Software Reliability Prediction using Neural Networks

V.Ramakrishna
KL University
Guntur, AP
India

MR Narasinga Rao
KL University
Guntur, AP
India

TM Padmaja
KL University
Guntur, AP
India

ABSTRACT

Predicting the Software reliability is a pertinent issue and it is a major concern of software developers and engineers in changing environment considerations. Software reliability models are developed to estimate the probability of failure free operation of the software for a long time. Many Software Reliability Growth Models (SRGM) were developed to give the latent number of faults in the software product. However none of these models performing to the expectations of the developers of the software. In this paper, A research is made using artificial neural network models to monitor the performance of the software that leads to predict the software reliability. The MLP model outperforms SVR model, and based on the results, these models can be considered to be a reasonable alternative for software quality prediction.

Keywords

SoftwareQuality, Software Reliability, MLP Neural Network, Support Vector Regression, Back-propagation algorithm.

1. INTRODUCTION

Quality of software is the key concern in industry and extensively studied through software reliability [1]. Software reliability is defined as the probability of failure free operation of the functionality of the software for long time under the designed environmental conditions and it is measured in terms of failure of the software. A number of software reliability models developed for the quantification of software quality prediction. These models can be grouped into Software Reliability Growth Models (SRGM's) and Data-Driven models.

SRGMs use probability models to describe the failure process under a set of assumptions to provide mathematical ease and these assumptions limits the models [2]. Data-Driven models use time series analysis including autoregressive methods [3, 4, 5, and 6]. These models are developed from past software failure history data. These models help to identify error prone programs and make the developers to focus on maintenance [7, 8].

In this paper an attempt is made to investigate the performance of two different connectionist paradigms given below for modeling the prediction of software reliability. We try to compare the performance of the two individual neural network models, one is an Multi Layer Perceptron (MLP) Neural Network with back-propagation algorithm and the other is a variant of a support vector machine called Support Vector Regression (SVR) model. These two models were employed on a software reliability data set which is obtained from project-5 of Bell-Telephone laboratories.

2 Methodology

This section presents the detailed theoretical descriptions of the algorithms used for proposing the models for Software Reliability Prediction.

2.1 Multi Linear Perceptron (MLP) The MLP network is constructed with back propagation algorithm with multiple hidden layers. The summary of the operation of the MLP with back propagation algorithm is given below[9]. The operation of the typical MLP with back propagation algorithm is as follows.

The operation of the typical back propagation network occurs as follows.

1. After presenting input data to the input layer , information propagates through the network to the output layer (forward propagation). During this time input and output states for each neuron will be set.[12]

$$x_j^{[s]} = f(I_j^{[s]}) = f(\sum (w_{ij}^{[s]} * x_i^{[s-1]}))$$

Where $x_j^{[s]}$ denotes the current input state of the jth neuron in the current [s] layer. $I_j^{[s]}$ Denotes the weighted sum of inputs to the jth neuron in the current layer[s].f is conventionally the sigmoid function. $w_{ij}^{[s]}$ denotes the connection weight between the ith neuron in the current layer [s] and jth neuron in the previous layer [s-1]

2. Global error is generated based on the summed difference of required and calculated output values of each neuron in the output layer. The Normalized System error E (glob) is given by the equation

$E(glob) = 0.5 * \Sigma (r_k - o_k)^2$ and $(r_k - o_k)$ denotes the difference of required and calculated output values.

3 Global error is back propagated through the network to calculate local error values and delta weights for each neuron. Delta weights are modified according to the delta rule that strictly controls the continuous decrease of synaptic strength of those neurons that are mainly responsible for the global error. In this manner the regular decrease of global error can be assured[9].

$$E_j^{[s]} = x_j^{[s]} * (1.0 - x_j^{[s]}) * \sum (e_k^{[s+1]} * w_{kj}^{[s+1]})$$

Where $E_j^{[s]}$ is the scaled local error of the jth neuron in the current layer [s] layer.

