American-Eurasian Journal of Scientific Research 11 (4): 240-249, 2016
ISSN 1818-6785
© IDOSI Publications, 2016
DOI: 10.5829/idosi.aejsr.2016.11.4.22726

Application of Backpropagation Algorithms in Predicting the Quality of Component Based Software Systems

Mwalili Tobias, Waweru Mwangi and Kimwele Michael

School of Computing and Information Technology,
Jomo Kenyatta University of Agriculture and Technology, Kenya

Abstract: Component-based software engineering is a development approach that uses existing software components as a means of accelerating delivery of quality software systems, within time and budget constraints. Metrics for evaluating the quality of component-based systems have been developed and validated. However, most of the existing metrics are based on aggregation models, linear regression and multivariate modeling techniques. Artificial Neural Network techniques have been found powerful in modeling Software quality metrics compared to traditional statistical techniques. This study aims at researching on the applicability of neural network ensembles for analyzing and validating the interface complexity metrics for JavaBeans components. This study utilizes the Backpropagation training algorithms to predict quality attributes for component-based systems. Specifically the study the performs an empirical analysis of the performance of the Backpropagation training methods based on algorithm's rate of convergence and the network's root means square error.

Key words: CBSE · Complexity metrics · Quality metrics · Backpropagation · Training methods

INTRODUCTION

Predicting the quality of Component-Based Systems (CBS) has generated a lot of research interest in the recent past. This is because of the risks associated the process of component selection and integration of components. These risk factors are numerous because the components are delivered and handled as black boxes. Quality metrics for CBS have been proposed and developed. However, most of these metrics are based on aggregation models, linear regression and multivariate modeling techniques. Artificial Neural Network (ANN) techniques have been found more adaptive in modeling Software quality metrics compared to traditional statistical techniques [1, 2]. Theoretically ANNs have the capability to approximate an arbitrary continuous function to a specified accuracy on any compact set [3].

Chauvin [4] introduced the Backpropagation (BP) rule, which is widely used in training a multilayer feed-forward ANNs. The gradient descent algorithm is the most basic training rule for the Backpropagation neural network (BPNN), where the weights are modified in the steepest descent direction (i.e., negative of the slope). The BPNN has been shown to be slow in learning convergence and may be trapped in local minima. The performance of the network is dependent on other factors such as the selected learning rate parameter the complexity of the classification problem. For faster convergence and improved performance, other training rules have been proposed and implemented. The improved algorithms fall into two broad categories. The rules in the first category use a heuristic technique based on the performance of the standard steepest descent algorithm. Under this category, we have gradient descent with momentum, dynamic propagation that has a gradient descent with a momentum and an adaptive learning rate and the resilient algorithm. The second category includes conjugate gradient, quasi-Newton and Levenberg-Marquardt (LM) algorithm. In this work we shall focus on the the performance of the first category of training rules in predicting the quality characteristics of software components.

Corresponding Author: Mwalili Tobias, School of Computing and Information Technology, Jomo Kenyatta University of Agriculture and Technology, Kenya.

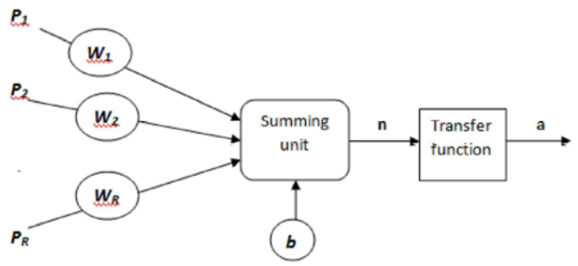


Fig. 1: The architecture of a single generic neuron

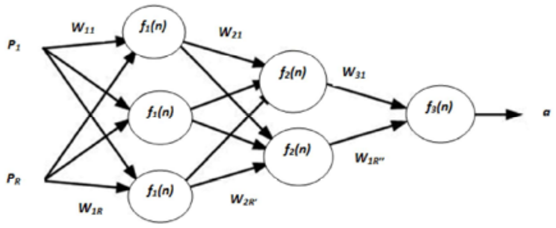


Fig. 2: The architecture of a MLP neural network

The rest of the paper is organized as follows. Section 2 and 3 gives an overview of CBSE and Backpropagation training rules. Section 4 outlines related research works, section 5 deals with research experiment design and finally section 6 summarizes results, discussions and conclusion.

Bpnn and Training Algorithms: A neural network is essentially an assembly of basic computing elements referred to as neurons. Each neuron is composed of two units, a summing unit and a transfer function unit. R inputs, each with a weight (**w**$_i$) plus a bias component (**b**) are summed up in the summing unit. The output (**n**) of the summing unit is then fed into the transfer function (**f**) to give the final output (**a**) of the neuron. Figure 1 below shows the architecture of a single generic neuron.

