Journal of Computer Science

Original Research Paper

Optimizing Software Effort Estimation Models Using Back-Propagation Versus Radial Base Function Networks

Abdel Karim Baareh

Computer Science Department, Ajloun University College, Al-Balqa Applied University, Ajloun, Jordan

Article history
Received: 21-11-2018
Revised: 11-01-2019
Accepted: 09-03-2019

Email: baareh@bau.edu.jo

Abstract: Software development effort estimation becomes a very important and vital tool for many researchers in different fields. Software estimation used in controlling, organizing and achieving projects in the required time and cost to avoid the financial punishments due to the time delay and other different circumstances that may happen. Good project cost estimation will lead to project success and reduce the risk of project failure. In this paper, two neural network models are used, the Back-propagation algorithm versus the redial base algorithm. A comparison is done between the suggested models to find the best model that can reduce the project risks related to time and increase the profit by achieving the demands of the required project in time. The two models are implemented on a 60 of NASA public dataset, divided into 45 data samples for training and 15 data samples for testing. From the result obtained we can clearly say that the performance of the back-propagation neural network in training and testing cases is actually better than the radial base function, so the back-propagation algorithm can be recommended as a useful tool in the software effort and cost estimation.

Keywords: Effort Estimation, NASA Software, Artificial Neural Network, Back-Propagation, Radial Base Function

Introduction

Building and estimating successful software is an important task that attracted many software developers (Boraso *et al*., 1996; Dolado, 2011). Bidding, budgeting and planning are very important factors that affect project success. Accurate defining of these factors will reflect on the project size, time, efforts, complexity and the different required tools to avoid the sudden and unexpected events that may happen during the project duration, that cause a project loss. Good software estimation gives exact feedback about the project progress that allows better resource utilization, allocation and use (Boehm, 1981).

In Software Technology Conference held in 1998, Dr. Patricia Sanders, Director of Test Systems Engineering and Evaluation at OUSD, stated that 40% of the DoD's software development costs are wasted and paid on reworking the software, that caused an annual loss of $18 billion on the year of 2000. Dr. Patricia added that only 16% of the developed software could finish in the accurate time and budget.

Effort estimation was mainly affected by the Developed Line of Code (DLOC), where the instructions of the program and statements were included. This model worked on 63 software projects and its core function based on finding and determining the arithmetical relationship between three important variables; the time of software development, human efforts during the work months and effort of maintenance (Kemere, 1987).

The Constructive Cost Model (COCOMO) is considered as one of the most important, popular and famous models used to estimate the software effort which is developed by Boehm (1981; Boehm *et al*., 1995).

Numerous techniques were used by different researchers for building an efficient estimation models structure to process the software cost estimation problem. Artificial neural network with different architecture was one of these models that proved its solidity and efficiency in this field (Shepper and Schoeld, 1997) moreover, the fuzzy logic used by (Kumar *et al*., 1994; Kaushik *et al*., 2012) and evolutionary algorithms such as genetic algorithm and genetic programming was also strongly used to deal with such types of problems.

Artificial neural network algorithm with back-propagation algorithm versus the radial base function is used in this paper. The comparison between the two models is presented. This comparison will contribute in

selecting the best neural network model for solving the software effort estimation problem. Artificial Neural Networks (ANN) works in a similar way as the human biological neural system acts, exactly comparable to the way how the brain operates and process information (Negnevitsky, 2005). The brain consists of large numbers of small cells that are fully interconnected to process the data. Also, the Artificial Neural Networks (ANN) consists of a great amount of fully and strongly interconnected cells called neurons, all working to gather in a systematic manner to solve specific problems, which also learn by example similar to the way the human biological systems do. Learning in Neural networks means readjusting the synaptic relations that existed between the variously connected neurons until we reach the optimal solution. In 1943 the first artificial neuron was introduced by the neurophysiologist Warren McCulloch and the logician Walter Pits. This research paper is formulated according to the following. Section 2 describes the related work. Section 3 represents the back-propagation learning algorithm. Section 4 presents the redial base function. Section 5 describes the constructive models. Section 6 discusses the experimental results. Finally, Section 7 discusses the conclusion and future work.

Literature Review

The Soft Computing technique is recently used in many research fields. These techniques are developed by (Zadeh, 1994) which contain different algorithms architecture such as the Fuzzy Logic, neurocomputing like neural networks and probabilistic reasoning. Later, the field is extended to include other new techniques such as genetic algorithms, genetic programming, swarm intelligence…etc. All these techniques played a vital role in developing and improving the research area (Kaushik *et al*., 2012; Huang *et al*., 2003). Soft Computing techniques were also used in processing and treating software cost estimation problems. The authors (Feng *et al*., 2010) implemented the genetic programming algorithm to optimize and improve the performance of back-propagation neural network to reduce the construction cost for software estimation. The authors (Shepper and Schofield, 1997) also used neural network in optimizing and treating the cost estimation models. A fuzzy logic using different techniques was also used to process the famous COCOMO model (Kaushik *et al*., 2012). The Fuzzy Logic technique using Takagi Sugeno model was used to find out how the rules can contribute in solving the software effort estimation problem as presented by the authors (Sheta and Aljahdali, 2013). Moreover, the author (Sheta, 2006) used the genetic programming to estimate the COCOMO model parameters for the NASA software project. The authors (Ghatasheh *et al*., 2015) used the firefly algorithm for optimizing the software effort estimation models. A neural network was also strongly presented in solving the software cost estimation problem as presented by (Singh *et al*., 2011). The authors (Oliveira *et al*., 2010) used the Hybrid method for selecting parameters and optimizing the model in order to clarify the impact of using GA in feature selection and effort estimation. The authors (Sehra *et al*., 2011) used the soft computing techniques for software project effort estimation where the NN, FL and GP were used for estimating the project efforts. In this paper, the interest and motivation of processing such type of problem comes from the real and historical importance in processing the software cost estimation problem as presented in the above - related works.

