Cost prediction using Neural Network Learning Techniques

Jitendra[1], Vikas[2], Kuldeep[3], Samiksha[4]

[1] **Asst. Professor, RIMT, Chidana**
jitendrakumar.03@gmail.com

[2]**Asst. Professor, NCCE, Israna**
beniwal_vikas23@rediffmail.com

[3]**Asst. Professor, RIMT, Chidana**
malik.4084@gmail.com

[4]**Asst. Professor, HIT, Asodha**
ersam_mehta@yahoo.co.in

ABSTRACT

The continuous hardware and software development, jointly with the world economical interaction phenomenon has contributed to the competitiveness increase between producing and delivering companies of software product and services. Cost prediction is the process of estimating the effort required to develop a software system. Cost prediction is an important part of software development. In recent years, software has become the most expensive component of computer system. The bulk of the cost of software development is due to the human effort and most cost estimation methods focus on this aspect and give estimate in terms of person –months. In this paper, we will discuss how we can predict the cost using Neural Network learning techniques.

Keywords: *Algorithmic models, Cost Prediction, SLOC.*

I. INTRODUCTION

Cost prediction is the process of estimating the effort required to develop a software system. Accurate cost prediction is important because:

- ➢ It can help to classify and prioritize development projects with respect to an overall business plan.
- ➢ It can be used to determine what resources to commit to the project and how well these resources will be used.
- ➢ It can be used to assess the impact of changes and support re-planning.
- ➢ Projects can be easier to manage and control when resources are better matched to real needs.
- ➢ Customers expect actual development costs to be in line with estimated costs.

Software cost prediction involves determination of effort, time and cost. Effort is measured in person-months of the programmers, analysts etc.. This effort estimate can be converted into cost by calculating an average salary per unit time of the staff involved, and then multiplying this by the estimated effort required.
Accurate estimation of software development effort has major implications for the management of software development. If management's estimate is too low, then the software development team will be under considerable pressure to finish the product quickly, and hence the resulting software may not be fully functional or tested. Thus, the product may contain residual errors that need to be corrected during a later part of the software life cycle, in which the cost of corrective maintenance is greater. On the other hand, if a manager's estimate is too high, then too many resources will be committed to the project. Furthermore, if the company is engaged in contract software development, then too high an estimate may fail to secure a contract.

Algorithmic models

Algorithmic models (AM) "calibrate" prespecified formulas for estimating development effort from historical data. Inputs to these models may include the experience of the development team, the required reliability of the software, the programming language in which the software is to be written, and an estimate of the final number of delivered source lines of code (SLOC). Basically algorithmic model is formula based model which takes historical cost information and which is based on the size of the software. Algorithmic model includes two most popular models used as follows:

a) COCOMO Model
b) Putnam's Model and SLIM

II. LITERATURE

2.1 Neural Network Learning Technique

Learning is a ability that is very similar to adaptation. Learning through adaptation to the external environment conditions, results in behavioral alterations and performance improvement of the learning system.
According to Britannica Dictionary learning refers to "the modification of behavior following upon and induced by interaction with the environment and as a result of experiences leading to the establishment of new pattern of response to external stimuli".
In the case of ANNs the meaning of learning can be given as "a process by which the free parameters of a neural network are adapted through a process of stimulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the parameter changes take place".
Learning is the process by which a neural network modifies its internal structure in response to external stimuli. According to this definition learning procedure of a neural network can be described by the three events: the stimulation of the networks by some environmental input, the adjustment of network's free parameters as a result of this stimulation (adaptation to the external stimuli), and finally exhibition of an altered response to the environment due to the changes that occurred in its internal structure and function.
The prescribed set of well-defined rules that specifies these structural modifications in the network for the solution of a learning problem is called a learning algorithm, learning law, or learning rule. Due to wide variation in nature and requirements of specific applications in biological systems there is a requirement of learning process to solve each type of problem. So there is diverse variety of learning algorithms for the design of neural networks.

