# SOFTWARE COST ESTIMATION MODELS USING ELMAN NEURAL NETWORKS

**Praynlin E[1]., Latha P[2].**

Government college of Engineering, Tirunelveli, India
Email: [1]praynlin25@gmail.com, [2]latha.muthuraj@yahoo.com

## Abstract

*Software cost estimation involves the estimation of cost required to develop a software. Cost overrun, schedule overrun occur in the software development due to the wrong estimate made during the initial stage of software development. So proper estimation is very essential for successful completion of software development. Lot of estimation techniques available to estimate the effort in which neural network based estimation method play a prominent role.. ELMAN neural network a recurrent type network can be used to estimate the cost. For a good predictor system the difference between estimated cost and actual cost should be as low as possible. To control better the time, cost and resource assigned to software project, organization need proper estimate of their size even before the project actually start. The development of a software system is an inherently complex process Estimating the cost needed to run a large software development project is doubly so and notoriously difficult.*

**Keywords:** ELMAN Network, Mean Magnitude of Relative Error (MMRE),

## I. INTRODUCTION

Estimating software development cost remains a complex problem, and the one which continues to draw significant research attention and it is very difficult to achieve accurate estimates[9,12]. Software project managers usually estimate the software development effort, cost and duration in the early stages of a software life cycle in order to appropriately plan, monitor and control the allocated resources. Correctness in estimating the required software development effort plays a critical factor in the success of software project management Right amount of resource should be allocated to a project[11]. If too much of resource is allocated to the project to is called as overestimation. If insufficient resource is allocated to the project is called underestimation. Software development activity involves lot of uncertainties the requirement will change, the developing platform will change, the developers capability to vary from one person to another lot of uncertainties are involved in the contributing factors which decides the effort required to develop the software. Hence soft computing frame work can be used that are good in handling the uncertainty. For good software estimation tool the estimated effort should be equal to the actual effort. Accurate estimation allows manager to allocate the resource to plan and coordinate all activities.

Accurate Software cost estimation is always a difficult task. Estimation by experts, analogy-based estimation and soft computing methods are some of the effort estimation methods. In estimation by experts, the entire project is subdivided into small activities and with previous experience in effort estimation the developer of software estimate the effort depending on the type of task under consideration [8]. In analogy based estimation is a form of CBR. Cases are defined as abstractions of events that are limited in time and space [13].In soft computing based approach several technique like neural network fuzzy logic, genetic engineering are used either individually or combinely as hybrid approaches to predict the effort [6].Soft computing based approach play a prominent role because the ability of the soft computing frame work to learn from previous projects especially neural network is good in learning. Now a day's estimation method using neural network is the interesting area for research compared to Theoretical estimation methods [14]. While considering the neural network lot of neural network architecture are available. Among which the most widely used method was Back propagation network. Elman network is a type of recurrent network that is equally as important as Back propagation algorithm. In our experiment both methods are used for estimating software development effort and their performance characteristics are analysed.

The paper is organized as follows. Section 2 gives the detail about the related works and section 3 talks about the research methodology and the brief description ELMAN neural network. Section 4 gives detail about the dataset used. Section 5 gives detail about the experimentation, section 6 about evaluation criteria. Results and conclusion are given in section 7 and section 8 repectively.

## II. RELATED WORKS

There are so many methods to estimate the cost they are analogy based methods[5,15], Bayesian methods[3,18]The use of Artificial Neural Networks to predict software development effort has focused mostly on the accuracy comparison of algorithmic models rather than on the suitability of the approach for building software effort prediction systems. Use of back propagation learning algorithms on a multilayer perceptron in order to predict development effort was described by Witting and Finnie [20,19]. The study of Karunanithi [10] reports the use of neural networks for predicting software reliability; including experiments with both feed forward and Jordon networks. The Albus multiplayer perceptron in order to predict software effort was proposed by Samson [16]. They use Boehm's COCOMO dataset. Srinivazan and Fisher [17] also exhibit the use of a neural network with a back propagation learning algorithm. But how the dataset was divided for training and validation purposes is not clearly mentioned. Iris febine et al[7] compares regression technique with artificial neural networks and found artificial neural network to be better than regression. Finally in the last years, a abundant interest on the use of ANNs has grown. ANNs have been fruitfully applied to several problem domains. They can be used as predictive models because they are modeling techniques having the capability of modeling complex functions

## III. RESEARCH METHODOLOGY

**Problem Statement:** Understanding and calculation of models based on historical data are difficult due to inborn complex relationships between the related attributes, are unable to handle categorical data as well as lack of reasoning abilities. Besides, attributes and relationships used to estimate software development effort could change over time and differ for software development environments. In order to overcome to these problems, a neural network based model with accurate estimation can be used.

