An Exploration of PNN and GRNN Models For Efficient Software Development Effort Estimation

B V Ajay Prakash, D V Ashoka, V N Manjunath Aradhya
SJBIT, JSSATE, SJCE
India
ajayprakas@gmail.com, dr.ashok_research@hotmail.com, aradhya1980@yahoo.co.in

ABSTRACT: *Increasing demand for software made IT industries to develop high quality software within predetermined time and budget. In order to accomplish these challenges, the software development process needs to effectively managed and planned. In software development process, effort estimation is very important activity to manage and plan for effective development of software projects. If estimation of software development effort is not accurately measured then entire software project may lead to failure and dissipate the IT industry budget. Machine learning and data mining techniques have been explored as an alternative to existing model COCOMO. This paper aims to explore artificial neural network models such as probabilistic neural networks (PNN) and generalized regression neural networks (GRNN) model on various datasets to accurately estimate the software development effort. The results are evaluated using Mean Magnitude of Relative Error (MMRE), Magnitude of Relative Error (MRE).*

Keywords: Effort Estimation, Neural Networks, PNN, GRNN, Data Mining Techniques

Received: 6 January 2015, Revised 12 February 2015, Accepted 17 February 2015

1. Introduction

Estimating size or resources is one of the vital topics in software engineering and IT. During software development, effort estimation is the basic step taken in budgeting, planning and development of the software project. Exact estimation of software development effort, predicting and scheduling are the essential components to deliver the software product on time, to produce superior quality and contained by estimated cost. Effort estimation inaccuracy can be harmful to an IT industry's economics and dependability due to behind schedule release, poor quality and stakeholder's displeasure with the software product. So in current years effort estimation has motivated as an extensive research. The key in component is developed kilo line of code (KLOC), which affects the software effort estimation which includes all program instructions [1]. Many software estimation models have been proposed to support project manager in accurate decision making about their projects [2, 3]. One of the known mathematical approaches for software cost estimation is the COnstructive COst MOdel (COCOMO) model was first provided by Boehm [2], it was built based on 63 software projects. COCOMO model helps in finding the development time, the effort. The COCMO model equation is given in the form as follows:

$$Effort\,(E) = a(KLOC)^b \qquad (1)$$

Here E represents the evaluated software effort in man-months. In equation 1 a and b parameters depend mainly on the type of software project. Software projects were categorized based on the complexity of the project into three categories. They are:

Model Name	Effort (E)	Time (D)
Organic Model	$E = 2.4\,(KLOC)^{1.05}$	$D = 2.5\,(E)^{0.38}$
Semi-Detached Model	$E = 3.0(KLOC)^{1.12}$	$D = 2.5\,(E)^{0.35}$
Embedded Model	$E = 3.6\,(KLOC)^{1.20}$	$D = 2.5\,(E)^{0.32}$

Table 1. Basic Cocomo Models

2. Related Work

Quite a lot of research works has been carried out on building efficient effort estimation using soft computing techniques [4, 5]. Kelly et. al. [6] provides methodology for exploring software cost estimation using Neural Networks (NNs), Genetic Algorithms (GAs) and Genetic Programming (GP). Artificial neural network are used in effort estimation due to its ability to learn from previous data [7, 8] It is also complex relationships between the dependent (effort) and independent variables (cost drivers). Many authors encompass neural network to effort estimation using feed-forward multi-layer Perceptron, Back propagation algorithm and sigmoid function [6]. Idri et.al [9] has made research on estimating software cost using the neural network and fuzzy logic rules on COCOMO'81 dataset. Samson et. al [10] applied multilayer perception in software effort prediction for boehm's COCOMO dataset also compares neural network with linear regression. Radlinki et. al. [11], analyses the accuracy of predictions for software development effort using different machine learning techniques. Parag [12] proposed a Probabilistic Neural Networks (PNN) approach for simultaneously estimating values of software development parameters (either software size or software effort) and probability that the actual value of the parameter will be less than its estimated value. Attarzadeh et. al. [13], the author's presents that the one of the greatest challenges for software developers is predicting the development effort for a software system based on developer abilities, size, complexity and other metrics for the last decades. Stamatia et. al. [14], comparative research has been done by using three machine learning methods such as Artificial Neural Networks (ANNs), Case-Based Reasoning (CBR) and Rule Induction (RI) to build software effort prediction systems. Kalichanin et. al [15] estimated software development effort of short-scale projects using Feed forward neural network. Proposed method used132 projects verify. Accuracy is measured in terms of MER, i.e., MMER is 0.26, LRM is 0.26 and NN is 0.25. Kumar et. al. [16] used wavelet neural network (WNN) to estimate software development effort with four models, i.e., WNN-morelet, WNN-guassian, TAWNN-guassian, and TAWNN-morelet. TAWNN. WNN-Morelet and WNN-Guassian outperformed all other techniques. Reddy and Raju [17] proposed a multilayer feed forward neural network to improve the performance of the neural network that suits to the COCOMO model. Publicly available COCOMO 81 dataset is used which consisting 63 projects. Data set is divided into training set and validation set in the ratio of 80 %: 20 %. Training set consists of 50 projects selected randomly and validation set consists of remaining 13 projects. Pichai Jodpimai et. al. [18] proposed a neural functional approximation function to estimate the software effort with minimal features.

