INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN: 2278 – 0211 (Online)

Testing The Artificial Inteligence Application Using Feedback Testing Method -: An Empirical View

Manoj kumar panda

Researcher & the Computer Science & Engineering & Computer Application

Professor And Head In Computer Engineering And Computer Application Dept.

Software Engineering ,Software Testing Methodology ,Human Computer Interaction, Software Project Management ,Neural Network And Artificial Inteligence

Abstract:

In the software engineering there is the renaissance and the software testing methodology is also following the same path as the software engineering , and the testing is also following the same footsteps but many of the software testing strategists testing the conventional software but our main and primary focus is on artificial intelligence software testing and one of the method in this system we propose the testing artificial intelligence application using feedback testing method .

Keywords: *feedback testing method (FTM),Weights of neuron(won)*

1.Introduction

In the aforesaid method the testing shall be done on the basis of information provided in input neurons .And the information pass through the hidden neurons in various stages and in between it would add the weights 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 .

Input Neuron	Stage 1 (Hidden Layer)	Stage 2 (Hidden Layer)	Output Neurons
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

Table 1

This feedback testing method requires one or more completed projects that are similar to the new project and derives the estimation through reasoning by analogy using the actual costs of previous projects. Estimation by analogy can be done either at the total project level or at subsystem level. The total project level has the advantage that all cost components of the system will be considered while the subsystem level has the advantage of providing a more detailed assessment of the similarities and differences between the new project and the completed projects. The strength of this method is that the estimate is based on actual project experience. However, it is not clear to what extend the previous project is actually representative of the constraints, environment and functions to be performed by the new system. Positive results and a definition of project similarity in term of features were reported in. Expert judgment: This method involves consulting one or more experts. The experts provide estimates

using their own methods and experience. Expert-consensus mechanisms such as PERT will be used to resolve the inconsistencies in the estimates. The Delphi technique works as follows:

The coordinator presents each expert with a specification and a form to record estimates.
Each expert fills in the form individually (without discussing with others) and is allowed to ask the coordinator questions.
The coordinator prepares a summary of all estimates from the experts (including mean or median) on a form requesting another iteration of the experts' estimates and the rationale for the estimates.
Repeat steps 2-3 as many rounds as appropriate.
Before the estimation, a group meeting involving the coordinator and experts is arranged to discuss the estimation issues. In step 3), the experts do not need to give any rationale 6 for the estimates. Instead, after each round of estimation, the coordinator calls a meeting to have experts discussing those points where their estimates varied widely. Parkinson: Using Parkinson's principle "work expands to fill the available volume", the cost is determined (not estimated) by the available resources rather than based on an objective assessment. If the software has to be delivered in 12 months and 5 people are available, the effort is estimated to be 60 person-months. Although it sometimes gives good estimation, this method is not recommended as it may provide very unrealistic estimates. Also, this method does not

2.Promote Good Software Engineering Practice

Price-to-win: The software cost is estimated to be the best price to win the project. The estimation is based on the customer's budget instead of the software functionality. For example, if a reasonable estimation for a project costs 100 person-months but the customer can only afford 60 sperson-months, it is common that the estimator is asked to modify the estimation to fit 60personmonths' effort in order to win the project. This is again not a good practice since it is very likely to cause a bad delay of delivery or force the development team to work overtime. Bottom-up: In this approach, each component of the software system is separately estimated and the results aggregated to produce an estimate for the overall system. The requirement for this approach is that an initial design must be in place that indicates how the system is decomposed into different components. Top-down: This approach is the opposite of the bottom-up method. An overall cost estimate for the system is derived from global properties, using either algorithmic or non-algorithmic methods. The total cost can then be split up among the various components. This approach is more suitable for cost estimation at the early stage. Algorithmic methods The algorithmic methods are based on mathematical models that

produce cost estimate as a function of a number of variables, which are considered to be the major cost factors. Any algorithmic model has the form:

Effort = f(x1, x2, …, xn)

where {x1, x2, …, xn} denote the cost factors. The existing algorithmic methods differ in two aspects: the selection of cost factors, and the form of the function f.

We will first discuss the cost factors used in these models, then characterize the models according to the form of the functions and whether the models are analytical or empirical.

3.Cost Factors Based On Technicalities

Besides the software size, there are many other cost factors.

These cost factors can be divided into four types: Product factors: required reliability; product complexity; database size used; required reusability; documentation match to life-cycle needs; Computer factors:

execution time constraint; main storage constraint; computer turnaround constraints; platform volatility; Personnel factors: analyst capability; application experience; programming capability; platform experience; language and tool experience; personnel continuity; Project factors: multisite development; use of software tool; required development schedule. The above factors are not necessarily independent, and most of them are hard to quantify. In 7 many models, some of the factors appear in combined form and some are simply ignored. Also, some factors take discrete values, resulting in an estimation function with a piece-wise form.Feedback tesing method requires one or more completed projects that are similar to the new project and derives the estimation through reasoning by analogy using the actual costs of previous projects. Estimation by analogy can be done either at the total project level or at subsystem level. The total project level has the advantage that all cost components of the system will be considered while the subsystem level has the advantage of providing a more detailed assessment of the similarities and differences between the new project and the completed projects. The strength of this method is that the estimate is based on actual project experience. However, it is not clear to what extend the previous project is actually representative of the constraints, environment and functions to be performed by the new system. Positive results and a definition of project similarity in term of features were reported in. Expert judgment: This method involves consulting one or more experts. The experts provide estimates using their own methods and experience. Expert-consensus mechanisms such

as Delphi technique or PERT will be used to resolve the inconsistencies in the estimates. The Delphi technique works as follows:

The coordinator presents each expert with a specification and a form to record estimates.

