedforward Networks with the Marquardt Algorithm", IEEE Trans. Software Engineering, Vol. 5, No. 6, 1994