A New Approach For Estimating Software Effort Using RBFN Network

Ch. Satyananda Reddy, P. Sankara Rao, KVSVN Raju, V. Valli Kumari

Department of Computer Science and Systems Engineering, College of Engineering,
Andhra University, Visakhaptnam-530003, India.

Summary
The prediction of software development effort has been focused mostly on the accuracy comparison of algorithmic models rather than on the suitability of the approach for building software effort prediction systems. Several estimation techniques have been developed to predict the Effort estimation. In this paper the main focus is on investigating the accuracy of the prediction of effort using RBFN network which can be used for functional approximation. The use of RBFN to estimate software development effort requires the determination of its architecture parameters according to the characteristics of COCOMO, especially the number of input neurons, no of hidden neurons, centers c_i, width σ_1 and weight w_i. In the aspect of learning, the RBFN network is much faster than other network because the learning process in this network has two stages and both stages can be made efficient by appropriate learning algorithms. The proposed network is empirically validated using COCOMO'81 dataset which is used to train and test the designed RBFN network and found that the RBFN designed with the K-means clustering algorithm performs better, in terms cost estimation accuracy.

Key words:
Radial Basis Function Neural Network, K-means clustering algorithm, Effort, COCOMO.

1. Introduction

Effort estimation consists the prediction of how many hours of work and how many workers are needed to develop a project. Predicting software development effort with high precision is still a great challenge for project managers. The majority of software cost estimation techniques falling into three general categories [9] includes Algorithmic Models (AM) [11] and Expert Judgment (EJ)[10], and Machine Learning(ML) [11] techniques. Among estimation techniques, COCOMO (**Co**nstructive **Co**st **Mo**del) is the most commonly used algorithmic cost modeling technique because of its simplicity for estimating the effort in person-month for a project at different stages. COCOMO uses the mathematical formulae to predict project cost estimation.

An RBFN architecture configured for software development effort is a three-layer feed forward network consisting of one input layer, one middle layer and an output layer. The RBFN generates output (effort) by propagating the initial inputs (cost drivers) through the middle-layer to the final output layer. The activation functions of each middle layer neurons are usually the Gaussian function. The output layer consists of one output neuron that computes the software development effort as a linear weighted sum of the outputs of the middle layer. Clustering has been often exercised as a preprocessing phase used in the design of the RBF neural networks. The primary aim of this algorithm is to set up an initial distribution of the respective fields (hidden neurons) across the space of the input variables. In particular, this implies a location of the nodal values of these fields (e.g., the nodes of Gaussian function).

K-means clustering algorithm has many successful applications such as pattern recognition and data compression. It is a multi-pass and time-consuming clustering algorithm. The research method of an RBFN for COCOMO and the use of RBFN to estimate software development effort require the determination of its architecture parameters according to the charactestics of COCOMO.

In this paper, the main focus is on investigating the accuracy of the prediction using RBFN network. The aim of this study is to verify if RBFN network can be used for prediction of effort on the basis of effort multipliers, size of the project, scale factor used in project development. To empirically evaluate the training algorithms and to find which training algorithm is suitable for the estimation purpose.

This paper is composed of 6 sections. Section 2 briefly describes the related work done for estimating the

Manuscript received July 5, 2008.
Manuscript revised July 20, 2008.

effort through different neural network approaches. Section 3 gives an overview of proposed RBFN network for COCOMO. Section 4 presents experimental design and application of RBFN network structure based on K-means for the dataset of COCOMO. Section 5 summarizes the results obtained by using RBFN approach discussed in earlier sections. The final section concludes that RBFN network is suited for calibrating COCOMO model.

2. Related Work

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.

Witting and Finnie [4], [6] describe their use of back propagation learning algorithms on a multilayer perceptron in order to predict development effort. The study of Karunanithi [3] reports the use of neural networks for predicting software reliability; including experiments with both feed forward and Jordon networks. The study of Samson [1] uses an Albus multiplayer perceptron in order to predict software effort. They use Boehm's COCOMO dataset. Srinivazan and Fisher [2] also report the use of a neural network with a back propagation learning algorithm. However it is not clear how the dataset was divided for training and validation purposes. Khoshgoftaar [5] presented a case study considering real time software to predict the testability of each module from source code static measures. They consider ANNs as promising techniques to build predictive models. Finally in the last years, a great interest on the use of ANNs has grown. ANNs have been successfully applied to several problem domains. They can be used as predictive models because they are modeling techniques capable of modeling complex functions.