The output of the neuron can therefore be expressed as in matrix form as equation 1 below

$$a = f(n) = f(wp + b) \tag{1}$$

The transfer (activation) function (f) is usually a nonlinear differentiable function function. The commonly used activation functions include are Log-Sigmoid and Hyperbolic Tangent Sigmoid, which are given by the Equations (2) and (3) respectively.

$$a = \frac{1}{1 + e^{-1}} \tag{2}$$

$$a = \frac{e^{n} - e^{-n}}{e^{n} - e^{-n}} \tag{3}$$

The Multi-layer Perceptron (MLP) is the most popular neural network architecture in prediction applications [5]. The MLP architecture has an input layer, one or more hidden layers and an output layer. The input layer captures input variables, he hidden layers model a non-linear relationship between the variables and the output layer gives the predicted values. Figure 2 below shows the architecture of a MLP with three hidden layers.

The process of training the MLP, involves presenting the network with two vectors, the input vector **(p)** and the desirable target output vector **(t)**. During an iteration the output of the network **(a$_i$)** is compared with target output **(t$_i$)**. The weights of the network are then adjusted according to a Backpropagation algorithm so as to minimize the network error given by

$$E(e^2) = E(t - a)^2 \tag{4}$$

There are various algorithms applicable for adjusting the weights of the MLP network. Which include;

The standard Backpropagation: This method uses the Gradient descent algorithm where the weights are adjusted based on the equation

$$\Delta w_k = -a_k . g_k \tag{5}$$

where, Δw_k represents the the change in weights, g_k is the gradient at any given iteration and g_k is an adjustable the learning rate. This relation implies that the network weights and biases are adjusted in the direction of the negative gradient of the performance function.

Backpropagation with Momentum: This method is an improvement of the standard Backpropagation by adding a momentum term such that the weights are now adjusted according to the equation.

$$\Delta w_k = -a_k . g_k + p\Delta w_{k-1} \tag{6}$$

This modification avoids oscillations and provides a mechanism for escaping from small local minima on the error surface. Furthermore the introduction of the momentum term reduces the sensitivity of the network to fast changes of the error surface [6].

Dynamic Propagation: This method uses the Gradient descent algorithm with momentum with a combination of an adaptive (dynamic) learning rate. The initial network output and error first calculated and then using the current learning rate, new weights and biases are then recalculated at each epoch as given by:

$$\Delta w_k = p\Delta w_{k-1} + \alpha.p.\frac{\Delta E_k}{\Delta w_k} \tag{7}$$

Resilient Propagation: This method modifies the network weight by considering the sign of the partial derivative to determine the direction of the weight update then multiply it by the step size as given by:

$$\Delta w_k = -sign\left(\frac{\Delta E_k}{\Delta w_k}\Delta_k\right) \tag{8}$$

Other MLP training methods include include, quasi-Newton, conjugate gradient, Polak-Ribiére update and Levenberg-Marquardt algorithm. This work presents a comparative the performance of the standard backpropagation, backpropagation with momentum, dynamic propagation and resilient and propagation in predicting the quality the quality of component based systems. The performance parameters that will be used are, rate of convergence (iterations before convergence) and the network Root Mean Square Error (RMSE) given by:

$$RMSE = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(y(k) - \tilde{y}(k))^2} \tag{9}$$

where $y(k)$ and $\tilde{y}(k)$ are respectively the expected and predicted value for a software component quality characteristic.

Related Work: Boetticher *et al* [1] introduced the concept of neural networks approach software metrics by modeling the McCabe and Halstead metrics. The study focused on the performance Backpropagation neural network with the quick-drop algorithm in modeling the McCabe and Halstead metric. Training data sets were derived from reuse based Ada programs repositories, Conn [2]. The performance of the algorithm was analyzed for varying network architectures on fixed size training sets. Results from this study indicated that neural network approach was feasible in modeling software metrics.

Lanubile *et al*., [6] applied neural network applied pattern recognition techniques classify modules as low or high risk. The outcome of this study showed that neural networks produced better results than statistical techniques such as regression modeling.

Kurfess *et al*., [7] performed neural network experiments on categorization and assessment of the quality of legacy systems subprograms. The study aimed at establishing whether a neural network can be used to categorize the quality of a legacy sub-programs in terms of its object-orientedness. Results from this study demonstrated the viability of neural network techniques in validating metrics for object oriented software systems.

Thwin *et al*., [8] applied the The Ward neural network and General Regression Neural Network (GRNN) predict the quality of communications software modules, using various object-oriented metrics, which included, Depth of Inheritance Tree (DIT), Number of Children (NOC), Coupling Between Objects (CBO), Response For a Class (RFC) and Inheritance Coupling (IC).

Furthermore Aljahdali *et al*. [9] demonstrated the viability of back-propagation and other training techniques in predicting reliability of legacy software modules, based on experimental data.

