COCOMO II Implementation Using Perceptron Learning Rule

Ridhika Sharma[1] , Dr. R. Rama Kishore[2]
[1]M.Tech (IT) Scholar , [2]Asstt. Prof.
Guru Gobind Singh Indraprastha University , New Delhi

Abstract:

Software cost and effort estimation is the most critical task in handling software projects. Since it is very difficult to bridge the gap between estimated cost and actual cost, hence the accurate cost estimation is one of the challenging tasks in maintaining software projects. In software industry the most widely used model for effort estimation is Constructive Cost Model (COCOMO). In this paper, the author explores the use of perceptron learning rule to implement COCOMO II for effort estimation. This work proposes an estimation model that incorporates COCOMO II with perceptron learning rule to provide more accurate software estimates at early phase of software development, so that the estimated effort is more close to the actual effort.

Keywords: COCOMO II, Neural Networks, Perceptron learning rule.

1.Introduction

Software cost and effort estimate is one of the most important activities in software project management [35]. It is the accuracy of cost and effort calculation that enable quality growth of software at a later stage [1, 9]. With an effective estimate of software cost and effort, software developers can efficiently decide what recourses are to be used frequently and how efficiently these resources can be utilized. For efficient software, accurate software development parameters are required, these include effort estimation, development time estimation, cost estimation, team size estimation, risk analysis, etc. .Since the effort and cost estimation is done at an early stage of software development; hence a good model is required to calculate these parameters accurately [19].

In past few decades several researchers have worked in the field of software effort estimation, and many conventional models were designed to estimate software, size and effort [6]. The models developed were based on mathematical formula and software development factors. One of the most frequently used model to estimate software effort is COCOMO developed by Berry Boehm. These models require inputs which are difficult to obtain at early stages of software development. Moreover these models take Values of software development factors based on experience and approximation, with zero reasoning Capability [2, 3]. Due to few such limitations of conventional algorithmic models,non-algorithmicmodels [21, 22, 23, 24] based on Soft Computing came into picture, which include Neural Network, Fuzzy logic and Genetic algorithms.

The non-algorithm based algorithm [10,12,and 14] work with real life situations and a vast flexibility for software development factors was provided. In this paper a neural network technique using perceptron learning algorithm for software cost estimation which is based on COCOMO II model is proposed. Perceptron model is supervised model of neural network where weights are updated depending on the teacher's response. Many researchers are working in implementing software effort and cost estimation in neural networks [4,5,and 13].

The paper is organized in following sections: section 1 describes introduction, sections 2 and 3 describes COCOMO II model and neural network using perceptron learning rule. Section 4 discusses the related work and proposed neural network model and its algorithm is described in section 5. Experimental results and evaluation criteria are shown in section 6. Section 7 ends the paper with a conclusion.

2. COCOMO II Model

There are many software cost estimation techniques **[27]** and models which are classified as algorithmic and non-algorithmic approach **[14,25,and 26].**Software development efforts estimation is the process of predicting the most realistic use of effort required to develop or maintain software based on incomplete, uncertain and/or noisy input. Effort estimates may be used as input to project plans, iteration plans, budgets, and investment analyses, pricing processes and bidding rounds.The use of a repeatable, clearly defined and well understood software development process has, in recent years, shown itself to be the most effective method of gaining useful historical data that can be used for statistical estimation. In particular, the act of sampling more frequently, coupled with the loosening of constraints between parts of a project, has allowed more accurate estimation and more rapid development times. Estimating is defined as [35]

“The process of forecasting or approximating the time and cost of completing project deliverables.” Or “The task of balancing the expectations of stakeholders and the need for control while the project is implemented “

COCOMO (Constructive Cost Model) is a model that allows software project managers to estimate project cost and duration. It was developed initially (COCOMO ’81) by Berry Boehm in early eighties. The COCOMO II model is a COCOMO’81 update for software development during 1990’s and 2000’s.

The COCOMO II Post Architectural Model [7, 8, and 11] predicts software development effort, Person Month (PM) as shown in equation 1.

$$PM = A.\ (Size)^{1.01 \sum_{j=1}^{5} SF_i} \prod_{i=1}^{17} EM_I \ldots\ldots\ldots\ldots (1)$$

It has a set of 17 multiplicative cost drivers (EM)[31, 32] and a set of 5 scaling cost drivers to determine the project’s scaling exponent (SF). These scaling cost drivers replace the development modes (Organic, semidetached, or Embedded) in the original COCOMO 81 model, and refine the four exponent-scaling factors in Ada COCOMO. All of the cost drivers are described below. There are multiple factors that affect project cost. COCOMO II model defines 17 parameters called cost drivers that have a major influence on project cost.

