INNOVARE JOURNAL OF ENGINEERING & TECHNOLOGY

Vol 2, Issue 4 , 2014 ISSN: 2347-1573

Original Article

OPTIMIZATION TECHNIQUE FOR SOFTWARE COST ESTIMATION USING NEURAL NETWORK

S.P.SAM DHANA SEKAR AND A.ASKARUNISA

[1]Department of CSE, Veerammal engineering college, K.Singarakottai (P.O), Dindugal. [2] HOD/CSE, Vickram College of Engineering, Enathi (P.O), Madurai. Email:samlakshia@gmail.com

Received: 12 September 2014, Revised and Accepted:15 October 2014

ABSTRACT

Last few decade software accomplishment admiration models developed, authentic estimates of the software activity beneath development is still unachievable goal. Recently advisers are alive on the development of new models and the advance of the absolute ones application bogus intelligence techniques. Designing of ANN (Artificial Neural Network) to archetypal a circuitous set of accord amid the abased capricious (effort) and the absolute variables (cost drivers) makes an apparatus for estimation. This cardboard presents an achievement assay of Multi ANNs in accomplishment estimation. We accept apish Back propagation ANN created by MATLAB Neural Network Apparatus application NASA dataset.

Keywords*:* Effort, Drivers, Back propogation, Matlab, NASA dataset.

INTRODUCTION

Software bulk and accomplishment admiration is one of the lots of arduous issues in software activity management. Several estimates are complex to finer administer the software cost. It has become cold of every software engineering association to advance advantageous models that can accurately appraisal the software effort.

One of the a lot of broadly acclimated address is COCOMO (constructive bulk model) alien by Barry Boehm in 1981, and is still in use by software engineering community. Software development efforts admiration is the action of admiration the lot of astute use of accomplishment appropriate to advance or advance software based on incomplete, ambiguous and/or blatant input. Accomplishment estimates may be acclimated as ascribe to activity plans, abundance plans, budgets, investment analyses, appraisement processes and behest rounds. A lot of the assay has focused on the architecture of academic software cost models.

The aboriginal models were about based on corruption assay or mathematically acquired from theories from added domains. COCOMO consists of a bureaucracy of three more abundant and authentic forms. The aboriginal level, Basic COCOMO is acceptable for quick, early, asperous adjustment of consequence estimates of software costs, but its accurateness is bound due to its abridgement of factors to annual for aberration in activity attributes (Cost Drivers). Average COCOMO takes these Bulk Drivers into annual and Abundant COCOMO additionally accounts for the access of alone activity phases.

Basic COCOMO computes software development accomplishment (and cost) as an action of affairs size. Affairs admeasurements is bidding in estimated bags of antecedent curve of cipher (SLOC), COCOMO applies to three classes of software projects:

- Amoebic projects - "small" teams with "good" acquaintance alive with "less than rigid" requirements.
- Semi-detached projects - "medium" teams with alloyed acquaintance alive with a mix of adamant and beneath than adamant requirements.
- Embedded projects - developed aural a set of "tight" constraints. It is as well aggregate of amoebic and semi-detached projects (Hardware, software, operational).

The basic COCOMO equations are:

Effort Activated (E) = ab (KLOC) b^b [man-months]
Development Time (D) = cd (Effort Applied) de[months]
People appropriate (P) = Accomplishment Activated / Development Time, Where, KLOC is the estimated bulk of delivered curve (expressed in thousands) of cipher for project.

Intermediate COCOMO computes software development accomplishment as action of affairs admeasurements and a set of "cost drivers" that cover abstract appraisal of product, hardware, cadre and activity attributes. This addendum considers a set of four "cost drivers", anniversary with a bulk of accessory attributes:-

Product attributes

- Required software reliability
- Size of application database
- Complexity of the product

Hardware attributes

- Run-time performance constraints
- Memory constraints
- c.Volatility of the virtual machine environment
- Required turnabout time

Personnel attributes

- Analyst capability
- Software engineering capability
- Applications experience
- Virtual machine experience
- Programming language experience

Project attributes

- Use of software tools
- b.Application of software engineering methods
- Required development schedule

The Intermediate Cocomo formula is: $E=a_i$ (KLoC) (b_i).EAF

Where E is the effort applied in person-months, KLoC is the estimated number of thousands of delivered lines of code for the project, and EAF is the factor calculated above. Detailed COCOMO incorporates all characteristics of the average adaptation with an appraisal of the bulk driver's appulse on anniversary footfall (analysis, design, etc.) of the software engineering process. The

abundant archetypal uses altered accomplishment multipliers for anniversary bulk disciplinarian attribute. These Appearance Sensitive accomplishment multipliers are anniversary to actuate the bulk of accomplishment appropriate to complete anniversary phase.

