

Software Reliability Prediction Using Neural Network

**Thesis submitted in partial fulfillment of
the requirements for the degree of**

Bachelor of Technology in Computer Science and Engineering

by

Rosalin Maharana

Roll No-110CS0303

under the guidance of

Dr. Durga Prasad Mohapatra



Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769008, Odisha, India
May 2014



Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769008, Odisha, India

Certificate

This is to certify that the work in the thesis entitled “*Software Reliability Prediction Using Neural Network*” by *Rosalin Maharana* is a record of an original research work carried out under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science. The thesis fulfills all requirements as per the regulations of this Institute and has reached the standard needed for submission. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Dr. Durga Prasad Mohapatra

Department of Computer Science and Engineering
National Institute of Technology Rourkela

Acknowledgement

On the submission of the Thesis report, we would like to extend our gratitude and sincere thanks to our supervisor Dr. D. P. Mohapatra, for his constant motivation and support during the course of this work in the last one year. We truly appreciate and value his esteemed guidance and encouragement from the beginning to the end of this thesis. He has been our source of inspiration throughout the thesis work and without his invaluable advice and assistance it would not have been possible for us to complete this thesis.

We would also like to give our most sincere thanks to Dr. S. K. Rath, Head of the Department of Computer Science and Engineering for his support during our work. A special acknowledgement goes to Mr Manmath Kumar Bhuyan, a PhD scholar for his guidance throughout the thesis. We would also like to express our thanks to all who extended their unlimited help to us during our project work and its subsequent documentation.

Last but not least we would like to thank all professors and members of the department of Computer Science and Engineering, NIT Rourkela for their generous help in various ways in completion of this thesis. Furthermore, we would like to take the name of our parents and God who directly or indirectly encouraged and motivated us during this dissertation.

Rosalin Maharana

Declaration

I hereby declare that all the work contained in this report is my own work unless otherwise acknowledged. Also, all of my work has not been previously submitted for any academic degree. All sources of quoted information have been acknowledged by means of appropriate references.

Rosalin Maharana

NIT Rourkela

Abstract

Software engineering is incomplete without Software reliability prediction. For characterising any software product quality quantitatively during phase of testing, the most important factor is software reliability assessment. Many analytical models were being proposed over the years for assessing the reliability of a software system and for modeling the growth trends of software reliability with different capabilities of prediction at different testing phases. But it is needed for developing such a single model which can be applicable for a relatively better prediction in all conditions and situations. For this the Neural Network (NN) model approach is introduced. In this thesis report the applicability of the models based on NN for better reliability prediction in a real environment is described and a method of assessment of growth of software reliability using NN model is presented. Mainly two types of NNs are used here. One is feed forward neural network and another is recurrent neural network. For modeling both networks, back propagation learning algorithm is implemented and the related network architecture issues, data representation methods and some unreal assumptions associated with software reliability models are discussed. Different datasets containing software failures are applied to the proposed models. These datasets are obtained from several software projects. Then it is observed that the results obtained indicate a significant improvement in performance by using neural network models over conventional statistical models based on non homogeneous Poisson process.

Contents

Chapter 1	1
Introduction	1
1.1 Motivation of Our Work	2
1.2 Objective of Our Work.....	3
1.3 Organisation of The Thesis	4
1.4 Literature Review	4
Chapter 2	7
Background.....	7
2.1 Software Reliability.....	7
2.2 Artificial Neural Network	8
2.3 Neural Network Modeling.....	8
2.4 Transfer Function	9
Chapter 3	11
Work Details	11
3.1 Back Propagation Learning Algorithm	11
3.2 Approach for Feed Forward Neural Network.....	12
3.3 Approach for Recurrent Neural Network	14
Chapter 4	16
Implementation & Results.....	16
4.1 Implementation setup	16
4.2 Different Performance Measures.....	18
4.3 Prediction Types.....	18
4.4 Results and Discussion.....	20
4.5 Graphs and Screenshots	22
Chapter 5	33
Conclusion and Future Work.....	33
5.1 Conclusion	33
5.2 Future Work.....	34
References.....	35
Appendix A.....	38
Datasets	38

List of Figures

Figure 2.1: A simple model of artificial neuron-----	9
Figure 3.1: Flowchart for back propagation algorithm-----	12
Figure 3.2: A sample feed forward network-----	13
Figure 3.3: A sample recurrent neural network-----	15
Figure 4.1: Performance of FFNN for Dataset1-----	22
Figure 4.2: Snapshot of performance of FFNN for Dataset1-----	23
Figure 4.3: Performance of FFNN for Dataset2-----	24
Figure 4.4: Snapshot of performance of FFNN for Dataset2-----	25
Figure 4.5: Performance of FFNN for Dataset3-----	26
Figure 4.6: Snapshot of performance of FFNN for Dataset3-----	27
Figure 4.7: Performance of FFNN for Dataset4-----	28
Figure 4.8: Snapshot of performance of FFNN for Dataset4-----	29
Figure 4.9: Performance of RNN for Dataset1-----	30
Figure 4.10: Performance of RNN for Dataset2-----	31
Figure 4.11: Performance of RNN for Dataset6-----	32

List of Tables

Table 4.1: Table of different software failure datasets used-----	17
Table 4.2: Feed forward neural network model results -----	20
Table 4.3: Recurrent neural network model results-----	20
Table 4.4: Comparison with analytical models-----	21
Table 4.5: Comparison between LTP and STP for feed forward neural network-----	21

Chapter 1

Introduction

Software is playing an ever-increasing role in our real time systems. Therefore there has been a gradual growth of concern over quality of software products and reliability has become a main concern from both software user's point of view and software developers' point of view. Also the rapid growth of software products in size and complexity has drawn the attention of researchers to be more focused on quality assessment by the estimation of the time of software testing period quantitatively to avoid any unwanted and unforeseen situation during operational phase. In this thesis report the applicability of neural network models for better reliability prediction in real environment are explored empirically and an assessment method of growth of software reliability using artificial neural network (ANN) mode is presented.

Artificial neural networks are generally known as "Neural Networks" and act in a way similar to the human brain. Non linearity and complexity of the brain is very high and behaves like a parallel computer. It has the ability for organizing its structural constituents known as neurons; hence it performs certain computation very quickly than the fastest computer present on earth. The brain structure is very intense and it builds up its own rules through experiences. Experiences are built up over time with the development of the human brain through many stages. A developing neuron is as similar as a plastic brain. To adapt with the surrounding environment the developing nervous system has the property of plasticity. Plasticity appears to be essential to the functioning of neurons as information processing units in the human brain. Similarly this same thing happens with neural networks made up of artificial neurons. A neural network is a machine that is designed to model the way in which the brain performs a particular task. To achieve good performance, neural networks should have a massive interconnection of simple computing cells referred to as

“neurons” or “processing units”. Neural networks perform essential computations through a process of learning.

Thus a neural network consists of simple processing units and big parallel distributed processors. The ability of storing experiential data and making it available for use comes naturally to it. Artificial neural network do not approach the complexity of the brain. It is similar to brain in two ways: 1. A learning process is used to acquire knowledge from its surrounding by the network. 2. The acquired knowledge is stored by the interneuron connection strengths known as synaptic weights. The procedure used to perform the process of learning is called learning algorithm. Function of learning algorithm is to modify the synaptic weights of the networks in an orderly manner in order to attain a desired design objective.

1.1 Motivation of Our Work

The software market is very competitive in this dynamic world. Software industries attempt to release software to grab the market as soon as it is ready. Now it is a challenge for software developers to rapidly design, implement, test, and maintain complex hardware or software systems as per the demands of the users. Also it is a challenge for software companies to deliver good quality and error free software in right time. The impact of the failures produces severe consequences such as environmental impact, inconvenience, economical losses, loss of human life etc. Needless to say, the reliability of computer systems has become a major concern for our society. Software reliability is an important facet of software quality characteristic. Many researchers have used neural networks to predict software reliability. Different neural networks with different learning methods also have been modelled. It is also observed that connectionist models perform better than the previous parametric models. Prediction of software reliability using computational intelligence (CI) can be very accurate and significant compared to traditional statistical methods. CI can offer promising approaches to software reliability prediction and modeling.

With this motivation, we implemented different neural network models with different learning algorithms and compared their performance results for software reliability prediction with the statistical methods and observed that neural networks perform better than the analytical models. The details of the work are described in the next chapter.

1.2 Objective of Our Work

The main objective of this research work is to implement different connectionist models with different learning regimes. Different datasets containing software failures are applied to the proposed models. These datasets are obtained from several software projects. Then different issues related to method of data representation, some unrealistic assumptions incorporated with software reliability models, and network architecture are discussed.

We have tried to implement the feed forward neural network architecture first with back propagation learning method for reliability prediction. As no work is done regarding the implementation of recurrent neural network with back propagation algorithm till now, so mainly our objective is to implement recurrent neural network architecture with back propagation learning algorithm. Followings are the key points of our implementation.

- Feed Forward Neural Network with one hidden layer and multiple hidden layer along with back propagation learning method
- Recurrent Neural Network with back propagation learning method
- Long term predictability and Short term predictability of feed forward neural networks
- Evaluation of effectiveness of the above proposed models by using different performance parameters

1.3 Organisation of The Thesis

The rest of this thesis report is organised into chapters as follows.

- Chapter 2 describes about the related work done and gives an overall literature review.
- Chapter 3 provides the background concepts used in the remaining part of the thesis. Some theoretical concepts regarding software reliability measures, artificial neural network and back propagation learning algorithm are described. Some basic concepts of feed forward and recurrent neural network are presented.
- Chapter 4 provides a brief review and implementation details of the project work.
- Chapter 5 describes the experimental results of the implemented network models and their performance results.
- Chapter 6 concludes the thesis report with a summary and possible future extension of this work.

1.4 Literature Review

Artificial Neural Network (ANN) is a powerful technique for Software Reliability Prediction.

Werbos [9] proposed back-propagation learning as an alternative to regression technique to identify sources of forecast in uncertainty in a recent gas market model. Thus it can be concluded that neural network models are very useful for regression techniques of forecasting in uncertainty of any data.

Shadmehr et al. [10] estimated model parameters of pharmacokinetics system using feed forward multilayered network and predicted the noise residues in the measured data sample. The authors compared the results with that of the optimal

Bayesian estimator and found the performance was better than the maximum likelihood estimator [11].

