Volume 5, Issue 7, July 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Estimating Software Development Effort using Neural Network Models

B. V Ajay Prakash*
Dept. of CSE, SJBIT,
Bengaluru, India

D V Ashoka
Dept. of CSE, JSSATE,
Bengaluru, India

V N Manjunath Aradhya
Dept. of MCA, SJCE,
Mysore, India

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

I. INTRODUCTION

In software development process project planning, effort calculation and scheduling are the initial steps taken in order to develop the software product. Software development effort calculation plays an important in IT industries to calculate how much resources are required. It is very much required for project managers to take initial decisions to start the software project Sometimes estimation of software development effort may exceed the estimated effort and can leads to software failures. These software failures can result in economical losses, poor quality software and stakeholders may disapprove with software product. Accurate estimation of software development effort is essential to deliver the software product on time and to improve the productivity of the software. One of the main parameter to estimate effort is calculate kilo line of code (KLOC) which includes number of program instructions. There are many software effort estimation models have been proposed to take right decision in initiating the software project by project managers [2, 3]. Boehm first introduced a mathematical model for software effort estimation called Constructive COst MOdel (COCOMO) model. The COCMO model equation is given in the form as follows:

$$Effort(E) = a(KLOC)^b \text{ -------- (1)}$$

Here E, represents the calculated software effort in person per months. In equation 1 a and b parameters depend mainly on the type of software project. Recent advances in data analysis techniques made many researchers to apply machine learning algorithms on software engineering data to get better results. Artificial Neural Networks (ANN) is inspired by the early models of information processing by the brain. ANN is an information processing model which is replicated by the genetic nervous system with learning process and can be configured for a specific applications such as image processing, patter recognition and 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]. The problem in statistics is to estimate a function from instance of input and output parameters with some or no knowledge of the estimation 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, GRNN and PNN models are used to estimate the software development effort.

II. RELATED WORKS

In building efficient effort estimation model several researchers have used machine learning algorithms [4, 5]. Kelly et. al. [6] provides methodology for exploring software cost estimation using Neural Networks (NNs), Genetic Programming (GP) and Genetic Algorithms (GAs). Effort estimation can be calculated using artificial neural network because of its capability to be trained from prior results [7, 8]. Feed forward multi-layer perceptron, back propagation algorithm and sigmoid function neural network models are used to effort estimation [6]. Idri et.al [9] used neural network and fuzzy logic rules to estimate software cost on COCOMO'81 dataset. Samson et. al [10] used multilayer perception in software effort prediction for boehm's COCOMO datasets also compares neural network with linear regression. Radlinki et. al. [11], analysed prediction accuracy, of effort using different machine learning techniques. Parag [12], applied Probabilistic Neural Networks (PNN) for estimating values of software development parameters such as software size or

effort. Stamatia et. al. [14], built software effort prediction systems by comparing three machine learning methods such as Artificial Neural Networks (ANNs), Rule Induction (RI) and Case-Based Reasoning (CBR). Kalichanin et. al [15] applied feed forward neural network to accurately estimate software development effort of short-scale projects. Proposed method used132 projects verify. Kumar et. al. [16] estimate software development effort using wavelet neural network (WNN). 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. Pichai Jodpimai et. al. [18] proposed a neural functional approximation function to estimate the software effort with minimal features.

III. NEURAL NETWORK MODELS

A. 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.

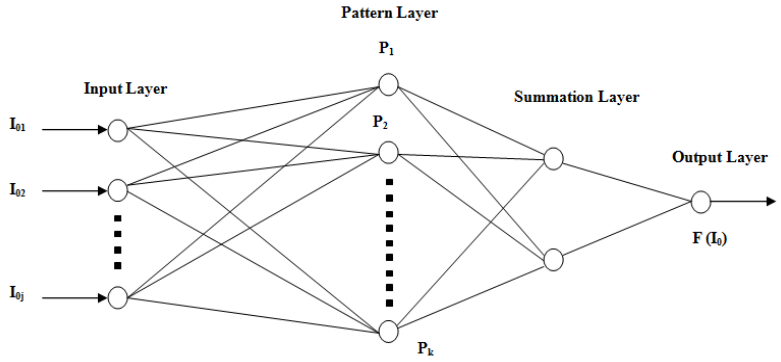

Fig. 1 Generalized Regression Neural Network (GRNN) Architecture

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.

