A New Architecture Based on Artificial Neural Network and PSO Algorithm for Estimating Software Development Effort

Amin Moradbeiky, Amid Khatibi Bardsiri
Islamic Azad University, Kerman Branch, Kerman, Iran
moradbeigi@itrc.ac.ir

***Abstract*— Software project management has always faced challenges that have often had a great impact on the outcome of projects in future. For this, Managers of software projects always seek solutions against challenges. The implementation of unguaranteed approaches or mere personal experiences by managers does not necessarily suffice for solving the problems. Therefore, the management area of software projects requires tools and means helping software project managers confront with challenges. The estimation of effort required for software development is among such important challenges. In this study, a neural-network-based architecture has been proposed that makes use of PSO algorithm to increase its accuracy in estimating software development effort. The architecture suggested here has been tested by several datasets. Furthermore, similar experiments were done on the datasets using various widely used methods in estimating software development. The results showed the accuracy of the proposed model. The results of this research have applications for researchers of software engineering and data mining.**

***Index Terms*— Development Effort Estimation; Neural Networks; Particle Swarm Optimization; Software Project.**

I. INTRODUCTION

Due to the intangible nature of software, software companies often have difficulty estimating the effort required to complete software projects [1]. Software project managers have always tried, in one way or another, to direct and respond to challenges facing software projects. In this regard, utilizing devices that would enable the managers of these projects to predict the forthcoming situations of projects or to assess the impact of decisions on the future of a project has been of special interest to researchers. Such instruments can play an important role in better understanding the future conditions of projects, and they usually operate in algorithmic or non-algorithmic ways. Algorithmic methods are neatly formulated and work with in specific framework. Regression-based approaches and COCOMO method are among methods included in this group. Non-algorithmic methods belong to another group and they work in a more flexible way. In this way, we try to predict future conditions with respect to the present situation. Expert judgment method (EJM) is the first method introduced in 1960 for estimating software development effort [2]. Other methods such as COCOMO [3], Coco 2 [4], SLIM [5], and function points analysis [6] have been formulated since then. These methods follow an algorithmic manner. A number of studies have used linear regression [8] [7], non-linear regression [7], and regression tree [9] [10] methods. Including among algorithmic methods are attribute-based estimation (ABE) [11] and its associated compound methods [12] [13] [14] [15] [16].

Using artificial neural network is one of the simplest and most applicable methods of data modeling. In this paper, we have employed artificial neural network for modeling and estimating software projects. In the next section, neural network and its mathematical concept have been explained. Afterwards, the criteria for evaluating the precision of the estimation have been presented. Then, the proposed architecture estimator, which is based on neural network, has been described, and, eventually, tested.

II. NEURAL NETWORK

Neural networks are simplified modeling of real neural systems that are widely used in solving various scientific problems. The scope of these networks is quite vast, ranging from classificatory applications to applications such as interpolation, estimation, detection, etc. Perhaps the most important advantage of these networks is their multiple capabilities, along with their ease of use.

A. The Concept of Network

One of the most efficient methods to solve complex problems is breaking them down into simpler sub-problems, such that each of these sub-sectors could be easier to understand and describe. In fact, a network is a collection of simple structures that together describe the final complex system. There are different types of networks, but they all have two components in common:

1) A set of nodes, with each node being the computing unit of the network which receives the inputs and processes them so as to obtain the required outputs. The processes performed by the nodes vary from simple ones - such as input collection - to the most complex computations. In special cases, a node may itself include a network.

2) Connections between nodes; these connections determine how information will pass between the nodes.

The interaction between the nodes, resulted from these connections, can lead to a general behavior displayed by the network; this behavior is such that it cannot be observed in any of the individual elements per se. The comprehensive character of this general behavior, compared with the performance of each single node, turns the network into a powerful instrument. In short, when a simple set of elements are combined in a network, they are able to exhibit a behavior which none of the elements is able to produce alone.