Washizaki's [10] proposed the several metrics for measuring reusability of software components, which include:-

- Rate of Component Observability (RCO) given by

$$RCO(c) = \frac{\text{No of readable propoerties in class C}}{\text{No of fields in C's facade class}} \tag{10}$$

- Rate of component customizability (RCC)

$$RCC(c) = \frac{\text{No.of writable propoerties in class C}}{\text{No.of fields in C's facade class}} \tag{11}$$

- Self-Completeness of Component's Return Value (SCCr)

$$SCCr(c) = \frac{\text{No.of void methods in class C}}{\text{No.of Methods in class C}} \tag{12}$$

Self-Completeness of Component's Parameter (SCCp)

$$SCCp(c) = \frac{\text{No of methods with parameters in class C}}{\text{No. of Methods in class C}} \tag{13}$$

Sharma [11] proposed the Interface Complexity Metric (ICM). The ICM models the external behavior of the component as aggregation components methods and properties complexity factors given by Equation (5)

$$ICM(C) = A\sum_{i=1}^{m} CIM_i + B\sum_{j=1}^{n} CP_j \tag{14}$$

where, CIM_i is the complexity of the ith interface method and CP_j is the complexity of the jth property. A and B are the weight values for methods and properties respectively.

The complexity of an interface method is computed based weighed values that are assigned to each return values or argument according to its data type.

Mwalili *et al*. [12] proposed the Bounded Interface Complexity Metric (BICM), for evaluating the complexity of a software component, given by the equation

$$BICM(C) = A\frac{\sum_{i=1}^{m} CM_i}{M} + B\frac{\sum_{j=1}^{n} CP_j}{N} \tag{15}$$

where, CM_i is the complexity of the ith interface method and CP_j is the complexity of the jth property. M and N represents the count of component methods and properties respectively while A and B are the weight values.

To analyze the consistency the components interface parameter types, we propose a new metric called interface surface consistency (ISC) that is based on the variability of the components methods and property complexity and given as

$$ISC = \frac{1}{m}\sum_{i=1}^{i=m} SDM(i) + SDP \tag{16}$$

where $SDM_{(i)}$ is the standard deviation for complexity of the ith method and SDP is the standard deviation for the property complexity. This metric carries overall information about the consistency of a components interface.

A high value indicates a rough interface surface and such a component would be difficult to integrate and maintain.

The next section describes the research experiment for carried out to establish performance Backpropagation training algorithms in predicting quality of software components based on the proposed interface metrics **BICM** and **ISC**.

Table 1: Network input data set

SNO	COMPONENT	ISC	BICM	SCCR	SCCP
1	Gmail	0.58769	0.58	0.5091	0.5273
2	Gcalender	0.5788	0.47	0.5167	0.4500
3	Gdata	0.68442	0.57	0.5287	0.4828
4	Gdocuments	0.62953	0.54	0.5490	0.5098
5	Gcontacts	0.39845	0.47	0.0631	0.4775
6	Gspreadsheets	0.77473	0.59	0.5217	0.4435
7	Gstorage	0.50522	0.46	0.6176	0.5686
8	Paapi	0.27386	0.39	0.4722	0.5000
9	Amazonrequest	0.4427	0.49	0.5600	0.5500
10	S3	0.5004	0.48	0.6154	0.5769
11	Sqs	0.36929	0.56	0.5522	0.6269
12	Ec2	1.55238	0.76	0.6508	0.6349
13	Simpledb	0.29006	0.45	0.5965	0.5439
14	Blob	0.73334	0.51	0.6636	0.5727
15	Table	0.30318	0.45	0.5854	0.4878
16	Queue	0.53955	0.48	0.6316	0.5000
17	Azurerequest	0.27386	0.47	0.5567	0.5567
18	Sapclient	0.14618	0.39	0.5849	0.4528
19	Ftp	0.23112	0.40	0.6429	0.5143
20	Rss	0.79353	0.52	0.6125	0.6125
21	Smtp	0.15233	0.35	0.6000	0.5333
22	Soap	0.38082	0.39	0.4904	0.4712
23	Mime	0.22685	0.53	0.6452	0.5484
24	Pop	0.20733	0.47	0.5211	0.4225
25	Syslog	1.35463	0.73	0.6190	0.6667
26	Telnet	0.21275	0.46	0.7500	0.7333
27	Whois	0.17239	0.44	0.6071	0.6071
28	Webupload	0.46281	0.44	0.5915	0.5634
29	Webdav	0.55864	0.47	0.6344	0.6237
30	Xmpp	0.98112	0.61	0.6739	0.6304
31	atePicker	0.16701	0.49	0.5419	0.5484
32	PVTree	0.3558	0.51	0.5313	0.6354
33	PVTable	0.51112	0.48	0.5182	0.5992
34	PVCalculator	0.17544	0.38	0.6364	0.5844
35	PVCalendar	0.98199	0.62	0.5598	0.5837
36	PVChoice	0.29792	0.40	0.5484	0.5806

Research Experiment: The research experiment involved training and testing of the MLP. The objective of the experiment is establishing the performance of the various Backpropagation training algorithms vs. the architecture of the MLP. The performance of the network will be evaluated in terms of rate of convergence (iterations before convergence) and network Root Mean Square Error (RMSE). The two network performance parameters will be analyzed against varying architecture of the MLP. The variations of the MLP architecture are in terms of the number of hidden layers in the network and the number of neurons per hidden layer. The specific research questions that we seek to answer are:

- What is the relationship between variation of the MLP network and the performance of the individual Backpropagation training algorithms?
- Is there a significant difference between the performances of various Backpropagation training algorithms as the architecture of the MLP network varies?
- Which Backpropagation training algorithm is most suitable for using interface complexity metrics to predict quality characteristics component based systems?