Learning methods can be broadly classified into two basic types:

2.1.1 Supervised Learning Method

In the supervised learning a teacher is required to grades the performance of the neural network during learning. In this learning method a teacher may be a set of data, a part of them to play the role of the external inputs (stimuli) to the networks, and the rest to correspond to the desired outputs for that inputs.
By giving the number of layers, the number of neurons per layer of the network and the type of activation function used, the synaptic weights, which were set in the starting, are then adjusted so that at the next iteration the output produced by the network be closer to the desired output. The goal of the learning process is to minimize the error between the desired output and the actual output produced by the network. At the termination of the learning process, the neural network has learned to produce an output that closely matches to the desired output. Then the network's structure is fixed and becomes operational, ready to fulfill its objective. Supervised learning method has following learning algorithms:

Gradient Descent Learning

This algorithm tries to minimize the error E between actual and desired output by adjusting the synaptic weights by an amount proportional to the first derivative of the mean squared error with respect to the synaptic weight.
Thus if ΔW_{ij} is the weight update of the link connecting the ith and jth neuron of the two neighboring layers, then ΔW_{ij} is defined as

$$\mathbf{\Delta W_{ij} = \eta\, \partial E/\partial W_{ij}}$$

Where, η is the learning rate parameter and $\partial E/\partial W_{ij}$ is the error gradient with reference to the weight W_{ij}.

Delta Rule
Delta rule is the special case of Gradient Descent Learning. Delta rule is also referred as the Widrow-Hoff Learning Rule. According to this learning rule the mechanism for synaptic modification during the training process acts in an appropriate way in order to reduce or minimize the difference between the desired output and the actual output produced by the processing elements. It is also called the Least Mean Square Learning Rule, because it tries to minimize the mean squared error of that difference.

Back-propagation Learning rule
Back-propagation learning (BPL) algorithm was invented in 1969 for learning in multilayer network. The back-propagation algorithm trains a given feed-forward multilayer neural network for a given set of input patterns with known classifications. When each entry of the sample set is presented to the network, the network examines its output response to the sample input pattern. The output response is then compared to the known and desired output and the error value is calculated. Based on the error, the connection weights are adjusted. The back-propagation algorithm is based on *Widrow-Hoff delta learning rule* in which the weight adjustment is done through *mean square error* of the output response to the sample input .The set of these sample patterns are

repeatedly presented to the network until the error value is minimized. The back-propagation neural network is shown in figure 1 As shown in figure it has one input layer, one hidden layer and one output layer. Input signals transmitted from input to hidden and hidden to output layer and error signal from output to hidden and hidden to input.

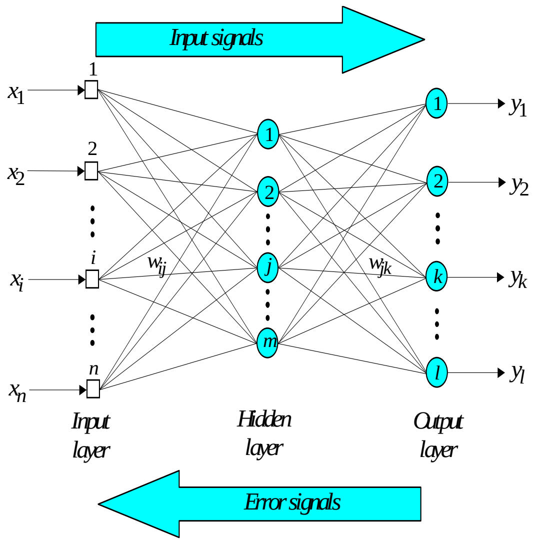

Figure 1 Three layer back-propagation neural network

Back-propagation learning algorithm uses training data to adjust the weights and threshold of neurons so as to minimize the error. It is based on the differences between the actual and the desired output. It works by applying the gradient descent rule to feed-forward network. The algorithm involves two phases, the forward phase that occurs when the inputs (external stimuli) are presented to the neurons of the input layer and are propagated forward to compute the output and the backward phase, when the algorithm performs modifications in the backward direction.

Steps of the algorithms are the following:

Step 1: Initialize weights with small, random values

Step 2: While stopping condition 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.

