I.J. Intelligent Systems and Applications, **2012, 9, 22-28**
Published Online August 2012 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2012.09.03

COCOMO Estimates Using Neural Networks

Anupama Kaushik,
Assistant Professor, Dept. of IT, Maharaja Surajmal Institute of Technology, GGSIP University, Delhi, India
anupama@msit.in

Ashish Chauhan, Deepak Mittal, Sachin Gupta
Dept. of IT, Maharaja Surajmal Institute of Technology, GGSIP University, Delhi, India
Ashish.chauhan004@gmail.com; deepakm905@gmail.com; sachin.gupta_15@yahoo.com

***Abstract*—** Software cost estimation is an important phase in software development. It predicts the amount of effort and development time required to build a software system. It is one of the most critical tasks and an accurate estimate provides a strong base to the development procedure. In this paper, the most widely used software cost estimation model, the Constructive Cost Model (COCOMO) is discussed. The model is implemented with the help of artificial neural networks and trained using the perceptron learning algorithm. The COCOMO dataset is used to train and to test the network. The test results from the trained neural network are compared with that of the COCOMO model. The aim of our research is to enhance the estimation accuracy of the COCOMO model by introducing the artificial neural networks to it.

***Index Terms*—** Artificial Neural Network, Constructive Cost Model, Perceptron Network, Software Cost Estimation

I. Introduction

Software cost estimation is one of the most significant activities in software project management. Accurate cost estimation is important because it can help to classify and prioritize development projects to determine what resources to commit to the project and how well these resources will be used. The accuracy of the management decisions will depend on the accuracy of the software development parameters. These parameters include effort estimation, development time estimation, cost estimation, team size estimation, risk analysis, etc. These estimates are calculated in the early development phases of the project. So, we need a good model to calculate these parameters. An early and accurate estimation model reduces the possibilities of conflicts between members in the later stages of project development.

In the last few decades many software cost estimation models have been developed. The algorithmic models also known as conventional models use a mathematical formula to predict project cost based on the estimates of project size, the number of software engineers, and other process and product factors [1]. These models can be built by analysing the costs and attributes of completed projects and finding the closest fit formula to actual experience. COCOMO (Constructive Cost Model), is the best known algorithmic cost model published by Barry Boehm in 1981 [2]. It was developed from the analysis of sixty three software projects. These conventional approaches lacks in terms of effectiveness and robustness in their results. These models require inputs which are difficult to obtain during the early stages of a software development project. They have difficulty in modelling the inherent complex relationships between the contributing factors and are unable to handle categorical data as well as lack of reasoning capabilities [3]. The limitations of algorithmic models led to the exploration of the non-algorithmic models which are soft computing based.

Non algorithmic models for cost estimation encompass methodologies on fuzzy logic (FL), artificial neural networks (ANN) and evolutionary computation (EC).These methodologies handle real life situations by providing flexible information processing capabilities. This paper proposed a neural network technique using perceptron learning algorithm for software cost estimation which is based on COCOMO model. Neural networks have been found as one of the best techniques for software cost estimation. Now-a-days many researchers and scientists are constantly working on developing new software cost estimation techniques using neural networks [4, 5, 6, 7].

The rest of the paper is organized as follows: section 2 and 3 describes the COCOMO model and artificial neural network concepts respectively.Section3 and 4 discusses the related work and proposed neural model. Section 4 and 5 presents the proposed model and the training algorithm implemented. Section 6 discusses the experimental results and evaluation criteria. Finally Section 7 concludes the paper.

