ISSN:0975-9646

Impact of Training Function Based Neural Network on Reusable Software Modules

Anupama Kaur

Computer Science Department, Swami Vivekanand Institute of Engineering and Technology
Punjab, India

Abstract— **For the enhancement of quality in the software development, to eliminate the repeated work and to improve the efficiency, we require a effective solution that is software reuse. But how to identify and evaluate the performance of reusable software from the existing systems has remained the task for developers. Defining metrics to reusable components has given the structural analysis to the different procedures. Neural Network's training functions gives the oppurtutinity to evaluate the attributes of reusability and work as the automatic tool to define reusability of procedures by calculation based on training. Metrics's values turn as input dataset for the Neural Network systems. In this paper, different training functions of algorithms have been experimented and results are recorded in terms of Accuracy, Mean Absolute Error (MEA) and Root Mean Square Error (RMSE). The proposed model defined in the future scope can become a effective and efficient way to state the reusability of software components.**

Keywords— **Software Reusability, Software Metric, Neural Network, training function, Accuracy, MAE, RMSE**

I. INTRODUCTION

Reusable software components have been promoted in recent years. The software development community is gradually drifting toward the promise of widespread software reuse, in which any new software system can be derived virtually from the existing systems. There are two approaches for reuse of code: develop the code from scratch or identify and extract the reusable code from already developed code. With the existence of the software there is less uncertainty in the cost of reusing which is an important factor for project management as it reduces the margin of error in project cost estimation. This is particularly true when relatively large software components as sub-system reused.

Reusing software can speed up system production because both development and validation time should be reduced. Thus the reuse of software in systems development is a strategy that increases productivity and quality. Code reuse is the idea that a partial or complete computer program written at one time can be, should be, or is being used in another program written at a later time. The reuse of programming code is a common technique which attempts to save time and energy by reducing redundant work [5].

Tracz observed that for programmers to reuse software they must first find it useful [2]. Experimental results confirm that prediction of reusability is possible but it involves more than the set of metrics that are being used [3]. According to Poulin [4], in some sense, researchers have fully explored most traditional methods of measuring reusability: complexity, module size, interface characteristics, etc., but the ability to reuse software also depends on domain characteristics. It means we should concentrate on evaluating the software in terms of its relevancy to a particular domain [1].

Major challenge is to develop a robust framework for software reuse. There are two approaches for reuse of code: develop the reusable code from scratch or identify and extract the reusable code from already developed code. The organizations that has experience in developing software, but not yet used the software reuse concept, there exists extra cost to develop the reusable components from scratch to build and strengthen their reusable software reservoir [6]. The cost of developing the software from scratch can be saved by identifying and extracting the reusable components from already developed and existing software systems or legacy systems [7]. But the issue of how to identify reusable components from existing systems has remained relatively unexplored. In both the cases, whether we are developing software from scratch or reusing code from already developed projects, there is a need of evaluating the quality of the potentially reusable piece of software [1].

Neural networks have seen an explosion of interest over the years, and are being successfully applied across a range of problem domains, in areas as diverse as finance, medicine, engineering, geology and physics. Indeed, anywhere that there are problems of prediction, classification or control, neural networks are being introduced. It can learn by example. In order to make a neural network useful, the user needs to gather representative data**,** and then invokes training algorithms to train the neural network. Neural network learns about its environment through a set of input-output training samples and is an interactive process of adjustment applied to its synaptic weights and bias levels [8].

The learning algorithm involves the following steps:

- ❖ The neural network receives the normalized inputs that are available in the input-output training data samples.
- ❖ The output of the artificial neural network is then computed.
- ❖ The output of the network is then compared with that given in the training data samples. The error in the output is computed by taking the difference of the desired output and computed output from the network.

- ❖ The synaptic weights and biases are then changed so as to decrease the error based on the error gradient with respect to the different synaptic weights.

The process is repeated until the desired error goal is achieved [9].

II. Proposed Methodology

The following steps are proposed in the methodology:

A. *Selection of metric suit for function oriented paradigm*

A framework of metrics is proposed for structural analysis of procedure or function-oriented. The code of software is parsed to calculate the metric values. The following suits of metrics are able to target those the essential attributes of function oriented features towards measuring the reusability of software modules, so it tried to analyze, refine and use following metrics to explore different structural dimensions of Function oriented components [9].