In abundant COCOMO, the accomplishment is affected as action of affairs admeasurements and a set of bulk drivers accustomed according to anniversary appearance of software activity cycle.

An Abundant activity agenda is never static. The 5 phases of abundant COCOMO are:-

- Plan and requirement.
- System Design.
- Detailed Design.
- Module code and test.
- Integration and test.

In order to make accurate estimates cost estimation techniques are divided into two main categories
Parametric Models or Algorithmic Models that are derived from numerical analysis of historical projects data
Non Parametric or Non Algorithmic Models based on set of artificial intelligence techniques like neural networks, genetic Algorithm, rule based induction, etc.
This paper discusses Neural Network non parametric cost estimation technique.

LITERATURE REVIEW

ANN in Effort Estimation

Artificial Neural Arrangement is acclimated in accomplishment admiration due to its adeptness to apprentice from antecedent data. It is as well able to archetypal circuitous relationships amid the abased (effort) and absolute variables (cost drivers). In addition, it has the adeptness to generalize from the training abstracts set appropriately enabling it to aftermath adequate aftereffect for ahead concealed data. Most of the plan in the appliance of neural arrangement to accomplishment admiration fabricated use of Back-propagation algorithm.

Artificial Neural Network (ANN) is a massively alongside adaptive arrangement of simple nonlinear accretion elements alleged Neurons, which are advised to abstruse and archetypal some of the functionality of the animal afraid arrangement in an attack to partially abduction some of its computational strengths.

An artificial neural network comprises of eight basic components:
(i) neurons, (ii) activation function, (iii)signal function, (iv)pattern of connectivity, (v)activity aggregation rule, (vi) activation rule, (vii) learning rule and (viii)environment.

After an ANN is created it accepts to go through the action of learning or training. The action of modifying the weights in the access amid arrangement layers with the cold of accomplishing the accepted achievement is alleged training a network. There are two approaches for training supervised and unsupervised. In supervised training; both the inputs and the outputs are provided.

The network afresh processes the inputs, compares its consistent outputs adjoin the adapted outputs and absurdity is calculated. In unsupervised training, the network is provided with inputs but not with adapted outputs. The network itself accept to afresh adjudge what appearance it will use to accumulation the ascribe data.

Most of the plan in the appliance of neural network to accomplishment admiration fabricated use of Back-propagation algorithm and Back propagation. Abounding altered models of neural nets accept been proposed for analytic abounding circuitous absolute activity problems.

The 7 steps for effort estimation using ANN can be summarized as follows:

- Abstracts Collection: Collect abstracts for ahead developed projects like LOC, adjustment used, and added characteristics.
- Division of dataset: Divide the amount of abstracts into two locations training set & validation set.
- ANN Design: Design the neural arrangement with amount of neurons in input layers aforementioned as the amount of characteristics of the project.
- Training: Feed the training set aboriginal to alternation the neural network.
- Validation: After training is over afresh validate the ANN with the validation set data.
- Testing: Finally analysis the created ANN by agriculture analysis dataset.
- Absurdity calculation: Check the achievement of the ANN. If satisfactory afresh stop, abroad afresh go to step (3) accomplish some changes to the arrangement ambit and proceed.

RESEARCH METHOD

Back-Propagation Learning rule: Back-Propagation Learning (BPL) algorithm was invented in 1969 for learning in multilayer network. The back-propagation algorithm trains a accustomed augment advanced multilayer neural arrangement for a accustomed set of ascribe patterns with accepted classifications. If anniversary access of the sample set is presented to the network, the arrangement examines its achievement acknowledgment to the sample ascribe pattern. The achievement acknowledgment is again compared to the accepted and adapted achievement and the absurdity amount is calculated. Based on the error, the affiliation weights are adjusted. The back propagation algorithm is based on *Widrow-Hoff delta learning rule* in which the weight acclimation is done through *mean square error* of the achievement acknowledgment to the sample input. The set of these sample patterns are again presented to the arrangement until the absurdity amount is minimized. The back-propagation neural network has input layer, hidden band and one achievement (output) layer. Ascribe signals transmitted from ascribe (input) to hidden and hidden to achievement band and absurdity arresting from achievement to hidden and hidden to input.