1. Personnel
   a.ACAP (Analyst Capability)
   b. APEX(Application Experience)
   c. PCAP(ProgrammerCapability)
   d. PLEX(Platform Experience)
   e. LTEX (Language and Tool Experience)
   f. PCON(PersonnelContinuity)
2. Platform
   a. TIME(Time Constraint)
   b. STOR(Storage Constraint)
   c. PVOL(Platform Volatility)
3. Product
   a. RELY(Required Software)
   b. DATA(Database Size)
   c. CPLX(ProductComplexity)
   d. RUSE(Required Reusability)
   e. DOCU(Documentation match to life cycle needs)
4. Project
   a. TOOL ((Use of Software Tools)
   b. SCED (Required Development Schedule)
   c. SITE(Multisite Development Schedule)

Scale factors are new in COCOMO II. They modify second coefficient in formula 1 (coefficient b). The effect of scale factor is in 1.01 – 1.26 range.

1.PREC (Precedence)
2. PMAT(Process Maturity)
3. TEAM(Team Cohesion)
4. FLEX (Development Flexibility)
5. RESL (Architectural and Risk Resolution)

Each driver can accept one of the six possible ratings : Very Low(VL) , low(L) , Nominal (N), High(H) , Very High(VH) , and extra high (XH). Table 1 [11] shows the apriority values assigned to each rating before calibrating.

Driver	Sym	VL	L	N	H	VH	XH
PREC	SF1	0.05	0.04	0.03	0.02	0.01	0.0
FLEX	SF2	0.05	0.04	0.03	0.02	0.01	0.0
RESL	SF3	0.05	0.04	0.03	0.02	0.01	0.0
TEAM	SF4	0.05	0.04	0.03	0.02	0.01	0.0
PMAT	SF5	0.05	0.04	0.03	0.02	0.01	0.0
RELY	EM1	0.75	0.88	1.00	1.15	1.40	
DATA	EM2		0.94	1.00	1.08	1.16	
CPLX	EM3	0.75	0.88	1.00	1.15	1.30	1.65
RUSE	EM4		0.89	1.00	1.16	1.34	1.56
DOCU	EM5	0.85	0.93	1.00	1.08	1.17	
TOME	EM6			1.00	1.11	1.30	1.66
STOR	EM7			1.00	1.06	1.21	1.56
PVOL	EM8		0.87	1.00	1.15	1.30	
ACAP	EM9	1.5	1.22	1.00	0.83	0.67	
PCAP	EM10	1.37	1.16	1.00	0.87	0.74	
PCON	EM11	1.26	1.11	1.00	0.91	0.83	
AEXP	EM12	1.23	1.10	1.00	0.88	0.80	
PEXP	EM13	1.26	1.12	1.00	0.88	0.80	
LTEX	EM14	1.24	1.11	1.00	0.9	0.82	
TOOL	EM15	1.20	1.10	1.00	0.88	0.75	
SITE	EM16	1.24	1.10	1.00	0.92	0.85	0.79
SCED	EM17	1.23	1.08	1.00	1.04	1.10	

Table 1. Apriori Model Values

3. Neural Network

A Neural Network (NN) is an artificial, computational model that simulates biological neural networks.

Basically, a Neural Network consists of linked, artificial neurons which are typically grouped to input, hidden, and output layers. Depending on the network structure, different network types can be identified. In contrast to recurrent networks, Feed-Forward Networks represent a directed acyclic graph. Information is forwarded in one direction only, consecutively processed by the input, hidden, and output neurons. [15]

Neural networks consist of layers of interconnected nodes, where each node produces a non-linear function of its input. The nodes in the network are divided into the ones from the input layer going through the network to the ones at the output layer through some nodes in a hidden layer. The NN process starts by developing the structure of the network and establishing the technique used to train the network with using an existing data set.