The proposed metrics for Function Oriented Paradigm are as follows:

- ➢ Cyclometric Complexity Using Mc Cabe's Measure [10][11]
- ➢ Halstead Software Science Indicator [10] [12]
- ➢ Regularity Metric [10][12]
- ➢ Reuse-Frequency Metric [10][12]
- ➢ Coupling Metric [10]

B. *Design & evaluate neural network system*

The following four Neural Network algorithms are experimented:

- ◆ Fletcher–Reeves Update Conjugate Gradient (FRUCG) algorithm
- ◆ Polak–Ribiere Update Conjugate Gradient (PRUCG) algorithm
- ◆ Powell-Beale Restarts Conjugate Gradient (PBRCG) algorithm
- ◆ Scaled Conjugate Gradient (SCG) algorithm

The following training functions are being used respectively for each algorithm above:

- ◆ tan-sigmoid transfer function (tansig) in hidden layer as Sigmoid output neurons are often used for pattern recognition problems.
- ◆ Linear transfer function (purelin) in output layer as linear output neurons are used for function fitting problems

The different Neural Network approaches are used for the modelling of the reusability data as generated from the previous step. For each approach following steps are used.

The following are the steps for each Neural Network based system:

1) Phase I

The following steps will be followed to train a Neural Network:

•Load the data
•Divide data into Training, Validation and Test data
•Set number of hidden neurons
•Training is accomplished by sending a given set of inputs through the network and comparing the results with a set of target outputs.
•If there is a difference between the actual and target outputs, the weights are adjusted to produce a set of outputs closer to the target values.
•Network weights are determined by adding an error correction value to the old weight.
•The amount of correction is determined
•This Training procedure is repeated until the network performance no longer improves.

2) Phase II

After training test the Neural Networks on the basis of Accuracy, MAE and RMSE. The details of the MAE and RMSE are given below:

- **Mean absolute error (MAE)**

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 [13]. The formula for calculating MAE is given in equation shown below:

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

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

- **Root mean-squared error (RMSE)**

RMSE is 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 [14]. It is just the square root of the mean square error as shown in equation given below:

$$RMSE = \frac{|a_1 - c_1| + |a_2 - c_2| + \cdots + |a_n - c_n|}{n} \quad (2)$$

C. *Conclusions drawn*

The conclusions are made on the basis of the results calculated in the previous section.

III. Implementation and Results

In this paper, the implementation of the algorithm is done in Matlab 7.4 environment and Neural Network toolbox for Matlab is used. The dataset is collected and Fletcher –Reeves Update Conjugate Gradient (FRUCG) algorithm, Polak–Ribiere Update Conjugate Gradient (PRUCG) algorithm, Powell-Beale Restarts Conjugate Gradient (PBRCG) algorithm, Scaled Conjugate Gradient (SCG) algorithm, based Neural Networks are experimented with training functions tansig and purelin to obtain the results in terms of *Accuracy*, *MAE* and *RMSE* values. The same neural network is run for five times and the following tables 1-4 are showing the Results of five different iterations of four different Neural Network Based algorithms for Identification of Reusable_Modules in the function based software systems.

The table 5 shows the Mean Values of the Results of tables means mean value of the results of five iterations. Fig 1-4 shows the training performance of the four different Algorithms for Reusability dataset. The mean or average result values of four algorithms under study as shown in table 5 depict that the *Accuracy*, *MAE* and *RMSE* values of the Fletcher–Reeves Update Conjugate Gradient (FRUCG) algorithm is the best among four neural network based algorithms experimented in the study with 100%, 0.013 and 0.0197 as *Accuracy*, *MAE* and *RMSE* values respectively.

TABLE I
RESULTS OF FLETCHER–REEVES UPDATE CONJUGATE GRADIENT ALGORITHM

Algorithm	Iterations	Accuracy	MAE	RMSE
FRUCG	1	100	0.0076	0.0107
	2	100	0.0366	0.0562
	3	100	0.0080	0.0115
	4	100	0.0065	0.0100
	5	100	0.0063	0.0101

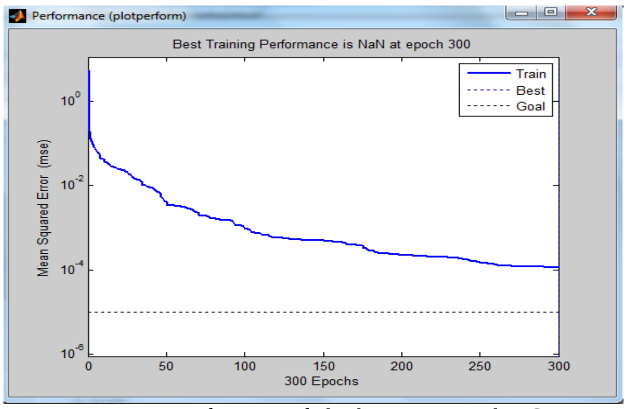

Fig 1.1 Training Performance of Fletcher–Reeves Update Conjugate Gradient Algorithm