II. COCOMO Model

The COCOMO model [2] is the most widely used algorithmic cost estimation technique due to its

simplicity. This model provides us with the effort in person months, the development time in months and the team size in persons. It makes use of mathematical equations to calculate these parameters. The COCOMO model is a hierarchy of software cost estimation models and they are:

1. Basic Model- It estimates effort for the small to medium sized software projects in a quick and rough fashion and takes the form

$$E = a\,(SIZE)^{b} \quad (1)$$

where E is effort applied in Person-Months and SIZE is measured in thousand delivered source instructions. The coefficients a and b are dependent upon the three modes of development of projects. Boehm proposed three modes of projects:

(a) Organic mode- It is for small sized projects upto 2-50 KLOC (thousand lines of code) with experienced developers in a familiar environment.

(b) Semi detached mode- It is for medium sized projects upto 50-300 KLOC with average previous experience on similar projects.

(c) Embedded mode- It is for large and complex projects typically over 300 KLOC with developers having very little previous experience.

2. Intermediate Model- The Basic COCOMO does not take account of the software development environment. Boehm introduced a set of 15 cost drivers in the Intermediate COCOMO that adds accuracy to the Basic COCOMO. The cost drivers are grouped into four categories:

1. Product attributes
   (a) Required software reliability (RELY)
   (b) Database size (DATA)
   (c) Product complexity (CPLX)

2. Computer attributes
   (a) Execution time constraint (TIME)
   (b) Main storage constraint (STOR)
   (c) Virtual machine volatility (VIRT)
   (d) Computer turnaround time (TURN)

3. Personnel attributes
   (a) Analyst capability (ACAP)
   (b) Application experience (AEXP)
   (c) Programmer capability (PCAP)
   (d) Virtual machine experience (VEXP)
   (e) Programming language experience (LEXP)

4. Project attributes
   (a) Modern programming practices (MODP)
   (b) Use of software tools (TOOLS)
   (c) Required development schedule (SCED)

Table I Coefficients for Intermediate COCOMO

MODE	a	b
Organic	3.2	1.05
Semi-Detached	3	1.12
Embedded	2.8	1.20

The Cost drivers have up to six levels of rating: Very Low, Low, Nominal, High, Very High, and Extra High. Each rating has a corresponding real number known as effort multiplier, based upon the factor and the degree to which the factor can influence productivity. The estimated effort in person-months (PM) for the intermediate COCOMO is given as:

$$\text{Effort} = a \times [SIZE]^{b} \times {}_{i=1}\Pi^{15}\, EM_{i} \quad (2)$$

In equation (2) the coefficient "a" is known as productivity coefficient and the coefficient "b" is the scale factor. They are based on the different modes of project as given in Table I. The contribution of effort multipliers corresponding to the respective cost drivers is introduced in the effort estimation formula by multiplying them together. The numerical value of the ith cost driver is EM_i (Effort Multiplier).

3. Detailed Model- Boehm introduced two more capabilities in this model and they are, Phase sensitive effort multipliers which help in determining the manpower allocation for each phase of the project and three level product hierarchy. These are module, subsystem and system levels. The ratings of the cost drivers are done at appropriate level.

This research used intermediate COCOMO model because it has estimation accuracy that is greater than the basic version, and at the same time comparable to the detailed version.

III. Artificial Neural Networks

Artificial neural networks (ANN) [8] 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. These networks focus on hypothetical matters from an information processing point of view. ANN's possess large number of highly interconnected processing elements called neurons. Each neuron is connected with the other by a connection link. Each connection link is associated with weights which contain information about the input signal. This information is used by the neuron net to solve a particular problem. Each neuron has an internal state of its own. This internal state is called the activation level of neuron, which is the function of the inputs the neuron receives. There are a number of activation functions that can be applied over net input such as Gaussian, Linear, Sigmoid and Tanh. Figure 1 shows the structure of a basic neural network.