Therefore, there are three main entities:

1. the neurons (nodes),
2. the interconnection structure,
3. the learning algorithm

Artificial neural networks are the interconnection of the artificial neurons. They are used to solve the artificial intelligence problems without the need for creating a real biological model. The neural network used in our approach is perceptron neural network [34].The perceptron is a network that learns concepts, i.e. it can learn to respond with true (1) or False (0) for inputs presented to it, by repeatedly studying examples provided to it. This network weights and biases could be trained to produce a correct target vector when presented with the corresponding input vector. The training technique used is called the perceptron learning rule. Perceptron Neural network is selected due to its ability to generalize from its training vectors and work with randomly distributed connections. Vectors from a training set are presented to the network one after another. If the network's output is correct, no change is made. Otherwise, the weights and biases are updated using the perceptron learning rule. An entire pass through all of the input training vectors is called an epoch. When such an entire pass of training set has occurred without an error, training is complete. At this time any input training vector may be presented to the network, the network will tend to exhibit generalization by responding with an output similar to the target vectors close to the previously unseen input vectors.
The activation function is one of the key components of the perceptron as in the most common neural network architectures. It determines based on the inputs, whether the perceptron activates or not. The perceptron takes all of the weighted input values and adds them together. If the sum is above or equal to some value (called the threshold) then the perceptron fires. Otherwise, the perceptron does not [19].

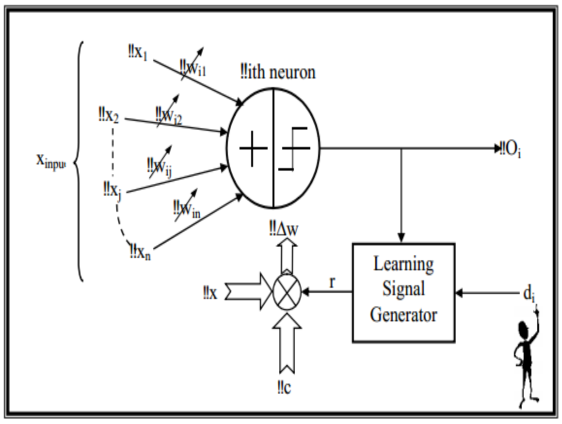


Figure 1.Neural Network Model

4.Related Work

Many researchers used their different non algorithmic models and different data sets to predict the software effort more correctly [28,29,and 32]. Most of the work in the application of neural network to estimate effort use backpropogation algorithm and cascade correlation network. [24]. ANN is a network of nonlinear computing elements called neurons which model the functionality of human brain. Anjana Bawa [24] proposed a general ANN architecture composed of 8 basic components. (i) Neurons, (ii) Activation function, (iii) Signal function, (iv) Pattern of connectivity, (v) Activity aggregation rule, (vi) Activation rule, (vii) Environment. The model implemented by Anupama Kaushik, et al. [19], is trained using perceptron learning algorithm. The test results from the trained neural network are compared with COCOMO model. Nasser Tadayon [17] explained the use of expert judgment and machine learning technique using neural network as well as referencing COCOMO II approach to predict the cost of software. Ch.Satyananda Reddy [20] adopted feed forward multilayer perceptron with linear activation function to avoid slow convergence problem that is a drawback of sigmoid activation function.

5. Proposed Neural Network

The main objective of the software cost and effort estimation using perceptron learning rule is to enhance the cost and effort estimation accuracy by introducing the concept of perceptron learning rule [33, 34] on COCOMO II model.
The proposed structure of the neural network with perceptron learning is shown in figure 2.
Neural networks consist of layers of interconnected nodes, where each node produces a non-linear function of its input. The neural network structure,

as shown in figure 4, used in our work consists of three layers namely:

1.Input layer: The use of the neural network to estimate PM (person-month) requires twenty-four input nodes in the input layer in the proposed neural network that corresponds to all EM and SF as well as two bias values.

2. Hidden layer: In order to structure the network to accomplish the COCOMO II post-architecture model, a specific hidden layer and a sigmoid activation function with some pre-processing of data for input layer is considered

3. Output layer: there is only one neuron at the output layer that will output the effort calculated from the network in terms of PM (Person/month).

The proposed structure of neural network is customized to accommodate the COCOMO II post architectural model. There are 5 scale factors denoted by SF and 17 effort multipliers denoted by EM. These inputs enter the network as weighted inputs. The effort is calculated using equation (1). The weights are initialized as w_i=1 for i=1 to 17 and v_j=0 for j=1 to 5. The values of bias1 is log (a) and bias2=1.01. All the inputs of Scale factors and effort multipliers are provided through the neurons of input layer as shown in figure 4 with bias.