The ANN tools and feed forward network using back propagation algorithm are applied for reliability and software quality prediction [12–14]. The authors developed a connectionist model and took failure data set as input to produce reliability as output. These papers describe network architecture, method of data representation and some unrealistic assumptions associated with software reliability models.

Karunanithi et al. [15] predicted software reliability using feed forward network and recurrent network. The authors compared the result with 14 different literature representative data sets and suggested that neural network produced better predictive accuracy compared to analytical models at end-point predictions.

Sitte [16] analyzed two methods for software reliability prediction: 1) neural networks and 2) parametric recalibration models. These approaches differentiate the neural networks and parametric recalibration models in the context of software reliability prediction and conclude that neural networks are much simpler and better predictors.

Tian et al. [7] predicted software reliability using recurrent neural network. Bayesian regularization is applied to train the network. The authors commented that their proposed approach produced less average relative prediction error than well known prediction techniques.

RajKiran et al. [17] implemented the use of wavelet neural networks (WNN) to predict software reliability. In this paper, the authors employed two kinds of wavelets i.e. Morlet wavelet and Gaussian wavelet as transfer functions. They made a comparison on test data with multiple linear regression (MLR), multivariate adaptive regression splines (MARS), back-propagation trained neural network (BPNN) and threshold accepting trained neural network (TANN), pi-sigma network (PSN), general regression neural network (GRNN) and found that its performance is better than others.

Lo [18] designed a model for software reliability prediction using artificial neural networks. This approach examines several conventional software reliability growth models without assuming some unrealistic things.

Fuzzy Wavelet Neural Network (FWNN) is used for phase space reconstruction technology and for software reliability prediction [19]. In this work, the network architecture is designed easily by taking the failure data as input.

Chapter 2

Background

2.1 Software Reliability

The probability that a software will perform a required function under stated conditions for a specified period of time is known as software reliability. Software reliability assessment is a very vital factor to characterise the quality of any software product quantitatively during testing phase.

Software Reliability Measures

- **Failure Rate:** It is the rate of occurrence of failures. It also represents number of failures in specified period of time.
- **Mean Time Between Failures (MTBF):** It is the average time between failures. No of hours taken to pass before a failure occurs is the MTBF. It is the inverse of failure rate.
- **Reliability:** The probability that an item will perform a required function without failure under the stated conditions for a specified period of time is called reliability. It takes into account the mission time.
- **Availability:** The probability that an item is in operable state at any time is called availability. It accounts for repairs and down time.

Software Reliability Growth Models

It includes two types of models

- Parametric models
- Nonparametric models

Parametric models are based on non homogeneous Poisson process. Neural network is non parametric model and based on statistical failure data. Nonparametric models are more flexible.

Different Reliability Metrics

- Failure rate
- Next time to failure
- Time between failures
- Cumulative failures detected

2.2 Artificial Neural Network

- It can be defined as a system where data can be processed through a number of nodes similar to neurons in brain.
- Each node is assigned with a function and it determines the node output with the help of some parameters available locally to it for a set of given input.
- By adjusting weight of these parameters the node function can be altered as intended.

2.3 Neural Network Modeling

- Like a brain, a neural network also performs in similar fashion. It has some learning mechanism designed within it for modelling the reliability.
- A number of neurons constitute NN which are simple processing elements. These neurons are connected to each other directly through communications links associated with some weight.
- Supervised learning method is used to train the NN with a series of sample input and to compare the responses overall for the pre specified period of time with the expected sample output.
- The training procedure is carried out until expected and convincing responses are provided by the network. The neurons are arranged layer by layer and the connection patterns within and in-between layers make the network architecture.
- The network can be either single-layered or multi-layered; layers of interconnected links between the neuron slabs determine it.

2.4 Transfer Function

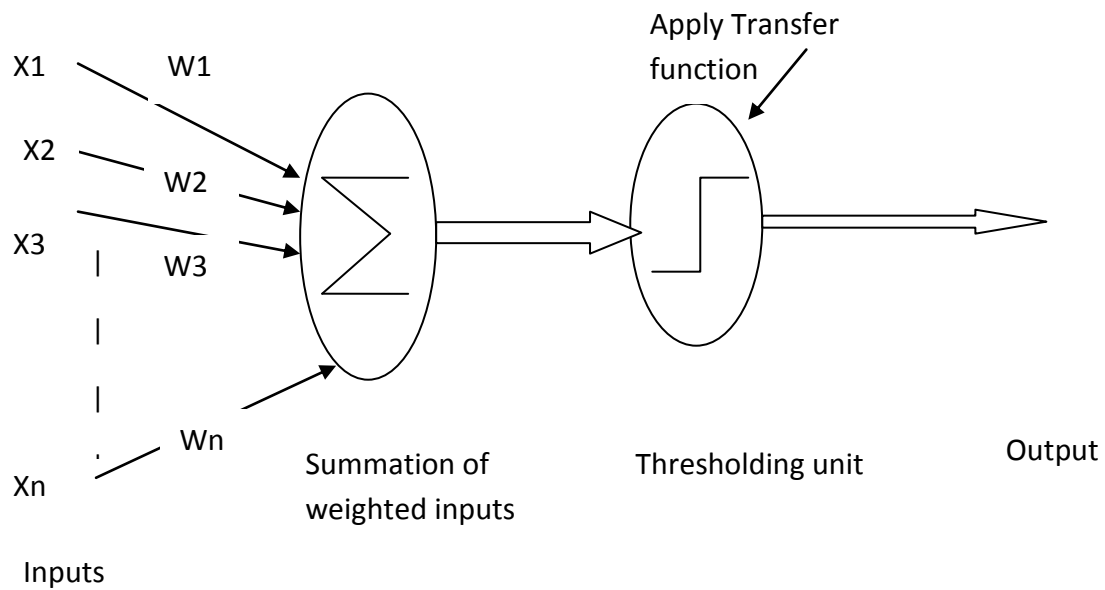


Figure 2.1: A simple model of artificial neuron

Let I = input to the neural network

Where $I = \sum_{i=1}^n W_i X_i$

Then $Y = F(I)$

where Y is the output of the neural network and F is the transfer function.

Hyperbolic Tangent Transfer Function

$$Y = F(I) = \frac{e^I - e^{-I}}{e^I + e^{-I}}$$

Y varies between -1 and +1.

Log Sigmoid Transfer Function

$$Y=F(I)=\frac{1}{1+e^{-I}}$$

Y varies between 0 and +1.

Both log sigmoid and hyperbolic tangent functions are continuous. In this thesis report we have used log sigmoid as transfer function.

Chapter 3

Work Details

We have implemented feed forward neural network and recurrent neural network with back propagation learning algorithm.

3.1 Back Propagation Learning Algorithm

Algorithm:

1. Initialize the weights
2. Repeat
3. For each training pattern
4. Train on that pattern
5. Find error for each pattern and mean square error for total no of patterns
6. Update the connecting weights by calculating errors layer by layer backward
7. End
8. Until the error is acceptably low.

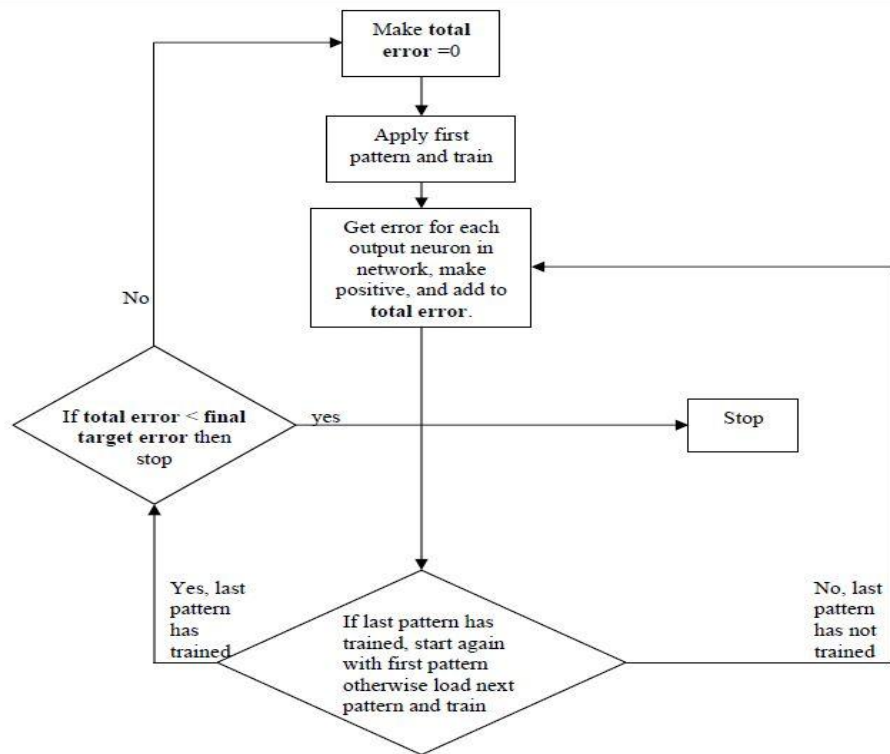


Figure 3.4: Flowchart for back propagation algorithm

3.2 Approach for Feed Forward Neural Network

- Here Back Propagation Learning rule is applied to a feed forward network.
- The basic feed forward neural network architecture comprises in two steps.
 - 1) feed forward NN
 - 2) back propagation
- Here the input vector is propagated through a weight layer. It is combined with the previous state activation as it depicted in next Figure 3.2.
- The conventional feed-forward neural network consists of two-layered network. The network comprises of two steps mapping.

$$y(t) = G(F(x(t)) \dots\dots\dots(1)$$

- The back-propagation learning techniques are used in the above equation 1 to update the weights of the network (F and G) for training the feed forward back propagation network. The operation is restricted in this paper to (“hidden/state” layer and “output” layer).

- The input vector 'x' is propagated with a layer associated with weight V as depicted in equation 3.

$$y_j(t) = f(\text{net}_j(t)) \text{-----}(2)$$

n

$$\text{net}_j(t) = \sum_i (v_{ji})(x_i(t)) + \theta_j \text{-----}(3)$$

i

where n is the number of inputs nodes, θ_j is a bias and f is an activation function.

- The output of the network is calculated by state and weight W associated with that output layer.

$$y_k(t) = g(\text{net}_k(t)) \text{-----} (4)$$

m

$$g(\text{net}_k(t)) = \sum_j y_j(t)w_{kj} + \theta_k \text{-----} (5)$$

j

where m is the number of states or 'hidden' nodes, θ_k is a bias and g is an activation function.