Acquisition of Training Data Set: The primary data for input into the network is obtained from Mwalili [10], which is a set of computed interface complexity metrics (ISC, BICM, SCCR and SCCP) for sample JavaBeans

Table 2: Components quality characteristics evaluation criteria

Metric	Evaluation Criteria
ISC	This metric indicates the consistency of the components interface parameter types. A high value could imply a comportment that is difficult to integrate and maintain
BICM	This metric indicates the complexity of the components interface. A high value could imply a component that is difficult to maintain
SCCR	This metric indicates the self-completeness of the information dealt by the component. A low value could imply a high degree of component independence from the exterior.
SCCP	This metric shows the degree of component's self-completeness and independence. The higher the value is, the higher the component portability.

Table 3: Expected network output data set.

SNO	COMPONENT	Maintainability	Indepedence	Portability
1	Gmail	0.75	0.25	0.5
2	Gcalender	0.75	0.75	0.5
3	Gdata	0.5	0.25	0.75
4	Gdocuments	0.5	0.5	0.75
5	Gcontacts	0.75	0.75	0
6	Gspreadsheets	0.5	0.25	0.5
7	Gstorage	0.75	0.75	1
8	Paapi	1	1	0.5
9	Amazonrequest	0.75	0.75	0.75
10	S3	0.75	0.75	1
11	Sqs	0.75	0.5	0.75
12	Ec2	0	0	1
13	Simpledb	1	0.75	0.75
14	Blob	0.5	0.5	1
15	Table	1	0.75	0.75
16	Queue	0.75	0.75	1
17	Azurerequest	1	0.75	0.75
18	Sapclient	1	1	0.75
19	Ftp	1	1	1
20	Rss	0.5	0.5	1
21	Smtp	1	1	0.75
22	Soap	0.75	1	0.5
23	Mime	1	0.5	1
24	Pop	1	0.75	0.5
25	Syslog	0	0	1
26	Telnet	1	0.75	1
27	Whois	1	0.75	1
28	Webupload	0.75	0.75	0.75
29	Webdav	0.75	0.75	1
30	Xmpp	0.25	0.25	1
31	atePicker	1	0.75	0.75
32	PVTree	0.75	0.5	0.75
33	PVTable	0.75	0.75	0.5
34	PVCalculator	1	1	1
35	PVCalendar	0.25	0.25	0.75
36	PVChoice	1	1	0.75

components downloaded from the components super store, ComponentSource.com. The values for these metrics are presented in Table 1 [13].

The expected outputs from the network are quantities for three software component quality characteristics of maintainability, independence and portability.

To acquire data for the expected output, software developers with experience in software component technology where randomly selected and asked to evaluate a set of software components based on provided fuzzy guidelines (Table 2) and their experience. The evaluators where required to give a score of 1 to 5 for each of the three quality attributes for all components in the provided set. The data collected data was then aggregated and is presented in Table 3

Data Normalization and Network Training: The input and output datasets where normalized using the normalization Equation (X) and separated into training and testing sets.

$$X_n = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (X)$$

The normalized dataset for training the network will therefore have seven fields derived from four input variables (**ISC, BICM, SCCR and SCCP**) and and three expected output variables (*Maintainability, Independence and Portability*).

The objective therefore is to do a comparative analysis of the performance of the standard backpropagation (**SBP**), backpropagation with momentum (**BPM**), dynamic propagation (**DYP**) and resilient propagation (**RSP**) in predicting the components quality characteristcs Maintainability, Independence and Portability.

The performance will be measured interms of rate of convergence (iterations before convergence) and the network Root Mean Square Error (RMSE). Fig 3 shows the conceptual framework for setting up, training and testing the MLP neural network.

Based on the above conceptual framework, we constructed two sets of physical MLP networks. In the first Set, all networks have one hidden layer each, while individual networks have 1, 2, 3 … 36 neurons in the hidden layer, giving a total of 36 physical networks in the set.. In the second Set, all networks have two hidden layers each, while individual networks have 1, 2, 3 … 36 neurons in the each hidden layer, giving a total of 36 physical networks in the set. All networks have 4 neurons in the input layer which corresponds to the 4 complexity metric values and 3 neurons in the output layer which

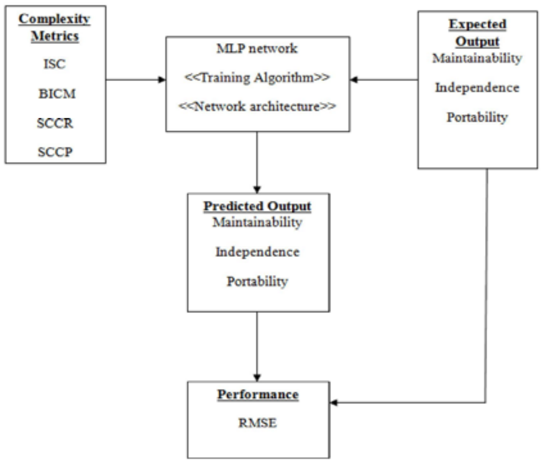


Fig. 3: Conceptual framework for MLP network training

Table 4: MLP Network Architecture configuration parameters

Method	SBP	BPM	DYP	RSP
Training function	traingd	traingdm	traingdx	trainrp
Activation function	Log-Sigmoid	Log-Sigmoid	Log-Sigmoid	Log-Sigmoid
Convergence goal	1.00E-03	1.00E-03	1.00E-03	1.00E-03
Hidden Layers	1 , 2	1, 2	1, 2	1, 2
Neurons per layer	1-36	1-36	1-36	1-36
Learnning Rate	0.2	0.2	0.05-0.2	-
Momentum	-	0.2	0.05-0.2	-
Max iterations	1000	1000	1000	1000