As the propogation network uses summation of the inputs but the COCOMO II model uses its multiplication, a log function is used to neutralize them. So, the equation obtained by Berry Boehm model of effort estimation is modified as:

Log (Effort) = log (a*[size]^b * $_{i=1}\pi^{15}$EM$_i$)

The output obtained by the above equation [20], is compared using the activation function and the output signal is sent forward. According to the output of the activation function, the weights applied on the inputs are modified. When the output of activation function is 1, the difference between the actual effort and the effort calculated is found to check if it is the permissible limit or not. If it is in the permissible limit, the output is accepted else weights are adjusted. This completes one epoch of the project.

This work proposes an estimation model that incorporates Constructive Cost Model (COCOMO II) with perceptron learning rule to provide more accurate software estimates at the early phase of software development. There are several on-going researchers working on implementing COCOMO using neural networks [16 , 17 , 18], but in this research a neural network model is trained using Perceptron learning approach to implement COCOMO II post architectural model.

This model uses the advantages of artificial neural networks such as learning ability and good interpretability, while maintaining the merits of the COCOMO II model. The aim of this study is to enhance the estimation accuracy of COCOMO model, so that the estimated effort is more close to actual effort. The proposed structure of neural network is customized to accommodate the COCOMO II post architectural model. There are 5 scale factors denoted by SF and 17 effort multipliers denoted by EM. The use of neural All the inputs of Scale factors and effort multipliers are provided through the neurons of input layer as shown in figure 4 with bias. The net input of scale factors and effort multipliers is calculated at each node of hidden layer.

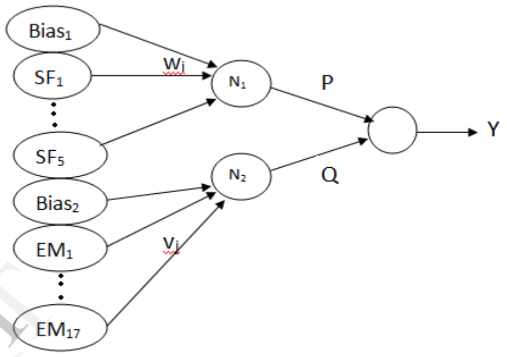


Figure 2: architecture of neural network.

Initialization: The weights associated with effort multipliers are initialized as w_i = 1 for I = 1 to 17, learning rate α = 0.001 and bias1 =log (A). The inputs are received and multiply to the weights and provided to the network. The weights associated with scale factors v_j = 0for j = 1 to 5 and bias 2 is 1.01.

Abbreviations used:

PM	: Person per month
A	:
SIZE	: Line of Code in KLOC
SF	: Scale factors
EM	: Effort Multipliers
Q_0	: Initial weight associated with scale factors
P_0	: Initial weight associated with scale factors

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

$PM_d = A. (Size)^{1.01} \sum_{j=1}^{5} SF_i . \prod_{i=1}^{17} EM_i$

Step 2: Calculate output of hidden layer neuron as:

Net input to hidden layer node 1 (for scale factors (w_i is the weights)) = N1 ((q0 + log (size)) Bias1 + $\sum_{i=1}^{5}(w_i + \log(size))(SF_i)$) = P

F (net) i.e. output of hidden layer node 1 (for scale factors) = F (N1)

F (N1) = 1/ 1 + exp (-N1) = S

Net input to hidden layer node 2 (for effort multiplier (v_j are the weights)) = N2

(P_0 Bias +$\sum_{j=1}^{17}(V_j \log(EMj))$)

F (net) i.e. output of hidden layer node 2 (for effort multiplier) = F (N_2)

F (N_2) = 1/ 1 + exp (-N_2) = 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.
Go to step 2

For flowchart see Annexure 1.

6. Estimation Criteria and Results

The experiments are done with the proposed neural network and are implemented in Visual Studio 2010. In this thesis, a cost estimation model based on artificial neural networks is constructed.

The evaluations consist of comparing the accuracy of the estimated effort with the actual effort. There are many evaluation criteria for software effort estimation among them here MRE (Magnitude Of relative Error) [36, 37] is used which is defined as:

$$\text{MRE} = \frac{(\text{Actual Effort} - \text{Estimated Effort})}{\text{Actual Effort}} * 100$$

The MRE was calculated for each software project based on the above equation. Table 5 [Annexure 2] shows some of the experimental values which weretested. These values are then compared with the actual effort of the model. The comparison tells us about the efficiency of our network. Each row of the table corresponds to a project data which specifies the size of the project, actual effort of the project, the cost driver values and finally the effort calculated by our project. The input values are entered in the project through a GUI (Graphical User Interface). The model is implemented in Visual Studio.