$$\Delta w_{ji}^{[s]} = lcoef * e_j^{[s]} * x_i^{[s-1]}$$

Where $\Delta w_{ji}^{[s]}$ denotes the delta weight of the connection between the current neuron and the joining neuron. Here, lcoef denotes the learning coefficient/ learning constant of the

training parameters.[12] Synaptic weights are updated by adding delta weights to the current weights. Neural network simulate neurotransmission by changing the strength of inter neural connections. Positive synaptic weights provide amplified neural signal and stronger effect to the joining neuron . No modification in the information flow is modeled by zero weight. Negative weights mean inhibition. The learning process of a neural network is similar to the learning function of the human brain. The learning takes place by providing data for both inputs and outputs. The calculated output value is compared to the required value that is also given in the training set. Depending on the difference between the required and the calculated output values, the network adjusts the synaptic weights whose distribution constitutes the basis of the problem-solving algorithm. The network processes the elements of the training set in cyclical order until the difference becomes lower than a given value. In the second part of the training process, the system is tested. The test set is fundamentally similar to the training set, but it contains different data. If testing fails network structure or the learning parameters are then modified.

2.2 Support Vector Regression

The Support Vector Machines have found applications in the domain of function approximation and regression [10][11]. Although its initial saplings have been found from optical character recognition, it was soon applied to object recognition tasks [12]. Later on, these machines were applied to regression and time series prediction tasks [10][13][14][15]. There are many industrial applications using support vector regression method. It is a constructive learning procedure based on the statistical learning theory (Vapnik, 1995) [16]. It is an inductive machine learning technique based on the structural risk minimization principle that aims at minimizing the true error. These type of machines performs classification by constructing an N-dimensional hyper plane that optimally separates the data into two categories. The prime objective of SVM is to find an optimal separating hyper-plane that correctly classifies data points as much as possible.SVM's also separates the points of two classes as far as possible by minimizing the risk of misclassifying the training samples and unseen test samples. In a typical regression problem, we are given a training set $\{x_i , y_i \}_{i=1}^{N} \subset R^d$ X R where x_i and y_i are the input and output variable vector of the ith pair. Support vector regression (Scholkopf & Smola, 2002) is based on a kernel method that performs nonlinear regression based on the kernel trick. Essentially, each input $x_i \in R^d$ is mapped implicitly via a nonlinear feature map Φ (.) to some kernel-induced feature space F where linear regression is performed.

In SVR (Smola & Schölkopf, 2004; Vapnik, 1995), the goal is to get a function f (X) that has at most ϵ deviation from the actually obtained targets y_i for all the training data[16]. Deviation larger than ϵ is not accepted. In the case of linear functions f taking the form

$$f(x) = \sum_{i=1}^{n} w_i x_i + b \text{ with w} \in \text{ R , b} \in \text{R----(1)}$$

one way to ensure minimum ϵ deviation is to minimize the norm

i.e.,

$$\|w\|^2 = \sum_{i=1}^{n} w_i^T w_i$$

The problem can be written as a Convex optimization problem

$$\text{Minimize} \quad \frac{1}{2}\|w\|^2$$

$$\text{Subject to} \quad \begin{cases} y_i - \sum_{i=1}^{n} w_i x_i - b \le \in \\ \sum_{i=1}^{n} w_i x_i + b - y_i \le \in \end{cases} \quad (2)$$

The assumption in (2) is that such a function actually exists that approximates all pairs (x_i,y_i) with ∈ precision. In the case where constraints are not feasible, slack variables ξ_i , ξ_i* were introduced. This case is called soft margin formulation (Bennett & Mangasarian, 1992) and is described by the following problem;

$$\text{Minimize} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*)$$

Subject to

$$\begin{cases} y_i - \sum_{i=1}^{n} w_i x_i - b \le \in + \xi_i \\ \sum_{i=1}^{n} w_i x_i + b - y_i \le \in + \xi_i^* \end{cases} \quad where \xi_i, \xi_i^* \ge 0 (3)$$