B. Artificial Neural Network

As mentioned earlier, there are various types of networks. Out of these variations, there is one which considers a node as an artificial neuron. Technically, artificial neural network (ANN) is the name applied to this computational approach. An artificial neuron is actually a computational model that is based on the nerve neurons of human being. Natural neurons receive their input through the synapse. These synapses are located on the dendrites or the neuronal membrane. In a real nerve, dendrites change the amplitude of the received pulses. This alteration is not of the same type across time. Indeed, it is learned by the nerve. In case the signal is sufficiently strong (i.e. if it surpasses the threshold value), the nerve is activated and sends a signal across the axon. This signal, in turn, could enter a synapse and stimulate other nerves. Figure 1 illustrates a real nerve.

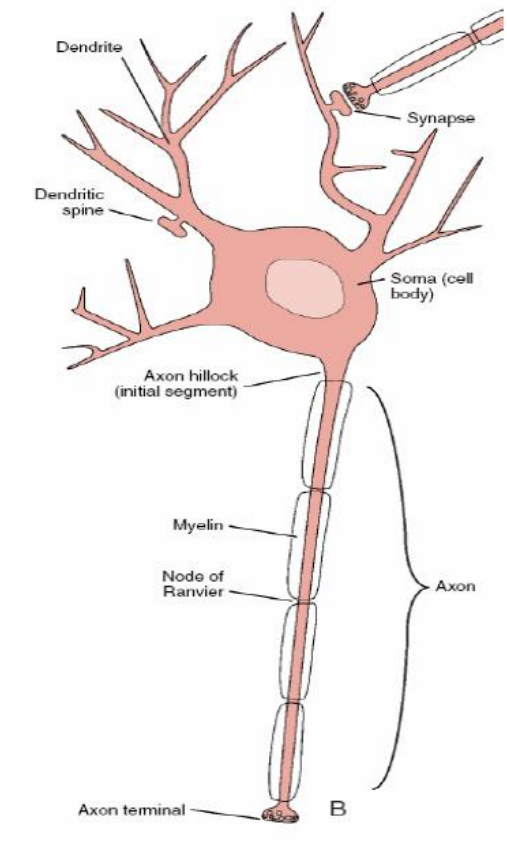

Figure. 1: A real nerve.

C. Mathematical Model of Artificial Neural Network

When modeling the nerves, one avoids their complexities and pays attention only to their basic concepts; otherwise, the modeling procedure will be very difficult. Apart from the applied simplifications, the main difference between this model and reality is that in the real network, inputs are temporal signals while they are real numbers in this model.

There are many variations in the model presented in Figure 2. For instance, the weights of a neural network, which transmit the output, can be positive or negative. On the other hand, there are diverse functions that can be used for thresholding. Among the most famous of these functions are arcsin, arctan, and sigmoid. These functions must be continuous, smooth, and differentiable. Also, the number of input nodes can be variable. Obviously, as the number of nodes augments, it becomes difficult to determine the weights. Therefore, one has to look at new ways of solving this problem. The process of determining optimal weights and setting their values is mainly recursive. For this purpose, the network is trained by rules and data; and using network learning capability, a variety of algorithms are recommended, all of which aim to approximate the produced output to the ideal and expected one.

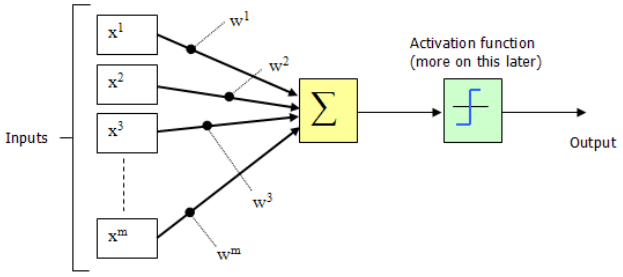

Figure. 2: Mathematical Model of Artificial Neural Network

Equation 1 is the total equation that the neural network follows. In this equation, X is the input vector, W is the weight vector, and m is the input data dimension. The value obtained from this equation was inserted into the activation function, through which the output value was determined.