correspond to 3 components quality attributes values. Using neural network notation the architecture for networks can be summarized as

4-> [1-36] ->3 (for Set 1) and
4-> [1-36] -> [1-36]->3 (for Set 2)

The aim is to establish how the various BP training methods will behave as the network architecture varies in terms of hidden layers and neurons per hidden layer. Network training and testing was performed for the various combinations of training methods, network architecture and network training parameters such as learning rate and momentum as per Table 4.

During the training and testing step, output data for rate of convergence and RMSE was captured and and tabulated in Tables A1-A4 in the appendix section. In the next section we now give the experiment results and discussion.

RESULT AND DISCUSSION

This contains the results and discussion of the study.

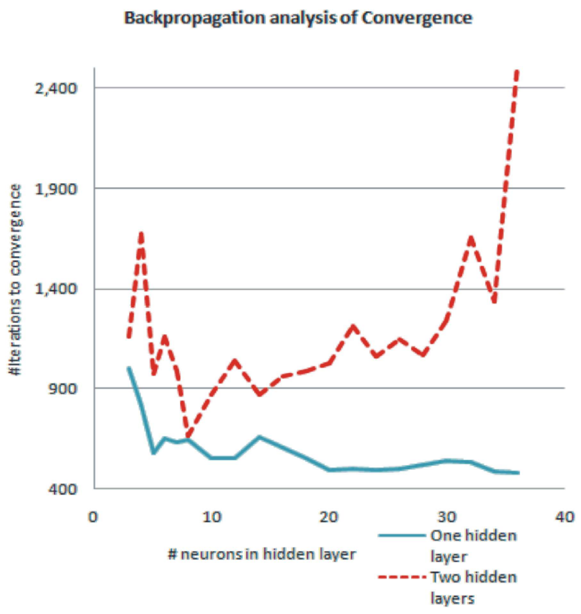


Fig. 4: Standard Backpropagation rate of convergence for MLP with 1 and 2 hidden layers.

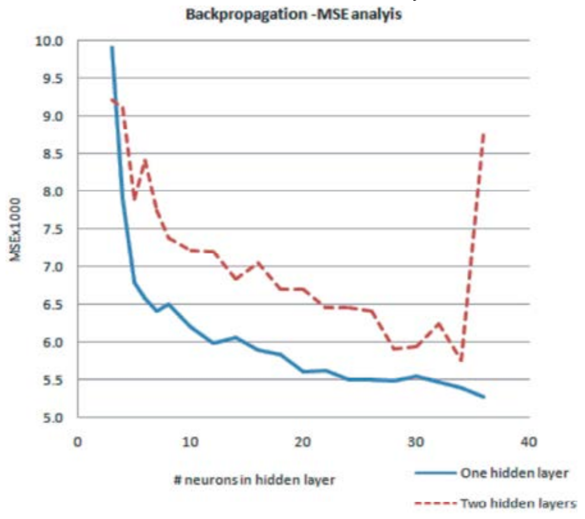


Fig. 5: Standard Backpropagation RMSE for MLP with 1 and 2 hidden layers

Analysis of 1 and 2 Hidden Layers Architecture: From the graph of Backpropagation-analysis of convergence (Figure 4), it is clear that the MLP with 1 hidden layer is converging faster than the one with 2 hidden layers. Also, as the number of neuron per hidden layer increases, the rate of convergence for network with 2 hidden layers deteriorates, while it improves for the network with 1 hidden layer.

When we consider (Figure 5) the graph of Backpropagation RMSE analysis it can also be noted that the network with 1 hidden layer yields a RMSE that is consistently lower that with 2 hidden layers. The two outcomes imply that a network with one hidden layer is more efficient and suitable for performing prediction in the problem at hand

Naturally one would have expected the network with 2 hidden layers to be more efficient and robust in prediction. However this is not the case. The poor performance of the MLP with 2 hidden could be attributed to problem data over-fitting which occurs when a network that has already memorized training examples is presented with new data not able to generalize to new situations leading to larger error values. Methods for preventing over-fitting are discussed in [13, 14]. One approach for avoiding this problem is to use a smaller network that does not have power enough to over-fit the problem. For the problem at hand, therefore conclude that a network with 1 hidden layer is more suitable that with 2 hidden layers. In subsequent analysis we will therefore focus on analyzing performance of other training methods based for network with 1(One) hidden layer.

