Empirical Testing Of The Neural Network Application Using Feedforward Testing Method

Manoj Kumar Panda
Prof & Head Nuva College of Engineering & Technology ,Nagpur

Abstract

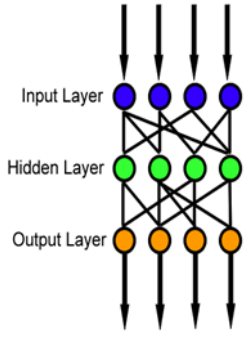
In the sofware testing methodology there is a revolution and in the field of testing that how to conduct a successful testing and the testing shall be done in such a way that error free product to be launched in the market and the contemporary software services and product organizations should follow a different kind of software testing methodologies . we propose the feedforward testing mothod which can be used extensively in the field the field of neural network softwares in the proposed system first we should understan the feed forward neural network here each nuron has capacity to give a positive and negative signals to to the middle layer and in the middle layer the actual processing work must happen and all the posibility of output must be stored in a vector and all the possibility of vectors the strongest output including the threshold must be taken into account and the out put will come out whether positive or negative .hence the testing must consider all the possible patterns

Keywords

Feedforward testing method , lines of code (LOC), function points (FP),Err , WBS , FR,NFR

INTRODUCTION

In recent years, software has become the most expensive component of computer and various robots and other neural network systems projects. The bulk of the cost of software development is due to the human effort, and most cost estimation methods focus on this aspect and give estimates in terms of person-months. Accurate software cost estimates are critical to both developers and customers. They can be used for generating request for proposals, contract negotiations, scheduling, monitoring and control. Underestimating the costs may result in management approving proposed systems that then exceed their budgets, with underdeveloped functions and poor quality, and failure to complete on time. Overestimating may result in too many resources committed to the project, or, during contract bidding, result in not winning the contract, which can lead to loss of jobs. Accurate cost estimation is important because: in the above
It can help to classify and prioritize development projects with respect to an overall business plan.



(Fig. 1)A Feedforward Testing Method

A multiple-layer neural network can calculate and compute a continuous in put output instead of a function. because the network itself is in the continuous process ,hence the work should not stop and the testing also should be continuous in nature A common choice is the so-called logistic function:

$$y = \frac{1}{1 + e^{-x}}$$ and another equation also

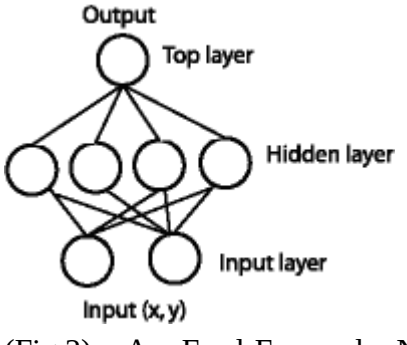
$$net_i = \sum_{j=1}^{n} wijxj, for\ i + 1,2, \ldots n$$

In the above equation each nuron has to consider the weight factor and in the testing also sall the weights are represented and output came out successfully without any errors

(In general form, f(X) is in place of x, where f(X) is an analytic function in set of x's.) With this choice, the single-layer network is identical to the logistic regression model, widely used in statistical modeling. The logistic function is also known as the sigmoid function. It has a continuous derivative, which allows it to be used in backpropagation. This function is also preferred because its derivative is easily calculated:and the output is more acceptable

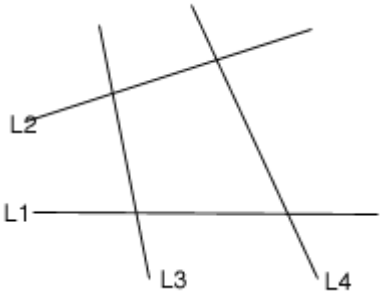
$$y' = y(1 - y)$$ and further the output is $o_i = f(w_i^t x, for\ i\ 1,2 \ldots m$

Recall that a single perceptron can classify points into two regions that are linearly separable. Now let us extend the discussion into the separation of points into two regions that are not linearly separable. Consider the following network:



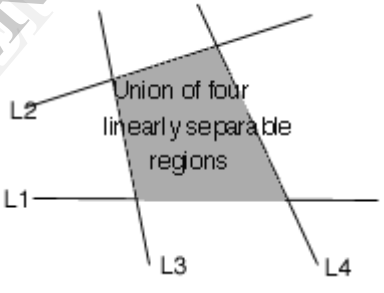
(Fig.2) A Feed-Forward Network Testing Method With One Hidden Layer.