**The COCOMO II model:** The COCOMO model is a software cost estimation model based on regression. It was developed by Barry Bohem the father of software cost estimation in 1981. Among of all traditional cost prediction models. COCOMO model can be used to calculate the amount of effort and the time schedule for software projects. COCOMO 81 was a stable model on that time. One of the problems with using COCOMO 81 today is that it does not match the development environment of the late 1990's.

Therefore, in 1997 COCOMO II was published and was supposed to solve most of those problems. COCOMO II has three models also, but they are different from those of COCOMO 81. They are

- Application composition model-mostly suitable for projects built with modern GUI-builder tools. Based on new Object Points
- Early Design Model-To get rough estimates of a project's cost and duration before have determined its entire architecture. It uses a small set of new Cost Drivers and new estimating equations. Based on Unadjusted function Points or KSLOC
- Post-Architecture Model-The most detailed on the three, used after the overall architecture for the project has been designed. One could use function points or LOC as size estimates with this model. It involves the actual development and maintenance of a software product COCOMO II describes 17 cost drivers that are used in the Post-Architecture model [2]. The cost drivers for COCOMO II are rated on a scale from Very Low to Extra High in the same way as in COCOMO 81. COCOMO II post architecture model is given as:

$$\text{Effprt } = A \times [\text{size}]^B \times \prod_{i=1}^{17} \text{Effort Multiplier}_\text{i}$$

Where

$$B = 1.01 + 0.01 \times \sum_{j=1}^{5} \text{Scale factor}_\text{j}$$

$A =$ Multiplicative constant

Size = Size of the software project measured in terms of KSLOC (thousands of source lines of code, function points or object points) The selection of Scale Factors (SF) is based on the rationale that they are a significant source of exponential variation on a project's effort or productivity variation. The standard numeric values of the cost drivers are given in Table 1.The cost drives and scale factors are given as input to the neural network with effort as the networks output. Two type of network used for analysis are discussed below.

*A. Elman Network*

Elman Network was first proposed by Jeffrey *L*. Elman in 1990. Elman neural network is feed forward network with an input layer, a hidden layer, an output layer and a distinct layer called context layer A recurrent network is one in which there is a

feedback from neuron's output to its input. The input

to the network is $x_1, x_2, x_3 — x_n$ and the output of the network is taken as $y_1, y_2, y_3 - y_n$. The output of the hidden layer (h1, h2, h3---hn) are fed back again to hidden layer neuron using context node ($c_1, c_2, c_3 — c_n$). Unlike feed forward neural networks, Recurrent Neural Networks can use their internal memory to process arbitrary sequences of inputs. The output of each hidden neuron is copied into a specific neuron in the context layer. The value of the context neuron is used as an extra input signal for all the neurons in the hidden layer one time step later. In an Elman network, the weights from the hidden layer to the context layer are set to one and are fixed because the values of the context neurons have to be copied exactly as shown in Fig. 1.

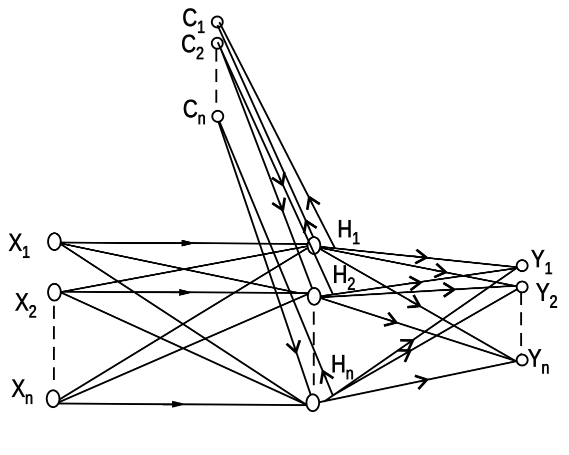

Fig. 1. ELMAN Network

## IV. DATASET DESCRIPTION

Here two types of datasets are used for analysis one is COCOMO dataset which is the historic project of nasa and Desharnais dataset, (i). COCOMO DATASETS

Dataset used for analysis and validation of the model can be got from historic projects of NASA. One set of dataset consists of 63 projects and other consists of 93 datasets. The datasets is of COCOMO II format. In our experiment 93 datasets are used for training and 63 data is used for testing.