Analysis for the Backpropagation Algorithms for 2 Hidden Layer Architecture: Having isolated the the MLP architecture with two hidden layers, we now turn our attention to the MLP with one hidden layer. The objective now is to establish how the various BP algorithms are going to perform in terms rate of convergence and RMSE. The performance of the various Backpropagation training methods is presented of Figures 6 and 7 below from the graph of convergence analysis, it can be that:-

- Resilient propagation converges faster that other methods but it’s very unpredictable and inconsistent method.
- The other three methods give a predictable trend.
- Dynamic propagation gives a convergence rate that is on average consistently lower compared to the other methods.

From the graph RMSE analysis, It can be noted that:

- Resilient propagation yields a very high RMSE and is very unpredictable. Its therefore out rightly a very poor method of predicting quality characteristics for CBS
- The other three methods give a very predictable downward trend for RMSE with Dynamic propagation being marginally lower compared to the other two

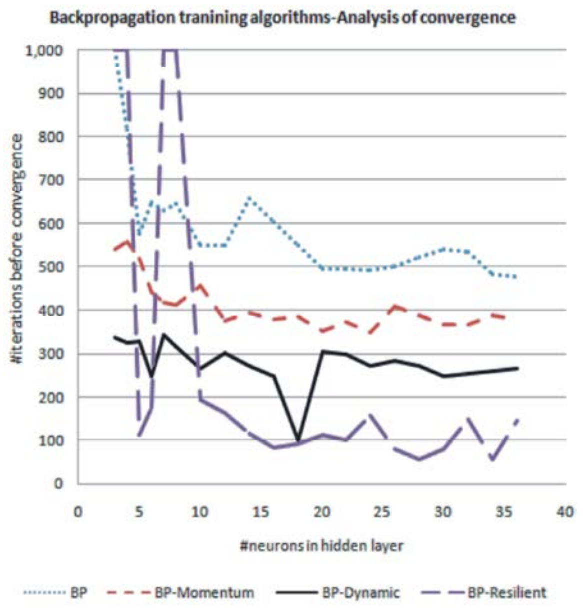


Fig. 6: All BP training methods rate of convergence for MLP with 1 hidden layer

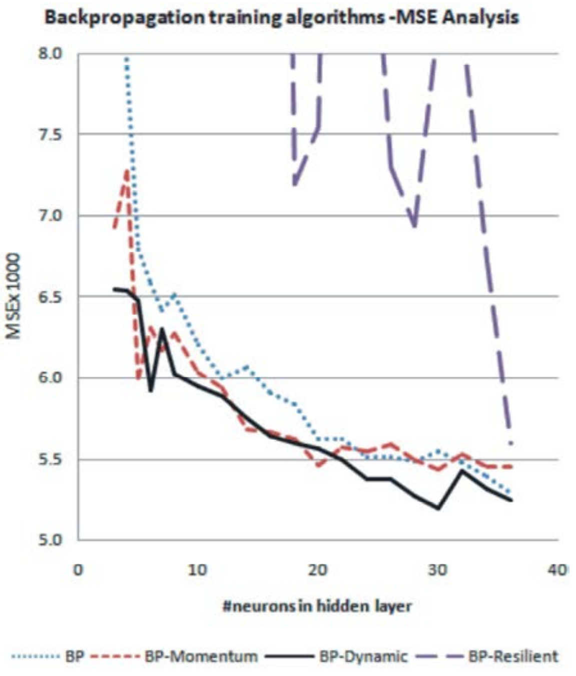


Fig. 7: All BP training methods RMSE for MLP with 1 hidden layer

- Even thought dynamic propagation appears to lower than the other two methods; we note that that visual inspection of the plot may not give a statistically sound conclusion as to which method is more suitable for predicting the problem at hand.

We now must do statistical analysis to so as to discriminate between the various methods of training. We will use Analysis of Variance (ANOVA) and regression analysis.

Analysis of Variance (ANOVA) for Backpropagation Algorithms Performance: Analysis of Variance (ANOVA) provides methods for testing the equality of three or more population means, that is, the means of a single variable from several populations. One-way analysis of variance is a procedure used to test if the means of several (say k) independent random samples are equal or not all equal to each other. In our case, we will be testing if the means of the rate of convergence and RMSE derived from the various Backpropagation training methods are equal to each other.

To test for equality of means for the rate of convergence, we let μR_{***} represent the sample mean for rate of convergence and setup the hypothesis,

$$H_0: \mu C_{SBP} = \mu C_{BPM} = \mu C_{DYP} = \mu C_{RSP}$$

$$H_1: Atleast\ one\ of\ the\ means\ for\ convergence\ is$$

$$different\ from\ the\ others$$

From Table 5 at 95% confidence interval we note that F_Test> F_Critical so we reject the null hypothesis and conclude that at least one of the means Convergence is different from the others.