The constant C > 0 determines the amount up to which deviations larger than e are tolerated. This is called ϵ-insensitive loss function $|\xi_\varepsilon|$ and is described by

$$\begin{cases} |\xi_\varepsilon| = 0, if\ |\xi| \le \varepsilon \\ |\xi| - \varepsilon, otherwise \end{cases} (4)$$

It is observed that in most cases the optimization problem (3) can be solved more easily in its dual formulation [6]. Hence, Lagrange multipliers are used to get the dual formulation as described in Fletcher (1989), which is as follows:[16]

$$L = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) - \sum_{i=1}^{l}(\eta_i\xi_i + \eta_i^*\xi_i^*) - \sum_{i=1}^{l}\alpha_i(\varepsilon + \xi_i - y_i + \sum_{i=1}^{n} w_i x_i + b) - \sum_{i=1}^{l}\alpha_i^*(\varepsilon + \xi_i^* + y_i - \sum_{i=1}^{n} w_i x_i - b) - \sum_{i=1}^{l}(\eta_i\xi_i + \eta_i^*\xi_i^*) \quad (5)$$

Here, L is the Lagrangian and α_i, α_i*,η_i,η_i* are lagrange multipliers. Hence the dual variables in (5) have to satisfy positivity constraints i.e.,

$\alpha_i, \alpha_i^*, \eta_i, \eta_i^* \geq 0$ (6)

According to the saddle point condition the partial derivatives of L with respect to primal variables (w, b, ξ_i, ξ_i^*) have to vanish for optimality. Therefore, we get

$$\frac{\partial L}{\partial b} = \sum_{i=0}^{l} (\alpha_i^* - \alpha_i) = 0 (7)$$

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) x_i = 0 (8)$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \eta_i = 0 (9)$$

$$\frac{\partial L}{\partial \xi_i^*} = C - \alpha_i^* - \eta_i^* = 0 (10)$$

Substituting (7) to (10) into (5), yields the dual optimization problem;

Maximize -

$$-\frac{1}{2} \sum_{i,j=1}^{l} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) \sum_{i,j=1}^{n} x_i x_j - \varepsilon \sum_{i=1}^{l} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{l} y_i (\alpha_i - \alpha_i^*) (11)$$

Subject to

$$\sum_{i=1}^{l} (\alpha_i - \alpha_i^*) = 0 \text{ and } \alpha_i, \alpha_i^* \in [0, C]$$

To make the SV algorithm nonlinear all training patterns xi are mapped $\Phi : \kappa \to \Lambda$ into some feature space Λ as described in (Aizerman, Braverman, & Rozonoer, 1964; Nilsson, 1965) and then using the standard SV regression a linear hyperplane is constructed in the feature space. Using the trick of kernel functions (Cortes & Vapnik, 1995) following QP problem is formulated.

Maximize

$$\begin{cases} \frac{-1}{2} \sum_{i,j=1}^{l} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) K(x_i - x_j) \\ -\in \sum_{i=1}^{l} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{l} y_i (\alpha_i - \alpha_i^*) \end{cases}$$

Subject to $\sum_{i=1}^{l} (\alpha_i - \alpha_i^*) = 0 and \alpha_i, \alpha_i^* \in [0, C]$

The optimal solution obtained

$$w = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) \phi(x_i) and$$

$$f(x) = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) K(x_i, x) + b$$

Where K(.,.) is a kernel function. ---------(12)

Usually more than one kernel used in the literature to map the input space into feature space (Cristianini & Shawe-Taylor, 2000). The question is to find a kernel functions that provides good generalization for a particular problem. One has to try more than one kernel function for a particular problem in order to resolve this issue. Because of the approximate mapping of input space to higher dimensional feature space using different kernel functions, support vectors extracted are different and the number of support vectors varies as well for each kernel. A Gaussian kernel has been used in this research.