The same (x, y) is fed into the network through the perceptrons in the input layer. With four perceptrons that are independent of each other in the hidden layer, the point is classified into 4 pairs of linearly separable regions, each of which has a unique line separating the region.



(Fig.3) 4 lines each dividing the plane into 2 linearly separable regions.

The top perceptron performs the various logical operations on the outputs of the hidden layers so that the whole network classifies input points in 2 regions that might not be linearly separable. For instance, using the AND operator on these four outputs, one gets the intersection of the 4 regions that forms the center region.



(Fig.4) Intersection of 4 linearly separable regions forms the center region.

By varying the number of nodes in the hidden layer, the number of layers, and the number of input and output nodes, one can classification of points in arbitrary dimension into an arbitrary number of groups. Hence feed-forward networks are commonly used for classification.

Backpropagation -- learning in feed-forward networks:

Learning in feed-forward networks belongs to the realm of supervised learning, in which pairs

of input and output values are fed into the network for many cycles, so that the network 'learns' the relationship between the input and output.

We provide the network with a number of training samples, which consists of an input vector i and its desired output o. For instance, in the classification problem, suppose we have points (1, 2) and (1, 3) belonging to group 0, points (2, 3) and (3, 4) belonging to group 1, (5, 6) and (6, 7) belonging to group 2, then for a feed-forward network with 2 input nodes and 2 output nodes, the training set would be:

{ i = (1, 2) , o =(0, 0)
i = (1, 3) , o = (0, 0)
i = (2, 3) , o = (1, 0)
i = (3, 4) , o = (1, 0)
i = (5, 6) , o = (0, 1)
i = (6, 7) , o = (0, 1) }

The basic rule for choosing the number of output nodes depends on the number of different regions. It is advisable to use a unary notation to represent the different regions, i.e. for each output only one node can have value 1. Hence the number of output nodes = number of different regions -1.

In backpropagation learning, and finding errors every time an input vector of a training sample is presented, the output vector o is compared to the desired value d.

The comparison is done by calculating the squared difference of the two i.e. the desired output and the actual output :

$$Err = (d-o)^2$$

The value of Err tells us how far away we are from the desired value for a particular input. The goal of backpropagation is to minimize the sum of Err for all the training samples, so that the network behaves in the most "desirable" way.

Minimize $\sum Err = (d-o)^2$

We can express Err in terms of the input vector (i), the weight vectors (w), and the threshold function of the perceptions. Using a continuous function (instead of the step function) as the threshold function, we can express the gradient of Err with respect to the w in terms of w and i.

Given the fact that decreasing the value of w in the direction of the gradient leads to the most rapid decrease in Err, we update the weight vectors every time a sample is presented using the following formula:

$$w_{new} = w_{old} - n \frac{\delta Err}{\delta w}$$ where n is the learning rate (a small number ~ 0.1)

Using this algorithm, the weight vectors are modified so that the value of Err for a particular input sample decreases a little bit every time the sample is presented. When all the samples are presented in turns for many cycles, the sum of Err gradually decreases to a minimum value, which is our goal as mentioned above.

The testing methods of feed forward software's

In the given table we have shown the input

values will be passed through the neurons and successfully passed through the stages it must be remembered that the stages can be multiple one as the neural network can accommodate multiple neuron

Input	Stage 1 hidden layer	Stage 2 hidden layer	Output
1,1	1,0	0,0	1
0,1	1,0	0,1	0
1,0	0,1	1,0	1
0,0	0,0	0,1	0
1,1	1,1	0,1	1
1,0	1,1	1,1	0
1,1	1,1	1,0	0

From the above table we have taken all the logical parameters which are helpful in the testing of feedforward method
Testing – find out whether the application is working according to requirements or not
Testing objectives – to find out the differences between expected values and actual values
To find out the syntax errors which are unidentified by development team .whether our application is maintaining company standards or not
.

Testing Principles for neural network application

1 Testing should be conducted according to requirements
2 Testing should be started on smaller modules then we have to turn on to large ones
3 Testing should be started much before so that the should be accomplished much before

EXHAUSTIVE TESTING INPUT (ETI)

Testing the application with all possible combination of values
Examples –find the values of
A, b, c in $ax2 + bx + c$ with respect to $x = z$
Where $x = 2$ and a, b, c are 8bit registers

The Cost Factor Of The Feed Forward Software Testing Method

Estimates:

effort (usually in person-months)□project duration (in calendar time)

cost (in Rupees)

Most cost estimation models attempt to generate an effort estimate, which can then be converted into the project duration and cost. Although effort and cost are closely related, they are not necessarily related by a simple transformation function. Effort is often measured in person months of the programmers, analysts and project managers. This effort estimate can be converted into a dollar cost figure by calculating an average salary per unit time of the staff involved, and then multiplying this by the estimated effort required. Practitioners have struggled with three fundamental issues:

Software cost estimation model to use

Software size measurement to use – lines of code (LOC), function points (FP), or feature point.