$$\sum_{i=1}^{m} bias + (W^{i} X^{i}) \tag{1}$$

III. Equations for Estimation Error Calculation

In this study, to determine the estimation error, we have employed certain equations that can be used by many researchers in the field. Using these equations enables one to compare the results of this study with other similar works. The equations used in this article include relative error (RE), magnitude of relative error (MRE), median magnitude of relative error (MdMRE), and prediction percentage (PRED), as shown in equations 2 to 5.

$$RE = \frac{Estimate - Actual}{Actual} \tag{2}$$

$$MRE = \frac{|Estimate - Actual|}{Actual} \tag{3}$$

$$MdMRE = median(MRE) \tag{4}$$

$$PRED(X) = \frac{A}{N} \tag{5}$$

IV. The Proposed Model

The purpose of this research is to use neural network for data modeling and then to use the model for prediction. Given that setting properly the parameters of an artificial neural network helps the developed network to have a more accurate model of its source data, we have suggested a method for making such a model using neural network. This new method makes use of the artificial intelligence algorithm of PSO to accurately model data using artificial neural network. The type of the neural network studied in the present article wasfeedforward. PSO algorithm configuration affects the accuracy of the results of the

model. Different researchers have proposed different configurations for PSO algorithm. In the present study, the proposal of Russell et al was used to configure the PSO algorithm [17]. According to this proposal, the best values for parameters of C1, C2, and W are respectively 2, 2, and 1. The model proposed in the present study consisted of two sections: training and testing. The training section of the proposed method tried to propose an accurate model of the data. In order to evaluate the accuracy of the model obtained from the training section, a separate architecture was used in the testing section. The architecture of the testing section estimated the amount of software development effort by employing a model obtained from the training section. The architecture of the training and testing sections is explained in the following sections. The accuracy of the proposed model was determined based on the accuracy of the model in the testing section. To determine the accuracy of the model, the formulas introduced in Section III were used. In order to increase the reliability of the results obtained from the proposed model, different datasets were utilized.

In the proposed model, the data have initially been divided into the training and testing sections; in the training section, as shown in Figure 3, PSO algorithm attempts to search the best settings for building the network. Whenever a specific setting is offered by the PSO algorithm, it is used for prediction, and its associated error is calculated; then, it is returned to the PSO algorithm as the setting feedback. Searching continues until the predetermined termination condition is fulfilled. The aim of this stage is to discover the best settings of the neural network to generate a prediction model with minimum error. The settings provided by PSO algorithm for the neural network configuration include determining the vector of bias values, weight, and the best number of the hidden layers.

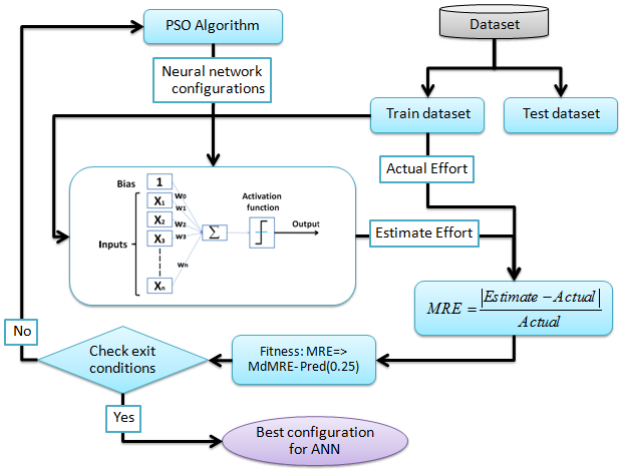

Figure. 3. Architecture of the train stage

In the training stage, a model was developed for prediction; now, we need to test this model in order to assess its accuracy. To test the model by the neural network, we make use of the data considered for this stage. Figure 4 displays the architecture for the testing stage. The data of the testing stage are estimated by the network one by one, and the estimation error is calculated for each datum. The total error of estimation process is also measured based on the error of each datum, and it is introduced as the test result. In the end, the total error of the estimation process was calculated based on the estimation error of each datum. Calculation of the error of each datum and the total error of the model was conducted based on formulas introduced in Section III. The distribution of the data in the training and testing sections is a very important issue. The method of data distribution indicates the reliability of the data obtained from the model [18]. The method used in the present study is explained in Section VI.