Back-Propagation Learning Algorithm

ANN with back-propagation algorithm considered as one of the most important learning algorithms used tell now. Back-propagation (BP) was introduced by David Rumelhart, Geoffrey Hinton and Ronald Williams 1986. It is considered as the fastest and the workhorse of learning in a neural network. The working mechanism of the back-propagation neural network is based on the concept of learning by example. This means that the user should give the neural network the examples of what he wants (desired output) and the network change the weights of the network's related to that, when training is completed, the output will be estimated according to the desired one which is called the (target output) for a particular input. The Back-propagation Artificial Network still proves its efficiency in a variety of application solving numerous serious real-life problems in finance sectors, cancer disease recognition (Braik and Sheta, 2011), science, forecasting (Baareh *et al*., 2006; Sheta *et al*., 2015; 2018), feature extraction (Al-Batah *et al*., 2010), classifications (Seethe *et al*., 2007; Hongjun *et al*., 1996; El-Sayyad *et al*., 2015), face recognition (Radha and Nallammal, 2011), Fingerprint recognition (Al-Najjar and Sheta, 2008) etc. The back-propagation artificial neural is used in this paper to solve the software cost estimation problem. ANN mostly, formulated from three layers, i.e. the input, hidden and the output. The weighted sum of the input neurons specifies the nonlinear activation (i.e. sigmoid) function argument (Baareh *et al*., 2006). Let $x_1(p)$, $x_2(p)$....$x_n(p)$ be the inputs of the network and let $y_1(p)$, y_2(p),…,$y_n(p)$ be the required output. The iteration number is defined by P. The function of the back-propagation neural network is illustrated as in (El-Sayyad *et al*., 2015):

1. Equation 1 represents the output calculated from the hidden layer:

$$y_j(p) = sigmoid[\sum_{i=1}^{n} x_i(p) w_{ij}(p) - \theta_j] \quad (1)$$

The number of input neurons is represented by n, j which represents the hidden layer number, w_{ij} represent the calculated mapped weights between

the inputs layer to the hidden layer and from the hidden to the output layer, θ is a threshold value:

2. Equation 2 represents the implemented sigmoid function:

$$y_j(p) = \frac{1}{1 + e^{-x_j(p)}} \quad (2)$$

3. Equation 3 represents the calculated output of the output layer:

$$y_k(p) = sigmoid[\sum_{j=1}^{m} x_{jk}(p) w_{jk}(p) - \theta_j] \quad (3)$$

where, *m* is the number of inputs of neuron *k* in the output layer

4. Equation 4 represents the Error Gradient calculated from the output layer:

$$\delta_k(p) = y_k(p)[1 - y_k(p)] e_k(p) \quad (4)$$

where $e_k(p)$ is the output layer error:

$$e_k(p) = y_{d \cdot k}(p) - y_k(p) \quad (5)$$

5. Equation 6 represents the ANN calculated weights:

$$\Delta w_{jk}(p) = \alpha y_j(p) + \delta_k(p) \quad (6)$$

6. Readjust the ANN weights using Eq. 7.

$$w_{jk}(p+1) = w_{jk}(p) + \Delta w_{jk}(p) \quad (7)$$

7. Equation 8 represent the calculated gradient error of the hidden layer:

$$\delta_j(p) = y_j(p)[1 - y_j(p)] \sum_{k=1}^{i} \delta_k(p) w_{jk}(p) \quad (8)$$

8. The weights are recalculated further using Eq. 9.

$$\Delta w_{ij}(p) = \alpha x_i(p) + \delta_j(p) \quad (9)$$

9. Equation 10 represents the readjusted weights:

$$w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p) \quad (10)$$

The structure of the back-propagation neural network is shown in Fig. 1.

Radial Basis Function (RBF) Networks

Radial Basis Function (RBF) Networks derives from the theory of function approximation, it is a way of learning, very fast and very good in interpolation (Harikumar and Vijayakumar, 2013). The constructed Radial Basis Function (RBF) consists of two feed-forward networks layers, input, hidden and output layers. The input layers used to read the inputs into the network for a process, a series of radial basis functions (e.g. Gaussian functions) were executed by the hidden nodes and the linear summation functions were also executed by the output nodes. When the network process started, the weights from both layers (input-hidden) and (hidden-output) were calculated. Moreover, if we have N data set of points in a space of multi-dimensional, this requires that every input vectors of the D dimension $x^p = \{x_i^p : i = 1,..., D\}$ to be related to its corresponding target output t^p.