To test for equality of means for the RMSE, we let μC_{***} represent the sample mean for RMSE and setup the hypothesis,

$$H_0: \mu R_{SBP} = \mu R_{BPM} = \mu R_{DYP} = \mu R_{RSP}$$

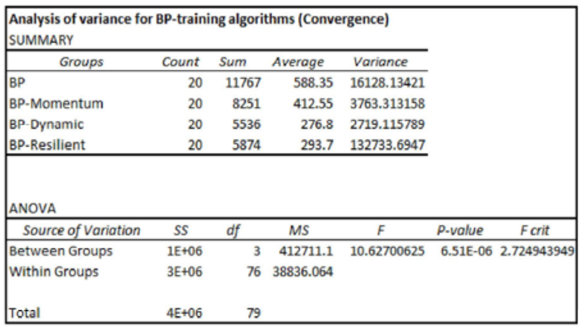
$$H_1: Atleast\ one\ of\ the\ means\ for\ RMSE\ is$$

$$different\ from\ the\ others$$

From Table 6 at 95% confidence interval we note that F_Test> F_Critical so we reject the null hypothesis and conclude that at least one of the means for RMSE is different from the others

From the above analysis, we have determined that the there is a significant difference between the four Backpropagation training methods. When we consider the rate to convergence, Table 4 reveals that Dynamic Backpropagation yields the least mean among the four methods. Similarly, when we consider the RMSE, Table 5 also reveals that Dynamic Backpropagation yields the least mean among the four methods.

Table 5: Analysis of variance for Backpropagation algorithms (rate of convergence)



Analysis of variance for BP-training algorithms (Convergence)

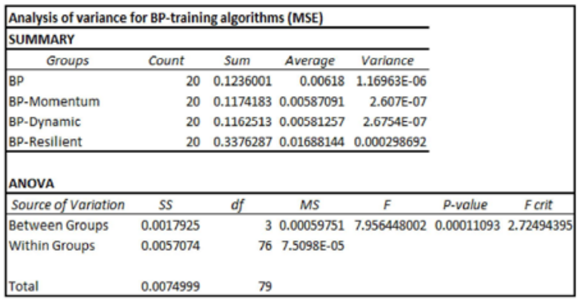
SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
BP	20	11767	588.35	16128.13421
BP-Momentum	20	8251	412.55	3763.313158
BP-Dynamic	20	5536	276.8	2719.115789
BP-Resilient	20	5874	293.7	132733.6947

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	1E+06	3	412711.1	10.62700625	6.51E-06	2.724943949
Within Groups	3E+06	76	38836.064			
Total	4E+06	79				

Table 6: Analysis of variance for Backpropagation algorithms (RMSE).



Analysis of variance for BP-training algorithms (MSE)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
BP	20	0.1236001	0.00618	1.16963E-06
BP-Momentum	20	0.1174183	0.00587091	2.607E-07
BP-Dynamic	20	0.1162513	0.00581257	2.6754E-07
BP-Resilient	20	0.3376287	0.01688144	0.000298692

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.0017925	3	0.00059751	7.956448002	0.00011093	2.72494395
Within Groups	0.0057074	76	7.5098E-05			
Total	0.0074999	79				

Regression Analysis for Backpropagation Algorithms Performance: Regression is a technique that calculates the best line that predicts a dependent variable from an independent variable. The objective is to make an inference about the relationship between the pair of variables based on sample data. In linear regression, the relationship between x (the independent or predictor variable) and y (the dependent or response variable) is expressed using the regression equation

$$y = b_0 + b_1 x \tag{17}$$

In our case, we want to make inference about the relationship between x (No. of hidden neurons per layer) and y (the RMSE) for the various Backpropagation Algorithms.

In comparing the regression lines for the various algorithms, we will be able to establish the trend of RMSE as the number of neurons in the hidden layer varies.

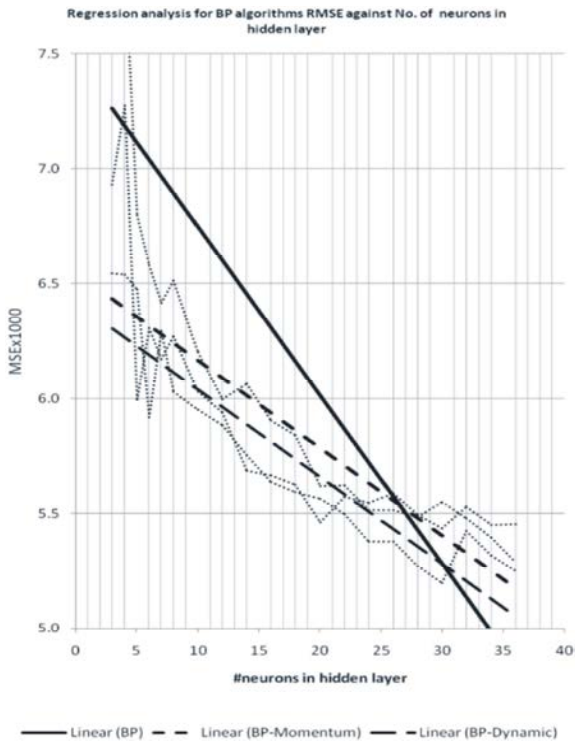


Fig. 8: Regression analysis for RMSE against No. of Hidden neurons in the hidden layer

Table 7: Regression Equations for RMSE against No of neurons in the hidden layer.