A Good Estimate for the Feed Forward Software Testing Method

The widely practiced cost estimation for the feed forward testing method is expert judgment. For many years, project managers have relied on experience and the prevailing industry norms as a basis to develop cost estimate. However, basing estimates on expert judgment is problematic:

This approach is not repeatable and the means of deriving an estimate are not explicit. □It is difficult to find highly experienced estimators for every new project. The relationship between cost and system size is not linear. Cost tends to increase exponentially with size. The expert

judgment method is appropriate only when the sizes of the current project and past projects are similar. Budget manipulations by management aimed at avoiding overrun make experience and data from previous projects questionable.

In the last three decades, many quantitative software cost estimation models have been developed. An *empirical model* uses data from previous projects to evaluate the current project and derives the basic formulae from analysis of the particular database available. An *analytical model*, on the other hand, uses formulae based on global assumptions, such as the rate at which developer solve problems and the number of problems available. Most cost models are based on the size measure, such as LOC used in the software project and FP the various function points , obtained from size estimation. The accuracy of size estimation directly impacts the accuracy of cost estimation which is necessary for the evaluating the feed forward soft ware testing project .

Although common size measurements have their own drawbacks, an organization can make good use of any one, as long as a consistent counting method is used. A good software cost estimate should have the following attributes. It is conceived and supported by the project manager and the development team. It is accepted by all stakeholders as realizable.

It is based on a well-defined software cost model with a credible basis.

It is based on a database of relevant project experience (similar processes, similar technologies, similar environments, similar people and similar requirements). It is defined in enough detail so that its key risk areas are understood and the probability of success is objectively assessed. Software cost estimation historically has been a major difficulty in software development. Several reasons for the difficulty have been identified: Lack of a historical database of cost measurement Software development involving many interrelated factors, which affect development effort and productivity, and whose relationships are not well understood Lack of trained estimators and estimators with the necessary expertise Little penalty is often associated with a poor estimate.

1.2. Process Of Estimation for the Feed Forward Software Testing Method

Estimation is an important part of the planning process. For example, in the top-down planning approach, the cost estimate is used to derive the project plan:

1.2.1.The project manager develops a characterization of the overall functionality, size, process, environment, people, and quality required for the project.

1.2.2 A macro-level estimate of the total effort and schedule is developed using a software cost estimation model.

1.2.3 The project manager partitions the effort estimate into a top-level work breakdown structure. He also partitions the schedule into major milestone dates and determines a staffing profile, which together forms a project plan.

1.2.4The actual cost estimation process involves seven steps:

1.2.5 Establish cost-estimating objectives

1.2.6 Generate a project plan for required data and resources

1.2.7. Pin down software requirements

1.2.8. Work out as much detail about the software system as feasible

1.2.9. Use several independent cost estimation techniques to capitalize on their combined strengths

1.2.10. Compare different estimates and iterate the estimation process

1.2.11. After the project has started, monitor its actual cost and progress, and feedback results to project management.

No matter which estimation model is selected, users must pay attention to the following to get best results: coverage of the estimate (some

models generate effort for the full life-cycle, while others do not include effort for the requirement stage)□calibration and assumptions of the model □sensitivity of the estimates to the different model parameters □deviation of the estimate with respect to the actual cost.

2 The Outputs Of This Steps Are As Follows:

2.1 Assumptions made to revise estimates
2.2□ Methods used to revise estimates
2.3 □Revised size, effort, schedule, and cost estimates
2.4 □Revised functionality and procurements
2.5□ Updated WBS
2.6 □Revised risk assessment
Review and Approve the Estimates
The purpose of this step is to review the software estimates and to obtain project and line management approval.
Software Requirement Specification
After gathering all requirements then the management team starts the development process from SRS. In SRS they specifies all requirements for developing the current application
➔FR (Functional requirements) for developing the current application all the technical requirements with according to hardware and
➔NFR (NON FUNCTIONAL)requirements in this management team specifies all requirements other than functional as follows

➔ Cost –total cost incurred for the testing activities

➔ Time – time duration for coducting all the testing activities for the feedforward testing method

➔Standards to maintain-Every company maintain its own standards. The standards of every company are depends on the market value in current trend .some of the common standards which are maintained by every company are

➔ N no of defects per SG Σ LOC.