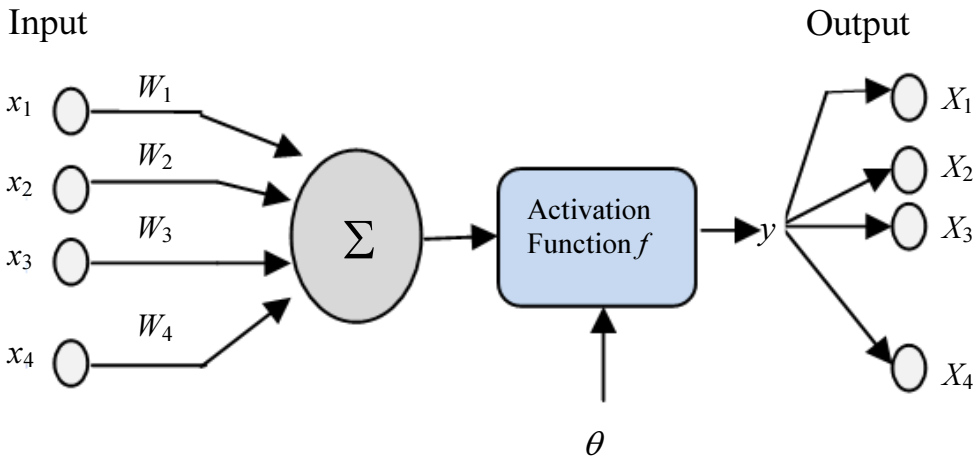

Fig. 1: Back-propagation neural network

The purpose is to get a function $f(x)$ such that $f(x^p) = t^p \forall\, p = 1,...,N$. The weights between the input-hidden layers and the hidden-output layers were determined using Equation 11:

$$\Phi_w = t \tag{11}$$

where, this can be written in a matrix form by defining the vectors $t = \left\{t^p\right\}, w = \{w_p\}$ and the matrix as:

$$\Phi = \{\Phi_{pq} = \phi(\| x^q - x^p \|)\} \tag{12}$$

where, this can finally be written as:

$$f\left(x^q\right) = \sum_{p=1}^{N} w_p \phi\left(\| x^q - x^p \|\right) = t^p \tag{13}$$

This operation can be clearly shown in Fig. 2.

Constructive Models

Two neural network algorithms were used in this paper, back-propagation algorithm compared to radial bases function. The constructed back-propagation network consists of three layers as shown in Fig. 3, an input layer, two hidden layers and one output layer. The input layer consists of three inputs that are product of complexity (CPLX), programmer capability (PCAP) and the thousand of source lines code (KSLOC), the first hidden layer consists of twenty fully interconnected neurons, the second hidden layer consists of ten fully interconnected neurons also and the output layer consists of one output that is the measured efforts.

The radial base function consists of three layers, the input layer, one hidden layer and, one output layer. The input layer consists of three inputs as mentioned above that is CPLX, PCAP and the KSLOC, the hidden layer constituted from four fully interconnected neurons and finally, one output layer is produced i.e., the estimated efforts as shown in Fig. 4.