3. Research Methodology

In this work, the Radial Basis Neural Network method is used to predict software development effort (in person month) using popular algorithmic method called COCOMO II.

$$\text{Effort} \;=\; A..\,(\text{Size})^{SF}\,.\,\Pi\ EM$$

Where A is constant factor that depends on local organization practices and the type of software that is developed, size is the assessment of the code size of the software, SF is usually lies between 1 and 1.5,and EM is a effort multiplier made by combining process product and development attributes. A complete description of COCOMO II and its predecessors is given by Boehm [7] or in the COCOMO II model definition manual [12].

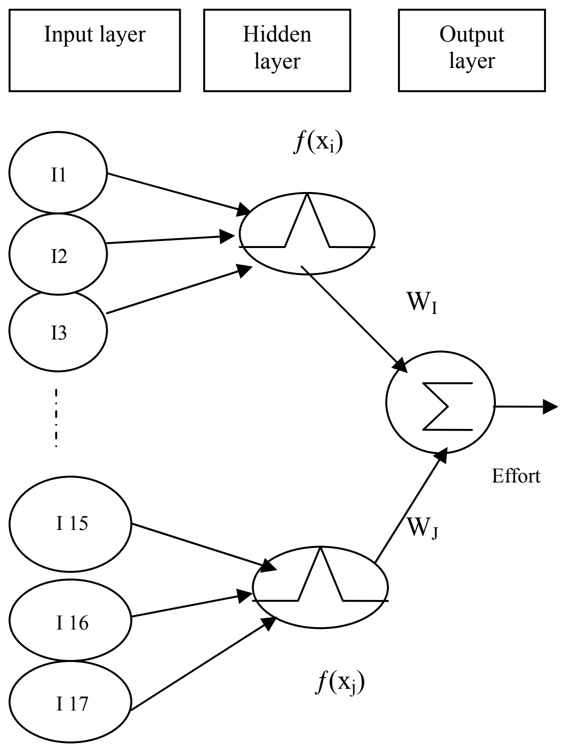
3.1. Radial Basis function Network

An RBFN architecture configured for software development effort is a three-layer feed forward network consisting of one input layer, one middle layer and an output layer. The RBFN generates output (effort) by propagating the initial inputs (cost drivers) through the middle-layer to the final output layer. Each input neuron corresponds to a component of an input vector. The input-layer contains M neurons, plus, eventually, one bias neuron. Each input neuron is fully connected to the middle-layer neurons. The activation function of each middle neuron is usually the Gaussian function. The Gaussian function decreases rapidly if the width σ_i is small, and slowly if it is large. The output layer consists of one output neuron that computes the software development effort as a linear weighted sum of the outputs of the middle layer:

3.2. Proposed Architecture

The RBFN network can be used for both classification and functional approximation. This method is mainly focused on functional approximation. In the aspect of learning, the RBFN network is much faster than other network [8]. The primary reason for this is that the learning process in RBFN network has two stages and both stages can be made efficient by appropriate learning algorithms. Although finding the optimal k clusters for a set of data is NP-complete, k-means is quite efficient and generally produces good results. Especially their middle layer composed of receptive fields, using k-means clustering technique. The use of on RBFN to estimate software development effort requires the determination of its architecture parameters according to the characteristics of COCOMO, especially the number of input neurons no of hidden neurons. Centers c_i, width σ_i and weight w_i and the use of some clustering technique to analyze and find clusters in the training data. The results of this grouping are establishing prototypes of the receptive fields.

The proposed RBFN network model is shown in Fig 3. All the values of size and effort multipliers used in COCOMO are preprocessed to natural logarithms and they are used as inputs to the hidden layer. The activation function in the hidden layers is Gaussian function represented by $f(x) \;=\; e\text{-}\,(x^2)$.



INPUTLAYER
I1-Bias, I2-ln(EM1), I3-ln(EM2), I4-ln(EM3), I5-ln(EM4), I6- ln(EM5), I7- ln(EM6), I8- ln(EM7), I9-ln(EM8), I10-ln(EM9), I11-ln(EM10), I12-ln(EM11), I13-ln(EM12),I14-ln(EM13),I15-ln(EM14),I16- ln(EM15), I17- Ln(Size)*SF.

HIDDEN LAYER

$$f\,(x) = e^{-}\left(\frac{\| x - c_i \|^2}{\sigma_i^2}\right)$$

OUTPUT LAYER:

$$\Sigma\, f(x_i)W_i$$

Fig 3. RBFN Network

The weights W_i, W_j denotes the weights of the arcs from the hidden layers nodes to the output node. Note that the initial values of W_i, Wj are adjusted so that the nodes in the hidden layer have same contribution to the output node effort. Using the above neural network with the clustering algorithm, initially form different Clusters and output is calculated for each cluster using Gaussian function with the set of known initial weights.