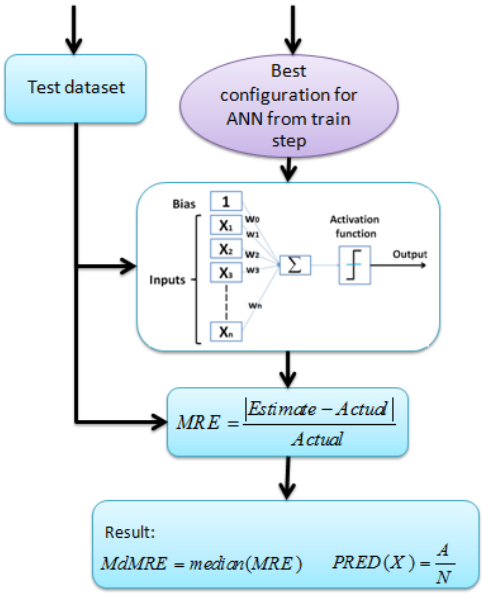

Figure. 4. Architecture of the testing stage

V. Assessment Method

In the estimation method via neural network, the arrangement of samples in the testing or training groups has a considerable impact on the obtained error as well as the quality of network training [18]. Therefore, to demonstrate the sustainability of the results of the proposed architecture, we need a method to indicate the independence of results from the location of samples. To achieve this end, there are various assessment methods such as 3 fold, 10 fold, etc. In this regard, the present study has used LOO method. In this method, each time a project is considered as a test, and it is estimated using the best parameters resulted from the testing stage. In this method, the number of projects corresponds to the number of running the testing stage. The value of MdMRE is equal to the median error derived from estimating each project.

VI. Introducing Datasets

Three datasets, including COCOMO, Desharnais, and Maxwell have been employed to test the proposed model. These datasets have been variously used by researchers. In the following sections, they have been statistically analyzed and tested.

A. Data analysis of COCOMO dataset

COCOMO dataset consists of 63 projects, each having 17 features. Table 1 analyzes the data existing in this dataset. In this dataset, the last feature ('actual') is considered as the aim of estimation.

Table 1
COCOMO data analysis

Feature	Maximum	Minimum	Mean	Median
'rely'	1.4	0.75	1.036349	1
'data'	1.16	0.94	1.004444	1
'cplx'	1.65	0.7	1.092063	1.07
'time'	1.66	1	1.11381	1.06
'stor'	1.56	1	1.14381	1.06
'virt'	1.3	0.87	1.008413	1
'turn'	1.15	0.87	0.971746	1
'acap'	1.46	0.71	0.905238	0.86
'aexp'	1.29	0.82	0.948571	1
'pcap'	1.42	0.7	0.93746	0.86
'vexp'	1.21	0.9	1.005238	1
'lexp'	1.14	0.95	1.001429	1
'modp'	1.24	0.82	1.004127	1
'tool'	1.24	0.83	1.016984	1
'sced'	1.23	1	1.048889	1
'loc'	1150	1.98	77.20984	25
'actual'	11400	5.9	683.527	98

B. Data analysis of Desharnais dataset

This dataset includes 77 projects, and 10 features have been evaluated numerically for each project. Table 2 presents the statistical characteristics of this dataset.