3.Results

In this research, Software Reliability Prediction models are constructed using MLP with back propagation and support vector regression algorithms. The dataset consists of two variables namely failure interval length and day of failure. Failure Interval Length and bias have been taken as inputs and day of failure has been taken as output for both the networks. Data consisting of 662 samples were used for training and data with 123 samples were used for testing purposes. We compare the performance of both of the networks for estimating the reliability of the software by considering the Normalized System Error of both the networks. There will be an interactive session for MLP network in which the user has to provide values for number of inputs like selecting learning or output generation, choosing the file from where the network gets it's data, number of inputs, number of outputs, number of samples to be considered, learning rate, number of hidden layers, number of units in the hidden layer, momentum rate, maximum total error, maximum individual error and finally the number of iterations. After providing all these values to the network, the network generates an error file showing the weights of the synaptic connections between individual layers along with the normalized system error of all the input samples. All the results have been generated based on the Normalized System Error of the MLP neural network. The Normalized System Error values along with the values of the other network parameter have been given in the table 3.1 below for training the network. In this research, A single hidden layer is considered for the purpose. Different Normalized System Error values have been generated for training the network by changing the number of units in the hidden layer. Following abbreviations have been considered for network parameters. NI: Number of Inputs, NOUT: Number of Outputs, NIS: Number of Input Samples, MR: Momentum Rate, MTE: Maximum Total Error, MIE: Maximum Individual Error, NIT: Number of Iterations, NHL: Number of Hidden layer, NUHL: Number of Units in the Hidden Layer, NSE: Normalized System Error.

Table3.1: The values of the network parameters during training

NI	NOUT	NIS	MR	LR	MTE	MIE	NIT	NHL	NUHL	NSE
2	1	662	0.9	0.5	0.01	0.001	500	1	1	0.001745
2	1	662	0.9	0.5	0.01	0.001	500	1	2	0.001218
2	1	662	0.9	0.5	0.01	0.001	500	1	3	0.000988
2	1	662	0.9	0.5	0.01	0.001	500	1	4	0.000852
2	1	662	0.9	0.5	0.01	0.001	500	1	5	0.000661
2	1	662	0.9	0.5	0.01	0.001	500	1	6	0.001014
2	1	662	0.9	0.5	0.01	0.001	500	1	7	0.000668
2	1	662	0.9	0.5	0.01	0.001	500	1	8	0.000517
2	1	662	0.9	0.5	0.01	0.001	500	1	9	0.000950
2	1	662	0.9	0.5	0.01	0.001	500	1	10	0.000928

From the table 3.1, it is observed that, as the number of units in the hidden layer increases from one to five keeping the other parameters fixed, the Normalized System Error gradually decreased and there is a slight increase in the NSE when the number of units is six. when the number of number units in the hidden layer(NUHL) is eight keeping other parameters fixed, the Normalized System Error is 0.000517. This error has been considered as the net error generated by the network during training. Given below fig. 3.1 is the graph between numbers of units in the hidden layer versus Normalized System Error during training.

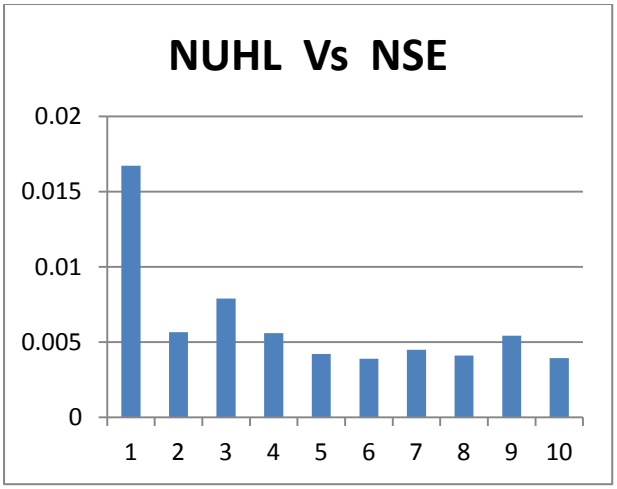

Fig3.1: The relationship between number of units in hidden layer and Normalized System Error