The actual effort is compared against the estimated value. Assume for a certain set of inputs the estimated output of the network as Effort_e. Furthermore Effort_a denotes the actual value of effort after the project is finished. The error is estimation is determined by

$\text{Err}_{\text{Effort}} = \text{Effort}_a - \text{Effort}_e$ which is used to determine the error of nodes in the hidden layer.

In the output layer:

1: Adjust weight by

$$W_i\ (\text{new}) = W_i\ (\text{old}) + \Delta W_i$$

2: Weight change is compute by

$$\Delta W_i = \delta .\ \text{Err}_{\text{Effort.}}\ f\ (x_i)$$

Where δ= learning rate

$$\text{Err}_{\text{Effort}} = \text{Effort}_a - \text{Effort}_e$$

$f(x_i)$ = output of Activation function in hidden layer

3: Repeat iteration until Convergence.

4. Experimental Design

The neural network used as RBFN with a single hidden layer using the neural network tool of MATLAB. The neurons in the hidden layer were depending upon the K-means clustering technique. Thus there are seventeen nodes in the input layer, clustering neurons in the hidden layer and one node in the output layer. The MATLAB adaptation learning function selected for this experiment was 'radbas', the performance function used was sum squared error (SSE). Transfer functions used were, radialbasis function in the hidden and purelinear function in the output layer. After training, testing was done on the network using the data set of COCOMO'81 and the output obtained were compared with the target values. The objectives of the experiments were:

- To verify if neural networks can be used for prediction of Effort on the basis of effort multipliers, size of the project, scale factor used in project development.
- To empirically evaluate the training algorithms and to find which training algorithm is suitable for the estimation purpose.

- The experiment was conducted using the advanced RBFN.

The output of the first layer for a feed forward network net can be obtained with the following code:

a = radbas (netprod (dist (net.X, Ci), net.σi))

The details of designing this network are built using functions newrbe and newrb, and their outputs can be obtained with newrb function to create a radial basis network that approximates a function defined by a set of data points. The weights and biases of each neuron in the hidden layer define the position and width of a radial basis function. Each linear output neuron forms a weighted sum of these radial basis functions. With the correct weight and bias values for each layer, and enough hidden neurons, a radial basis network can fit any function with any desired accuracy. The function NEWRB creates a radial basis network which approximates the function defined by P and T. In addition to the training set and targets, NEWRB takes two arguments, the sum-squared error goal and the spread constant.

newrb (P,T,GOAL,SPREAD)

5. Experimental Results

This section presents the results obtained when applying the RBFN to the artificial COCOMO'81 dataset. These experiments use the full COCOMO'81 dataset for training and testing. The effort value is 1597 when the input values substituted in COCOMO person-month formulae and the output value is 1537 when the input values implemented in MATLAB program. The function newrb iteratively creates a radial basis network one neuron at a time. Neurons are added to the network until the sum-squared error 0.5 falls beneath an error goal or a maximum number of neurons have been reached. The call for this function is:

net = newrb (P, T,GOAL,SPREAD)

The function newrb takes matrices of input and target vectors, P and T, and design parameters GOAL and, SPREAD, and returns the desired network graph.

Table.5.1 shows the comparison of effort estimation between COCOMO and RBFN network with actual effort. The effort obtained through RBFN network is very close to the actual effort when compared with the effort calculated through COCOMO.

Table 5.1 Effort Estimation Results

P.Id	Actual Effort	Effort using COCOMO	Effort using RBFN network
1	2040	2395	1983
2	1075	810	1023
3	243	246	240
4	218	298	203
5	1600	2776	1592

The mapping between input values when applied radial basis function and the target range of given input values forms the radial basis function graph by connecting clustering nodes is plotted in Fig 5. Simulate the network response for inputs over the same range.

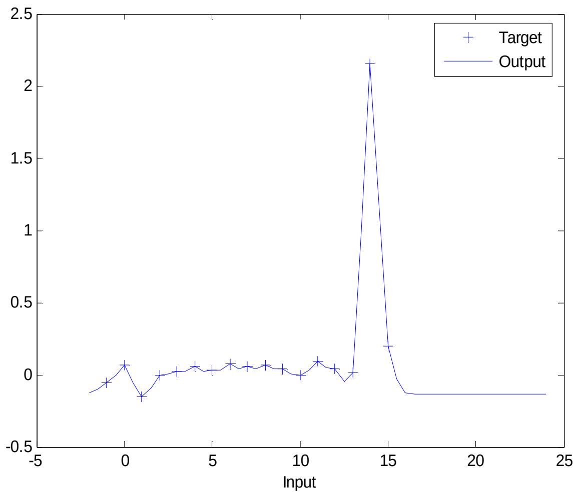


Fig .5. The Relationship between the accuracy of Radial Basis Function and K-means clustering.