Cascade Correlation Learning

Cascade-correlation (CC) is an architecture and generative, feed-forward, supervised learning algorithm for artificial neural networks. Cascade-Correlation begins with a minimal network, then automatically trains and adds new hidden units one by one creating a multi-layer structure.
Cascade-Correlation (CC) combines two ideas:

- The first is the cascade architecture, in which hidden units are fixed which do not change once added.
- The second is the learning algorithm, which creates and installs the new hidden units. To install new hidden unit, the algorithm maximize the magnitude of the correlation between the new unit's output and the residual error signal of the network.

Steps of the algorithms are following:

Step 1: CC starts with a minimal network consisting only of an input and an output layer. Both layers are fully connected.

Step 2: Train all the connections ending at an output unit with a usual learning algorithm until the error of the net no longer decreases.

Step 3: Generate the so-called candidate units. Every candidate unit is connected with all input units and with all existing hidden units. Between the pool of candidate units and the output units there are no weights.

Step 4: Try to maximize the correlation between the activation of the candidate units and the residual error of the net by training all the links leading to a candidate unit. Learning takes place with an ordinary learning algorithm. The training is stopped when the correlation scores no longer improves.

Step 5: Choose the candidate unit with the maximum correlation, freeze its incoming weights and add it to the net. To change the candidate unit into a hidden unit, generate links between the selected unit and all the output units. Since the weights leading to the new hidden unit are frozen, a new permanent feature detector is obtained. Loop back to step 2.

Step 6: This algorithm is repeated until the overall error of the net falls below a given value.

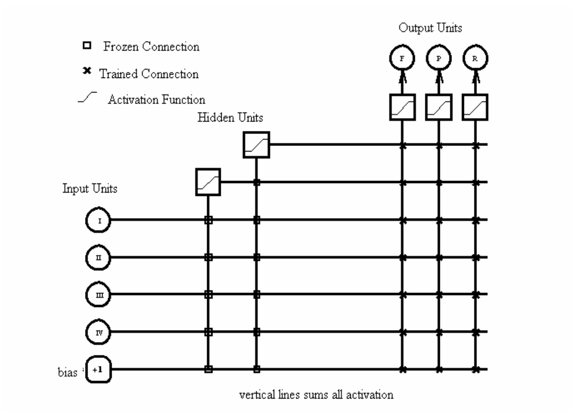

Figure 2 A neural networks trained with Cascade-correlation learning algorithm

2.1.2 Unsupervised Learning Method

In the unsupervised learning method, the target output is not presented to the network.. So the network follows a self-supervised method and makes no use of external influences for synaptic weight modification. Instead of an external teacher, there is an internal monitoring of the network's performance that makes adaptations in the input signals according to the function of the network. Unsupervised learning method has following algorithms:

Hebbian Learning
This algorithm was proposed by Hebb in 1949 and is based on correlation weight adjustment. This is the mechanism inspired by biology. According to Hebb "when an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic changes take place in one or both cells such that A's efficiency as one of the cells firing B, is increased". This basic rule simple states that if a processing element receives an input from another processing element, a local and strongly interactive mechanism modifies the synaptic efficiency according to the correlation of the presynaptic and postsynaptic activities. This type of synapse is referred as Hebbian synapse.

Competitive Learning
Competitive learning law, is inspired by learning in biological systems According to this algorithm the processing elements compete for the opportunity of learning. The processing element with the largest output has the capability of inhibiting its competitors as well as exciting its neighbors. Only the winner is permitted an output and only the winner plus its neighbors are permitted to adjust their weights (winner-takes-all).

Reinforced Learning Method
In reinforced learning method, a teacher though available but does not present the expected output but only indicates if the computed output is correct or not. The information provided helps the network in its learning process. A reward is given for a correct answer computed and a penalty for a wring answer.

III RESULTS AND ANALYSIS

The results for cost prediction using neural network learning techniques is presented and discussed. For this purpose neural network tool of MATLAB is used. Values using COCOMO Model is also predicted using Basic COCOMO Model.