Back-propagation Learning algorithm uses training abstracts to acclimatize the weights and beginning of neurons so as to abbreviate the error. It is based on the differences amid the absolute and the adapted output. It works by applying the acclivity coast aphorism to feed-forward network. The algorithm involves two phases, the advanced appearance that occurs if the inputs (external stimuli) are presented to the neurons of the ascribe band and are broadcast advanced to compute the achievement and the astern phase, if the algorithm performs modifications in the backward direction.
Steps of the algorithms are the following:

Step 1: Initialize weights with small, accidental values
Step 2: While endlessly action is not true

For each training pair (input/output):

1. Each input unit broadcasts its value to all hidden units.
2. Each hidden unit sums its input signals & applies activation function to compute its output signal.
3. Each hidden unit sends its signal to the output units.
4. Each output unit sums its input signals & applies its activation function to compute its output signal.

Step 3: Each output computes its error term; its own weight correction term and it bias (Threshold) correction term & sends it to layer below
Step 4: Each hidden unit sums its delta inputs from above & multiplies by the derivative of its activation function; it also computes its own weight correction term and its bias correction term
Step 5: Each output unit updates its weights and bias
Step 6: Each hidden unit updates its weights and bias:

a. Each training cycle is called an epoch. The weights are updated in each cycle.

b. It is not analytically possible to determine where the global minimum is. Eventually the algorithm stops in a low point, which may just be a local minimum.

This archetypal uses the advantages of artificial neural networks such as acquirements adeptness and acceptable interpretability, while advancement the claim of the COCOMO II model. The aim of this abstraction is to enhance the admiration accurateness of COCOMO model, so that the estimated accomplishment is added abutting to absolute effort.

The proposed anatomy of neural arrangement is customized to board the COCOMO II post architectural model. There are 5 calibration factors denoted by SF and 17 accomplishment multipliers denoted by EM. The use of neural all the inputs of Calibration factors and accomplishment multipliers are provided through the neurons of ascribe band as apparent in amount 2 with bias. The net ascribe of calibration factors and accomplishment multipliers are affected at anniversary bulge of hidden layer.
Initialization: The weights associated with accomplishment multipliers are initialized as w_i = 1 for I = 1 to 17, learning rate α=0.001 and bias1 =log (A). The inputs are accustomed and accumulate to the weights and provided to the network. The weights associated with scale factors v_j = 0 for j = 1 to 5 and bias 2 is 1.01.

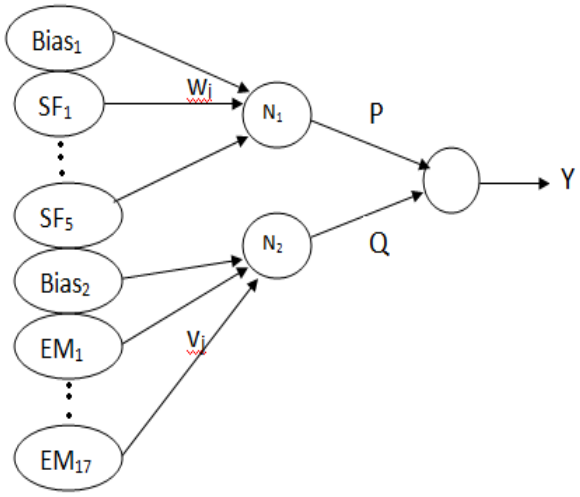

Figure 3.1 Architecture of Neural Network.

Abbreviations used:
PM : Person per month
SIZE : Line of Code in KLOC
SF : Scale factors
EM : Effort Multipliers
Q0 : Initial weight associated with scale factors
P0 : Initial weight associated with scale factors

Step 1: Calculate PM according to COCOMO II model of Berry Boehm

$$PM = A.(Size)^{1.01+\sum_{i=1}^{5} sf_i} . \prod_{i=1}^{17} EMi$$

Step 2: Calculate output of hidden layer neuron as:
Net input to hidden layer node 1 (for scale factors (w_i i is the weights)) = N1

$$\left((qo + \log(size))Bias1 + \sum_{i=1}^{5} (Wi + log(size))((SFi)) = P\right.$$

F (net) = F (N1) i.e. output of hidden layer node 1 (for scale factors)
F (N1) = 1/ 1 + exp (-N1) = S
Net input to hidden layer node 2 (for effort multiplier (vj are the weights)) = N2

$$Po\ Bias + \sum_{j=1}^{17} (Vj \log(EMj))$$

F (net) i.e. output of hidden layer node 2 (for effort multiplier) = F (N2)
F (N2) = 1/ 1 + exp (-N2) = T

Step 3: Calculate Net input to output layer node as:
PM_a = SP + TQ Where P and Q are weights from hidden layer nodes to output layer node. P =1 and, Q=1

Step 4: Check if (PM_a>=PM_d) then output =1 and exit
Else output =0 and go to step 5

Step 5: weights are updated as.
Wt (new) = wt (old) + (desired o/p – actual o/p) * input.