Table3.2: The values of the network parameters during testing

NI	NOUT	NIS	MR	LR	MTE	MIE	NIT	NHL	NUHL	NSE
2	1	123	0.9	0.5	0.01	0.001	500	1	1	0.016720
2	1	123	0.9	0.5	0.01	0.001	500	1	2	0.00567
2	1	123	0.9	0.5	0.01	0.001	500	1	3	0.007885
2	1	123	0.9	0.5	0.01	0.001	500	1	4	0.005599
2	1	123	0.9	0.5	0.01	0.001	500	1	5	0.004208
2	1	123	0.9	0.5	0.01	0.001	500	1	6	0.003896
2	1	123	0.9	0.5	0.01	0.001	500	1	7	0.004495
2	1	123	0.9	0.5	0.01	0.001	500	1	8	0.004109
2	1	123	0.9	0.5	0.01	0.001	500	1	9	0.005418
2	1	123	0.9	0.5	0.01	0.001	500	1	10	0.003949

From the table 3.2, it can be known that, when the number of number units in the hidden layer (NUHL) is six, keeping other parameters fixed, the Normalized System Error is 0.003896. This error has been considered as the net error generated by the network during testing. Given below fig 3.2 is the graph between numbers of units in the hidden layer versus Normalized System Error (NSE) during training.

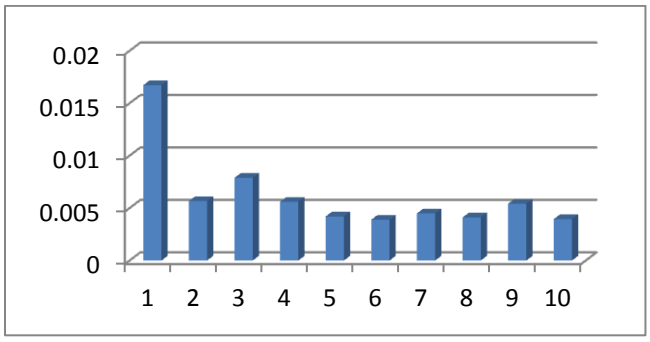

Fig 3.2: The graph between numbers of units in the hidden layer versus Normalized System Error (NSE) during training.

The following table gives the performance of both of the networks (MLP VS SVR) with respect to their NSE values during training and testing.

Table 3.3: The values of the network parameters during training and testing(For both of the networks)

Training				
NI	NOUT	NIS	NSE for MLP	NSE for SVR
2	1	662	0.000517	0.0444
Testing				
NI	NOUT	NIS	NSE for MLP	NSE for SVR
2	1	123	0.003896	0.0738

Experimental results show that MLP with back propagation algorithm performs much better than support vector regression model given the predefined set of variables as input to the above models. The MLP network outperforms SVR by 98.87% during training and 94.85% in testing phases of the networks which is clearly evident from the table 3.3.

Discussion and Conclusion:

The multi-layer perceptron neural network has been trained by back-propagation algorithm to predict the reliability of software along with the support vector regression model. It is essential to predict the reliability of software ,as more software's that are being developed are error prone. Although, there are different models have been proposed for predicting the reliability of the software, the technology of neural networks will definitely be an added advantage given the larger the data set. The development of our model is a practical tool to predict the reliability of software given the predefined variables as input to the networks. As more data are generated, Both the systems improve in precision and can be widely employed in predicting the reliability of software in different environments. In Summary, we developed two network models, one is an MLP with back-propagation algorithm and the other is a Support Vector regression model to predict the reliability of software using the available information. The outcomes of both the models are consistent with the knowledge of the domain expert. Thus in this proposed research, the results support the reliability of the software by both of the networks and also gives an importance of understanding the impact of neural networks for finding the reliability of software.

References

[1] Liang T., Afzel, N. "*On-line prediction of software reliability using an evolutionary*" connectionist model", Journal of Systems and Software, vol. 77,pp. 173–180, 2005.

[2] Hu Q., Xie M., and Ng S., *Software Reliability Predictions using Artificial Neural Networks*, Computational Intelligence in Reliability Engineering (SCI) 40, 197–222, 2007.