Here sigmoid function is taken as activation function.

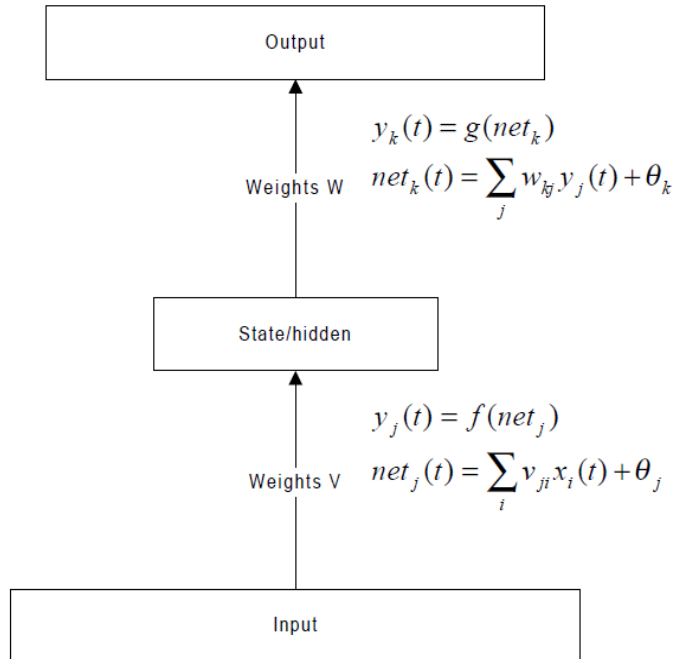


Figure 3.5: A sample feed forward network

3.3 Approach for Recurrent Neural Network

- Here Back Propagation Learning rule is applied to a recurrent network.
- The basic recurrent neural network architecture comprise in two steps.
 - 1) feed forward NN
 - 2) back propagation with recurrent.
- Here the input vector is propagated through a weight layer. It is combined with the previous state activation through an additional recurrent weight layer, R as it depicted in next Figure 3.3.
- The conventional feed-forward neural network consists of two-layered network. The network comprises of two steps mapping

$$y(t) = G(F(x(t))) \dots\dots\dots(1)$$

- The back-propagation learning techniques are used in the above equation 1 to update the weights of the network (F and G) for training the Recurrent Back Propagation Network. The operation is restricted in this paper to (“hidden/state” layer and “output” layer).
- The input vector 'x' is propagated with a layer associated with weight V and combined with previous state activation associated with recurrent weight U as depicted in equation 3.

$$y_j(t) = f(\text{net}_j)(t) \text{-----}(2)$$

$$\text{net}_j(t) = \sum_i (v_{ji})(x_i(t)) + \sum_h (u_{jh})(y_h(t-1)) + \theta_j \text{-----}(3)$$

$$\text{net}_j(t) = \sum_i (v_{ji})(x_i(t)) + \sum_h (u_{jh})(y_h(t-1)) + \theta_j \text{-----}(3)$$

where n is the number of inputs nodes, θ_j is a bias, m is the number of states or ‘hidden’ nodes, and f is an activation function.

- The output of the network is calculated by state and weight W associated with that output layer.

$$y_k(t) = g(\text{net}_k(t)) \text{-----} (4)$$

$$\text{net}_k(t) = \sum_j y_j(t)w_{kj} + \theta_k \text{-----} (5)$$

$$g(\text{net}_k(t)) = \sum_j y_j(t)w_{kj} + \theta_k \text{-----} (5)$$

where g is an activation function.

Here sigmoid function is taken as activation function.

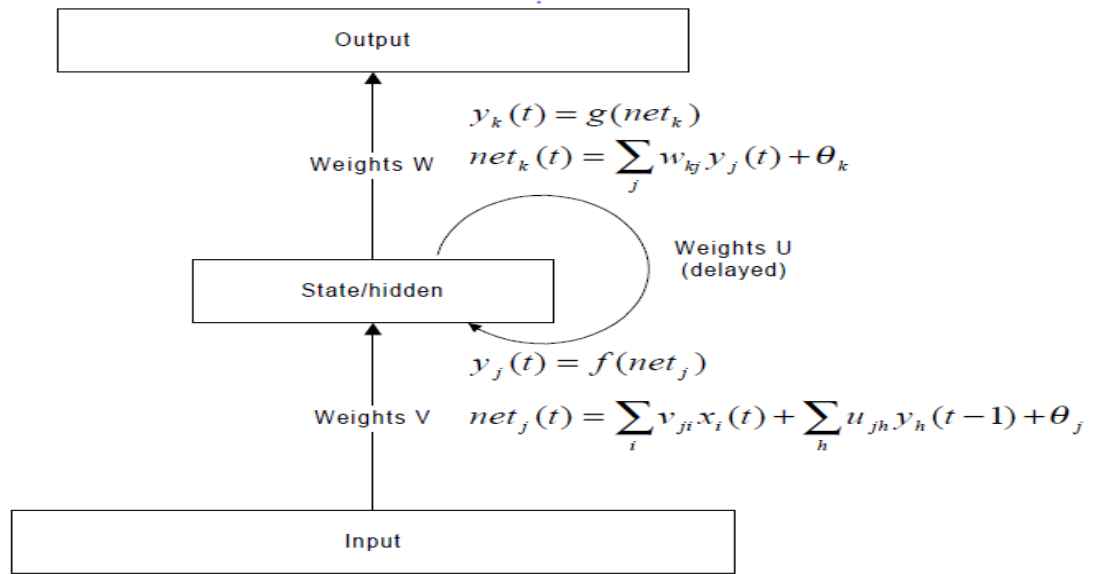


Figure 3.6: A sample recurrent neural network

Chapter 4

Implementation & Results

4.1 Implementation setup

- The FFNN and RNN with back propagation learning algorithm is implemented using MATLAB 7.10.0.499.
- In our prediction experiment, failure data during system testing phase of various projects collected at Bell Tele-phone Laboratories, **Cyber Security and Information Systems In-formation Analysis Centre(CSIAC) by John D. Musa** are considered.
- CSIAC provides software failure datasets in order to support the project manager to monitor testing, estimating the project schedule, and helping the researchers to evaluate the reliability model.
- The data set consists of
 - Failure Number
 - Failure Interval Lengths/Time Between Failures (TBF) in CPU secs
 - Day of Failure of software project
- We have taken 5 numbers of application software testing data set for demonstration of predictive performance and prediction accuracy as shown in Figure 5.1.
- 70% of each dataset is used for training the model and the rest failure data is used for validating the model.

Table 4.1: Table of different software failure datasets used

Project Code	Project Name	Number of Failures	Development Phases
DBS[1]	Real Time Command & Control System	136	System Test Operations
DBS[2]	Real Time Command & Control System	54	System Test Operations
DBS[3]	Real Time Command & Control System	38	System Test Operations
DBS[4]	Real Time Command & Control System	53	System Test Operations
DBS[6]	Commercial System	73	Subsystem Test

- For FFNN proposed model,
 - Cumulative Execution time is taken as input
 - No of Cumulative Failures is taken as output
 - Both cumulative execution time and no of cumulative failures are normalized in the range 0 to 1.
 - Graph is plotted with cumulative execution time in X-axis and no of cumulative failures in y-axis.
- For RNN proposed model
 - Cumulative execution time for nth failure is taken as input
 - Cumulative execution time for n+1th failure is taken as output

- Graph is plotted with no of failure taken in X-axis and cumulative execution time in y-axis.
- Cumulative execution time is normalized in the range 0 to 1.

4.2 Different Performance Measures

- The following performance measures are being used to validate the proposed models:

- Relative Error(%): $RE_i = (| (P_i - A_i) / A_i |) * 100$

- Average Relative Error(%): $1/n \sum_{i=1}^n RE_i$

- Root Mean Squared Error: $RMSE = \sqrt{ \left(\sum_{i=1}^n (P_i - A_i)^2 \right) / n }$

- Mean Absolute Error: $\left[\sum_{i=1}^n | P_i - A_i | \right] / n$

- Mean Error: $\left[\sum_{i=1}^n (P_i - A_i) \right] / n$

where

P_i =Predicted Value

A_i =Actual Value

n =total no of observations/patterns

4.3 Prediction Types

- Neural Networks can predict software reliability in two ways:
 - Long Term Prediction
 - Short Term Prediction

Short term prediction (STP)

Suppose we have a software failure dataset having the following format.

Cumulative

<u>execution time</u>	<u>No of failures</u>
t1	x1
t2	x2
t3	x3
tk	xk
t(k+1)	x(k+1)
tn	xn
t(n+1)	x(n+1)

We can interpret no of failures as a function of cumulative execution time .

Suppose y is the function, then y can be written as $y(t)=x$

- Then a neural network can be modeled by with 3 layers taking k input neurons in input layer, 1 output neuron in output layer and a hidden layer.
- Taking $y(t1), y(t2), y(t3) \dots y(tk)$ as inputs to the neural network and Predicting $y'(t(k+1))$ as output (where $y(t(k+1))$ is taken as target value) is known as short term prediction or 1-step ahead prediction .

Long term prediction(LTP)

Considering the neural network model as explained in previous section,

- $y(t1), y(t2), y(t3) \dots y(tk)$ are taken as inputs to the neural network and $y'(t(k+1))$ is predicted as output (where $y(t(k+1))$ is taken as target value).
- Then $y(t2), y(t3), y(t4) \dots y'(t(k+1))$ are taken as inputs to the neural network and $y'(t(k+2))$ is predicted as output (where $y(t(k+2))$ is taken as target value).
- Continuing like this up to nth pattern is known as long term prediction or n-step ahead prediction.

4.4 Results and Discussion

Different dataset are taken and results are shown in the following tables and figures.