3. Neural Network Models

Artificial Neural Networks (ANN) is inspired by the early models of information processing by the brain. ANN is an information processing paradigm that is simulated by the biological nervous systems towards learning process and is configured for a specific application, such as pattern recognition or data classification. A novel structure of large number of highly interconnected processing elements (neurons) and its synaptic connections are the key elements of this paradigm [20]. A main problem in statistics with applications in many areas is to estimate a function from some instance of input-output pairs with little or no knowledge of the form of the function. This form of problem is called function approximation, inductive learning, and nonparametric regression. In neural networks terms this can solved using supervised learning process. The function is learned from the instances which a teacher supplies. As neural networks are extremely fast and efficient, we have considered GRNN and PNN to estimate the software development effort.

3.1 Generalized Regression Neural Networks

Generalised Regression Neural Network is a type of supervised learning model based on radial basis function (RBF) which

can be used for regression, classification and time series predictions. The GRNN architecture is as shown in figure1.

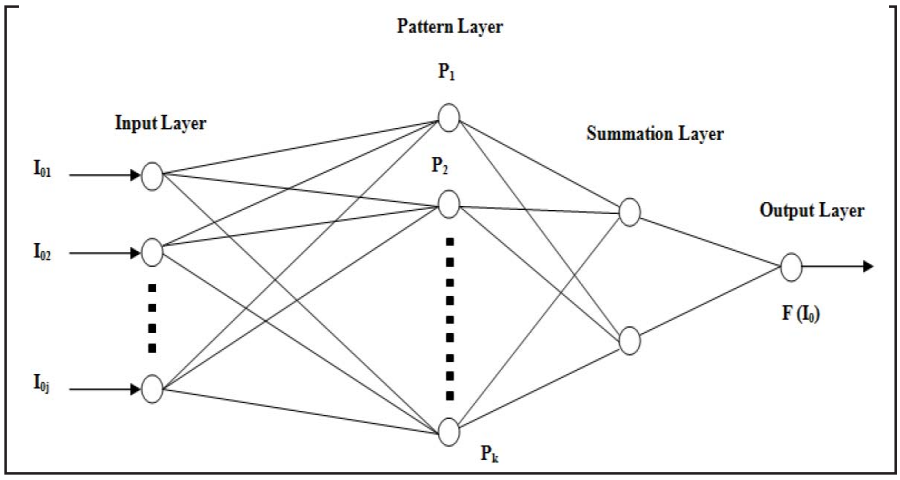

Figure 1. Generalized Regression Neural Network (GRNN) Architecture

GRNN consists of four layers, which are named as input layer, pattern layer, summation layer and output layer. The number of input units depends on the total number of observation parameters i.e. an input vector '*I*' (feature matrix F_i). The input layer connected to the pattern layer consists of neurons provides training patterns and its output to the summation layer to perform normalization of the resultant output set. Each of the pattern layers is connected to the summation neurons and calculates the weight vector using the following equations.

$$W_i = e^{\left[\frac{\|I - I_t\|^2}{2h^2}\right]}$$

$$F(I) = \frac{\sum_{i=1}^{n} T_i W_i}{\sum_{i=1}^{n} W_i}$$

Where the output $F(I)$ is weighted average of the target values T_i of training cases I_i close to a given input case *I*.