Table 6 [Annexure 3] shows the actual effort, the estimated effort and the MRE value for the experimented projects. Figure 4 is the graphical representation of the actual and calculated effort of the 15 projects. Through this graph it can be observed that the difference between the actual and the calculated effort is quite less which shows that the proposed algorithm is an accurate and precise algorithm.

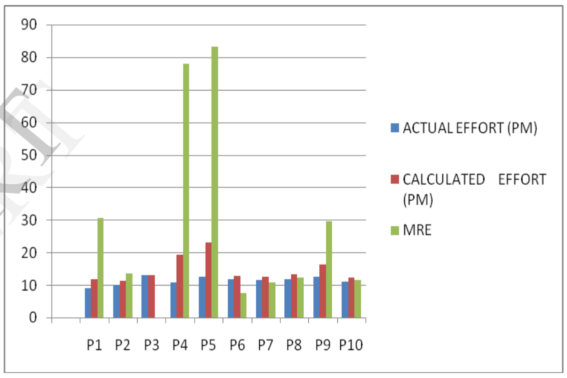


Figure 4: Actual and Calculated Effort

7. Conclusion

Neural network architecture for multilayer perceptron is used to implement COCOMO II model for software effort estimation and the learning rule used is Perceptron learning rule.
The architecture of the network is multilayer and network is trained using Perceptron learning rule.
Proposed algorithm takes inputs viz. Scale factors, Effort Multipliers and Size to calculate intermediate values of hidden layers and sigmoid activation function is applied to get the output of hidden neurons, and the output node produces 0 or 1 i.e. true or false based on the net input received at the output node. Final results are shown using Visual Studio 2010 and mean error is calculated.

Thus, it is concluded that the use of artificial neural network algorithm to model the COCOMO II estimation algorithm is an efficient way to find the values of project estimates.

8. Future Scope

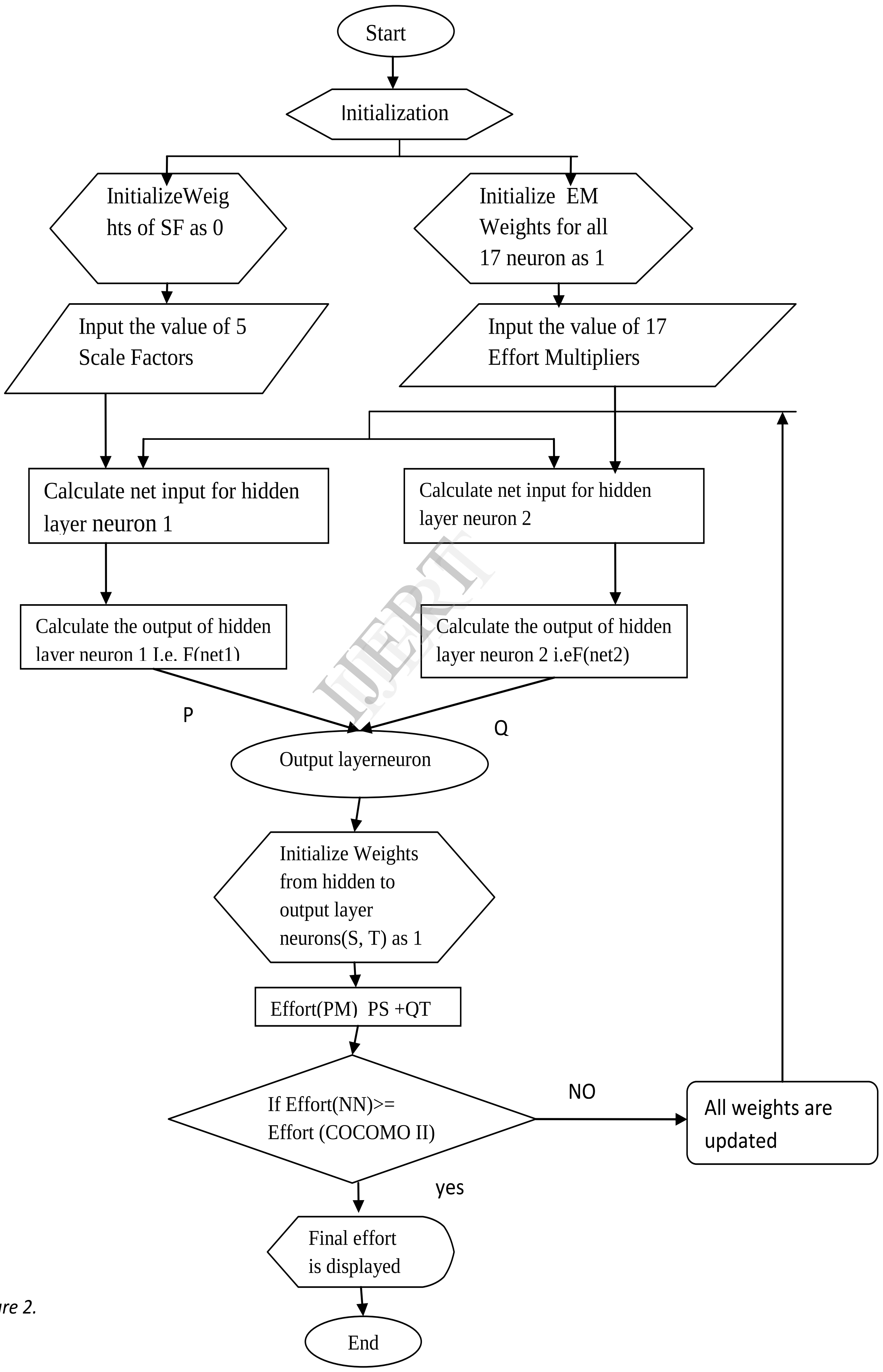
Here the most popular neural network approach viz. Perceptron learning rule is suggested to predict the software cost estimation. To get accurate results the proposed neural network depends on adjustment of weights from input to hidden layer of the perceptron model. The proposed network is validated using 10 sample of different projects which is used to train and test the designed neural network and found that the Neural Network designed with perceptron learning rule performs better in terms of efficiency and estimation accuracy. This work can be extended by integrating with various supervised learning algorithms.