Table 4.2: Feed forward neural network model results

Data set No	No of Input neuron	No of Output neuron	No of Hidden neuron	Average Relative Error (%)	RMSE	Mean Absolute error	Mean Error
1	2	1	9	3.0624	0.0045	0.0259	-0.0017
2	2	1	9	2.8719	0.0076	0.0244	-0.0090
3	2	1	3	8.8602	0.0256	0.0723	0.0623
4	2	1	5	4.9537	0.0122	0.0393	0.0342

Table 4.3: Recurrent neural network model results

Data set No	No of Input neuron	No of Output neuron	No of Hidden neuron	Average Relative Error (%)	RMSE	Mean Absolute error	Mean Error
1	2	1	3	6.7607	0.0127	0.0508	-0.0297
2	2	1	3	15.0380	0.0387	0.1159	-0.1122
6	2	1	3	8.9263	0.0232	0.0725	-0.0725

Table 4.4: Comparison with analytical models

Analytical Models	Average Relative Error(%)
Logarithmic	16.23
Exponential	17.93
Inverse Polynomial	18.45
Power	26.42
Delayed S-shape	25.61

Table 4.5: Comparison between LTP and STP for feed forward neural network

	DS1		DS2		DS3		DS4	
	STP	LTP	STP	LTP	STP	LTP	STP	LTP
No of input neuron	3	3	3	3	3	3	3	3
No of hidden layer	1	1	1	1	1	1	1	1
No of hidden neurons	5	5	5	5	5	5	5	5
Average relative error(%)	3.0249	4.4809	4.1843	9.9864	7.2198	8.3546	4.9705	5.9054
RMSE	0.0054	0.0082	0.0118	0.0220	0.0210	0.0240	0.0123	0.0153
Mean absolute error	0.0274	0.0410	0.0379	0.0852	0.0599	0.0693	0.0394	0.0513
Mean error	- 0.0207	- 0.0388	- 0.0321	- 0.0852	0.0560	0.0662	0.0342	- 0.0513

4.5 Graphs and Screenshots

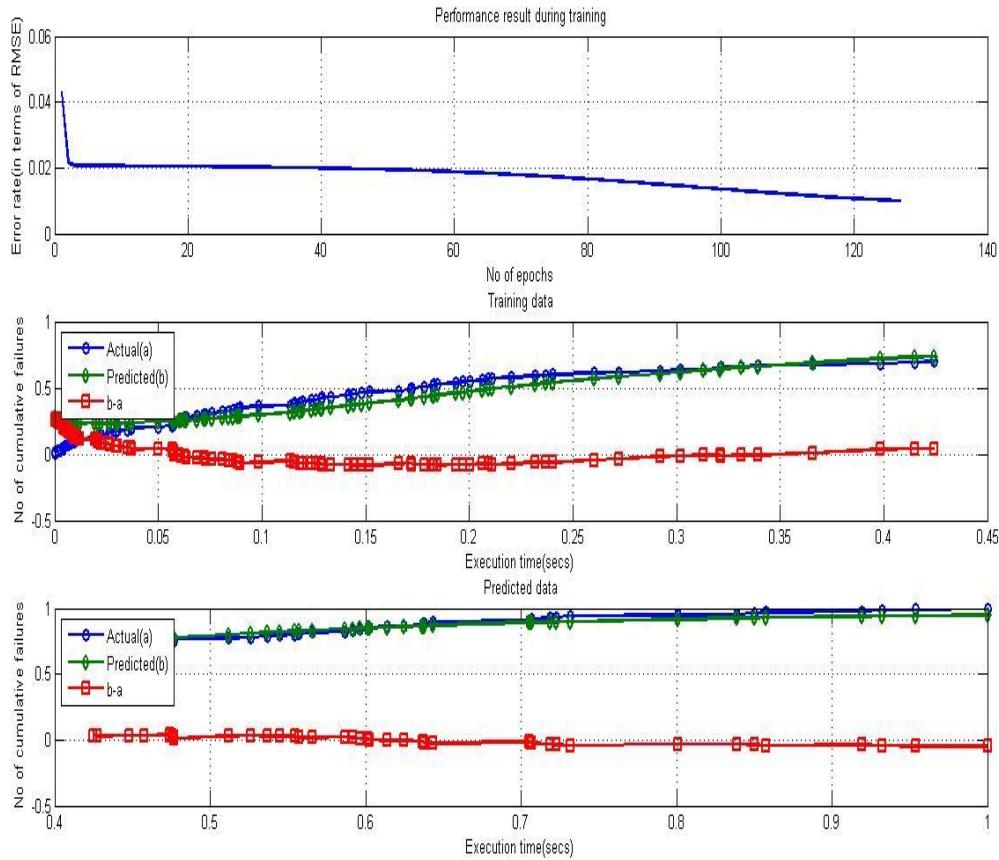


Figure 4.1: Performance of FFNN for Dataset1

In the above figure, plots for feed forward neural networks are shown. The first graph is the plot between number of epochs vs. error rate in terms of RMSE during training. It shows how the error rate gradually decreases with the no of epochs by the use of back propagation learning. The second and third graph is the plot between cumulative execution time vs. no of cumulative failures for Dataset1. Second graph is plotted for the training period and third graph is plotted for the prediction period. In both second and third graph blue colour line represents actual data, green colour line represents predicted data and red colour line represents difference between predicted data and actual data.

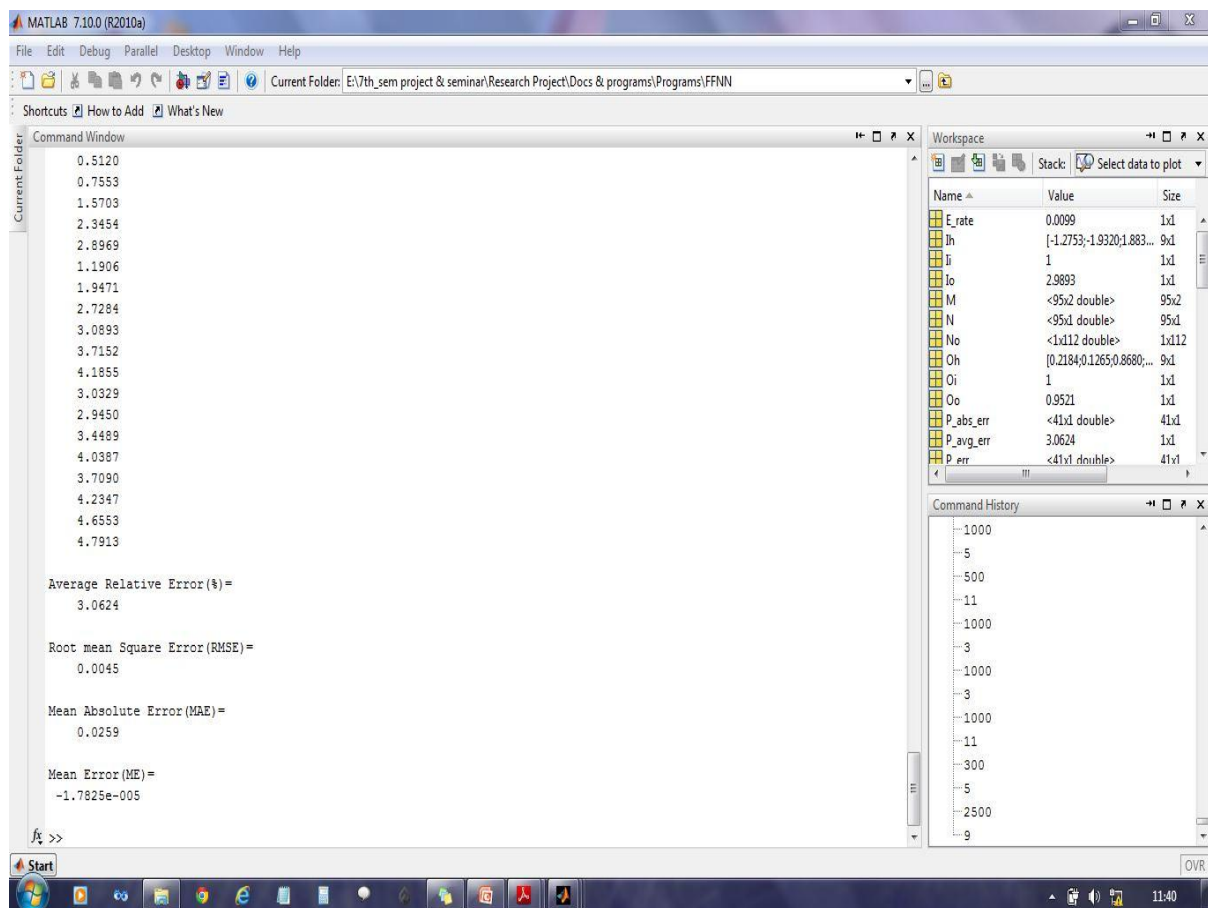


Figure 4.2: Snapshot of performance of FFNN for Dataset 1

In the above figure, the snapshot of different performance measure values using FFNN for Dataset1 is shown.

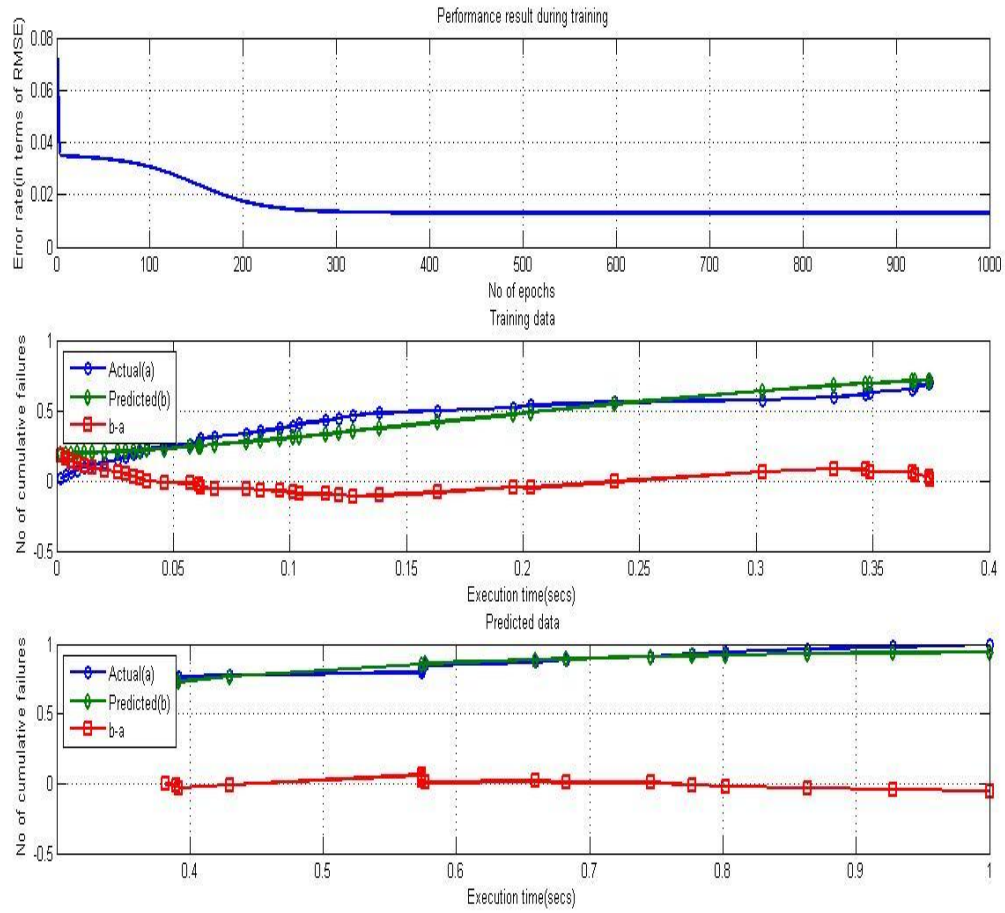


Figure 4.3: Performance of FFNN for Dataset2

In the above figure, plots for feed forward neural networks are shown. The first graph is the plot between number of epochs vs. error rate in terms of RMSE during training. It shows how the error rate gradually decreases with the no of epochs by the use of back propagation learning. The second and third graph is the plot between cumulative execution time vs. no of cumulative failures for Dataset2. Second graph is plotted for the training period and third graph is plotted for the prediction period. In both second and third graph blue colour line represents actual data, green colour line represents predicted data and red colour line represents difference between predicted data and actual data.