B. 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.

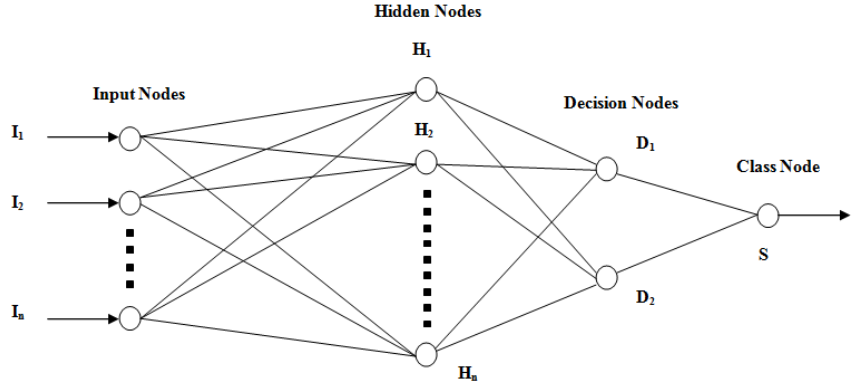

Fig. 2 Probabilistic Neural Network (PNN) Architecture

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

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

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

IV. 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.

A. Data sets preparation

Publicly accessible PROMISE software repository (http://promisedata.org/data) is used to apply the proposed neural network model. Most popular datasets NASA, china and desharnais is considered for the experimental purpose. In data sets there are two kinds of variables are available. One is dependent variable and another is independent variable, here dependent variable is effort and independent variable value decides the dependent variable value. So, to better estimation some of the independent variable can be eliminated. The variable in each data sets are as shown in table 1.

TABLE I: NASA, CHINA AND DESHARNAIS DATASETS VARIABLES

Sl. No.	Data Sets	Attributes	Features	Size
1	NASA projects	analysts capability (acap), programmers capability (pcap), application experience (aexp), modern programing practices (modp), use of software tools (tool), virtual machine experience (vexp), language experience (lexp), schedule constraint (sced), main memory constraint (stor), data base size (data), time constraint for cpu (time), turnaround time (turn), machine volatility (virt), process complexity (cplx), required software reliability (rely)	17	60
2	Chinese software companies projects data	ID, AFP, Input, Output, Enquiry, File, Interface, Added, Deleted, Changed, , NPDR_AFP, NPDU_UFP, Resource, Dev.Type, PDR_AFP, PDR_UFP Duration, N_effort, Effort	18	499
3	Canadian software projects (Desharnais)	Project, TeamExp (Team experience in years), ManagerExp (Experience of project manager in years), YearEnd, Length, Transactions, Entities, PointsAdjust, Envergure, PointsNonAjust (function points), Language, Effort,	12	81

B. Applying GRNN and PNN models

Publicly accessible popular COCOMO 81, China and Desharnais data sets are used for the experiment purpose. Coding is done using MATLAB 11, First classified the testing and training sets. In datasets independent variable are identified and stored in input vector. The output variable dependent variable effort is removed from the vector and the target vector class is prepared for predicting the effort. The value of the software development effort in each project varies. So, some projects are randomly selected for testing the accuracy of the proposed NN models. Table 2, 3, and 4 shows the comparative results.

TABLE II: COMPARATIVE RESULTS OF ACTUAL AND ESTIMATED EFFORT WITH MRE USING COCOMO DATA SETS

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 III: COMPARATIVE RESULTS OF ACTUAL AND ESTIMATED EFFORT WITH MRE USING CHINA DATA SETS

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 IV: 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

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:

$$MRE = \frac{|\, ActualEffort - \Pr edictedEffort \,|}{ActualEffort} x100$$

MMRE for N projects is defined as in

$$MMRE = \frac{1}{N}\sum_{i=1}^{N} MRE_i$$

A higher value means worst prediction accuracy for MRE and MMRE, table 5 shows the MMRE results

TABLE V: RESULTS OF MMRE OF GRNN AND PNN MODELS

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

C. Comparative Analysis

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 4.

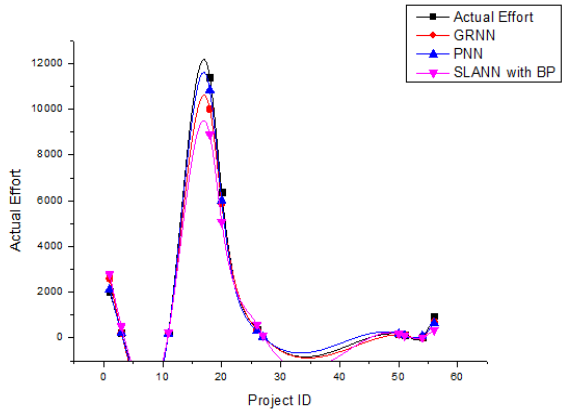

Fig. 3. Comparison of actual effort, GRNN, PNN and SLANN with BP [22].