The Dataset need for training as well as testing is available in www.promisedata.org/?p=6 and in www.promisedata.org/?p=35 . The dataset available is of COCOMO 81 format which is to be converted to COCOMO II by following the COCOMO II Model definition manual [1] and Rosetta stone [4] COCOMO 81 is converted to COCOMO II. COCOMO 81 is the earlier version developed by Barry Boehm in 1981 and COCOMO II is the next model developed by Barry Boehm in year 2000. Some of the attributes like TURN are used only in COCOMO 81 and some new attributes like RUSE, DOCU, PCON, SITE are introduced in COCOMO II.

**Table I. Effort Multipliers and their Range**

| Effort Multipliers | | Range |
|---|---|---|
| Required software Reliability | RELY | 0.82 – 1.26 |
| Data base size | DATA | 0.90 – 1.28 |
| Product Complexity | CPLX | 0.73 – 1.74 |
| Developed for Reusability | RUSE | 0.95 – 1.24 |
| Documentation Match to Life-Cycle needs | DOCU | 0.81 – 1.23 |
| Execution Time Constraints | TIME | 1.00 – 1.63 |
| Main storage Constraint | STOR | 1.00 – 1.46 |
| Platform Volatility | PVOL | 0.87 – 1.30 |
| Analyst capability | ACAP | 1.42 – 0.71 |
| Programmer capability | PCAP | 1.34 – 0.76 |
| Personal continuity | PCON | 1.29 – 0.81 |
| Applications Experience | APEX | 1.22 – 0.81 |

| Effort Multipliers | | Range |
|---|---|---|
| Platform Experience | PLEX | 1.19 – 0.85 |
| Language and Tool Experience | LTEX | 1.20 – 0.84 |
| Use of software tool | TOOL | 1.17 – 0.78 |
| Multisite Development | SITE | 1.22 – 0.80 |
| Required Development Schedule | SCED | 1.43 – 1.00 |

**Table II. Scale factors and their range**

| Scale Factor | | Range |
|---|---|---|
| Precedentedness | PREC | 0.00 – 6.20 |
| Development Flexibility | FLEX | 0.00 – 5.07 |
| Architecture/Risk Resolution | RESL | 0.00 – 7.07 |
| Team Cohesion | TEAM | 0.00 – 5.48 |
| Process Maturity | PMAT | 0.00 – 7.80 |

Every input as Effort multiplier has been tuned by following the COCOMO II model definition manual [1]. The scale factor and Effort multiplier and their range is given in Table I and II. One of such inputs RELY can be discussed below in Table III:

The Rating levels are fixed by the developer. If the failure of the software causes slight inconvenience and the corresponding rating level is very low, then the effort multiplier is fixed to be 0.82. In case of some software failure can easily be recoverable then the corresponding rating level is Nominal and rating level is fixed to be 1. If the failure of the software causes risk to human life the rating level given by the developer is very high then the effort multiplier is fixed to be 1.26

**Table III. Fixing Input attributes**

| RELY Descriptor | Rating Levels | Effort Multipliers |
|---|---|---|
| Slight Inconvenience | Very Low | 0.82 |
| Low, easily recoverable loss | Low | 0.92 |
| Moderate, easily, recoverable loss | Nominal | 1 |
| High financial | loss, High | 1.1 |
| Risk to Human life | Very High | 1.26 |

*(II) DESHARNAIS DATASET*

Deserharnais Datasets for experimentation purpose are available in the link: http://promise.site.uottawa.ca/SERepository/datasets/desharnais.arff, The dataset includes the parameters like Team expriance, Managers experience and totally eight input parameters and one effort. It totally consists of 77 projects and 62 is used for training and 15 is used for testing.

## V. EXPERIMENTATION

Neural Network Uses two set of datasets One set of dataset consists of 63 projects and other consists of 93 datasets both are from the historic projects of NASA. Here we use 93 projects for training the network and 63 projects for testing. The simulation is done in MATLAB 10b environment. Elman network the weight and bias are randomly fixed so each time there is a possibility of getting different result to avoid this problem the whole network is made to run for 50 iteration and their errors are summed up.