TABLE II
RESULTS OF POLAK–RIBIERE UPDATE CONJUGATE GRADIENT ALGORITHM

Algorithm	Iterations	Accuracy	MAE	RMSE
PRUCG	1	100	0.0081	0.0124
	2	100	0.0120	0.0192
	3	100	0.0287	0.0444
	4	100	0.0109	0.0150
	5	100	0.0108	0.0167

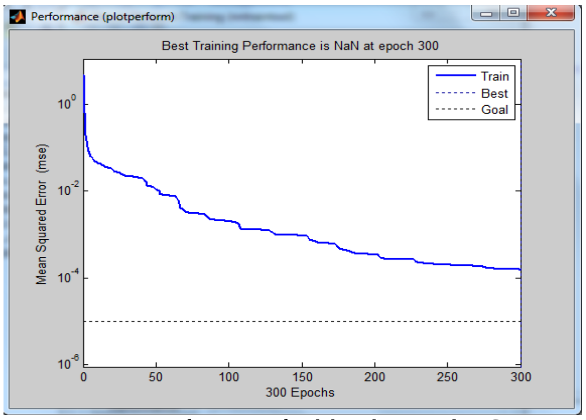

Fig 1.2. Training Performance of Polak–Ribiere Update Conjugate Gradient (PRUCG) algorithm

TABLE III
RESULTS OF POWELL-BEALE RESTARTS CONJUGATE GRADIENT ALGORITHM

Algorithm	Iterations	Accuracy	MAE	RMSE
PBRCG	1	100	0.0532	0.0791
	2	99.1	0.0533	0.0882
	3	100	0.0284	0.0428
	4	100	0.0162	0.0322
	5	94.5	0.1435	0.1982

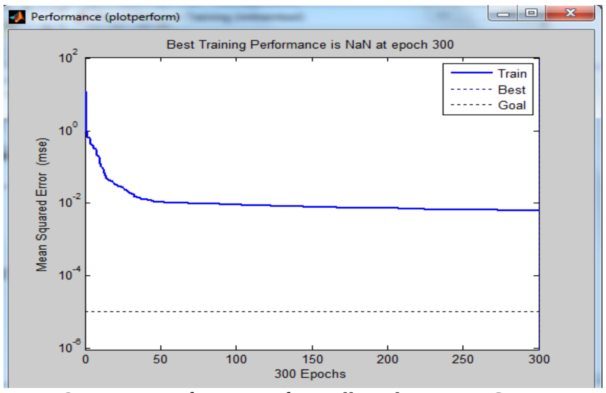

Fig 1.3 Training Performance of Powell-Beale Restarts Conjugate Gradient (PBRCG) algorithm

TABLE IV
RESULTS OF SCALED CONJUGATE ALGORITHM

Algorithm	Iterations	Accuracy	MAE	RMSE
SCG	1	99.1	0.0567	0.0949
	2	100	0.0566	0.0082
	3	100	0.0072	0.0113
	4	100	0.0034	0.0059
	5	100	0.0044	0.0074

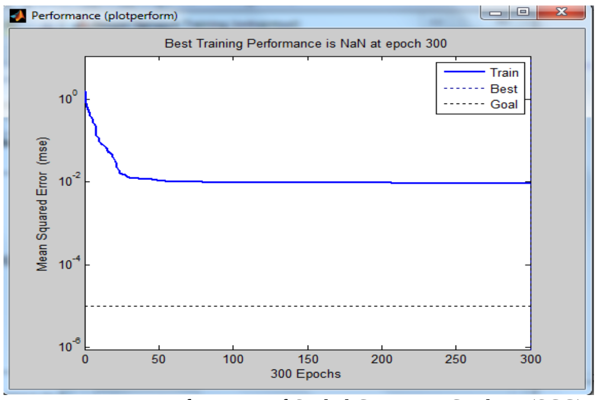

Fig 1.4 Training Performance of Scaled Conjugate Gradient (SCG) algorithm

TABLE V
MEAN VALUES OF THE RESULTS OF TABLE I TO TABLE IV

Algorithm	Mean Accuracy	Mean MAE	Mean RMSE
FRUCG	100	0.013	0.0197
PRUCG	100	0.0141	0.02154
PBRCG	98.72	0.05892	0.0865
SCG	99.82	0.02566	0.02554