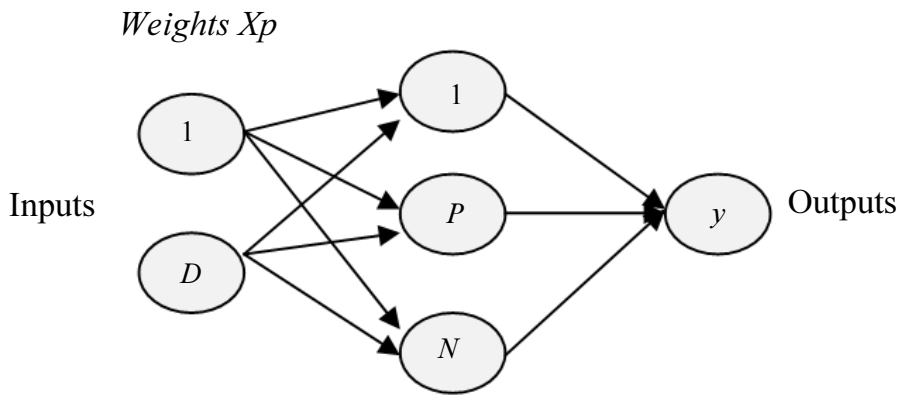

Fig. 2: Radial base function neural network

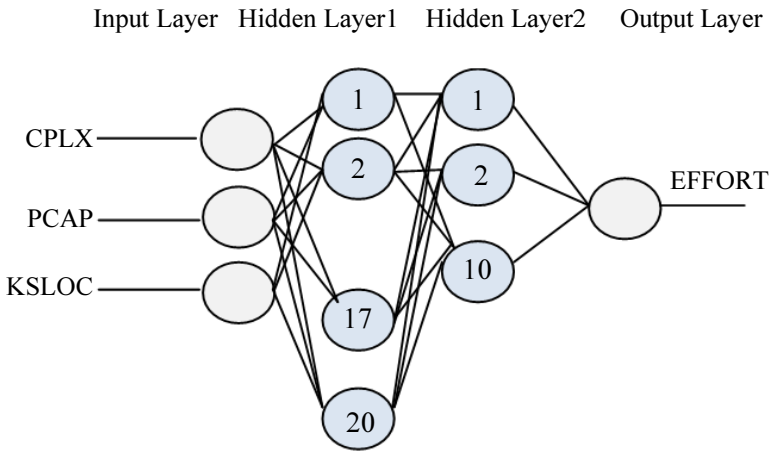

Fig. 3: Implemented back-propagation neural network

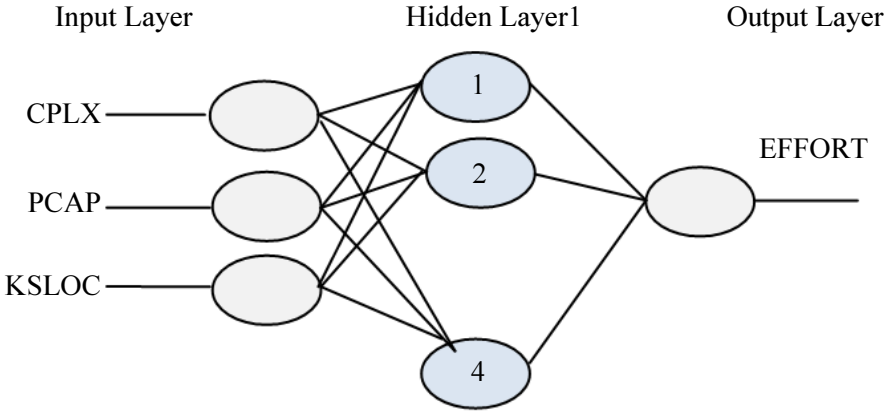

Fig. 4: Implemented radial base function neural network

Experimental Results

Data Collection

In this paper, NASA public dataset is used. The dataset consists of 60 projects data (Singh and Sahoo, 2011). The dataset consists of 17 attributes, but in this paper only four attributes are considered, three for the input, product complexity (CPLX), programmer capability (PCAP) and thousand of source lines of code (KLOC) and one for the output, that is efforts as shown in Table 1. The 60 datasets are actually divided into 45 for training and 15 for testing. This experiment is implemented using NN-Matlab toolbox.

Evaluation Criteria

In this paper, different evaluation criteria are used to measure and compare the actual and estimated efforts error as seen below:

Correlation coefficient (R):

$$R = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2 \sum_{i=1}^{n}(\hat{y}_i - \bar{\hat{y}})^2}} \quad (14)$$

Mean Absolute Error (MAE):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y - \hat{y}| \quad (15)$$

Root Mean Square Error (RMSE):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2} \quad (16)$$

- Relative Absolute Error (RAE):

$$\text{RAE} = \frac{\sum_{i=1}^{n}|y - \hat{y}|}{\sum_{i=1}^{n}|y - \hat{y}|} \quad (17)$$

Table 1: Data set with the actual and estimated efforts

Project No	CPLX	PCAP	KSLOC	Efforts
1	1	1	70.0	278.0
2	2	1	227.0	1181.0
3	2	2	177.9	1248.0
4	2	3	115.8	480.0
5	2	3	29.5	120.0
6	2	3	19.7	60.0
7	2	3	66.6	300.0
8	2	3	5.5	18.0
9	2	3	10.4	50.0
10	2	3	14.0	60.0
11	2	3	16.0	114.0
12	2	3	6.5	42.0
13	2	3	13.0	60.0
14	2	3	8.0	42.0
15	2	2	90.0	450.0
16	2	3	15.0	90.0
17	2	2	38.0	210.0
18	2	2	10.0	48.0
19	2	3	161.1	815.0
20	2	3	48.5	239.0
21	2	3	32.6	170.0
22	2	3	12.8	62.0
23	2	3	15.4	70.0
24	2	3	16.3	82.0
25	2	3	35.5	192.0
26	2	3	25.9	117.6
27	2	3	24.6	117.6
28	2	3	7.7	31.2
29	2	3	9.7	25.2
30	2	3	2.2	8.4
31	2	3	3.5	10.8
32	2	3	8.2	36.0
33	2	3	66.6	352.8
34	2	1	150.0	324.0
35	2	3	100.0	360.0
36	2	2	100.0	215.0
37	2	1	100.0	360.0
38	2	2	15.0	48.0
39	2	3	32.5	60.0

Table 1: Continue

40	2	2	31.5	60.0
41	2	2	6.0	24.0
42	2	3	11.3	36.0
43	2	1	20.0	72.0
44	2	2	20.0	48.0
45	2	2	7.5	72.0
46	2	2	302.0	2400.0
47	2	3	370.0	3240.0
48	2	3	219.0	2120.0
49	2	3	50.0	370.0
50	2	2	101.0	750.0
51	3	1	190.0	420.0
52	2	3	47.5	252.0
53	4	3	21.0	107.0
54	3	1	42.3	2300.0
55	3	2	79.0	400.0
56	5	3	284.7	973.0
57	5	3	282.1	1368.0
58	2	2	78.0	571.4
59	2	2	11.4	98.8
60	2	2	19.3	155.0

- Root relative Squared Error (RRSE):

$$\text{RRSE} = \sqrt{\frac{\sum_{i=1}^{n}(y-\hat{y})^2}{\sum_{i=1}^{n}(y-\hat{y})^2}} \quad (18)$$

where, y and $\hat{y}$ are the actual and the estimated efforts and n is the number of measurements used in the experiment.

Back-propagation performance: In this paper, the constructed back-propagation neural network consists of three layers as mentioned before three inputs that are the Product of Complexity (CPLX), Programmer Capability (PCAP) and thousand of source lines of code (KLOC), we have also two hidden layers the first one consists of 20 neurons and the second one consists of 10 neurons, in addition to one output layer. The correlation coefficient graph can be also shown in Fig. 5. The training and testing performance of the actual and estimated back-propagation neural network can be shown in Fig. 6 and 7. The different statistical results of error estimation functions for the back-propagation neural network at training and testing cases can be shown in Table 2 and 3.