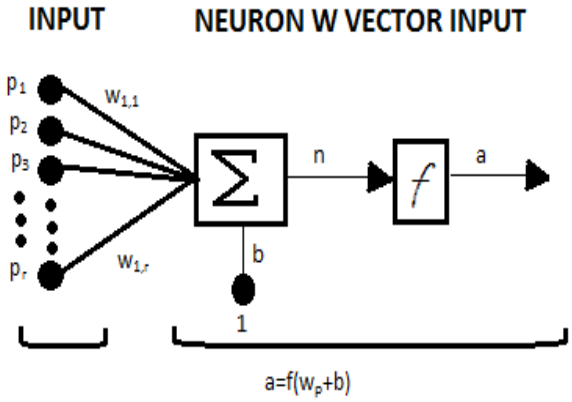

Fig.1 Basic Neural Network

A basic neural network consists of a number of inputs applied by some weights, combined together to give an output. The feedback from the output is again put into the inputs to adjust the applied weights and to train the network. This structure of the neural networks help to solve the practical, non linear, decision making problems easily.

The neural network used in our approach is perceptron neural network [9]. 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 the training set has occurred without error, training is complete. At this time any input training vector may be presented to the network and it will respond with the correct output vector. If a vector P not in the training set is presented to the network, the network will tend to exhibit generalization by responding with an output similar to target vectors for input vectors close to the previously unseen input vector P.

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. Basically, 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. The output of the perceptron network is given by

$$y = f(y_{in}) \quad (3)$$

where $f(y_{in})$ is activation function and is defined as

$$f(y_{in}) = \begin{cases} 1 & if \ y_{in} > \theta \\ 0 & if \ -\theta \le y_{in} \le \theta \\ -1 & if \ y_{in} < -\theta \end{cases} \quad (4)$$

IV. Relevant Work

Artificial neural networks are good at modeling complex non linear relationships. Since last many years, there have been many researchers who have worked upon the cost estimation of the projects using the artificial neural networks. Many researchers have applied the neural networks approach to estimate software development effort [10, 11, 12, 13, 14, 15 and 16]. A recent study by Jorgensen [17] provides a detailed review of different studies on the software development effort. Prasad Reddy P.V.G.D, Sudha K.R, Rama Sree P and Ramesh S.N.S.V.S.[18] explain the radial study of the Neural Network. Another study by Samson et al. [12] uses an albus multilayer perceptron in order to predict software effort. They use Boehm's COCOMO dataset. Srinivasan and Fisher [15] report the use of a neural network with a back propagation learning algorithm. They found that the neural network outperformed other techniques. K. Vinay Kumar, V. Ravi, Mahil Carr, and N. Raj Kiran [19] use the wavelet neural network for predicting software development cost. N. Tadayon [20] also reports the use of neural network with a back propagation learning algorithm. However it was not clear how the dataset was divided for training and validation purposes. B. Tirimula Rao et al. [21] provided a novel neural network approach for software cost estimation using functional link artificial neural network. COCOMO is arguably the most popular and widely used software estimation model, which integrates valuable expert knowledge [2].

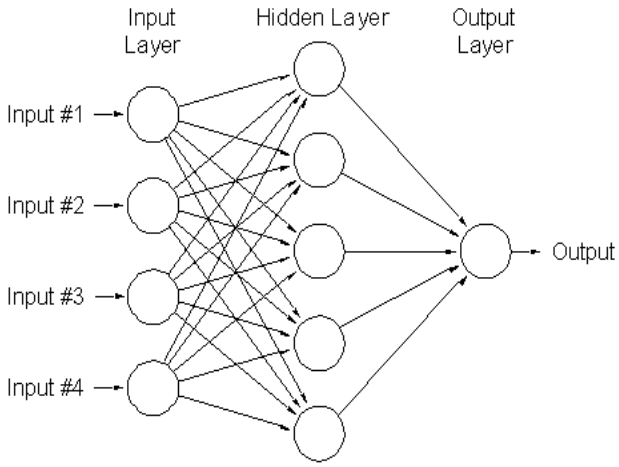

Fig. 2 Architecture of the proposed Neural Network

V. Proposed Neural Network

Figure 2 shows the basic structure of the proposed

network. The performance of a neural network depends on its architecture and their parameter settings. There are many parameters governing the architecture of the neural network including the number of layers, the number of nodes in each layer, the transfer function in each node, learning algorithm parameters and the weights which determine the connectivity between nodes. Inappropriate selection of network patterns and learning rules may cause serious difficulties in network performance and training. The problem is to decide the number of layers and number of nodes in the layers and the learning algorithm as well. However, the criterion is to select the minimum nodes which would not impair the network performance. The number of layers and nodes should be minimized to amplify the performance.