References

1. Boehm, B.W., “Software Engineering Economics”, IEEE Transactions on software Engineering, SE-10, 1, pp. 4-21, January 1984.
2. REVIC, Software Cost Estimating Model User's Manual, ver. 9, 1991
3. Jones, Capers, "Software Cost Estimation in 2002", Crosstalk, The Journal of Defence Software Engineering, Vol. 15 No.6, June 2002
4. Stamelos,etal.”Estimating the development cost of custom software, information and management”, V.40 n.8, pp729-741.
5. R.W.Jensom, “ Extreme Software Cost Estimating”, Crosstalk, The Journal of Defence Software Engineering, Vol. 15 No.6, June 2004
6. A.Albreeht, “Measuring Application Development Productivity”, in IBM Application Development Symp.1979, pp83-92.
7. Pressman, Roger, Software Engineering A Practitioner's Approach, McGraw Hill, Boston, MA, 2001
8. Boehm, Barry, and others Software Cost Estimation With COCOMO II, Prentice Hall, Upper Saddle River, NJ, 2000
9. Warburton, R. D. H. (1983). "Managing and predicting the costs of real-time software."IEEE Transactions on Software Engineering 9(5): 562-569.
10. Symons (1991), Software Sizing and Estimating – Mark II FPA, Wiley, UK.A.R.Venkatachalam, Software Cost Estimation Using Artificial Neural Networks. Proceedings of 1993 International Joint Conference on Neural Networks.
11. SunitaDevnani-Chulani, Bradford Clark, Barry Boehm, “Calibrating the COCOMO II Post – Architectural Model”.
12. B. Boehm., Cost Models for Future Life Cycle Process: COCOMO2. Annals of Software Engineering. 1995 [25] Pankaj jalote, ―An Integrated Approach for Software Engineering.‖ , Third Edition. ISBN: 978-81-7319-702-4.
13. J.P. Lewis, “Large Limits to software estimation”, Software Engineering Notes, Vol. 26, No. 4, July 2001Stamelos, et al. “Estimating The Development cost of custom software”, Information and Management, v.40 n.8, pp.729-741, 2003
14. Anthony Senyard et al., “Software Engineering Methods for neural Networks.” IEEE Proceedings of the tenth Asia Pacific Software Engineering Conference, (APSEC’03), PAGES468-477, 2003.
15. Karunanithi, N., etal. “Using neural networks in reliability prediction”, IEEESoftware, pp.53-59, 1992.
16. Ali Idri , Samir Mbarki , Alain Abran ,Validating and Understanding Software Cost Estimation Models based on Neural Networks
17. Nasser Tadayon, Neural Network Approach of Software Cost Estimation. Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC’05) 0-7695- -2315-3/05
18. Iman Attarzadeh , Amin Mehranzadeh , Ali Barati , Proposing an Enhanced Artificial Neural Network Prediction Model to Improve Accuracy in Software Effort Estimation , 2012 Fourth International Conference on Computational Intelligence, Communication Systems and Networks
19. Anupama Kaushik , Ashish Chauhan , Deepak Mittal, Sachin Gupta , COCOMO Estimates using Neural Networks , *I.J. Intelligent Systems and Applications,* 2012, 9, 22-28 Published Online August 2012 in MECS
20. Ch. Satyananda Reddy, KVSVN Raju,a Concise Neural Network Model for Estimating Software Effort,International Journal of Recent Trends in Engineering, Issue. 1, Vol. 1, May 2009