3.1 Experimental Setup

Data
Results in neural networks will be calculated by taking historical data of 50 projects which is divided into three parts: 20 projects data for training the network, 10 projects for validating the network and 10 projects for testing the 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 3.1 Data used for training 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 3.2 Data used for validating 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	98.70
47	130	673.7
48	165	246.9
49	200	130.3
50	214	86.9

Table 3.3 Data used for testing the Neural Network

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

Parameters	Back-propagation Learning Algorithms	Cascade Correlation Learning Algorithms
Network Type	Feed-forward backprop	Cascade-forward backprop
Training function	TRAINLM	TRAINLM
Performance function	MSE	MSE
Number of neurons	20	20
Transfer function	LOGSIG	LOGSIG
No. of epochs	46	46

Table 3.4 Parameters for Neural Networks learning algorithms in Neural Network Tool of MATLAB

3.2 Evaluations of Results

In this section we will analyze the results of neural network learning algorithms i.e. Back-propagation learning algorithms, Cascade Correlation learning algorithms and COCOMO Model of software engineering. Comparison of these cost prediction techniques will be done on the basis of results evaluated and then values of RMSE and MMRE will be calculated and compared.

3.2.1 Performance of Effort between COCOMO Model and Back-propagation Learning Algorithms.

In this section effort using COCOMO Model and Back-propagation learning algorithms will be compared. Values are given in the following table.

Table 3.5 Comparison between COCOMO Model and Back-propagation learning algorithms

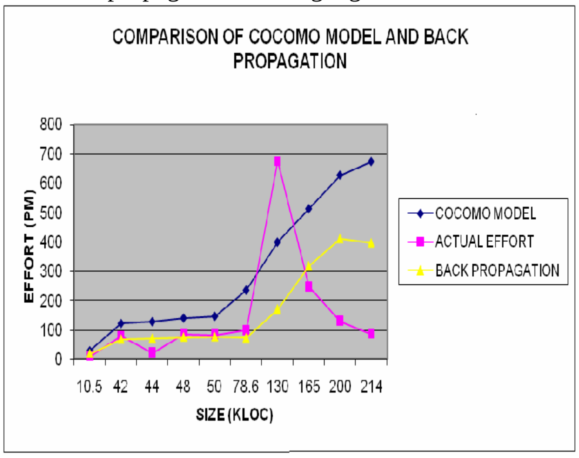

Figure 3.1 Comparison of COCOMO Model and Back-propagation learning algorithms

Above shown graph indicates that the effort calculated using Back-propagation is nearest to the actual effort as compared to the COCOMO Model.

3.2.2 Performance of Effort between COCOMO Model and Cascade Correlation learning Algorithm.

In this section effort using COCOMO Model and Cascade Correlation learning algorithm will be compared. Values are given in the following table.

Project No.	Size (KLOC)	Actual effort	COCOMO Model	Cascade Correlation
41	10.5	10.3	28.3	16.6
42	42	78.9	121.5	79.13
43	44	23.2	127.6	83.45
44	48	84.9	139.8	81.84
45	50	84	145.9	80.59
46	78.6	98.7	234.6	84.26
47	130	673.7	398	267.9
48	165	246.9	511.2	245.96
49	200	130.3	625.6	223.66
50	214	86.9	671.6	214.73

Project No.	Size (KLOC)	Actual effort	COCOMO Model	Back-propagationLearning Algorithms
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

Table 3.6 Comparison between COCOMO Model and Cascade Correlation learning Algorithm

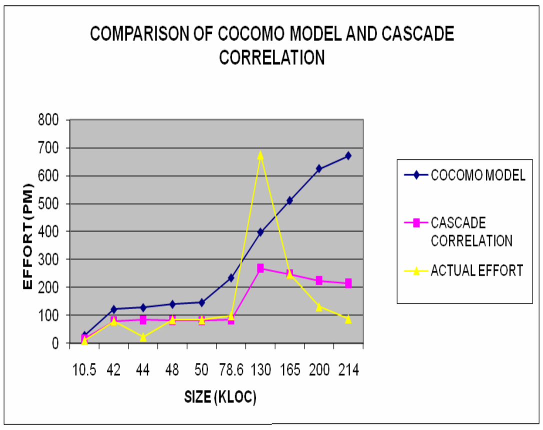

Figure 3.2 Comparison of COCOMO Model and Cascade Correlation learning algorithm Above shown graph indicates that effort calculated using Cascade Correlation learning algorithm is nearest to the actual effort as compared to the COCOMO Model.

3.2.3 Performance of Effort between Back-propagation and Cascade Correlation learning algorithms

In this section effort using COCOMO Model and Cascade Correlation learning algorithms will be compared. Values are given in the following table.