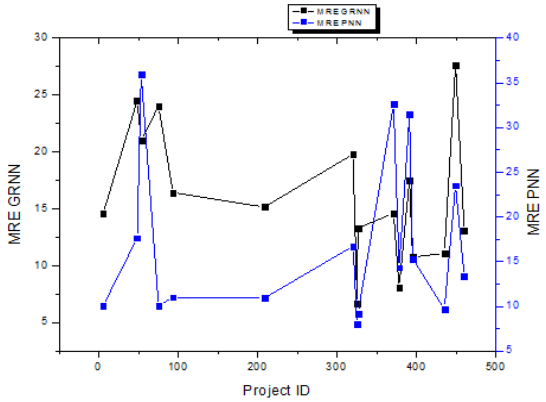

Fig 4. Comparative analysis of MRE results of GRNN and PNN

V. CONCLUSIONS

Software effort estimation plays an important role in project planning, schedule of any software development process. Project mangers use software metrics to measure the software effort. Accurate estimation is very essential to take the initial decisions to start developing the software project. Nowadays to reduce the development time and to improve the quality of software product machine learning algorithms are applied. In the proposed approach neural network models such as GRNN and PNN are applied on cocomo nasa, china and Desharnais data sets. GRNN and PNN gives the better result compared to the SLANN. Some of the independent variables are used to accurately estimate the dependent variable effort. In future, more focus will be on independent variable which is used to estimate the effort.

REFERENCES

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

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

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

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

[5] A. C. Hodgkinson and P. W. Garratt, "A neuro-fuzzy cost estimator," in Proceedings of the Third Conference on Software Engineering and *Applications*, 1999, pp. 401–406.

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

[7] A. J. Albrecht and J. R. Gaffney, "Software function, source lines of code, and development effort prediction: A software science validation," *IEEE Trans. Software Engineering*, 1983, vol. 9, no. 6, pp. 630–648.

[8] J. E. Matson, B. E. Barret, and J. M. Mellinchamp, "Software development cost estimation using function points," *IEEE Trans. Software Engineering*, 1994, vol. 20, no. 4, pp. 275–287.

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

[10] Samson, B., Ellison, D., Dugard, P., "Software cost estimation using an Albus Perceptron (CMAC)", Journal of Information and Software Technology, 1997, vol. 39, no.1, pp. 55–60.

[11] L. Radlinki and W. Hoffmann, "On Predicting Software Development Effort Using Machine Learning Techniques and Local Data," International Journal of Software Engineering and Computing, 2010, vol. 2, pp.123-136,

[12] Parag C. Pendharkar, "Probabilistic estimation of software size and effort," An International Journal of Expert Systems with Applications, 2010, vol. 37, pp.4435-4440.
[13] I. Attarzadeh and Siew Hock Ow, "Software Development Effort Estimation Based on a New Fuzzy Logic Model," International Journal of Computer Theory and Engineering, 2009, Vol. 1, No. 4, pp.1793-8201.
[14] Bibi Stamatia and Stamelos Ioannis, "Selecting the Appropriate Machine Learning Techniques for Predicting of Software Development Costs," Artificial Intelligence Applications and Innovations, 2006, vol. 204, pp.533-540.
[15] I. Kalichanin-Balich and C. Lopez-Martin, 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), 2010,pp. 269-275.
[16] K. Vinay Kumar, et al., Software development cost estimation using wavelet neural networks, Journal of Systems and Software, 81, 2008, pp. 1853-1867.
[17] C. S. Reddy and K. Raju, A concise neural network model for estimating software effort, International Journal of Recent Trends in Engineering, (1) 2009, pp. 188-193.
[18] P. Jodpimai, et al., Estimating software effort with minimum features using neural functional approximation, in International Conference on Computational Science and Its Applications (ICCSA), 2010, pp. 266-273.
[19] Masters, T., 1995. Advanced Algorithms for Neural Networks. Wiley, New York
[20] Patterson.D.W. Artificial neural networks. Prentice Hall, 1995.
[21] Donald F Specht. Probabilistic Neural Networks. Neural Networks, 1990, vol.3, pp.109-118,
[22] Ch.Satyananda Reddy and KVSVN Raju, An Optimal Neural Network Model for Software Effort Estimation, International Journal of Software Engineering, 2010, Vol.3 No.1 pp. 63-78.
[23] B V Ajay Prakash, D V Ashoka, V N Manjunath Aradhya, An Evaluation of Neural Networks Approaches used for Software Effort Estimation", In the Proceedings of ACEEE International Conference on Multimedia Processing Communication and Information Technology (MPCIT 2013), Shimoga, Karnataka, India, pp 292-296