21. Xishi Huang, Luiz F. Capretz, Jing Ren, Danny Ho, Neuro Fuzzy Model for Software Cost Estimation Proceedings of the Third International Conference On Quality Software (QSIC'03)
22. Ali Idri , Alain Abran , Samir Mbarki ,An Experiment on the Design of Radial Basis Function Neural Networks For Software Cost Estimation
23. Harish Mittal , Pradeep Bhatia ,Optimization Criteria for Effort Estimation using Fuzzy Technique , CLEI ELECTRONIC JOURNAL, VOLUME 10, NUMBER 1, PAPER 2, JUNE 2007.
24. Anjana Bawa , Mrs.Rama Chawala, Experimental Analysis of Effort Estimation Using Artificial Neural Network
25. Ali Bou Nassif , Luiz Fernando Capretz, Danny Ho ,Software Effort Estimation In Early Stages of the Software life Cycle Using a Cascade Correlation Neural Network Model , 2012 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing
26. L.A. Zadeh, 2002, From Computing with numbers to computing with words-from manipulation of measurements to manipulation of perceptions, Int. J. Appl. Math.Comut.Sci., 2002, Vol.12, No.3, 307-324.
27. Vahid Khatibi, Dayang N. A. Jawawi , Software Cost Estimation Methods: A Review , Journal of Emerging Trends in Computing and Information Sciences , Volume 2 No. 1, ISSN 2079-8407
28. Prasad Reddy P.V.G.D , Sudha K.R., Rama Shree P & Ramesh S.N.S.V.S.C, Software Effort Estimation using radial Basis and Generalized Regression Neural Networks, Journal of Computing, Vol2, Issue5, May2010'.
29. ZhiweiXu,TaghiM.Khoshgoftaar, Identification of fuzzy models of software cost estimation, Fuzzy sets and systems 145(2005) 141-163.
30. Mrinal Kanti Ghose, Roheet Bhatnagar and VandanaBhattacharjee, "comparing Some Neural Network Models for Software Development Effort Prediction", IEEE, 2011.
31. Ali Idri and Alan Abran, "COCOMO cost model using fuzzy logic" Proceedings of 7th International Confrence on Fuzzy Theory and Technology, Atlantic City, New Jersey, Feb27-Mar03,2000.
32. Boehm, B.W.,et al., "Cost models for future life cycle processes: COCOMO 2.0" Annals of Software Engineering on Software Process and Product Measurement, Amsterdam, 1995.
33. Stephen Marsland,Jonathan Shapiro and Ulrich Nehmzow. "Aself-organizing network that grows when required", Journal Neural Networks Vol15 Issue (8-9):1041-1058, 2002.
34. S.N.Sivanandam,S.N.Deepa, Principles of Soft Computing,Wiley, India (2007).
35. YogeshSingh,K.K.Aggarwal, Software Engineering and Estimation.
36. B.A.Kitchenham, L.M.Pickard, S.G.MacDonell and M.J Shepperd, "What Accuracy Statistics Really Measure." IEEE Proc. – Software, vol.148, no.3, pp. 81-85, June, 2001.
37. Braind L.C., Emam K.E. Surmann D. and Wieczorek I., "An assessment and comparison of common software cost estimation modelling techniques", ISERN – 1998-27.

Annexure 1.