The network designed uses only one hidden layer and that hidden layer has eight neurons and output layer has one neuron. The hidden layer uses sigmoidal transfer function. And output layer uses linear transfer function. The above consideration is used for ELMAN network for uniformity. Input fed to the neural network is normalised using Premnmx and the output is DE normalised using postmnmx. Premnmx normalises the value between (-1 to 1)

## VI. EVALUATION CRITERIA

For evaluating the different software effort estimation models, the most widely accepted evaluation criteria are the mean magnitude of relative error (MMRE), Probability of a project having relative error less than 0.25, Root mean square of error, Mean and standard deviation of error.

The magnitude of relative error (MRE) is defined as follows

$$MRE_i = \frac{[\text{actual effort}_i - \text{predicted effort}_i]}{\text{actual effort}_i} \quad ...(1)$$

The MRE value is calculated for each observation I whose effort is predicted. The aggregation of MRE over multiple observations (N) can be achieved through the mean MMRE as follows

$$MMRE = \frac{1}{N} \sum_{i}^{N} MRE_i \quad ...(2)$$

$$PRED\,(25) = \frac{MRE \leq 0.25}{N} \quad ...(3)$$

Consider $Y$ is the neural network output and $T$ is the desired target. Then Root mean square error (RMSE) can be given by

$$\text{RMSE} = \sqrt{(y - T)^2} \quad ...(4)$$

Error can be calculated by the difference between Y and T then mean and standard deviation is calculated by calculating the mean and standard deviation of the error

***ERROR=(Y-T)*** ...(5)

The standard deviation can be calculated by Standard deviation

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)^2} \quad ...(6)$$

The **skewness** of a random variable is the ratio of its third central moment $\mu_3$ to the cube of its standard deviation $\sigma$. Skewness is denoted as $Y_1$.

$$Y_1 = \frac{\mu_3}{\sigma^3} \quad ...(7)$$

The **kurtosis** of a random variable is the ratio of its fourth central moment $\mu_4$ to the fourth power of its standard deviation $\sigma$. Kurtosis is denoted as $Y_2$. Thus

$$Y_1 = \frac{\mu_4}{\sigma^4} \quad (8)$$

## VII. RESULTS

Results of ELMAN for training and testing for Desharnais dataset is given in Table IV and the dataset for training and testing for COCOMO dataset is given in Table.V . A model which gives lower MMRE is better than the model which gives higher MMRE. A model which gives high PRED(25) is better than the model which gives lower PRED(25). Similarly the model which gives lower RMSE is better than the model which gives higher RMSE. The model which mean and standard deviation is closer to zero is better than the models which gives mean and standard deviation far away from zero.

**Table IV. Training and Testing results of Desharnais dataset**

| Performance parameters | Training | Testing |
|---|---|---|
| MMRE | 0.3392 | $4.59E-01$ |
| MSE | 6.78E+06 | $2.36E+0$ |
| RMSE | $2.60E+03$ | $4.86E+03$ |
| PRED | 43.5484 | 26.6667 |
| Mean | $4.83E+02$ | $3.44E+03$ |
| Std.Dev | $2.58E+03$ | $3.55E+03$ |
| skewness | 0.1115 | 0.5801 |
| kurtosis | 5.21 | 2.4652 |

**Table V. Training and Testing Results of COCOMO Dataset**

| Performace paramaters | Training | Testing |
|---|---|---|
| MMRE | 0.3785 | 2.745 |
| MSE | $3.21E+05$ | $2.17E+06$ |
| RMSE | $5.66E+02$ | $1.47E+03$ |
| PRED | 62.3656 | 14.2857 |
| Mean | $-55.3205$ | $-393.067$ |
| Std.Dev | $5.67E+02$ | $1.43E+03$ |
| skewness | $-4.7549$ | $-4.7468$ |
| kurtosis | 41.2888 | 27.4722 |

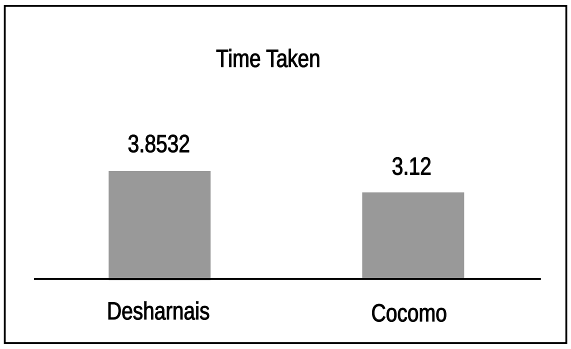

Fig. 2. Time taken for training and testing of ELMAN Network for Desharnais and COCOMO dataset.