In our network, there are 17 inputs to the network which are size of the project in KLOC, 15 effort multipliers, actual effort of the project and one bias value. These inputs enter the network as weighted inputs. The effort is calculated using equation (5).The weights are initialized as $W_i = 1$ for i = 1 to 17, learning rate, $\alpha = 0.001$ and bias b = 1. The inputs, as received, are multiplied to the weights and provided to the network. As the Propagation network uses summation of the inputs but the COCOMO model uses its multiplication, a log function is used to neutralize them. So, the equation (2) is modified as:

$$\log (\text{Effort}) = \log (a \times [\text{SIZE}]^b \times {}_{i=1}\Pi^{15} \text{EM}_i) \qquad (5)$$

The output obtained by the above equation, 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 actual effort and effort calculated is found to check if it is in permissible limit or not. If it is in the permissible limit, the output is accepted else the weights are adjusted. This completes with one epoch of the project.

The algorithm for training the above network and for calculating new set of weights is depicted in the following steps:

Step 1: Initialize the weights, bias and learning rate α.

Step 2: Perform steps 3-8 until stopping condition is false.

Step 3: Perform steps 4-7 for each training pair.

Step 4: The input layer receives input signal and sends it to the hidden layer by applying identity activation functions on all the input units from i=1 to 17.

Step 5: Each hidden unit j= 1 to 5 sums its weighted input signals to calculate net input given by:

$$y_{inj} = b_j + \sum_{i=1}^{n} x_i w_{ij}$$

The activation functions as given by equation (4) are applied over the above net input to calculate the output response: $y_j = f(y_{inj})$

Step 6: Calculate the output i.e. effort at the output layer using the same procedure as in step 5 and considering all the weights for j=1 to 5 as 1.

Step 7: Compare the actual effort with the computed effort, if the difference is within the permissible limit the output is accepted else the weights are updated as follow:

$$w_i(\text{new}) = w_i(\text{old}) + \alpha \times \text{input}(i)$$

Step 8: Check for the stopping condition i.e. if there is no change in weights then stop the training process, else start again from Step 3.

VI. Evaluation Criteria and Results

The experiments are done with the proposed neural network model by taking some of the original projects from COCOMO dataset. COCOMO dataset is publicly available which consists of 63 projects [22]. We have divided the entire dataset into two sets, training set and validation set to get more accuracy of prediction. The model is implemented in Matlab.

The evaluation consists in comparing the accuracy of the estimated effort with the actual effort. There are many evaluation criteria for software effort estimation among them we applied the most frequent one which is Magnitude of Relative Error (MRE) which is defined as in equation (6).

$$\text{MRE} = \frac{|\text{Actual Effort} - \text{Estimated Effort}|}{\text{Actual Effort}} \times 100 \qquad (6)$$

Table 1 shows some of the experimental values which were tested. 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, the actual effort of the project, the effort multiplier 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 Matlab.

Table 2 shows the actual effort, the estimated effort and the MRE value for the experimented projects. Figure 3 is the graphical representation of the actual and the calculated effort of 15 projects of COCOMO dataset [22]. 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.