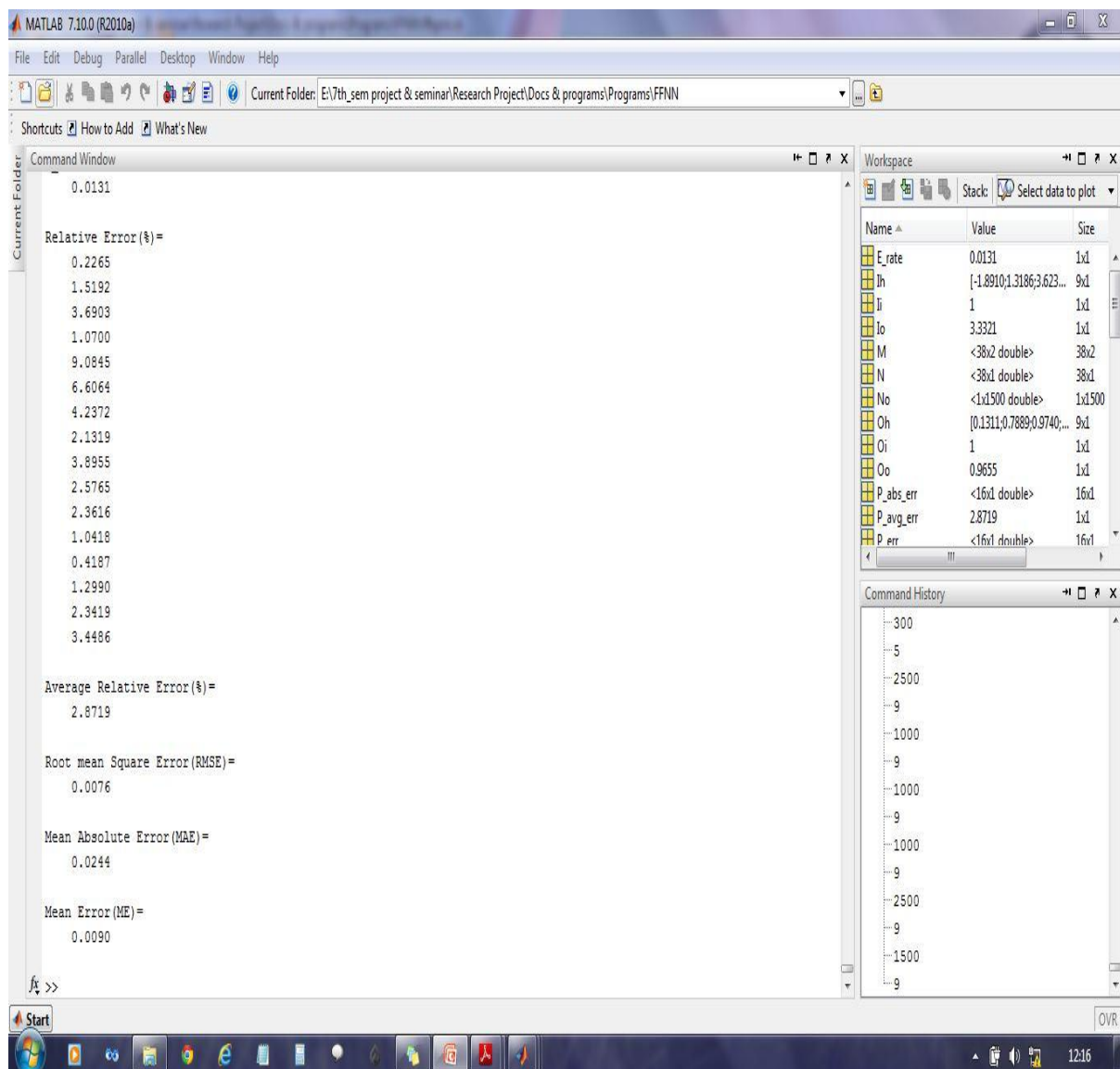


Figure 4.4: Snapshot of performance of FFNN for Dataset2

In the above figure, the snapshot of different performance measure values using FFNN for Dataset2 is shown.

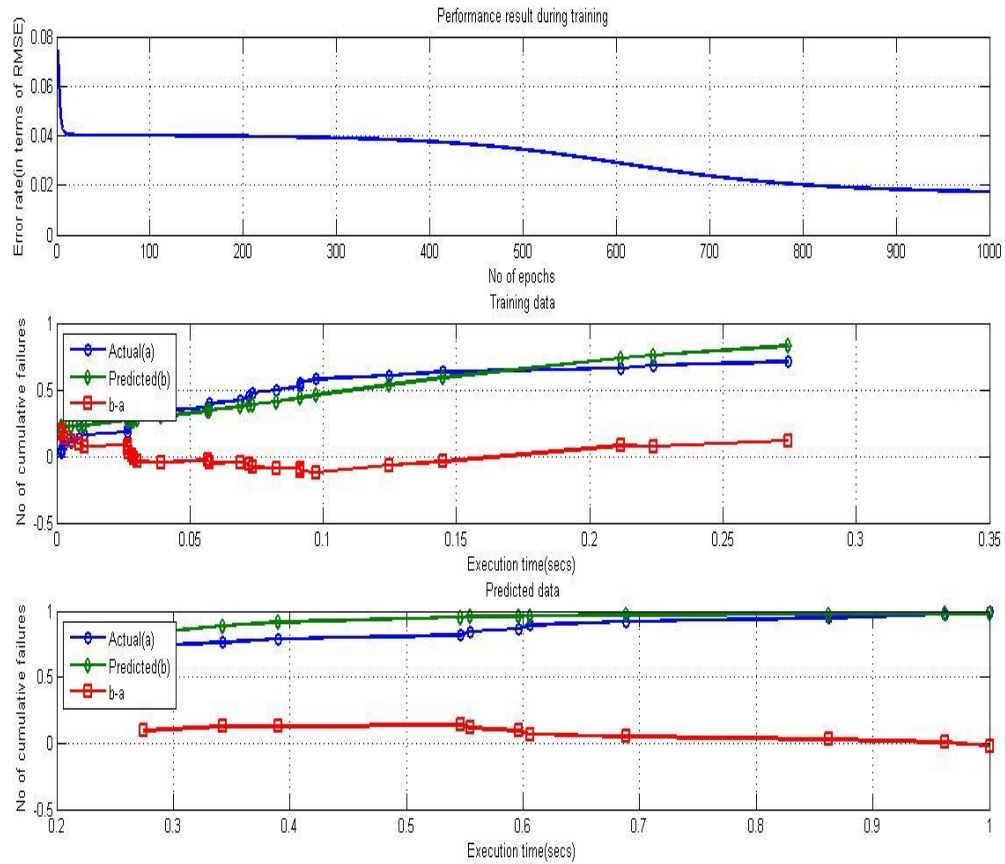


Figure 4.5: Performance of FFNN for Dataset3

In the above figure, plots for feed forward neural networks are shown. The first graph is the plot between number of epochs vs. error rate in terms of RMSE during training. It shows how the error rate gradually decreases with the no of epochs by the use of back propagation learning. The second and third graph is the plot between cumulative execution time vs. no of cumulative failures for Dataset3. Second graph is plotted for the training period and third graph is plotted for the prediction period. In both second and third graph blue colour line represents actual data, green colour line represents predicted data and red colour line represents difference between predicted data and actual data.