3.2 Probabilistic Neural Network (PNN)

The PNN [19] is a Bayes–Parzen classifier. The foundation of the approach is well known decades ago (1960s). It models the Bayesian classifier & minimizes the risk of misclassification. Bayes' classifier is usually criticized due to lack of information about the class probability distributions and makes use of nonparametric techniques, whereas the inherent advantage of PNN is the better generalization and convergence properties when compared to that of Bayesian classifier in classification problems [21]. PNN Architecture is as shown in figure 2.

PNN is similar to that of supervised learning architecture, but PNN does not carry weights in its hidden layer. Each node of hidden layer acts as weights of an example vector. The hidden node activation is defined as the product of example vector '*E*' &input feature vector '*F*' given as $h_i = E_i$ x *F*. The class output activations are carried out using the following equation

$$S_j = \frac{\sum_{i=1}^{n} e^{\left(\frac{h_i - 1}{\varphi^2}\right)}}{N}$$

Where '*N*' is example vectors belonging to class '*S*', 'h_i' is hidden node activation and ' φ ' is smoothing factor.

4. Proposed Methodology

Proposed Methodology consists of data sets preparation, selecting the features from the data sets which are relevant, preparing training and test data sets. Apply the GRNN and PNN model to estimate the software development effort.

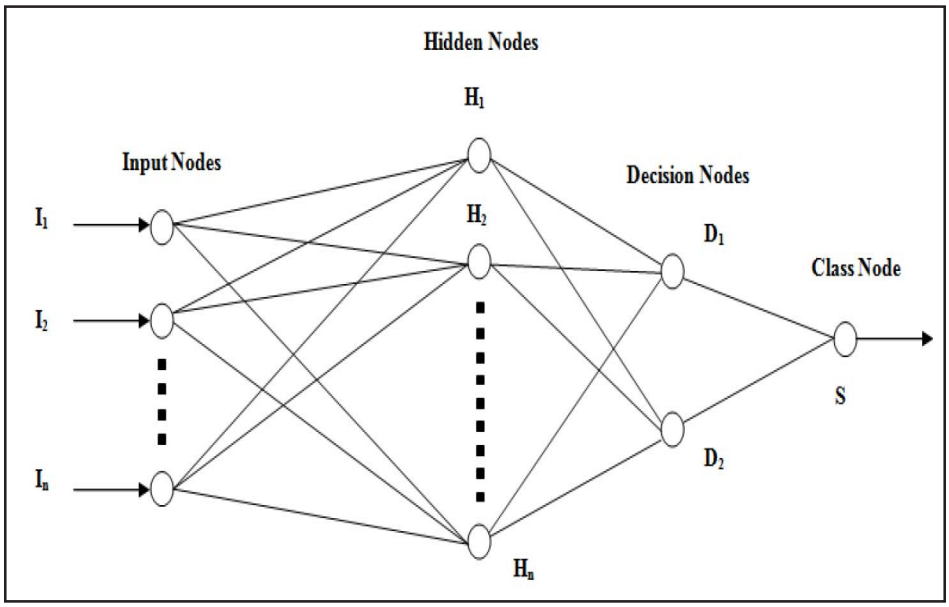

Figure 2. Probabilistic Neural Network (PNN) Architecture

4.1 Data sets preparation

In this research work publicly available PROMISE repository (http://promisedata.org/data) is used. Table 2 shows the different data sets used for effort estimation.

Dataset	Description	Features	Size
COCOMO NASA	NASA projects	17	60
Maxwell	Biggest commercial Banks projects in finland	27	62
China	Chinese software companies projects data	18	499
Nasa93	NASA projects	17	93
Desharnais	Canadian software projects	12	81
Cocomo81	Projects data from NASA	17	63
Kemerer	Large business applications	7	15
Telecom	Telecom companies projects	3	18
Total			890

Table 1. Eight Different Dataset And 890 Projects Used For Effort Estimation

4.2 Applying Grnn And Pnn Models

We used publicly available popular COCOMO 81, China and Desharnais data sets for the experiment purpose. Implementation is done using MATLAB 11, we first classified the testing and training sets. In datasets independent variable are identified and stored in input vector. And the target vector class is prepared for predicting the effort. The value of the software development effort in each project varies. So we randomly selected some projects for testing the accuracy.