Table 1 Experimental Studies

Project No.	Size (KLOC)	Actual Effort (person months)	Effort Multipliers				Calculated Effort (person months)
			Low Ems	Nominal Ems	High Ems	Very High EMs	
P1	29.5	120	DATA, VIRT, TURN, SCED	TIME, STOR, ACAP, AEXP, PCAP, VEXP, TOOL	RELY, CPLX, LEXP, MODP		104.17
P2	14	60	DATA, VIRT, TURN, SCED	TIME, STOR, ACAP, AEXP, PCAP, VEXP, TOOL	RELY, CPLX, LEXP, MODP		53.64
P3	5.5	18	DATA, VIRT, TURN, SCED	TIME, STOR, ACAP, AEXP, PCAP, VEXP, TOOL	RELY, CPLX, LEXP, MODP		18.69
P4	48.5	239	VIRT, VEXP, SCED	RELY, PCAP	CPLX, TURN, AEXP, LEXP	DATA, TIME, STOR, ACAP, MODP, TOOL	240.00
P5	32.6	170	VIRT, VEXP, SCED	RELY, VEXP	CPLX, TURN, AEXP, LEXP	DATA, TIME, STOR, ACAP, MODP, TOOL	154.87
P6	115.8	480	DATA, VIRT, TURN, SCED	TIME, STOR, ACAP, AEXP, PCAP, VEXP, TOOL	RELY, CPLX, LEXP, MODP		439.17
P7	66.6	300	DATA, VIRT, TURN, SCED	TIME, STOR, ACAP, AEXP, PCAP, VEXP, TOOL	RELY, CPLX, LEXP, MODP		245.39
P8	5.5	18	DATA, VIRT, TURN, SCED	TIME, STOR, ACAP, AEXP, PCAP, VEXP, TOOL	RELY, CPLX, LEXP, MODP		17.79
P9	10.4	50	DATA, VIRT, TURN, SCED	TIME, STOR, ACAP, AEXP, PCAP, VEXP, TOOL	RELY, CPLX, LEXP, MODP		38.42
P10	38	210		DATA, TIME, STOR, VIRT, TURN, ACAP, VEXP, LEXP, MODP, TOOL, SCED	RELY, CPLX, AEXP, PCAP		194.64
P11	12.8	62	VIRT, VEXP, SCED	RELY, PCAP	CPLX, TURN, AEXP, LEXP	DATA, TIME, STOR, ACAP, MODP, TOOL	54.22
P12	15.4	70	VIRT, VEXP, SCED	RELY, PCAP	CPLX, TURN, AEXP, LEXP	DATA, TIME, STOR, ACAP, MODP, TOOL	66.74
P13	16.3	82	VIRT, VEXP, SCED	RELY, PCAP	CPLX, TURN, AEXP, LEXP	DATA, TIME, STOR, ACAP, MODP, TOOL	71.13
P14	7.7	31.2	DATA, VIRT, TURN, SCED	TIME, STOR, ACAP, AEXP, PCAP, VEXP, TOOL	RELY, CPLX, LEXP, MODP	DATA, TIME, STOR, ACAP, MODP, TOOL	30.14
P15	9.7	25.2	DATA, VIRT, TURN, SCED	TIME, STOR, ACAP, AEXP, PCAP, VEXP, TOOL	RELY, CPLX, LEXP, MODP		20.55

Table 2 Comparisons of Results

Project No.	Actual Effort (person months)	Calculated Effort (person months)	MRE
P1	120	104.17	13.19
P2	60	53.64	10.6
P3	18	18.69	3.83
P4	239	240.00	0.41
P5	170	154.87	8.9
P6	480	439.17	8.5
P7	300	245.39	18.20
P8	18	17.79	1.16
P9	50	38.42	23.16
P10	210	194.64	7.31
P11	62	54.22	12.54
P12	70	66.74	4.65
P13	82	71.13	13.25
P14	31.2	30.14	3.39
P15	25.2	20.55	18.45