➔ Number of functions per module.

➔ Number of functional points.

Design- after completion of preparation of SRS development team divide the functionality of the application in to modules and sub modules by preparing two documents.

➔ HLD (high level design)-Dividing the functionality of the applications into modules and sub modules and defining data validations is called as HLD.

CONCLUSION-: from the above discussion we came to that the feedforward testing method is unique and using this method we can test various neural network methods , in the proposed system first we should understan the feed forward neural network here each nuron has capacity to give a positive and negative signals to to the middle layer and in the middle layer the actual processing work must happen and all the posibility of output must be stored in a vector and all the possibility of vectors the strongest output including the threshold must be taken into account and the out put will come out whether positive or negative .hence the testing must consider all the possible patterns

REFERENCES

1. ^ Rául Rojas (1996). *Neural networks: a systematic introduction*. Springer. p. 336. ISBN 978-3-540-60505-8.
2. ^ [a] [b] Cruse, Holk; *Neural Networks as Cybernetic Systems*, 2nd and revised edition
3. ^ H. Jaeger. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. Science, 304:78–80, 2004.
4. ^ W. Maass, T. Natschläger, and H. Markram. A fresh look at real-time computation in generic recurrent neural circuits. Technical report, Institute for Theoretical Computer Science, TU Graz, 2002.
5. ^ [a] [b] Hochreiter, Sepp; and Schmidhuber, Jürgen; *Long Short-Term Memory*, Neural Computation, 9(8):1735–1780, 1997
6. ^ Gers, Felix A.; and Schmidhuber, Jürgen; *LSTM Recurrent Networks Learn Simple Context Free and Context Sensitive Languages*, IEEE Transactions on Neural Networks, 12(6):1333–1340, 2001

7. ^ Graves, Alex; and Schmidhuber, Jürgen; *Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks*, in Bengio, Yoshua; Schuurmans, Dale; Lafferty, John; Williams, Chris K. I.; and Culotta, Aron (eds.), *Advances in Neural Information Processing Systems 22 (NIPS'22), December 7th–10th, 2009, Vancouver, BC*, Neural Information Processing Systems (NIPS) Foundation, 2009, pp. 545–552
8. ^ Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing, 45:2673–81, November 1997.
9. ^ A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Networks, 18:602–610, 2005.

A. J. Albrecht, and J. E. Gaffney, "Software function, source lines of codes, and development

10. effort prediction: a software science validation", *IEEE Trans Software Eng.* SE-9, 1983,
11. pp.639-648.
12. 2. U. S. Army, *Working Schedule Handbook, Pamphlet No.* 5-4-6, Jan 1974.
13. 3. J. D. Aron, *Estimating Resource for Large Programming Systems*, NATO Science Committee,
14. Rome, Italy, October 1969.
15. 4. R.K.D. Black, R. P. Curnow, R. Katz and M. D. Gray, *BCS Software Production Data*, Final
16. Technical Report, RADC-TR-77-116, Boeing Computer Services, Inc., March 1977.
17. 5. B. W. Boehm, *Software engineering economics*, Englewood Cliffs, NJ: Prentice-Hall, 1981.

18. 6. B.W. Boehm et al "The COCOMO 2.0 Software Cost Estimation Model", *American*

19. *Programmer*, July 1996, pp.2-17.
20. 7. L. C. Briand, K. El Eman, F. Bomarius, "COBRA: A hybrid method for software cost

21. estimation, benchmarking, and risk assessment", *International conference on software*

22. *engineering*, 1998, pp. 390-399.
23. 8. G. Cantone, A. Cimitile and U. De Carlini, "A comparison of models for software cost

24. estimation and management of software projects", in *Computer Systems*: *Performance and*

25. *Simulation*, Elisevier Science Publishers B.V., 1986.
26. Measuring, Monitoring & Testing the Quality of the Software Using Exhaustive Testing Input (ETI) & Software Test Resposibility Matrix (STRM)**,IJETAE,2013** Manoj Kumar Panda.
27. 'Pragmatic Peer Review Project Contextual Software Cost Estimation 泡 novel approach ' bearing paper id 'IJCSI-2011-8-6-982',Manoj Kumar Panda

About the author:-

Manoj kumar panda , The author is a prominent academic ,and well known researcher & the author worked in numerous research projects in the field of Computer Science & Engineering & Computer Application and a Professor And Head In Computer Engineering And Computer Application Dept. and the area of interest of author is in Software Engineering ,Software Testing Methodology ,Human Computer Interaction , Software Project Management ,Neural Network And Artificial Inteligence .