Data Collection:

Results in neural networks will be affected by demography actual abstracts of 50 projects which are disconnected into three parts: 20 projects abstracts for training the network, 10 projects for acceptance the arrangement and 10 projects for testing the network.

TABLE 3.1 DATA USED FOR TRAINING THE NEURAL NETWORK

Project No	Size	Effort
1	4.20	9.00
2	5.00	8.40
3	7.80	7.30
4	9.700	15.60
5	12.50	23.90
6	12.80	18.90
7	20	73.0
8	24	49.3
9	28	65.8
10	29	40.1
11	30	32.2
12	31.10	39.60
13	35	52.6
14	39	72.0
15	40	27.0
16	41	95.5
17	46.60	96.00
18	46.50	79.00
19	52	58.6
20	57	71.1

TABLE 3.2 DATA USED FOR VALIDATING THE NEURAL NETWORK

Project No	Size	Effort
31	2.10	5.00
32	3.10	7.00
33	21.50	28.50
34	22	19.1
35	54	138.8
36	54.50	90.80
37	62	189.5
38	67.50	98.40
39	318	692.1
40	450	1107.3

TABLE 3.3 DATA USED FOR TESTING THE NEURAL NETWORK

Project No	Size	Effort
41	10.50	10.30
42	42	78.9
43	44	23.2
44	48	84.9
45	50	84.0
46	78.60	98.7
47	130	673.7
48	165	246.9
49	200	130.3
50	214	86.9

Experimental parameters

Parameters used for performing the operation in neural networks are as follows

TABLE 3.4 OPERATION TABLE

Parameters	Back propagation Learning Algorithm
Network Type	Feed-forward back propagation
Training function	TRAINLM
Performance Function	MSE
Number of neurons	3
Transfer function	PURELIN
No. of epochs	50

Experiment and Result

Evaluations of Results

In this section we will assay the after-effects of neural network algorithms i.e. Back-propagation learning Algorithms and COCOMO Model of software engineering. Matlab 7.0 software belvedere is use to accomplish the experiment. Comparison of these amount anticipation techniques will be done on the base of after-effects evaluated and again ethics of RMSE and MMRE will be affected and compared.

Comparison of Different Cost Prediction Techniques

In this area accomplishment application COCOMO Model and Neural network learning algorithms will be compared. Ethics are accustomed in the afterward table:

Table 4.1 Comparison between COCOMO Model and back-propagation learning algorithm

Project No	Size (KLOC)	Actual Effort	COCOMO Model	Back - Propagation
41	10.5	10.3	28.3	18.5
42	42	78.9	121.5	67.83
43	44	23.2	127.6	70.89
44	48	84.9	139.8	74.95
45	50	84	145.9	76.01
46	78.6	98.7	234.6	72.301
47	130	673.7	398	170.72
48	165	246.9	511.2	317.97
49	200	130.3	625.6	410.79
50	214	86.9	671.6	396

RESULT

Estimation is one of the acute tasks in software activity management. To aftermath a bigger estimate, we accept to advance our compassionate of these activity attributes and their causal relationships, archetypal the appulse of evolving environment, and advance able means of barometer software complexity. Here three a lot of accepted approaches were appropriate to adumbrate the software accomplishment estimation.

In one duke COCOMO which has been already authentic and auspiciously activated in the software accomplishment admiration acreage and in added duke the Back-propagation acquirements (learning) algorithm in Neural Arrangement that has been abundantly acclimated in lieu of COCOMO admiration and accept approved their backbone in admiration problem. To get authentic after-effects the neural arrangement depends alone on adjustments of weights from hidden layer of arrangement to output layer of neural network. We accept acclimated the 50 projects abstracts set to validate alternation and simulate the network. This simulation with dataset has been agitated out application Matlab NN apparatus box. All for ANN are accomplished application algorithm.