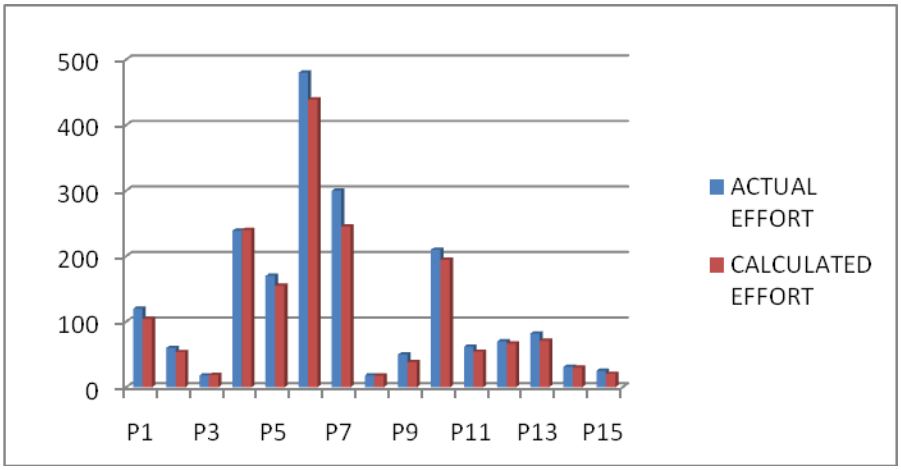

Fig. 3: Actual and Calculated Effort

VII. Conclusion

A reliable and accurate estimate of software development effort has always been a challenge for both the industrial and academic communities. There are several software effort forecasting models that can be used in forecasting future software development effort. We have constructed a cost estimation model based on artificial neural networks. Our idea consists in the use of a model that maps COCOMO model to a neural network with minimal number of layers and nodes to increase the performance of the network. The neural network that we have used to predict the software development effort is the Perceptron network. We have used the COCOM0'81 dataset to train and to test the network. It is observed that the obtained accuracy of the network is acceptable.

Thus, it is concluded that the use of the artificial neural network algorithm to model the COCOMO estimation algorithm is an efficient way to find the values of the project estimates. It provides us with nearly accurate values.

Acknowledgement

The authors would like to thank the anonymous reviewers for their careful reading of this paper and for their helpful comments.

References

[1] Ch. Satyananda Reddy, KVSN Raju,"An Improved Fuzzy Approach for COCOMO's EffortEstimation using Gaussian Membership Function," Journal of Software", Volume 4, No. 5, July 2009.

[2] Boehm, B.W., "Software Engineering Economics," Prentice-Hall, Englewood Cliffs, NJ, USA, 1994.

[3] M.O. Saliu, M.Ahmed, "Soft Computing based Effort Prediction Systems –A Survey, in : E.Damiani, L.C. Jain (Eds)," Computational Intelligence in Software Engineering, Springer-Verlag, July 2004, ISBN 3-540-22030-5.

[4] Dawson, C.W., "A neural network approach to software projects effort estimation," Transaction Information and Communication Technologies, Vol.16, pages 9, 1996.

[5] Idri, A. Khoshgoftaar, T.M. Abran, A., "Can neural networks be easily interpreted in software cost estimation?," Proceedings of the IEEE Internation Conference on Fuzzy Systems, FUZZ-IEEE'02, Vol.:2, 1162-1167, 2002.

[6] Finnie, G.R. and Wittig, G.E., "AI tools for software development effort estimation," In proceedings of the IEEE International Conference on Software Engineering: Education and Practice, Washington DC, pp 346-353, 1996.

[7] B. Tirimula Rao, B. Sameet, G. Kiran Swathi, K. Vikram Gupta, Ch. Ravi Teja, S. Sumana, "A novel neural network approach for software cost estimation using Functional Link Artificial Neural Network(FLANN)", International Journal of Computer Science and Network Society, Vol.9 No.6, June 2009.

[8] Stephen Marsland , Jonathan Shapiro, and Ulrich Nehmzow. "A self-organising network that grows when required", Journal Neural Networks, Vol. 15 Issue (8-9):1041- 1058, 2002.