Project No.	Size (KLOC)	Actual effort	Back-propagation	Cascade Correlation
41	10.5	10.3	18.5	16.6

42	42	78.9	67.83	79.13
43	44	23.2	70.89	83.45
44	48	84.9	74.95	81.84
45	50	84	76.01	80.59
46	78.6	98.7	72.301	84.26
47	130	673.7	170.72	267.9
48	165	246.9	317.97	245.96
49	200	130.3	410.79	223.66
50	214	86.9	396	214.73

Table 3.7 Comparison between Back-propagation learning algorithm and Cascade Correlation learning algorithm

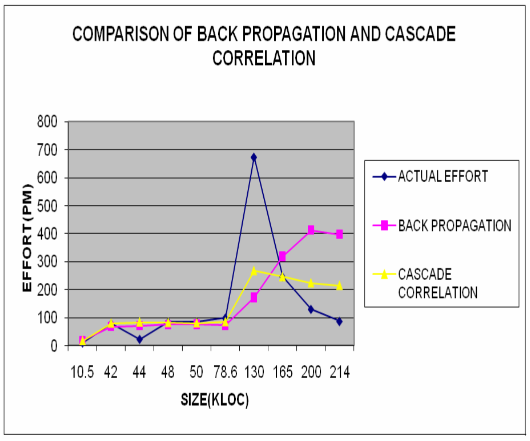

Figure 3.3 Comparison of Back-propagation and Cascade Correlation learning algorithms

Above shown graph indicates that effort calculated using Cascade Correlation learning algorithm is more nearest to the actual effort.

3.2.4 Comparison of Different Cost Prediction Techniques

In this section effort using COCOMO Model and Neural Network learning algorithms will be compared. Values are given in the following table.

Project No.	Size (KLOC)	Actual Effort	COCOMO Model	Back-Propagation	Cascade Correlation
41	10.5	10.3	28.3	18.5	16.6
42	42	78.9	121.5	67.83	79.13
43	44	23.2	127.6	70.89	83.45
44	48	84.9	139.8	74.95	81.84
45	50	84	145.9	76.01	80.59
46	78.6	98.7	234.6	72.301	84.26
47	130	673.7	398	170.72	267.9
48	165	246.9	511.2	317.97	245.96
49	200	130.3	625.6	410.79	223.66
50	214	86.9	671.6	396	214.73

Table 3.8 Comparison between COCOMO Model, Back-propagation learning algorithm and Cascade Correlation learning algorithm

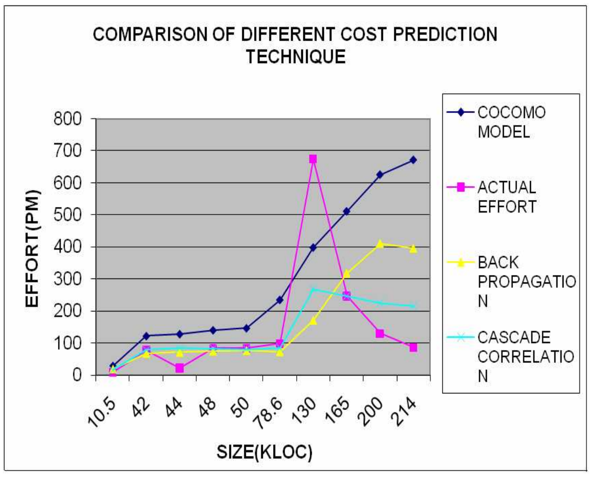

Figure 3.4 Comparison of COCOMO Model, Back-propagtion and Cascade Correlation learning algorithms

Above graph shows that effort calculated using Cascade Correlation learning algorithm is nearest to the actual effort as compared to other cost prediction techniques.

3.2.5 Error Prediction of Different Cost Prediction Techniques

In this section error will be calculated of different Cost Prediction Techniques.