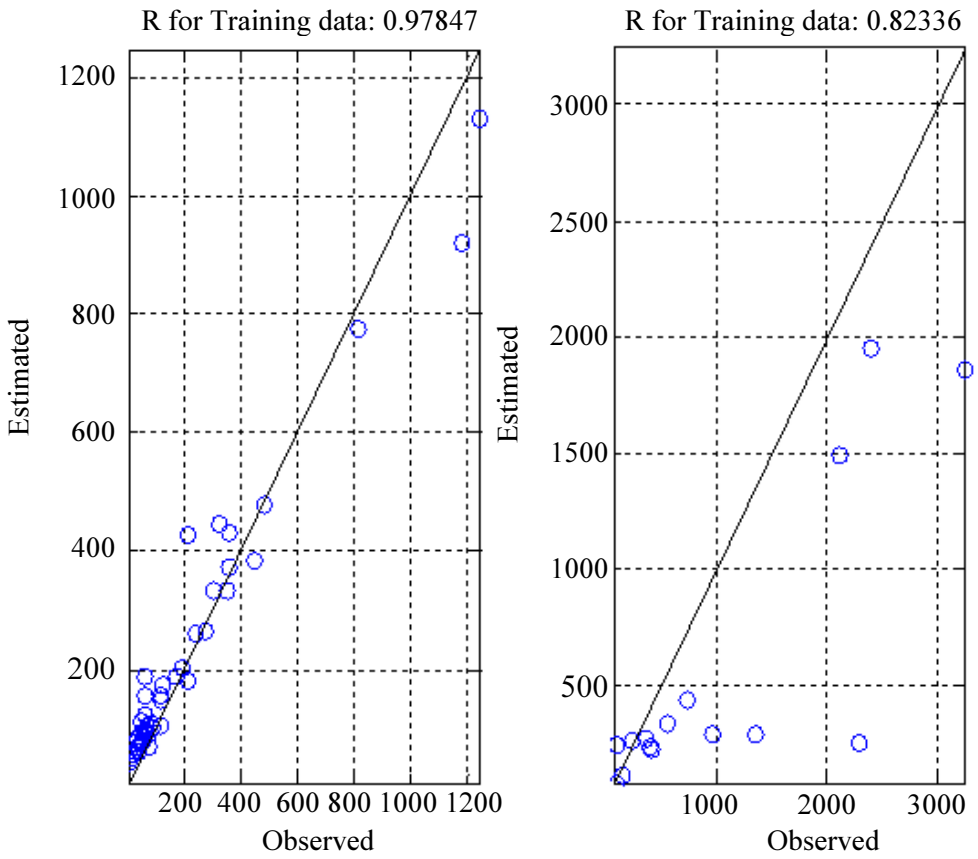

Fig. 5: Correlation Coefficient of the proposed NN

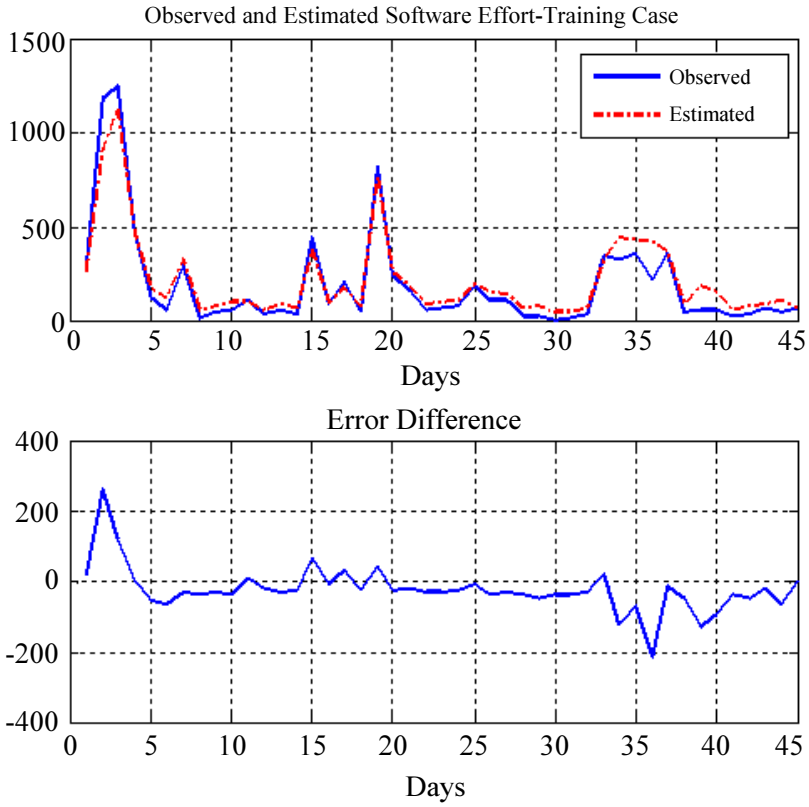

Fig. 6: BPNN Actual and estimated software effort-training

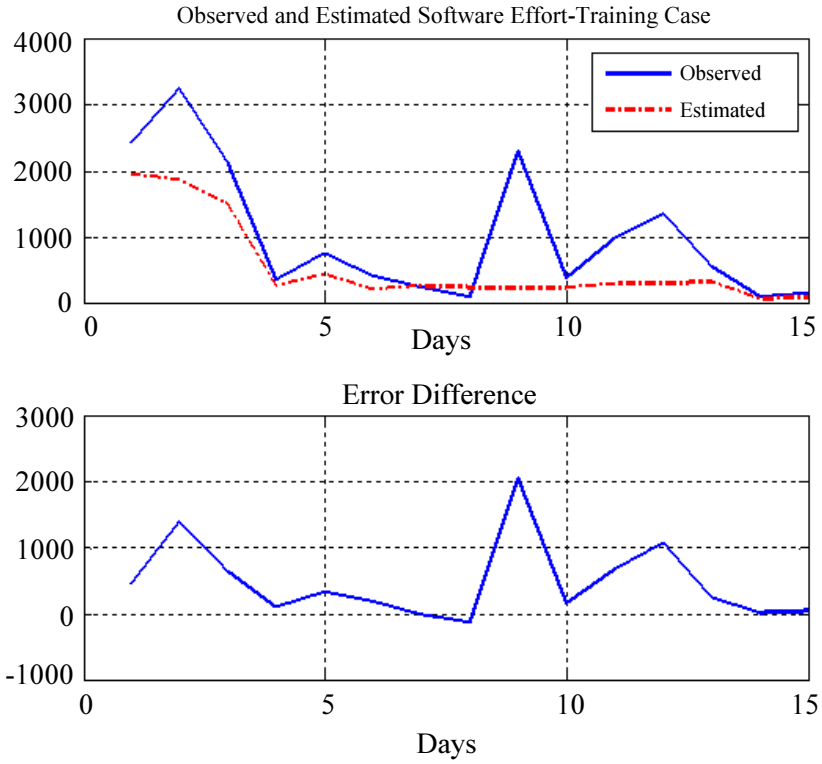

Fig. 7: BPNN Actual and estimated software effort-testing case

According to the results obtained from the plotted figures and the different error evaluation criteria, it's obvious that the performance of the back-propagation neural network in both training and testing cases is better than the redial base function.

Redial Base Function performance: The constructed radial base function consists of two layers, the input layer that contains three inputs; the hidden layer contains four interconnected neurons and the output layer. The RBF correlation coefficient diagram can be shown in Fig. 8.

Table 2: BPNN statistical measurements - training case

Evaluation Criteria	Results
R_Coff	0.978470
MAE	48.756835
RMSE	69.731815
RAE	27.043861
RRSE	25.752251

Table 3: BPNN statistical measurements- testing case

Evaluation Criteria	Results
R_Coff	0.823361
MAE	498.703556
RMSE	755.400655
RAE	59.816353
RRSE	77.468770

The training and testing performance of the actual and estimated radial base function network can be shown in Fig. 9. and 10. The different error estimation statistical functions results for the training and testing cases of the radial base function neural network can be shown in Table 4 and 5.

The performance of the radial base function was satisfactory but not as the back-propagation neural network according to the results obtained from the plotted graphs and the error evaluation criteria.

Table 4: RBF statistical measurements - training case

Evaluation Criteria	Results
R_Coff	0.932815
MAE	50.218568
RMSE	97.577038
RAE	27.854637
RRSE	36.035609

Table 5: RBF statistical measurements - testing case

Evaluation Criteria	Results
R_Coff	0.715623
MAE	502.337178
RMSE	793.533455
RAE	60.252183
RRSE	81.379411