Method	Regression equation	R Squared
SBP	$y = -0.073x + 7.484$	$R^2 = 0.538$
BPM	$y = -0.038x + 6.547$	$R^2 = 0.650$
DYP	$y = -0.037x + 6.417$	$R^2 = 0.852$

Via linear regression analysis, we will also be able to calculate the R-square, which gives the degree of fit, or rather by how much does RMSE depend on the number f neurons in the hidden layer.

Figure 8 and Table 6 present the graph of regression analysis and the regression equations respectively

From the graph of regression analysis, it is evident that Dynamic Backpropagation yields a trend that is consistently lower compared to the other training methods.

From Table 7 we note that the R-squared value **(0.852)** for Dynamic Backpropagation is higher compared to the other methods. This implies that DYP has the highest degree of fit and therefore Dynamic Backpropagation is the most responsive method of training for the varying size of network architecture.

The next section gives a summary of work done, overall conclusion and makes recommendations for further research work.

CONCLUSION

Predicting the quality of Component-Based Systems (CBS) has generated a lot of research interest in the recent past. This is because of the risks associated the process of component selection and integration of components.

In this work have have done a review of Backpropagation algorithms and their application in predicting metrics for software quality. We have also done a review the Bounded Interface Complexity Metric (BICM) and the Interface Surface Consistency (ISC) Metric proposed by Mwalili [12].

In the research experiment, we trained a MLP to predict software components characteristics of maintainability, portability and independence based on input data for the various complexity metrics (BICM, ISC SCCR and SCCP). The objective of the experiment was to establish the following

- The relationship between variation of the MLP network and the performance of the individual Backpropagation training algorithms
- Whether there is significant difference between the performances of various Backpropagation training algorithms as the architecture of the MLP network varies.
- The Backpropagation training algorithm that is most suitable for using interface complexity metrics to predict quality characteristics component based systems.

For the various Backpropagation algorithms, we captured output data for RMSE and number of iterations before convergence. The data was then tabulated and analyzed in terms of network size (no of neurons in hidden layer). The outcome of the study revealed the following:-

- Backpropagation algorithms have the potential for predicting quality characteristics for software components and component based systems
- MLP network with one hidden layer was found to be suited to solving the prediction problem
- There is a significant difference between the four training methods
- Dynamic Backpropagation was found to be the most efficient of the four methods.

In this work we have analyzed the performance of the various Backpropagation training methods in predicting the quality characteristics, using the ANN MLP architecture. There is need to further research on performance of other ANN architectures and other training methods. The analysis was based data derived from individual components; therefore, there is need for analysis of the quality target system, once the components have been composed to a system.

REFERENCES

1. Boetticher, G., K. Srinivas and D. Eichmann, 1993. A Neural Net-Based Approach to Software Metrics, Proceedings of the Fifth International Conference on Software Engineering and Knowledge Engineering, San Francisco, CA, pp: 271-274.
2. Conn, R., 1987. The Ada Software Repository and Software Reusability, Proc. of the Fifth Annual Joint Conference on Ada Technology and Washington Ada Symposium, pp: 45-53.
3. Esposito Anna, 2000. Approximation of continuous and discontinuous mappings by a growing neural RBF-based algorithm, Neural Networks, 13(6): 651-665.
4. Chauvin Yves and David E. Rumelhart, 1995. Backpropagation: theory, architectures and applications. Psychology Press.
5. Mellit, Adel and Alessandro Massi Pavan, 2010. A 24-h forecast of solar irradiance using artificial neural network: Application for performance prediction of a grid-connected PV plant at Trieste, Italy, Solar Energy, 84(5): 807-821.
6. Lanubile Filippo, A. Lonigro and Giuseppe Vissagio, 1995. Comparing models for identifying fault-prone software components, SEKE.
7. Kurfess Franz, J. and Lonnie R. Welch, 1996. Categorization of programs using neural networks, Engineering of Computer-Based Systems, 1996. Proceedings, IEEE Symposium and Workshop on. IEEE.
8. Thwin Mie, Mie Thet and Tong-Seng Quah, 2005. Application of neural networks for software quality prediction using object-oriented metrics, Journal of Systems and Software, 76(2): 147-156.
9. Aljahdali Sultan, H. and Khalid A. Buragga, 2008. Employing four ANNs paradigms for software reliability prediction: an Analytical Study, ICGST International Journal on Artificial Intelligence and Machine Learning, 8(2): 1-8.
10. Washizaki Hironori, Hirokazu Yamamoto and Yoshiaki Fukazawa, 2003. A metrics suite for measuring reusability of software components, Software Metrics Symposium, 2003. Proceedings. Ninth International. IEEE.
11. Sharma Arun and Rajesh Kumar, 2009. Design and analysis of metrics for component-based software systems. Diss. Ph. D thesis.
12. Mwalili Tobias, Waweru Mwangi and Kimwele Michael, 2015. Empirical Evaluation of Complexity Metrics for Component Based Systems, Journal of Theoretical & Applied Information Technology, 73(2).
13. Srivastava Nitish, 2014. Dropout: A simple way to prevent neural networks from over fitting, The Journal of Machine Learning Research, 15(1): 1929-1958.