[9] S.N. Sivanandam, S.N. Deepa, Principles of Soft Computing, Wiley India (2007).

[10] Ch. Satyananda Reddy and KVSVN Raju, " An Optimal Neural Network Model for Software Effort Estimation", Int.J. of Software Engineering, IJSE Vol.3 No.1 January 2010

[11] Jorgerson, M., "Experience with accuracy of software maintenance task effort prediction models," IEEE Transactions on Software Engineering, Volume 21 (8), 674–681, 1995.

[12] Samson, B., Ellison, D., Dugard, P., "Software cost estimation using an Albus perceptron (CMAC)," Journal of Information and Software Technology, Volume 39 (1), 55–60, 1997.

[13] Schofield, C., "Non-algorithmic effort estimation techniques," Technical Report TR98-01, 1998.

[14] Seluca, C., "An investigation into software effort estimation using a back propagation neural network," M.Sc.Thesis, Bournemouth University, UK, 1995.

[15] Srinivasan, K., Fisher, D., " Machine learning approaches to estimating software development effort," IEEE Transactions on Software Engineering, Volume 21 (2), 126–137, 1995.

[16] Wittig, G., Finnie, G., "Estimating software development effort with connectionist models," Journal of Information and Software Technology, Volume 39 (7), 469–476, 1997.

[17] Hughes, R.T., "An evaluation of machine learning techniques for software effort estimation," University of Brighton, 1996.

[18] Prasad Reddy P.V.G.D, Sudha K.R, Rama Sree P and Ramesh S.N.S.V.S., "Software Effort Estimation using Radial Basis and Generalized Regression Neural Networks", Journal of Computing, Volume 2, Issue 5, May 2010, ISSN 2151-9617

[19] K. Vinay Kumar, V. Ravi, Mahil Carr, N. Raj Kiran, "Software development cost estimation using wavelet neural networks", The journal of Systems and Software 81(2008) 1853-1867.

[20] N. Tadayon, "Neural Network Approach for Software Cost Estimation", proceedings of the International Conference on Information Technology: Coding and Computing(ITCC'05), Vol. 2, pp. 815-818, 2005.

[21] B. Tirimula Rao, B. Sameet, G. Kiran Swathi, K. Vikram Gupta, Ch. Ravi Teja, S. Sumana, "A novel neural network approach for software cost estimation using Functional Link Artificial Neural Network (FLANN)", International Journal of Computer Science and Network Society, Vol. 9 No.6, June 2009.

[22] www.promisedata.org

Anupama Kaushik received her B.E (Computer Science) from Bharathiyar University and M.Tech (Information Technology) from Tezpur University. She joined Department of Information Technology of Maharaja Surajmal Institute of Technology as an Assistant Professor in 2004. Her research area includes Software Engineering, Object Oriented Software Engineering and Soft Computing.

Ashish Chauhan is a student pursuing his B.Tech from Department of Information Technology of Maharaja Surajmal Institute of Technology. This work was a part of their project on Software Cost Estimation. His research area includes Software Engineering and Artificial Neural Networks.

Deepak Mittal is a student pursuing his B.Tech from Department of Information Technology of Maharaja Surajmal Institute of Technology. This work was a part of their project on Software Cost Estimation. His research area includes Software Engineering and Artificial Neural Networks.

Sachin Gupta is a student pursuing his B.Tech from Department of Information Technology of Maharaja Surajmal Institute of Technology. This work was a part of their project on Software Cost Estimation. His research area includes Software Engineering and Artificial Neural Networks.

How to cite this paper: Anupama Kaushik,Ashish Chauhan,Deepak Mittal,Sachin Gupta,"COCOMO Estimates Using Neural Networks", International Journal of Intelligent Systems and Applications(IJISA), vol.4, no.9, pp.22-28, 2012. DOI: 10.5815/ijisa.2012.09.03