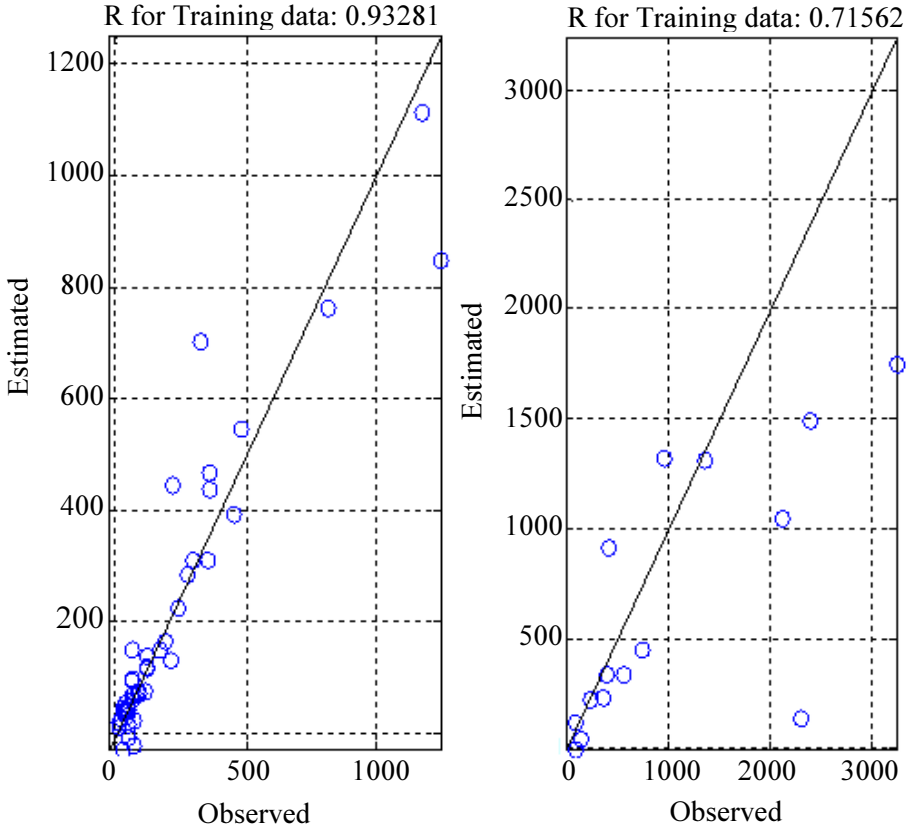

Fig. 8: Correlation Coefficient of the proposed RBF

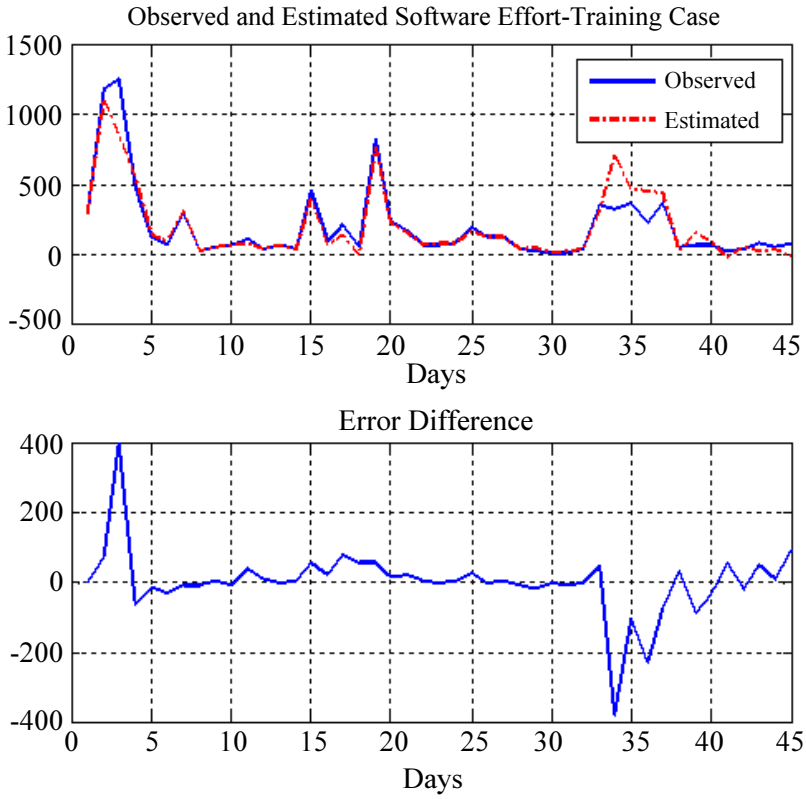

Fig. 9: Actual and estimated software effort-training case RBF

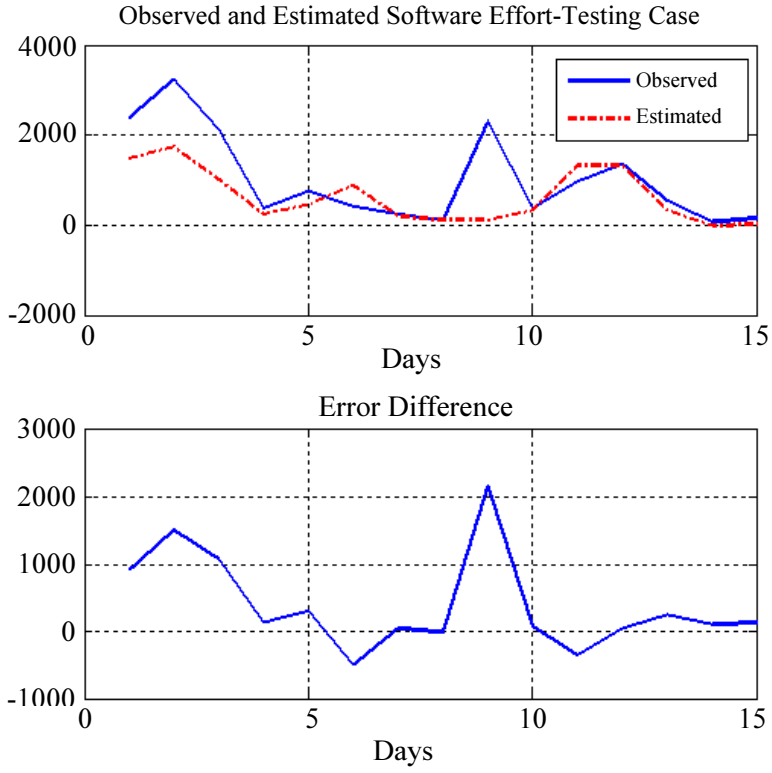

Fig. 10: RBF Actual and estimated software effort-testing case

Conclusion

This paper is introduced to contribute in solving the problem of software effort and cost estimation by comparing two well known neural networks models. These proposed models used to help the project managers in planning, managing and avoiding the risks resulted from the unexpected problems and the delay that may happen during the project period. Actually, a comparison between the both models, back-propagation algorithm (BP) and Redial Base Function (RBF) was presented. In both models, training and testing cases are developed and the Figures that show the actual and estimated efforts are plotted, furthermore, the errors measurements are calculated. According to the results obtained we can say that the performs of the back-propagation neural network in both cases training and testing are better than the radial base function, so the back-propagation algorithm proves, again and again, its efficiency in dealing with such types of problems. My future plan is to extend this work to cover other applications in soft computing techniques.