To calculate the error following formule will be used:

$$E_{rror} = | E - E' |$$

Where

E_{rror} is the output error,
E is the actual effort,

Performance Criteria	**COCOMO Model**	**Back-propagation**	**Cascade Correlation**
RMSE	277.74	208.71	139.15
MMRE	2.1557	1.1072	0.6228

E’ is the estimated effort
In this section
$E_1 = | E - E'_{coc} |$
$E_2 = | E - E'_{back} |$
$E_3 = | E - E'_{casc} |$

Where
E is the actual effort,
E_1 error of COCOMO Model
E'_{coc} is the estimated effort using COCOMO Model
E_2 error of Back-propagation learning algorithm
E'_{back} is the estimated effort using Back-propagation learning
E_3 error of Cascade Correlation learning algorithm
E'_{casc} is the estimated effort using Cascade Correlation learning

Project No.	**Size(KLOC)**	**Actual Effort**	**COCOMO Model (E_1)**	**Back-propagation(E_2)**	**Cascade Correlation(E_3)**
41	10.5	10.3	18	8.20	6.2923
42	42	78.9	42.6	11.07	0.22766
43	44	23.2	104.4	47.70	60.2523
44	48	84.9	54.9	9.95	3.0598
45	50	84	61.9	7.99	3.4101
46	78.6	98.7	135.9	26.40	14.14
47	130	673.7	275.7	502.98	405.8181
48	165	246.9	264.3	71.07	0.94078
49	200	130.3	495.3	280.49	93.3566
50	214	86.9	584.7	309.10	127.8354
Mean			203.8	127.5	**71.5**

Table 3.9 Output error of different cost prediction techniques

These error values will be used to calculate the RMSE and MMRE as follows:

$$RMSE = (\sqrt{((1/N)\ (\sum_{i=1}^{N}((E_{rror})_i)^2))}\ )$$

$$MMRE = ((1/N)\ (\sum_{i=1}^{N} |(E_{rror})_i| / E_i))$$

Where N is total number of projects

Table 3.10 Perforamnce Evaluation of COCOMO Model, Back-propagation and Cascade Correlation learning algorithms

Following graphs shows the comparison using RMSE(Root Mean Square Error) of different cost prediction techniques.

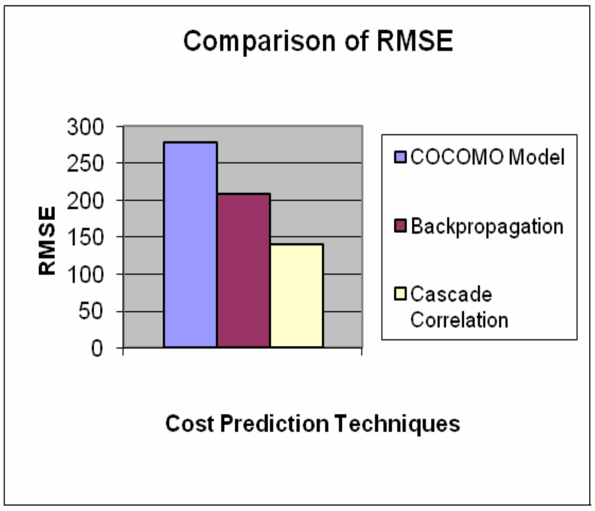

Figure 3.5 Comparison using RMSE
Following graphs shows the comparison using MMRE(Mean Magnitude Relative Error) of different cost prediction techniques.

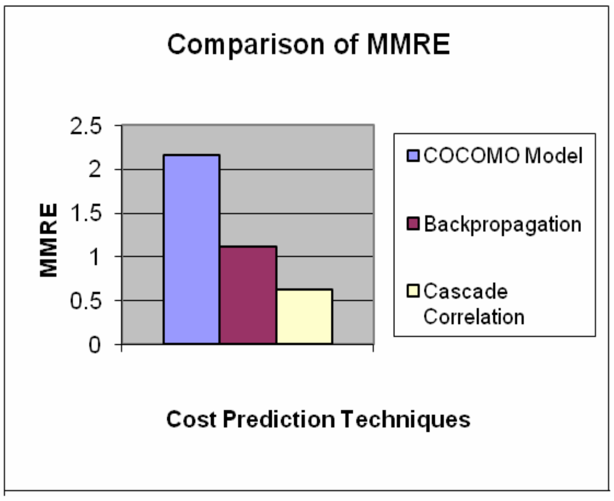

Figure 3.6 Comparison using MMRE

Comparison using RMSE and MMRE shows that Cascade Correlation is better than Back-propagtion and COCOMO Model. Accuracy is high in the Cascade Correlation.