Table 2
Desharnaisdata analysis

Feature	Maximum	Minimum	Mean	Median
F1	4	0	2.298	2
F2	7	0	2.649	3
F3	36	1	11.246	10
F4	886	9	179.805	134
F5	387	7	120.545	96
F6	793	92	285.35	259
F7	52	5	29.528	28
F8	698	83	272.509	247
F9	3	1	1.377	1
effort	14987	651	4795	3542

C. Data analysis of Maxwell dataset

Another dataset examined here is Maxwell, which is composed of 62 projects. This dataset has numerically defined 26 features for each project and has so far been investigated by many studies. Table 3 analyzes the data of this dataset.

Table 3
Data analysis of Maxwell dataset

Feature	Maximum	Minimum	Mean	Median
F1	5	1	2.354839	2
F2	5	1	2.612903	2
F3	4	0	1.032258	1
F4	2	1	1.935484	2
F5	2	1	1.870968	2
F6	1	0	0.241935	0
F7	4	1	2.548387	3
F8	5	1	3.048387	3
F9	5	1	3.048387	3
F10	5	2	3.032258	3
F11	5	2	3.193548	3
F12	5	1	3.048387	3
F13	4	1	2.903226	3
F14	5	1	3.241935	3
F15	5	2	3.806452	4
F16	5	2	4.064516	4
F17	5	2	3.612903	4
F18	5	2	3.419355	3
F19	5	2	3.822581	4
F20	5	1	3.064516	3
F21	5	1	3.258065	3
F22	5	1	3.33871	3
F23	54	4	17.20968	13.5
F24	3643	48	673.3065	385
F25	9	1	5.580645	6
effort	63694	583	8223.21	5189.5

VII. TESTING THE DATASETS

In this section, the proposed architecture has been tested. The purpose of testing this architecture has been to evaluate its accuracy. The tests have been conducted on the datasets discussed above. The results of the tests have been analyzed and presented based on the type of each dataset. Using the criteria and equations introduced in section III, we calculated the architecture accuracy in the tests.

A. Testing Desharnais dataset

In the first test, we dealt with Desharnais dataset. The characteristics of this dataset have been given in section VI.B. The MdMRE value obtained by running the proposed architecture through LOO evaluation method has been given in Table 4. In this test, MdMRE and PRED were 0.3252 and 0.3636, respectively.

Table 4
The effectiveness of different estimation methods in Desharnais dataset

Approach	MdMRE	Pred
ABE K=2	0.4295	0.2987
ABE K=3	0.3921	0.3117
ABE K=4	0.3333	0.3247
ABE K=5	0.3642	0.3636
CART	0.4280	0.2857
MLR	0.4140	0.2727
SWR	0.6557	0.1169
Proposed Model	**0.3252**	**0.3636**

B. Testing COCOMO dataset

A second test has been carried out on COCOMO dataset. The characteristics of this dataset were presented in section VI.A. The related MdMRE value resulted from employing the proposed architecture through LOO evaluation has been provided in Table 5. For this test, MdMRE and PRED amounted, respectively, to 0.7496 and 0.1905.

Table 5
Comparing the effectiveness of different estimation methods in COCOMO dataset

Approach	MdMRE	Pred
ABE K=2	0.8056	0.1270
ABE K=3	0.8013	0.1111
ABE K=4	0.7959	0.0952
ABE K=5	0.7679	0.1429
CART	0.8597	0.1587
MLR	1.0064	0.1746
SWR	10.6590	0.0476
Proposed Model	**0.7496**	**0.1905**

C. Testing Maxwell dataset

The next test was performed on Maxwell dataset. The characteristics of this dataset were explained in section VI.C. The associated MdMRE value derived from employing the proposed architecture via LOO evaluation is presented in Table 6. MdMRE and PRED values in this test were 0.42 and 0.27, respectively.