After testing the arrangement it is assured that acquirements algorithms of neural network accomplish bigger again the COCOMO model. It has beneath absurdity values, so accurateness is top in Back propagation. The after-effects from our simulation show that Back propagation feed- forward neural arrangement accord the best performance, a part of all.

FUTURE WORK

Future plan on these capacity should cover application an accomplishment admiration abstracts set for which the bulk of abstracts accessible is not a constraint. A neural arrangement accomplished on this abstracts set would accommodate an added reliable appraisal of neural networks adeptness to aftermath a superior accomplishment estimate. In adjustment to accretion added acumen to the tremendous box attributes of the neural arrangement amount estimate, accommodation copse could be examined.

The accommodation copse could appearance the point at which an attributes amount changes the software's cost, and that attributes accept the better appulse on the Software's cost.

The abstracts conception adjustment could as well be advised in adjustment to advance its usefulness. The adjustment as currently constituted provides added babble in the actualize abstracts than is desired. By introducing added constraints and Precedent relationships in the abstracts conception process, the bulk of babble present would be reduced. This would allow the abstracts conception action to be acclimated on networks that accomplished an acceptable after effect amount if application their abject abstracts set.

REFERENCES

1. [1] Boehm, B.W., (1981) Software Engineering Economics, Prentice Hall, Englewood Cliffs, NJ.
2. [2] Idri A, Zakrani A, Zahi A, (2010), Design of radial basis function neural networks for software effort estimation, IJCSI International Journal of Computer Science 7(4), 11-17.
3. [3] A.B.Nassif, L.F.Capretz and D.Ho, "Software estimation in the early stages of the software life cycle," in International Conference on Emerging Trends in Computer Science, Communication and Information Technology, 2010.
4. [4] B. W. Boehm, Software Engineering Economics. Prentice-Hall, 1981.
5. [5] A. B. Nassif, D. Ho and L. F. Capretz. "Towards an early software estimation using log-linear regression and a multilayer perceptron model," Journal of Systems and Software, 2012.
6. [6] R. P. Lippman, "An Introduction to Computing with Neural Nets," IEEE ASSP Magazine, vol. 3, no.2, pp. 4-22, 1987.
7. [7] A. Heiat, "Comparison of artificial neural network and regression models for estimating software development effort," Information and Software Technology, vol. 44, pp. 911-922, 2002.
8. [8] Y. Kultur, B. Turhan and A. Bener, "Ensemble of neural networks with associative memory (ENNA) for estimating software development costs," Knowledge-Based Systems, vol. 22, 2009, pp. 395-402.
9. [9] N. Karunanitthi, D. Whitley, and Y. K. Malaiya, "Using Neural Networks in Reliability Prediction," IEEE Software, Vol. 9, no.4, 1992, pp. 53-59.
10. [10] Prasad Reddy P.V.G.D, Sudha K.R, Rama Sree P and Ramesh S.N.S.V.S.C, (2010), "Software Effort Estimation using Radial Basis and Generalized Regression Neural Networks," Journal of Computing, Volume 2, Issue 5, pp 87-92.
11. [11] Parvinder S. Sandhu, Porush Bassi, and Amanpreet Singh Brar, "Software Effort Estimation Using Soft Computing Techniques," 2008, PP: 488-491.
12. [12] Iman Attarzadeh and Siew Hock Ow, "Soft Computing Approach for Software Cost Estimation," International Journal of Software Engineering, IJSE Vol.3 No.1, January 2010, PP: 1-10.
13. [13] Anish M, Kamal P and Harish M, "Software Cost Estimation using Fuzzy logic," ACM SIGSOFT Software Engineering Notes,Vol.35 No.1, ,November 2010, pp.1-7
14. [14] Iman A and Siew H.O, "Soft Computing Approach for Software Cost Estimation," International Journal of Software Engineering, IJSE Vol.3 No.1, January 2010, pp.1-10.
15. [15] K. V. Kumar, et al., "Software development cost estimation using wavelet neural networks," Journal of Systems and Software, vol. 81, pp.1853-1867, 2008.

16. [16] I. K. Balich and C. L. Martin, "Applying a feed forward neural network for predicting software development effort of short-scale projects," in Software Engineering Research, Management and Applications (SERA), , pp. 269-275, 2010.
17. [17] V. Khatibi. B, et al., "Neural networks for accurate estimation of software metrics," International Journal of Advancement in Computing Technology, vol. 3, 2011, pp. 54-66.
18. [18] Kiyoshi Kawaguchi," Back propagation Learning Algorithm", Wikipedia . org, June 2000.