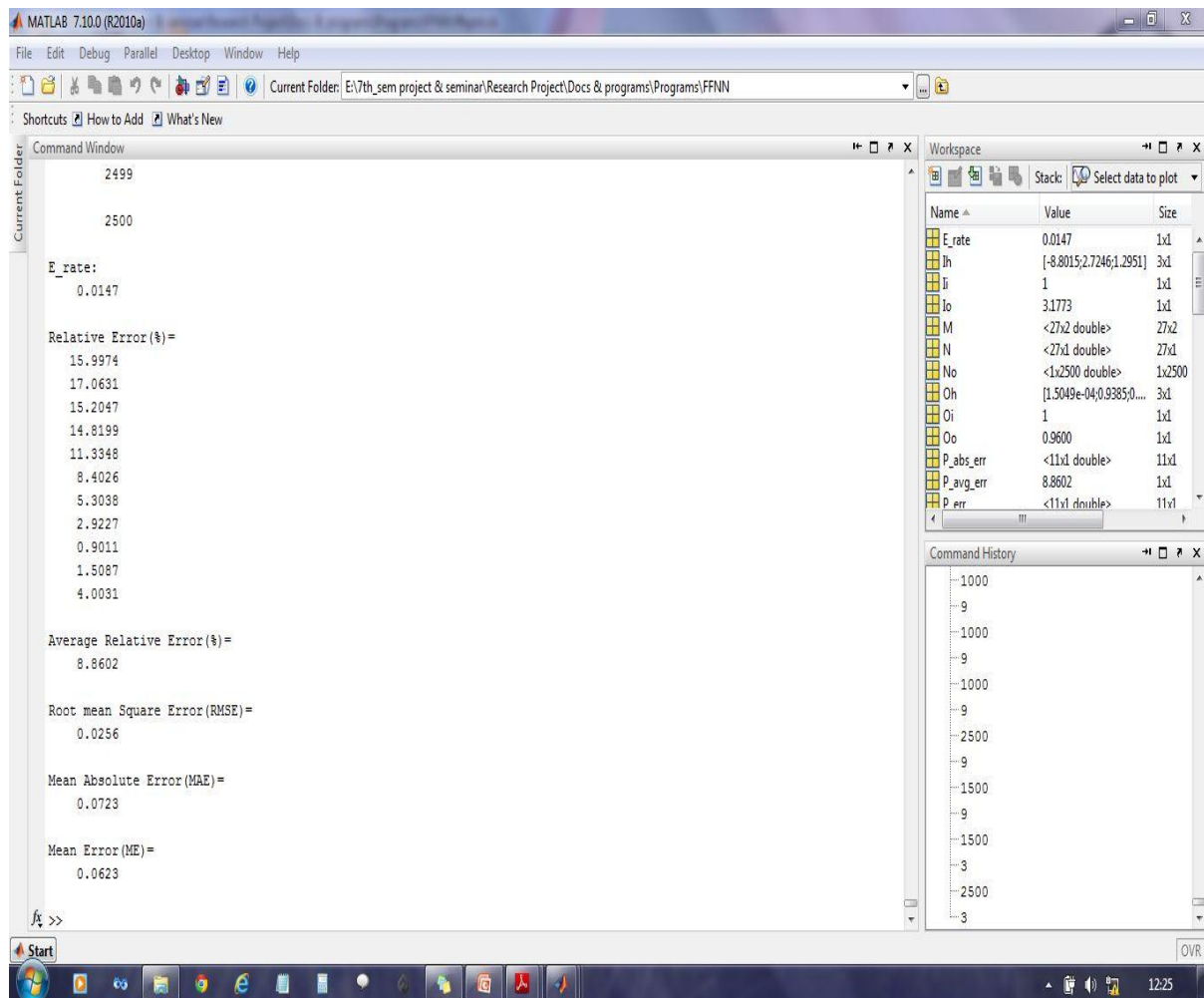


Figure 4.6: Snapshot of performance of FFNN for Dataset3

In the above figure, the snapshot of different performance measure values using FFNN for Dataset3 is shown.

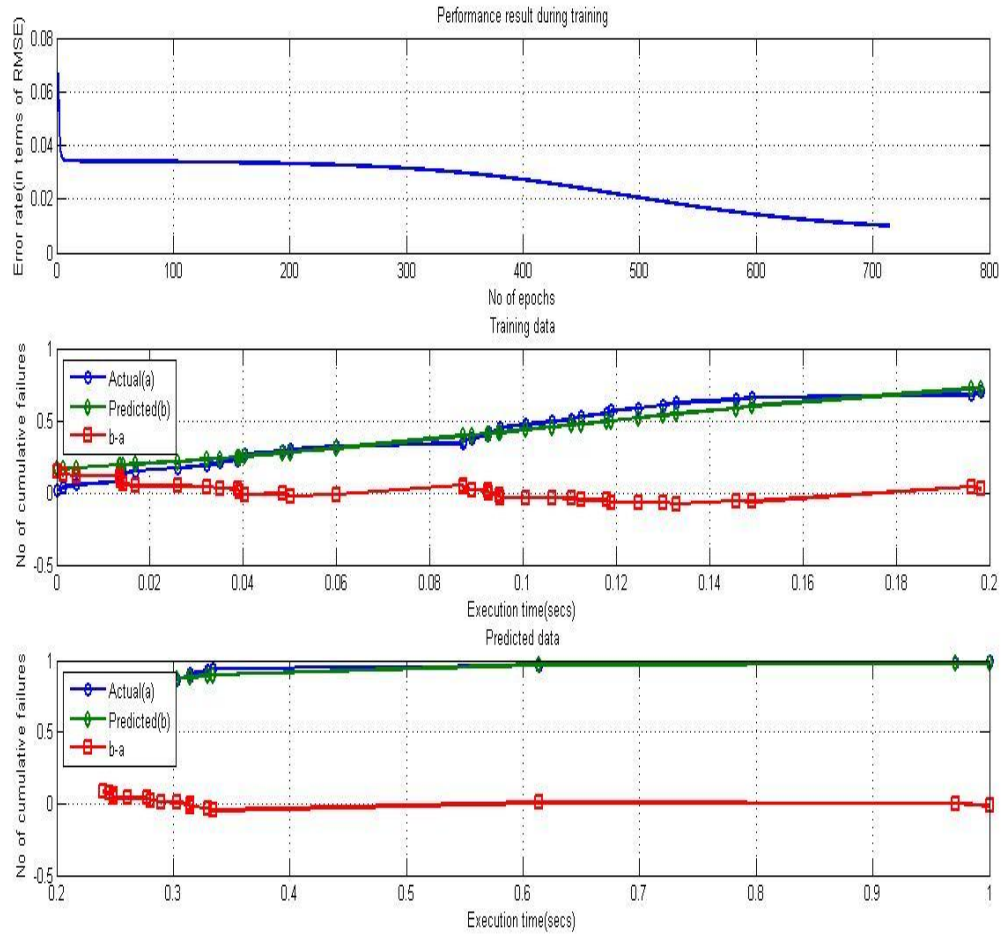


Figure 4.7: Performance of FFNN for Dataset4

In the above figure, plots for feed forward neural networks are shown. The first graph is the plot between number of epochs vs. error rate in terms of RMSE during training. It shows how the error rate gradually decreases with the no of epochs by the use of back propagation learning. The second and third graph is the plot between cumulative execution time vs. no of cumulative failures for Dataset4. Second graph is plotted for the training period and third graph is plotted for the prediction period. In both second and third graph blue colour line represents actual data, green colour line represents predicted data and red colour line represents difference between predicted data and actual data.

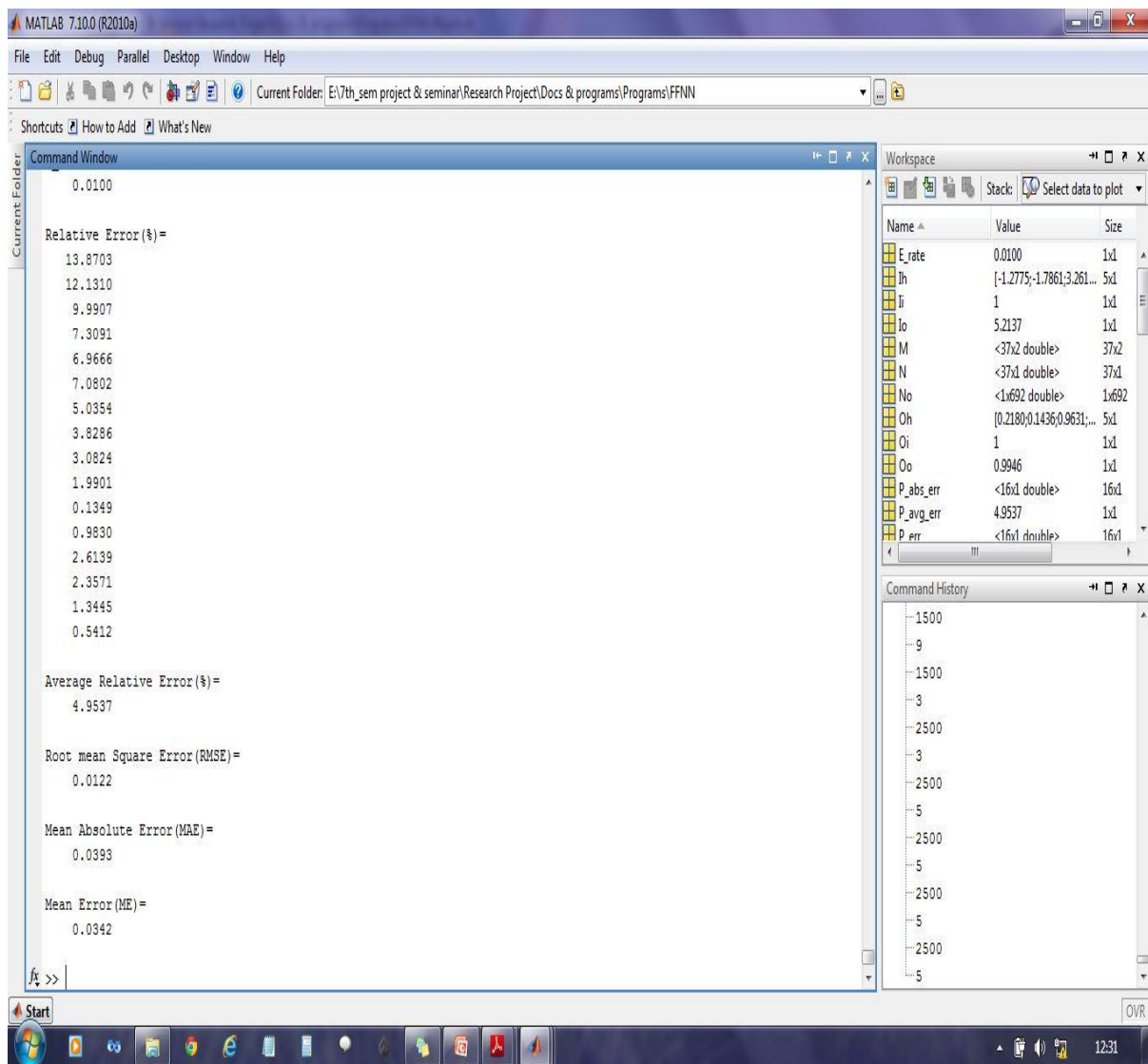


Figure 4.8: Snapshot of performance of FFNN for Dataset4

In the above figure, the snapshot of different performance measure values using FFNN for Dataset4 is shown.