IV. CONCLUSION

Today, almost no model can estimate the cost of software with a high degree of accuracy. This state of the practice is created because:

- There are a large number of interrelated factors that influence the software development process of a given development team and a large number of project attributes, such as number of user screens, volatility of system requirements and the use of reusable software components.
- The development environment is evolving continuously.
- The lack of measurement that truly reflects the complexity of a software system.

To produce a better estimate, we must improve our understanding of these project attributes and their causal relationships, model the impact of evolving environment, and develop effective ways of measuring software complexity.
We require estimating the cost at the initial stage of a project, but there is high uncertainty about these project attributes at initial stage. That's why the estimate produced at this stage is inevitably inaccurate, as the accuracy depends highly on the amount of reliable information available to the estimator.
Here three most popular approaches were suggested to predict the software cost estimation. In one hand COCOMO which has been already proven and successfully applied in the software cost estimation field and in other hand the Back-propagation learning algorithm and cascade-correlation learning algorithm in Neural Network that has been extensively used in lieu of COCOMO estimation and have demonstrated their strength in predicting problem. To get accurate results the neural network depends only on adjustments of weights from hidden layer of network to output layer of neural network. We have used the 50 projects data set to validate, train and test the network. After testing the network it is concluded that learning algorithms of neural network perform better then the COCOMO model and from learning algorithms Cascade correlation performs better then the Back-propagation learning algorithm. It has less error values, so accuracy is high in Cascade Correlation

V. FUTURE WORK

Future work on these topics should include using a cost estimation data set for which the amount of data available is not a constraint. A neural network trained on this data set would provide a more reliable estimate of a neural network's ability to produce a quality cost estimate.
In order to gain further insight to the black box nature of the neural network cost estimate, decision trees could be examined. The decision trees could show the point at which an attribute's value changes the software's cost, and what attributes have the largest impact on the software's cost.
The data creation method could also be examined in order to improve its usefulness. The method as currently constituted provides more noise in the created data than is desired. By introducing more constraints and precedent relationships in the data creation process, the amount of noise present would be reduced. This would allow the data creation process to be used on networks that achieved a good result value when using their base data set

VI. REFERENCES

[1] Clark, B., Chulani, S. and Boehm, B. (1998), "Calibrating the COCOMO II Post Architecture Model," International Conference on Software Engineering, Apr. 1998.

[2] Christos Stergiou and Dimitrios Siganos, “Neural Networks”

[3] Christopher M. Fraser(2000), “Neural Networks: A Review from a Statistical Perspective”, Hayward Statistics

[4] Nasser Tadayon, “Neural Network Approach for Software Cost Estimation”, IEEE proceedings of the International Conference on Information Technology: Coding and Computing(ITCC ’2005)

[5] S. Kanmani, J. Kathiravan, S. Senthil Kumar and M. Shanmugam, “Neural Networks Based Effort Estimation using Class Points for OO Systems”,IEEE proceedings of the International Conference on Computing: Theory and Applications(ICCTA’2007)

[6] Ch. Satyananda Reddy, P. Sankara Rao, KVSVN Raju, V. Valli Kumari, “A New Approach For Estimating Software Effort Using RBFN Network” , IJCSNS International Journal of Computer Science and Network Security, VOL.8 No. 7, July 2008

[7] Kiyoshi Kawaguchi,” Backpropagation Learning Algorithm”, Wikipedia.org, June 2000.

[8] Cascade Correlation Architecture and Learning Algorithms for Neural Networks”, Wikipedia.org, Nov. 1995
[9] Konstantinos Adamopoulos,” Application of Back Propagation Learning Algorithms on Multilayer Perceptrons”, Department of Computing May 2000.

[10] K. Srinivasan and D. Fisher, “Machine learning approaches to estimating software development effort”, *IEEE Trans. Soft. Eng*., vol.21, no.2, Feb. 1995, pp. 126-137.

[11] Hareton Leung, Zhang Fan, ”Software Cost Estimation”, Department of Computing, Hong Kong, 1999.