6. Conclusion and Future work

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 Radial Basis Function Neural Network that has been extensively used in lieu of COCOMO estimation and have demonstrated their strength in predicting problem. Further more, the K-means clustering algorithm is used to predict the mean value which is useful in Gaussian function for network. Note that the initial values of $W_{i,}$ W_j are adjusted so that the nodes in the hidden layer have same contribution to the output node effort. To get accurate results the proposed network depends only on adjustments of weights from hidden layer of network to output layer of RBFN network. The proposed network is empirically validated using COCOMO'81 dataset which is used to train and test the designed RBFN network and found that the RBFN designed with the K-means clustering algorithm performs better, in terms cost estimation accuracy. This work can be extended by integrating with Fuzzy C-means clustering algorithm.

References

[1] B. Samson, D. Ellison, and P. Dugard, "Software Cost Estimation Using Albus Perceptron (CMAC)," Information and Software Technology, 1997,vol.39,pp.55-60.
[2] K. Srinivazan, 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.
[3] N. Karunanitthi, D.Whitely, and Y.K.Malaiya, "Using Neural Networks in Reliability Prediction," IEEE Software, 1992. vol.9, no.4, pp.53-59.
[4] 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.
[5] T.M.Khoshgoftaar, E.B.Allen, and Z.Xu, "Predicting testability of program modules using a neural network," Proc.3rd IEEE Symposium on Application-Specific Systems and Sof. Eng. Technology, 2000,pp.57-62.
[6] G.Witting, and G.Finnie, "Estimating software development effort with connectionist models," Inf. Software Technology, 1997,vol.39, pp.369-476.
[7] Boem, "Cost Models for Future Software Live Cycle Processes:COCOMO 2.0", Annl Soft. Eng, 1995, pp.45-60.
[8] LiMin Fu, "Neural Networks in Computer Intelligence," Tata McGraw-Hill Edition 2003, pp.94-97.
[9] M.J.Shepperd, C.Schofield, and B.Kithenham, "Effort Estimation Using Analogy." Proc.ICSE-18, IEEE Computer Society Press,Berlin,1996.
[10] R.Gray, S.G.MacDonell, and M.J.Shepperd, "Factors Systematically associated with errors in subjective estimates of software development effort: the stability of expert judgement", IEEE 6th International Metrics Symposium, Boca Raton, November 5-6,1999.
[11] Mendas, E., Mosely N., Counsell, S.A Comparison of Development Effort Estimation Techniques for Web Hypermedia Applications.IEEE Symposium on software Metrics,2002.
[12] http://sunset.usc.edu/research/COCOMOII/index.htm.

Mr. Ch. Satyananda Reddy obtained his B.E. Degree in Computer Science and M.Tech Degree in Computer Science with specialization in Software Engineering from Jawaharlal Nehru Technological University, Hyderabad, India. He is currently a Research Scholar and working as Assistant Professor in the Department. of Computer Science and Systems Engineering at Andhra University College of Engineering, Visakhapatnam, India. His Research interests include Software Engineering, Software Metrics, Software Quality Assurance, Cost Estimation, Neural Networks and Fuzzy Systems.

Mr. P. Sankara Rao received his BCA Degree and Master of Science in Information Systems from Andhra University, Visakhapatnam, India. He is working as Assistant Professor in the Dept of Computer Science at AITAM Tekkali, AP, India. He is currently pursuing his Masters Degree in Computer Science and Technology at Andhra University College of Engineering, Visakhapatnam, India. His Research interests include Software Cost Estimation, Neural Networks.

Dr. KVSVN Raju holds a PhD degree in Computer Science from IIT, Kharagpur, and is presently working as Professor in the department of Computer Science and Systems Engineering at Andhra University Engineering College, Visakhapatnam. His research interests include Software Engineering Data Engineering and Data Security.

Dr. V. Valli Kumari holds a PhD degree in Computer Science and Systems Engineering from Andhra University Engineering College, Visakhapatnam and is presently working as Professor in the same department. Her research interests include Security and privacy issues in Data Engineering, Network Security and E-Commerce. She is a member of IEEE and ACM.