Table 6
Comparing the effectiveness of different estimation methods in Maxwell dataset

Approach	MdMRE	Pred
ABE K=2	0.5659	0.2258
ABE K=3	0.4777	0.2097
ABE K=4	0.5069	0.1774
ABE K=5	0.5536	0.2097
CART	0.5652	0.2581
MLR	1.7900	0.0484
SWR	1.3495	0.1129
Proposed Model	**0.42**	**0.27**

VIII. CONCLUSION

Artificial neural network has a simple operation, and one can use it for data modeling. The present study proposed an architecture based on artificial neural network for modeling and estimating software projects. The results of testing this architecture demonstrated the efficacy of this model. In this paper, PSO algorithm was used to configure the network. It is recommended that future studies also take advantage of artificial-intelligence-based methods to configure artificial neural networks.

REFERENCES

[1] The Standish Group, "Chaos Report," Technical report, http://www.standishgroup.com, 2009.
[2] E.A. Nelson, "Management Handbook for the Estimation of Computer Programming Costs," System Developer Corp., 1966.
[3] B. Boehm, "Software Engineering Economics," Prentice Hall, 1981.
[4] B. Boehm, R. Madachy, and B. Steece, "Software Cost Estimation with Cocomo II," Prentice Hall, 2000.
[5] L.H. Putnam, "A General Empirical Solution to the Macro Software Sizing and Estimation Problem," IEEE Trans. Software Eng., vol. 4, no. 4, pp. 345-361, July 1978.
[6] A.J. Albrecht, and J.E. Gaffney, "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation," IEEE Trans. Software Eng., vol. 9, no. 6, pp. 639-648, Nov. 1983.
[7] G. Finnie, G. Wittig, and J.-M. Desharnais, "A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case-Based Reasoning and Regression Models," J. Systems and Software, vol. 39, pp. 281-289, 1997.
[8] P. Sentas, L. Angelis, I. Stamelos, and G. Bleris, "Software Productivity and Effort Prediction with Ordinal Regression," Information and Software Technology, vol. 47, pp. 17-29, 2005.
[9] L. Briand, K.E. Emam, D. Surmann, and I. Wieczorek, "An Assessment and Comparison of Common Software Cost Estimation Modeling Techniques," Proc. 21st Int'l Conf. Software Eng., pp. 313-323, May 1999.
[10] L. Briand, T. Langley, and I. Wieczorek, "A Replicated Assessment and Comparison of Common Software Cost Modeling Techniques," Proc. 22nd Int'l Conf. Software Eng., pp. 377-386, June 2000.
[11] M. Shepperd and, C. Schofield, "Estimating software project effort using analogies," IEEE Trans Softw Eng 23 (11):736–743, 1997.
[12] L. Angelis and, I. Stamelos, "A simulation tool for efficient analogy based cost estimation," Empir Softw Eng 5(1):35–68, 2000.
[13] N. H. Chiu and, S. J. Huang, "The adjusted analogy-based software effort estimation based on similarity distances," J. Syst Softw 80(4):628–640, 2007.
[14] S. Gupta, G. Sikka, H. Verma, "Recent methods for software effort estimation by analogy," SIGSOFT Softw Eng Notes 36(4):1–5, 2011.
[15] E. Kocaguneli, T. Menzies, A. Bener and, J. W. Keung, "Exploiting the essential assumptions of analogy-based effort estimation," IEEE Trans Softw Eng 38(2):425–438, 2012.
[16] D. Milios, I. Stamelos and, C. Chatzibagias, "Global optimization of analogy-based software cost estimation with genetic algorithms," artificial intelligence applications and innovations. L. Iliadis, I. Maglogiannis and H. Papadopoulos, Springer Boston, 350–359, 2011.
[17] C. Eberhart-Russell, Y. Shi, and J. Kennedy. "Swarm intelligence," Elsevier, 2001.
[18] E. Kocaguneli and T. Menzies, "Software effort models should be assessed via leave-one-out validation," Journal of Systems and Software, 86(7), 1879-1890, 2013.