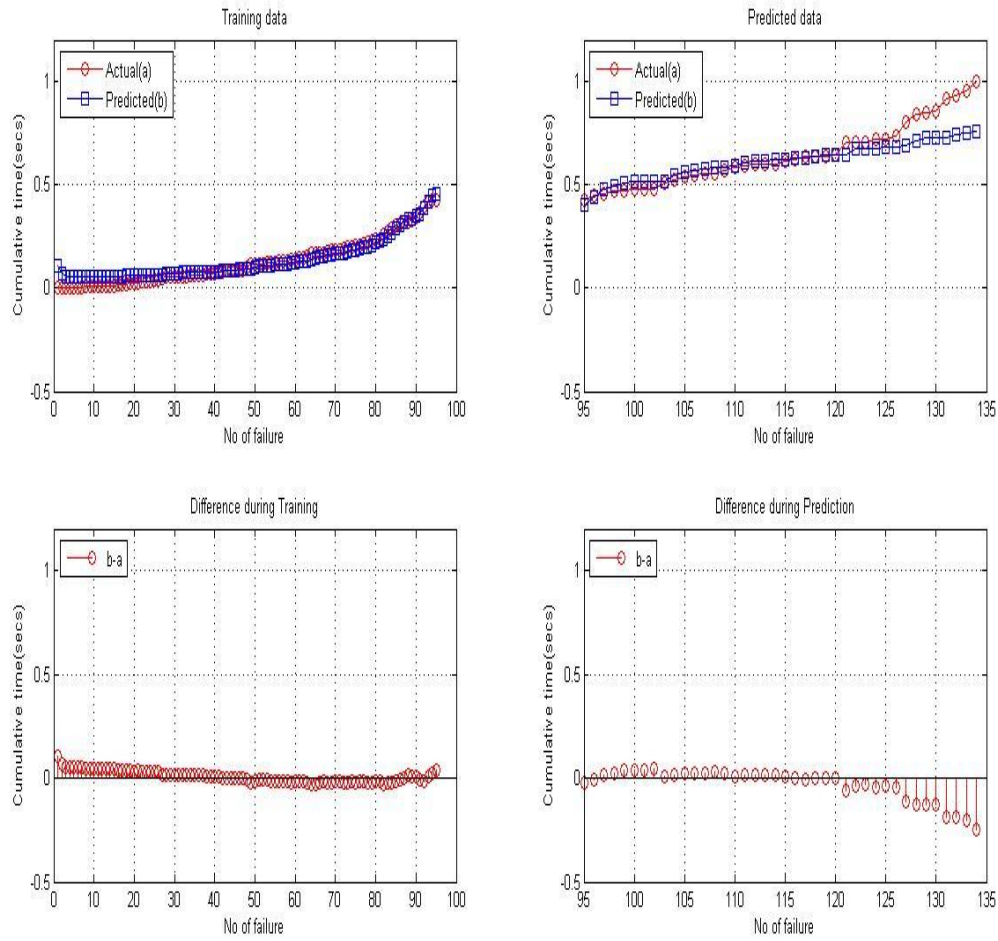


Figure 4.9: Performance of RNN for Dataset1

In the above figure, plots for recurrent neural networks are shown. The upper two graphs are the plot between no of cumulative failures vs. cumulative execution time for Dataset1. Upper left graph is plotted for the training period and upper right graph is plotted for the prediction period. In both upper graphs red colour line represents actual data; blue colour line represents predicted data. In the lower two graphs red colour line represents difference between predicted data and actual data.

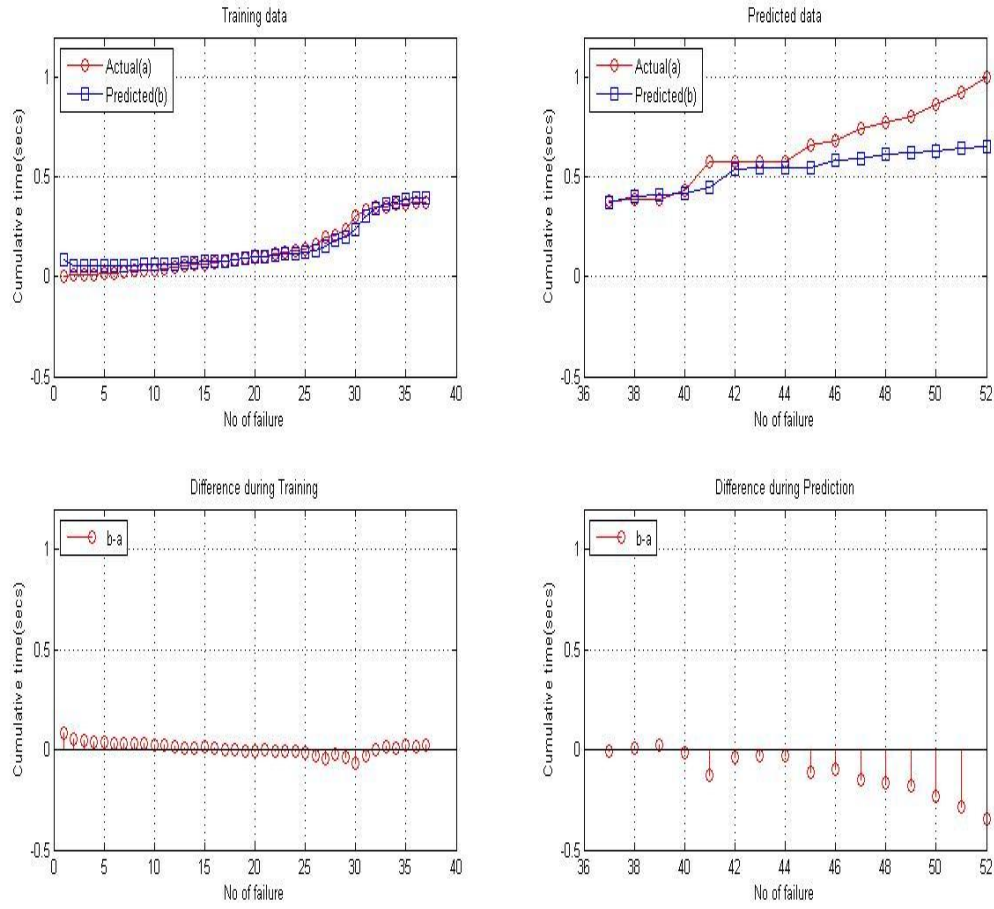


Figure 4.10: Performance of RNN for Dataset2

In the above figure, plots for recurrent neural networks are shown. The upper two graphs are the plot between no of cumulative failures vs. cumulative execution time for Dataset2. Upper left graph is plotted for the training period and upper right graph is plotted for the prediction period. In both upper graphs red colour line represents actual data; blue colour line represents predicted data. In the lower two graphs red colour line represents difference between predicted data and actual data.

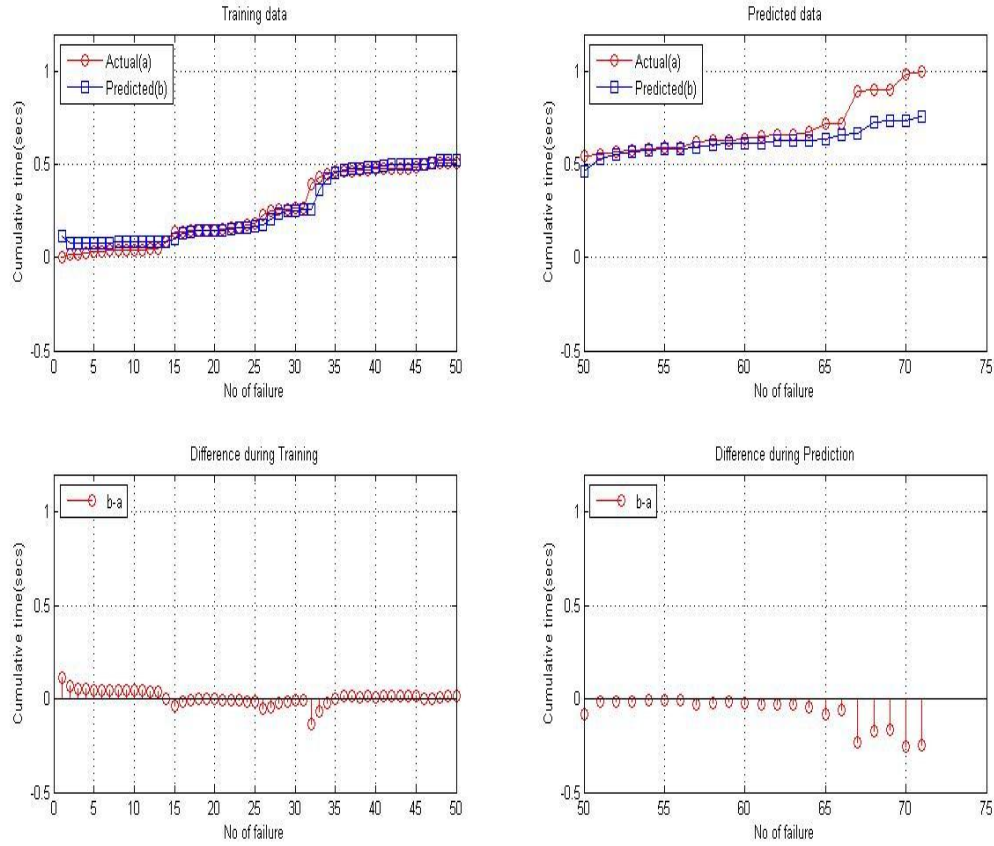


Figure 4.11: Performance of RNN for Dataset6

In the above figure, plots for recurrent neural networks are shown. The upper two graphs are the plot between no of cumulative failures vs. cumulative execution time for Dataset3. Upper left graph is plotted for the training period and upper right graph is plotted for the prediction period. In both upper graphs red colour line represents actual data; blue colour line represents predicted data. In the lower two graphs red colour line represents difference between predicted data and actual data.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

We have successfully implemented the feed forward and recurrent neural network with back propagation learning algorithm. The observations conclude that neural network model performs better in terms of less error in prediction as compared to existing analytical models and hence it is a better alternative to do software reliability test using neural network. However it can be seen from the figures that the NN method proposed in this paper using back propagation algorithm provides a good fit than analytical models. As the connection weights are randomly initialized, thus the neural network gives different results for the same datasets and thus the performance of the network varies. The usefulness of a Neural Network method is dependent on the nature of dataset up to a greater extent. In most cases STP gives better result than LTP. Neural Network model gives better result for larger datasets than smaller datasets. These models are easily compatible with different smooth trend data set and projects. We have implemented the program in MATLAB. But the programs can be implemented in other languages such as Java, Python etc.

5.2 Future Work

Software reliability can be predicted using hybrid intelligent system. In addition to neural network model genetic programming can be applied further. Novel recurrent architectures for Genetic Programming (GP) and Group Method of Data Handling (GMDH) to predict software reliability can be proposed.

Further, research can be extended by developing GP and GMDH based ensemble models to predict software reliability. In the ensemble models, GP and GMDH are considered as constituent models.