Evaluation criterion is used to assess and the compare the performance of the GRNN and PNN models. Magnitude of Relative

Error (MRE) and Mean Magnitude of Relative Error (MMRE) is used for evaluation of effort estimation.MRE is defined as in:
-

$$MRE = \frac{|\ ActualEffort - predictedEffort\ |}{ActualEffort} \times 100$$

MMRE for *N* projects is defined as in

A higher value means worst prediction accuracy for MRE and MMRE,

Project ID	Actual Effort	Software Development Effort Estimated using		MRE using	
		GRNN	PNN	GRNN	PNN
1	2040	2594	2124	17.35	4.12
3	243	212	221	12.75	9.05
11	218	202	192	7.33	11.92
18	11400	10002	10842	12.26	4.89
20	6400	5880	6002	8.12	6.21
26	387	352	312	9.04	19.37
27	88	62	54	29.54	38.63
50	176	152	186	13.63	5.68
51	122	142	104	16.39	14.75
54	20	13	89	35	55
56	958	702	645	26.72	32.67
31	1063	843	784	20.69	26.24
32	702	668	623	4.83	11.25
48	1272	1088	1024	14.46	19.49

Table 2. Comparative Results Of Actual And Estimated Effort With Mre Using Cocomo Data Sets

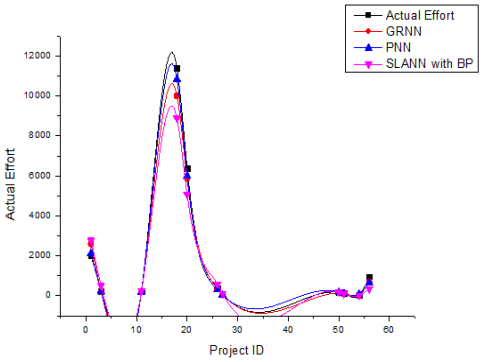

Figure 3. Comparison Of Actual Effort, Grnn, Pnn And Slann With Bp [22].

4.3 Comparative Analysis

Our results are compared with single layer artificial neural network (SLANN) with back propagation algorithm results which is obtained by [22]. The result is shown in Figure 3.

Project ID	Actual Effort	Software Development Effort Estimated using		MRE using	
		GRNN	PNN	GRNN	PNN
371	89	102	118	14.60	32.58
75	139	98	142	24.03	10.07
449	170	123	210	27.64	23.52
48	204	154	168	24.50	17.64
53	281	340	382	20.99	35.94
460	374	325	424	13.10	13.36
93	481	402	248	16.42	11.01
325	752	702	812	6.64	7.97
391	1210	998	829	17.52	31.48
395	1741	1554	2008	10.74	15.33
208	2520	2138	2245	15.15	10.91
5	2994	2558	2694	14.56	10.02
320	3877	3109	3228	19.80	16.73
378	15039	13821	12884	8.09	14.32
326	29399	25482	26700	13.32	9.18
435	54620	48553	49324	11.10	9.69

Table 3. Comparative Results Of Actual And Estimated Effort With Mre Using Desharnasis Data Sets

Project ID	Actual Effort	Software Development Effort Estimated using		MRE using	
		GRNN	PNN	GRNN	PNN
71	546	424	488	22.34	10.62
70	1155	927	842	19.74	27.09
5	2149	1814	2521	15.54	17.31
18	3437	2824	3124	17.83	9.10
41	4620	4112	3998	10.99	13.46
79	9520	8914	9142	6.36	3.97
46	14973	11593	11874	22.57	20.69
81	23940	21248	20997	11.24	12.29

Table 4. Comparative Results Of Actual And Estimated Effort With MRE Using Desharnasis Data Sets

Datasets	MMRE using	
	GRNN	PNN
COCOMO	16.29	18.51
China	16.13	16.85
Desharnasis	15.82	14.31

Table 5. Results of MMRE of GRNN and PNN models

5. Conclusion

Software effort estimation implementation in a large software development organization takes more than a year. Getting hold of estimation experience and integrating it into project management processes along with the consequent introduction of IT measurements for continuing improvement might require another couple of years. This paper has illustrated the application of different neural network models namely GRNN and PNN for software effort estimation to reduce time and effort in software development. From the results, GRNN and PNN models reduce the error and improve the accuracy of software effort. A neural network implementation is done in MATLAB 11.0 and used different dataset of 890 projects in order to accurately estimate efforts. The discussion made in this paper may use as a reference guide to software project managers in software effort estimation.