Each expert fills in the form individually (without discussing with others) and

is allowed to ask the coordinator questions.

The coordinator prepares a summary of all estimates from the experts

(including mean or median) on a form requesting another iteration of the experts' estimates and the rationale for the estimates.

Repeat steps 2-3 as many rounds as appropriate.A modification of the Delphi technique proposed by Boehm and Fahquhar [5]seems to be more effective: Before the estimation, a group meeting involving the coordinator and experts is arranged to discuss the estimation issues. In step3), the experts do not need to give any rationale 6 for the estimates. Instead, after each round of estimation, the coordinator calls a meeting to have experts discussing those points where their estimates varied widely.Parkinson: Using Parkinson's principle "work expands to fill the available volume", the cost is determined (not estimated) by the available resources rather than based on an objective assessment. If the software has to be delivered in 12 months and 5 people are available, the effort is estimated to be 60 person-months. Although it sometimes gives good estimation, this method is not recommended as it may provide very unrealistic estimates. Also, this method does not

4.Promote Good Software Engineering Practice

for Feedback testing method hence the out come or the result must be accurate

4.1.Price-To-Win

The software cost is estimated to be the best price to win theproject. The estimation is based on the customer's budget instead of thesoftware functionality. For example, if a reasonable estimation for a project costs100 person-months but the customer can only afford 60 sperson-months, it iscommon that the estimator is asked to modify the estimation to fit 60personmonths' effort in order to win the project. This is again not a good practice since it is very likely to cause a bad delay of delivery or force the development team to work overtime.

4.2.Bottom-Up

In this approach, each component of the software system is separately estimated and the results aggregated to produce an estimate for the overall system. The requirement for this approach is that an initial design must be in place that indicates how the system is decomposed into different components.

4.3.Top-Down

This approach is the opposite of the bottom-up method. An overall cost estimate for the system is derived from global properties, using either algorithmic or non-algorithmic methods. The total cost can then be split up among the various components. This approach is more suitable for cost estimation at the early stage.

5.Algorithmic Methods

The algorithmic methods are based on mathematical models that produce cost estimate as a function of a number of variables, which are considered to be the major cost factors. Any algorithmic model has the form:

Effort = f(x1, x2, …, xn)

where {x1, x2, …, xn} denote the cost factors. The existing algorithmic methods differ in two aspects: the selection of cost factors, and the form of the function f.

We will first discuss the cost factors used in these models, then characterize the models according to the form of the functions and whether the models are analytical or empirical.

6.Function Point Cost Factors

The most comprehensive set of cost factors are proposed and used

These cost factors can be divided into four types: Productfactors: required reliability; product complexity; database size used; requiredreusability; documentation match to life-cycle needs; Computer factors:execution time constraint; main storage constraint; computer turnaroundconstraints; platform volatility; Personnel factors: analyst capability;applicationexperience; programming capability; platform experience; language and tool experience; personnel continuity; Project factors: multisite development; use of software tool; required development schedule.The above factors are not necessarily independent, and most of them are hard to quantify. In 7 many models,

some of the factors appear in combined form and some are simply ignored. Also, some factors take discrete values, resulting in an estimation function with a piece-wise form.

7.Conclusion

From the above discussion we came to that the feedback testing method is unique and using this method we can test various neural network & artificicial intelligence softwares , in the proposed system first we should understan the feedback 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

8.Reference

1. J. Albrecht, and J. E. Gaffney, "Software function, source lines of codes, and development
2. effort prediction: a software science validation", IEEE Trans Software Eng. SE-9, 1983,
   pp.639-648.
3. U. S. Army, Working Schedule Handbook, Pamphlet No. 5-4-6, Jan 1974.
4. J. D. Aron, Estimating Resource for Large Programming Systems, NATO Science Committee,Rome, Italy, October 1969.
5. R.K.D. Black, R. P. Curnow, R. Katz and M. D. Gray, BCS Software Production Data, FinalTechnical Report, RADC-TR-77-116, Boeing Computer Services, Inc., March 1977.B. W. Boehm, Software engineering economics, Englewood Cliffs, NJ: Prentice-Hall, 1981.B.W. Boehm et al "The COCOMO 2.0 Software Cost Estimation Model", AmericanProgrammer, July 1996, pp.2-17.L. C. Briand, K. El Eman, F. Bomarius, “COBRA: A hybrid method for software cost
6. estimation, benchmarking, and risk assessment”, International conference on software
7. engineering, 1998, pp. 390-399.
8. G. Cantone, A. Cimitile and U. De Carlini, “A comparison of models for software cost
9. estimation and management of software projects”, in Computer Systems: Performance and
10. Simulation, Elisevier Science Publishers B.V., 1986.
11. Measuring, Monitoring & Testing the Quality of the Software Using Exhaustive Testing Input (ETI) & Software Test Resposibility Matrix (STRM),IJETAE,2013 Manoj Kumar Panda.
12. 'Pragmatic Peer Review Project Contextual Software Cost Estimation 泡 novel approach ' bearing paper id 'IJCSI-2011-8-6-982',Manoj Kumar Panda