[3] Aljahdali, S. "*Prediction of Software Reliability Using Neural Network and Fuzzy logic*", Ph.D. Dissertation presented to the faculty of College of Graduate Studies., Dept. of the Software Engineering and Info. System, George Mason University, Fairfax, Virginia, U.S.A, May 2003.

[4] Aljahdali, S., Sheta, A., and Rine, D., "*Prediction of Software Reliability: A Comparison between regression and neural network non-parametric Models*", Proceeding of the IEEE/ACS Conference, pp.470-471, 2000.

[5] Khoshgoftaar T.M., and Allen, E.B., "*Logistic Regression Modeling of Software Quality*", International Journal of Reliability, Quality and Safety Engineering, 6(4), 1999.

[6] Stewart, W., "*Collinearity and least squares regression*", Statistical Science, pp. 68-100, 1987.

[7] Aljahdali, S., Sheta, A., and Habib, M. "*Software Reliability Analysis Using Parametric and Non-Parametric Methods*", Proceedings of the ISCA 18th International Conference on Computers and their Application, March 26-28, 2003, pp. 63-66.

[8] Aljahdali, S., Sheta, A., and Rine, D., "*Predicting Accumulated Faults in Software Using Radial Basis Function Network*", Proceedings of the ISCA 17th International Conference on Computers and their Application, 4-6, April 2002, pp. 26-29.

[9] Narasinga Rao MR, Sridhar GR, Madhu K, Rao AA. *A Clinical Decision Support System using Multilayer Perceptron Neural Network to Assess Well Being in Diabetes*. J Assoc Physicians India 2009; 57:127–33.

[10] Drucker.H, Burges C.j, Kaufman L, Smola A, Vapnik V, *Support Vector Regression Machines*", in Advances in Neural Information Processing Systems (1997), M.C.Mozer, M.I.Jordon, and T, Petsche(eds.),Vol.9,The MIT press,pp.155-161.

[11] Vapnik V, Golowich S.E, and Smola A, '*Support vector method for function approximation, regression estimation and signal processing*', in Advances in Neural Information Processing Systems(1997), M.C.Mozer, M.I.Jordan, and T.petsche(eds.,),Vol.9, The MIT press, Cambridge, MA,pp.281-287.

[12] Scholkopf,B, Burges, C.,and Vapnik., V., '*Incorporating invariances in support vector machines', in Artificial Neural Networks*—ICANN '96, C.Vonder Malsburg, W.von Seelen,J.Vorbuggen, and B.Sendhoff(eds.,),Vol.1112 of Springer Lecture Notes in Computer Science, Springer- Verlag, Berlin,1996, pp.47-52.

[13] Mattera, D, Haykin.S., ' *Support vector machines for dynamic reconstruction of a chaotic system*', in Advances in Kernel Methods- Support Vector Learning, B.Scholkpf, C.J.C.Burges, and A.J.Smola(eds.,), MIT Press, Cambridge,MA,1999,pp.211-242.

[14] Muller. K.R, Smola A.J, Ratsch. G, Kohlmorgan.J, and Vapnik.V., '*Predicting time series with and support vector machines*', in Artificial Neural Networks-ICANN'97, W.Gerstner,A.Germond, M.Hasler,J.D.Nicoud,(eds.,), Springer Lecture Notes in Computer Science, Springer -Verlag, Berlin,1997, pp.999-1004,

[15] Stitson.M, Gammerman,A., Vapnik.V., Vovk.V., Watkins, C., and Weston, J., '*Support vector regression with anova decomposition kernels*', in Advances in Kernels Methods-Support Vector Learning, B.Scholkopf, C.J.C.Burges, and A.J.Smola (eds.,), MIT Press, Cambridge, MA, 1997,pp.285-292.

[16] M.A.H. Farquad, V.Ravi, S.Bapi Raju, " *Support Vector Regression based hybrid rule extraction methods for forecasting*", Expert Systems with Applications 37 (2010) 5577–5589.