References

[1] Menzies, T., Port, D., Chen, Z., Hihn. J., Stukes, S. (2005). Validation methods for calibrating software effort models, *In*: *Proceedings of* the 27th international conference on Software Engineering (ICSE'05), (New York, NY, USA), ACM Press, p. 587–595.

[2] Boehm, B. (1981). Software Engineering Economics. Englewood Cliffs, NJ, Prentice-Hall.

[3] Boehm, B. (1995). Cost Models for Future Software Life Cycle Process: COCOMO2. Annals of Software Engineering.

[4] Ryder, J. (1995). Fuzzy COCOMO: Software Cost Estimation. PhD thesis, Binghamton University.

[5] Hodgkinson. A. C., Garratt, P. W. (1999). A neuro-fuzzy cost estimator, *In*: Proceedings of the Third Conference on Software Engineering and *Applications*, p. 401–406.

[6] Kelly, M. A. (1993). A methodology for software cost estimation using machine learning techniques, Master's thesis, Naval Postgratuate School, Monterey, California.

[7] Albrecht. A. J., Gaffney, J. R. (1983). Software function, source lines of code, and development effort prediction: A software science validation., *IEEE Trans. Software Engineering*, 9 (6), p. 630–648.

[8] Matson, J. E., Barret. B. E., Mellinchamp, J. M. (1994). Software development cost estimation using function points, *IEEE Trans. Software Engineering*, 20 [4] , p. 275–287.

[9] Ali Idri.,Taghi, M., Khoshgoftaar., Alain Abran., (2002). Can Neural Networks be easily Interpreted in Software Cost Estimation , IEEE Transaction, p.1162-1167.

[10] Samson, B., Ellison, D., Dugard, P. (1997). Software cost estimation using an Albus Perceptron (CMAC) ., *Journal of Information and Software Technology*, 39 (1), p. 55–60. .

[11] Radlinki. L., Hoffmann, W. (2010). On Predicting Software Development Effort Using Machine Learning Techniquesand Local Data, *International Journal of Software Engineering and Computing*, 2, p.123-136.

[12] Parag, C., Pendharkar. (2010). Probabilistic estimation of software size and effort, An International *Journal of Expert Systems with Applicati* 37, p.4435-4440.

[13] Attarzadeh. I., Siew Hock Ow. (2009). Software Development Effort Estimation Based on a New Fuzzy Logic Model,

International Journal of Computer Theory and Engineering, 1, [4], p.1793-8201.

[15] Bibi Stamatia., Stamelos Ioannis. (2006), Selecting the Appropriate Machine Learning Techniques for Predicting of Software Development Costs, *Artificial Intelligence Applications and Innovations*, 204, p.533-540.

[16] Kalichanin-Balich, I., Lopez-Martin, C. (2010). Applying a feedforward neural network for predicting software development effort of short-scale projects, in Eighth ACIS International Conference on Software Engineering Research, *Management and Applications* (SERA), p. 269-275.

[17] Vinay Kumar, K., Ravi, V., Carr, M., NR Kiran. (2008). Software development cost estimation using wavelet neural networks, *Journal of Systems and Software*, (81) p. 1853-1867.

[18] Reddy, C. S., Raju, K. (2009). A concise neural network model for estimating software effort, *International Journal of Recent Trends in Engineering*, (1). p.188-193.

[19] Jodpimai., P., Sophatsathit. (2010). Estimating software effort with minimum features using neural functional approximation, in *International Conference on Computational Science and Its Applications* (ICCSA), p. 266-273.

[20] Masters., T. (1995). Advanced Algorithms for Neural Networks. Wiley, New York

[21] Patterson., D.W. (1995), Artificial neural networks. Prentice Hall.

[22] Donald F Specht., (1990). Probabilistic Neural Networks. Neural Networks, 3, p.109-118.

[23] Ch.Satyananda Reddy., KVSVN Raju. (2010). An Optimal Neural Network Model for Software Effort Estimation, *International Journal of Software Engineering*, 3 (1) p. 63-78.