## VIII. CONCLUSION

Most important thing in software effort prediction is its closeness to actual effort. we have analyzed the performance of both using historic dataset of NASA and Desharnais dataset in ELMAN neural network. The work can be extended by using different type of learning methods and learning algorithm. The experimentation can be further validates using some other datasets like kitchenham,Maxwell, myiazaki e.t.c

## REFERENCES

[1] Barry Boehm, COCOMO II: Model Definition Manuel. Version 2.1, Center for Software Engineering, USC,2000.

[2] Boehm B. W. “Software Engineering Economics”, Englewood Cliffs, NJ, Prentice-Hall, 1981

[3] Chikako van Koten,"Bayesian Statistical Models for Predicting Software Development Effort",The Information Science Discussion Paper Series, ISSN 1172-6024,October 2005.

[4] Donald J. Reifer, Barry W. Boehm and Sunitha chulani, “The Rosetta stone: Making COCOMO 81 Estimates work with COCOMO II”, CROSSTALK The Journal of Defence Software Engineering, pp 11 – 15, Feb.1999.

[5] FIONA WALKERDEN,ROSS JEFFERY,"An Empirical Study of Analogy-based Software Effort Estimation,Empirical Software Engineering", 4, 135–158 (1999), 1999 Kluwer Academic Publishers, Boston.

[6] Iman Attarzadeh, Siew Hock Ow, “A Novel Algorithmic Cost Estimation Model Based on Soft Computing Technique,” Journal of computer science ,pp. 117-125, 2010.

[7] Iris Fabiana de Barcelos Tronto , Jose Demisio Simo es da Silva, Nilson Sant’Anna,”An investigation of artificial neural networks based prediction systems in software project management”. The journal of system and software, June 2007, pp.356-367

[8] Jorgenson.M, “Forecasting of software development work effort: Evidence on expert judgement and formal models,” International Journal of forecasting 23 pp.449–462, 2007.

[9] Jo E. Hannay, "Better Software Effort Estimation— A Matter of Skill or Environment?",SIMULA Research Laboratory, Department of Software Engineering, Pb.134.

[10] Karunanitthi.N, D.Whitely, and Y.K.Malaiya, “Using Neural Networks in Reliability Prediction,” IEEE Software, 1992. vol.9, no.4, pp.53-59

[11] Katherine Baxter,"Understanding Software Project Estimates", Champlain College,CROSSTALK The Journal of Defense Software Engineering,March/April 2009

[12] Les Hatton,"How Accurately Do Engineers Predict Software Maintenance Tasks?", Kingston University London, IEEE computer society, 2007.

[13] Martin Shepperd And Chris Schofield, “Estimating Software Project Effort Using Analogies,” IEEE Transactions On Software Engineering, Vol. 23, No. 12, PP.736-743 November 1997.

[14] Magne Jorgenson and Martin Shepperd, “A Systematic Review of Software Development Cost Estimation Studies”, IEEE Transactions on software engineering, Vol.33, No.1,pp.33-53, January 2007

[15] Mohammad Azzeh, Daniel Neagu, Peter I. Cowling,"Analogy-based software effort estimation using Fuzzy numbers",The Journal of Systems and Software 84 (2011) 270–284

[16] Samson.B, D. Ellison, and P. Dugard, “Software Cost Estimation Using Albus Perceptron (CMAC),” Information and Software Technology, 1997,vol.39,pp.55-60.

[17] Srinivazan.K, and D. Fisher, “Machine Learning Approaches to Estimating Software Development Effort,”. IEEE Transactions on Software Engineering, February 1995, vol.21,no.2, pp. 126-137

[18] Sunitha Chulani, Barry Boehm, Bert steece,"Bayesian Analysis for Empirical software Engineering cost models", univeristy of southern california, USC-CSC-1999

[19] G.Witting, and G.Finnie, “Estimating software development effort with connectionist models,” Inf. Software Technology, 1997,vol.39, pp.369-476

[20] G.Witting, and G. Finnie, “Using Artificial Neural Networks and Function Points to Estimate 4GL Software Development Effort”, J.Information Systems,1994, vol.1, no.2, pp.87-94.