IV. Conclusions

In this research paper, five Neural Network based approaches are experimented to develop the reusability evaluation model for function oriented software systems. Metric based approach is used for structural analysis of a software module. McCabe's Cyclometric Complexity Measure for Complexity measurement, Regularity Metric, Halstead Software Science Indicator for Volume indication, Reuse Frequency metric and Coupling Metric are used. Fletcher–Reeves Update Conjugate Gradient (FRUCG) algorithm, Polak–Ribiere Update Conjugate Gradient (PRUCG) algorithm, Powell-Beale Restarts Conjugate Gradient (PBRCG) algorithm, Scaled Conjugate Gradient (SCG) algorithm are experimented with the training functions tansig and purelin . The Fletcher–Reeves Update Conjugate Gradient (FRUCG) algorithm is the best among five neural network based algorithms experimented in the study with 100%, 0.013 and 0.0197 as *Accuracy*, *MAE* and *RMSE* values respectively. The performance of the Fletcher–Reeves Update Conjugate Gradient (FRUCG) algorithm is found to be best as compare to other algorithms that are recorded to calculate the mean result values. So, algorithm with these training functions based approach can be used for the Modeling of the reusable component based on metrics discussed in this paper.

It can be further to use other programming languages and also more algorithms with other training functions such as logsig, etc can be experimented to find the best algorithm.

References

[1] Parvinder S. Sandhu, Harpreet Kaur, and Amanpreet Singh (2009)"Modeling of Reusability of Object Oriented Software System" World Academy of Science, Engineering and Technology 56 2009.

[2] W. Tracz, A Conceptual Model for Mega programming, SIGSOFT Software Engineering Notes, 16(3, July 1991) 36-45.

[3] Stephen R. Schach and X. Yang, Metrics for targeting candidates for reuse: an experimental approach, ACM, (SAC 1995) 379-38.

[4] J. S. Poulin, Measuring Software Reuse–Principles, Practices and Economic Models (Addison-Wesley, 1997).

[5] Ajay Kumar (2012) "measuring software reusability using svm based classifier approach", International Journal of Information Technology and Knowledge Management January-June 2012, Volume 5, No. 1, pp. 205-209.

[6] W. Lim, Effects of Reuse on Quality, Productivity, and Economics, IEEE Software, 11(5, Oct. 1994), 23-30.

[7] G. Caldiera and V. R. Basili, Identifying and Qualifying Reusable Software Components, IEEE Computer, (1991) 61-70.

[8] Sonia Manhas, Rajeev Vashisht, and Reeta Bhardwaj (2010) "Framework for Evaluating Reusability of Procedure Oriented System using Metrics based Approach", International Journal of Computer Applications (0975 – 8887), Volume 9– No.10, November 2010.

[9] Sonia Manhas, Rajeev Vashisht, Parvinder S. Sandhu and Nirvair Neeru (2010) "Reusability Evaluation Model for Procedure Based Software Systems", International Journal of Computer and Electrical Engineering, Vol.2, Dec, 2010.

[10] Parvinder Singh Sandhu and Hardeep Singh, "Automatic Reusability Appraisal of Software Components using Neuro-Fuzzy Approach", International Journal Of Information Technology, vol. 3, no. 3, 2006, pp. 209-214.

[11] T. MaCabe, "A Software Complexity measure", IEEE Trans. Software Eng., vol. SE-2 (December 1976), pp. 308-320.

[12] G. Caldiera and V. R. Basili, Identifying and Qualifying Reusable Software Components, IEEE Computer, (1991), pp. 61-70.

[13] Herenji, H. R. and Khedkar, P (1992), "Learning and Tuning Fuzzy Logic Controllers through Reinforcements", IEEE Transactions on Neural Networks, vol. 3, 1992, pp. 724-740.

[14] Challagulla, V.U.B., Bastani, F.B., I-Ling Yen, Paul, (2005), "Empirical assessment of machine learning based software defect prediction techniques", 10th IEEE International Workshop, Object-Oriented Real-Time Dependable Systems, WORDS 2005, 2-4 Feb 2005, pp. 263-270.