References

- [1] J. D. Musa, “Software Reliability Data,” Data & Analysis Centre for Software, January 1980.
- [2] R. Iyer and I. Lee, “Measurement-based analysis of software reliability,” Handbook of Software Reliability Engineering, McGraw-Hill, pp. 303 – 358, 1996.
- [3] J. D. Musa and K. Okumoto, “A Logarithmic Poisson Execution Time Model for Software Reliability Measurement,” in ICSE, IEEE Press Piscataway, NJ, USA: Proceedings of the 7th International Conference on software Engineering, pp. 230–238, 1984.
- [4] T. R. Benala, S. Dehuri, and R. Mall, “Computational Intelligence in Software Cost Estimation: An Emerging Paradigm,” ACM SIGSOFT Software Engineering, vol. 37, no. 3, pp. 1–7, May 2012.
- [5] IEEE, “Standard glossary of software engineering terminology,” Standards Coordinating Committee of the IEEE Computer Society, 1991.
- [6] Simon Haykin, “Neural Networks A Comprehensive Foundation”, Pearson Prentice Hall, 2nd Edition, 2001.
- [7] L. Tian and A. Noore, “Software Reliability Prediction Using Recurrent Neural Network with Bayesian Regularization,” International Journal of Neural Systems, vol. 14, no. 3, pp. 165–174, June 2004.
- [8] A. L. Goel, “Software reliability models: Assumptions, limitations, and applicability,” IEEE Transaction on Software Engineering, vol. 11, no. 12, pp. 1411–1423, December 1985.
- [9] P. Werbos, “Generalization of Back propagation with Application to Recurrent Gas Market Model,” Neural Network, vol. 1, pp. 339–356, 1988.
- [10] R. Shadmehr and D. Z. D'Sargenio, “A Comparison of a Neural Network Based Estimator and Two Statistical Estimators in a Sparse and Noisy Data Environment,” in IJCNN, vol. 1, Washington D.C, pp. 289–292, June 1990.

- [11] N. Karunanithi, Y. Malaiya, and D. Whitley, "Prediction of Software Reliability Using Neural Networks," in Proceedings IEEE International Symposium on Software Reliability Engineering. Austin, TX: IEEE, pp. 124–130, May 1991.
- [12] T. M. Khoshgoftaar, A. S. Pandya and H. More, "A Neural Network Approach For Predicting Software Development Faults." Research Triangle Park, NC: Proceedings of Third International Symposium on Software Reliability Engineering, pp. 83–89, October 1992.
- [13] Y. Singh and P. Kumar, "Prediction of Software Reliability Using Feed Forward Neural Networks," in Computational Intelligence and Software Engineering (CiSE), I. Conference, Ed. IEEE, pp. 1–5, 2010.
- [14] M. M. T. Thwin and T. S. Quah, Eds., Application of Neural Network for Predicting Software Development Faults using Object-Oriented Design Metrics, vol. 5. Proceedings of the 9th International Conference on Neural Information Processing (ICONIP'02), November 2002.
- [15] N. Karunanithi and D. Whitley, "Prediction of Software Reliability Using Feed forward and Recurrent Neural Nets," in Neural Networks, 1992. IJCNN, vol. 1. Baltimore, MD: IEEE, pp. 800–805, June 1992.
- [16] R. Sitte, "Comparison of software-reliability-growth predictions: neural networks vs. parametric-recalibration," Reliability, IEEE Transactions, vol. 48, no. 3, pp. 285–291, September 1999.
- [17] N. RajKiran and V. Ravi, "Software Reliability Prediction using Wavelet Neural Networks," in International Conference on Computational Intelligence and Multimedia Applications, vol. 1. Sivakasi, Tamil Nadu: IEEE, pp. 195 – 199, December 2007.
- [18] J. H. Lo, "The Implementation of Artificial Neural Networks Applying to Software Reliability Modeling," Control and Decision Conference, 2009. CCDC '09, Chinese, pp. 4349 – 4354, June 2009.
- [19] L. Zhao, J. pei Zhang, J. Yang, and Y. Chu, "Software reliability growth model based on fuzzy wavelet neural network," in 2nd International Conference on Future Computer and Communication (ICFCC), vol. 1. Wuhan: IEEE, pp. 664–668, May 2010.

- [20] R. Mohanty, V. Ravi, and M. R. Patra, “Hybrid Intelligent Systems for Predicting Software Reliability,” *Applied Soft Computing*, vol. 13, no. 1, pp. 189–200, August 2013.

Appendix A

Datasets

Cyber Security and Information Systems In-formation Analysis Centre(CSIAC) Software project failure datasets:

Dataset1:

Failure Number	Failure Interval Length(in cpu secs)	Day of Failure
1	3	1
2	30	2
3	113	9
4	81	10
5	115	11
6	9	11
7	2	17
8	91	20
9	112	20
10	15	20
11	138	20
12	50	20
13	77	20
14	24	20
15	108	20
16	88	20
17	670	30
18	120	30
19	26	30
20	114	30
21	325	30
22	55	30
23	242	31
24	68	31
25	422	31
26	180	32
27	10	32
28	1146	33

29	600	34
30	15	42
31	36	42
32	4	46
33	0	46
34	8	46
35	227	46
36	65	46
37	176	46
38	58	46
39	457	47
40	300	47
41	97	47
42	263	47
43	452	53
44	255	53
45	197	54
46	193	54
47	6	54
48	79	54
49	816	56
50	1351	56
51	148	56
52	21	57
53	233	57
54	134	57
55	357	57
56	193	59
57	236	59
58	31	59
59	369	59
60	748	59
61	0	59
62	232	59
63	330	59
64	365	61
65	1222	62
66	543	63
67	10	63

68	16	63
69	529	64
70	379	64
71	44	64
72	129	64
73	810	64
74	290	64
75	300	64
76	529	65
77	281	65
78	160	65
79	828	66
80	1011	66
81	445	66
82	296	66
83	1755	67
84	1064	67
85	1783	68
86	860	68
87	983	68
88	707	69
89	33	69
90	868	69
91	724	69
92	2323	70
93	2930	71
94	1461	72
95	843	72
96	12	72
97	261	72
98	1800	73
99	865	73
100	1435	74
101	30	74
102	143	74
103	108	74
104	0	74
105	3110	75
106	1247	76

107	943	76
108	700	76
109	875	77
110	245	77
111	729	77
112	1897	78
113	447	79
114	386	79
115	446	79
116	122	79
117	990	79
118	948	80
119	1082	80
120	22	80
121	75	80
122	482	80
123	5509	81
124	100	81
125	10	81
126	1071	82
127	371	83
128	790	83
129	6150	83
130	3321	83
131	1045	84
132	648	84
133	5485	87
134	1160	87
135	1864	88
136	4116	92

Dataset2:

Failure Number	Failure Interval Length(in cpu secs)	Day of Failure
1	191	1
2	222	2
3	280	11
4	290	11

5	290	14
6	385	23
7	570	23
8	610	23
9	365	23
10	390	23
11	275	23
12	360	27
13	800	27
14	1210	28
15	407	29
16	50	29
17	660	29
18	1507	31
19	625	31
20	912	32
21	638	32
22	293	32
23	1212	33
24	612	33
25	675	33
26	1215	33
27	2715	37
28	3551	37
29	800	38
30	3910	38
31	6900	38
32	3300	38
33	1510	41
34	195	42
35	1956	42
36	135	43
37	661	43
38	50	43
39	729	43
40	900	46
41	180	46
42	4225	46
43	15600	53

44	0	53
45	0	53
46	300	53
47	9021	57
48	2519	64
49	6890	64
50	3348	67
51	2750	69
52	6675	71
53	6945	71
54	7899	72

Dataset3:

Failure Number	Failure Interval Length(in cpu secs)	Day of Failure
1	115	1
2	0	1
3	83	3
4	178	3
5	194	3
6	136	3
7	1077	3
8	15	3
9	15	3
10	92	3
11	50	3
12	71	3
13	606	6
14	1189	8
15	40	8
16	788	18
17	222	18
18	72	18
19	615	18
20	589	26
21	15	26
22	390	26
23	1863	27

24	1337	30
25	4508	36
26	834	38
27	3400	40
28	6	40
29	4561	42
30	3186	44
31	10571	47
32	563	47
33	2770	47
34	652	48
35	5593	50
36	11696	54
37	6724	54
38	2546	55

Dataset4:

Failure Number	Failure Interval Length(in cpu secs)	Day of Failure
1	5	1
2	73	1
3	141	1
4	491	5
5	5	5
6	5	5
7	28	5
8	138	5
9	478	9
10	325	9
11	147	10
12	198	10
13	22	10
14	56	10
15	424	20
16	92	20
17	520	20
18	1424	26
19	0	26

20	92	26
21	183	26
22	10	26
23	115	27
24	17	27
25	284	27
26	296	27
27	215	27
28	116	27
29	283	31
30	50	31
31	308	31
32	279	31
33	140	32
34	678	32
35	183	32
36	2462	41
37	104	41
38	2178	42
39	285	43
40	171	44
41	0	44
42	643	46
43	887	46
44	149	48
45	469	48
46	716	48
47	604	48
48	0	48
49	774	50
50	256	50
51	14637	58
52	18740	70
53	1526	71

Dataset6:

Failure Number	Failure Interval Length(in cpu secs)	Day of Failure
1	3	1
2	14	1
3	59	1
4	32	2
5	8	2
6	52	2
7	2	2
8	25	2
9	2	2
10	3	5
11	4	6
12	1	6
13	30	6
14	21	7
15	196	12
16	265	12
17	6	12
18	3	12
19	8	12
20	1	12
21	12	12
22	36	13
23	38	13
24	1	13
25	74	14
26	43	14
27	236	14
28	121	15
29	18	16
30	9	16
31	23	16
32	1	16
33	672	24
34	189	24
35	83	26
36	52	26

37	8	26
38	1	26
39	41	27
40	7	27
41	43	28
42	1	28
43	4	28
44	5	28
45	1	28
46	16	28
47	70	29
48	60	30
49	2	30
50	2	30
51	3	30
52	169	31
53	29	32
54	88	33
55	55	35
56	27	35
57	24	35
58	27	35
59	140	37
60	33	37
61	5	37
62	36	37
63	74	38
64	40	39
65	2	39
66	86	40
67	221	42
68	6	42
69	891	52
70	23	53
71	4	53
72	437	58
73	66	58