Start
Initialization
InitializeWeights of SF as 0
Initialize EM Weights for all 17 neuron as 1
Input the value of 5 Scale Factors
Input the value of 17 Effort Multipliers
Calculate net input for hidden layer neuron 1
Calculate net input for hidden layer neuron 2
Calculate the output of hidden layer neuron 1 I.e. F(net1)
Calculate the output of hidden layer neuron 2 i.eF(net2)
P
Q
Output layerneuron
Initialize Weights from hidden to output layer neurons(S, T) as 1
Effort(PM) PS +QT
If Effort(NN)>= Effort (COCOMO II)
NO
All weights are updated
yes
Final effort is displayed
End

Annexure 2.

PROJE CT NO.	SIZE (LCOC)	ACTUA L EFFORT (PM)	SCALE FACTORS						EFFORT MULTIPLIERS					CALCU LATED EFFOR T (PM)
			VERY LOW SF	LOW SF	NOMIN AL SF	HIGH SF	VERY HIGH SF	EXTRA HIGH SF	VERY LOW	LOW EM	NOMIN AL EM	HIGH EM	VERY HIGH EM	
P1	3	9	TEAM	PMAT	RESL	FLEX	PREC		DATA PVOL SITE TOOL	RELY RUSE TIME	SCED, CPLX, STOR, PCAP	LTEX , AEXP , PEXP ,	ACAP , PCOM , DOCU	11.76 5
P2	6	10.03	PREC		FLEX	PMAT	RESL	TEAM	RELY RUSE TIME	DATA PVOL ,SITE, TOOL	ACAP , PCOM , DOCU	SCED, CPLX, STOR, PCAP	LTEX , AEXP , PEXP ,	11.38
P3	12	12.97	PMAT	REASL	TEAM	PREC		FLEX	SCED, CPLX, STOR , PCAP	LTEX , AEXP , PEXP ,	DATA , PVOL , SITE , TOOL	ACAP , PCOM ,DOCU	RELY , RUSE, TIME ,	12.97
P4	7	10.908	RESL	FLEX		TEAM	PMAT	PREC	LTEX AEXP PEXP	SCED, CPLX, STOR, PCAP	ACAP , PCOM , DOCU	RELY , RUSE, TIME ,	DATA , PVOL , SITE, TOOL	19.40 6
P5	10	12.554	FLEX	PREC	PMAT	RESL	TEAM		ACAP PCO M DOC U	LTEX , AEXP , PEXP ,	SCED, CPLX, STOR, PCAP	DATA , PVOL , SITE, TOOL	RELY , RUSE, TIME ,	22.98
P6	9	11.89	TEAM	REASL	PREC		FLEX	PMAT	TOOL DATA STOR , SITE	SCED TIME RELY PCON	RUSE , DOCU , PVOL , PCAP , LTEX	ACAP , AEXP ,	CPLX , PEXP	12.79
P7	8	11.44	PREC	PMAT	RESL	RESL	TEAM		CPLX TIME TOOL ,RELY	SITE , PVOL , SCED , DOCU	STOR , DATA , ACAP , PCOM	LTEX , AEXP , PEXP ,	RUSE , PCAP	12.66
P8	14	11.82		PMAT	TEAM	PREC	FLEX	RESL	STOR ,DAT A,AC AP , PLEX	DOCU ,RUSE, PCOM ,APEX	SCED , PVOL , PCAP , TOOL	RELY , CPLX	TIME , SITE , LTEX ,	13.28
P9	12	12.61	FLEX	RESL		PMAT	PREC	TEAM	TOOL DATA STOR , SITE	ACAP ,AEXP	CPLX , PEXP	SCED TIME RELY PCON	RUSE ,DOCU , PVOL , PCAP , LTEX	16.33
P10	10	11.01		FLEX	PMAT	TEAM	PREC	RESL	STOR DATA ACAP PCO M	RUSE ,PCAP	LTEX , AEXP , PEXP ,	CPLX , TIME , TOOL , RELY	SITE , PVOL , SCED , DOCU	12.27

Table 1. Experimental Evaluation

Annexure 3.

PROJECT NO.	ACTUAL EFFORT (PM)	CALCULATED EFFORT (PM)	MRE
P1	9.00	11.76	30.66
P2	10.03	11.38	13.45
P3	12.97	12.927	0
P4	10.90	19.406	78.03
P5	12.55	22.98	83.10
P6	11.89	12.79	7.56
P7	11.44	12.66	10.66
P8	11.82	13.28	12.35
P9	12.61	16.33	29.50
P10	11.01	12.27	11.44

Table 2. Evaluation Criteria