Acknowledgment

This work has been carried out during sabbatical leave granted to the author Abdel Karim Mohamed Ali Baareh from Ajloun College - Al-Balqa Applied University (BAU) during the first semester of the academic year 2018/2019 and the second semester of the academic year 2018/2019.

Ethics

The author testifies that this article is original and has not been published elsewhere and that ethics were considered for this research.

References

Al-Batah, M.S., N.A. Mat Isa, K.Z. Zamli and K.A. Azizli, 2010. Modified Recursive Least Squares algorithm to train the Hybrid Multilayered Perceptron (HMLP) network. Applied Soft Computing, 10: 236-244. DOI: 10.1016/j.asoc.2009.06.018

Al-Najjar, Y. and A.F. Sheta, 2008. Minutiae Extraction for Fingerprint Recognition. Proceedings of the 5th IEEE International Multi-Conference on System, Signal and Devices (SSD'08), Amman, Jordan. DOI: 10.1109/SSD.2008.4632892

Baareh, A.K.M., A.F. Sheta and K.A. Khnaifes, 2006. Forecasting River Flow in the USA: A Comparison between Auto-Regression and Neural Network Non-Parametric Models. J. Computer Sci., 2: 775-780.

Boehm, B., B. Clark, E. Horowitz and C. Westland, 1995. Cost models for future software life cycle process: COCOMO2. To appear in Annals of Software Engineering Special Volume on Software Process and Product Measurement, Science Publishers, Amsterdam, The Netherlands.

Boehm, W.B., 1981. Software Engineering Economics. Englewood Cliffs, NJ. Prentice-Hall, Australia. ISBN-10: 0138221227.

Boraso, M. and C. Montangero and H. Sedehi, 1996. Software Cost Estimation: An experimental study of model performances. Technical Report del Dipartimento di Informatica. Università di Pisa, Pisa, IT.

Braik, M. and A.F. Sheta, 2011. A new approach for potentially breast cancer detection using extracted features and artificial neural networks. J. Intelligent Computing, 2: 54-71.

Dolado, J.J., 2001. On the problem of the software cost function. Information Software Technol., 43: 61-72.

El-Sayyad, A., A.M. Nassef and A.K. Baareh, 2015. Cardiac arrhythmia classification using boosted decision trees. International Review Computers Software, 10: 280-289. DOI: 10.15866/irecos.v10i3.5359

Feng, W., W. Zhu and Y. Zhou, 2010. The application of genetic algorithm and neural network in construction cost estimate. Proceedings of the 3rd International Symposium on Electronic Commerce and Security Workshops (ISECS'10), July. 29-31, Guangzhou, P.R. China, pp: 151-155.

Ghatasheh, N., H. Faris, I. Aljarah and R.M.H. Al-Sayyed, 2015. Optimizing software estimation models using firefly algorithm. J. Software Eng. Applications, 8: 133-142.

Harikumar, R. and T. Vijayakumar, 2013. A comparison of Elman and Radial Basis Function (RBF) neural networks in optimization of fuzzy outputs for epilepsy risk levels classification from EEG signals. Int. J. Soft Computing Engineering, 2: 295-303.

Hongjun, L., R. Setiono and H. Liu, 1996. Effective data mining using neural networks. IEEE Trans. Knowledge Data Eng., 8: 957-961. DOI: 10.1109/69.553163

Huang, X., L.F. Capretz, J. Ren and D. Ho, 2003. A neuro-fuzzy model for software cost estimation. Proceedings of the 3rd Conference on Quality Software, Nov. 7-7, IEEE Xplore Press, USA. DOI: 10.1109/QSIC.2003.1319094.

Kaushik, A., A.K. Soni and R. Soni, 2012. A comparative study on fuzzy approaches for COCOMO"s effort estimation. Int. J. Computer Theory Engineering, 4: 990-993. DOI: 10.7763/IJCTE

Kemere, C.F., 1987. An empirical validation of Software cost estimation models. Communication ACM, 30: 416-429. DOI: 10.1145/22899.22906

Kumar, S., B.A. Krishna and P.S. Satsangi, 1994. Fuzzy systems and neural networks in software engineering project management. Applied Intelligent, 4: 31-52. DOI: 10.1007/BF00872054

Negnevitsky, M., 2005. Artificial Intelligence: A Guide to Intelligent Systems, 2nd Edn., Addison-Wesley, Boston. ISBN-10: 0-321-20466-2, pp: 435.

Oliveira, A.L.I, P.L. Braga, R.M.F. Lima and M.L. Cornélio, 2010. GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation. Information Software Technol., 52: 1155-1166. DOI: 10.1016/j.infsof.2010.05.009

Radha, V. and N. Nallammal, 2011. Neural network based face recognition using RBFN classifier. Proceedings of the World Congress on Engineering and Computer Science, Oct. 19-21, San Francisco, USA.

Seethe, M., I.V. Muralikrisha and B.L. Deekshatulu, 2007. Artificial neural networks and other methods of image classification. J. Theoretical Applied Information Technol.

Sehra, S.K., S. Brar and N. Kaur, 2011. Soft computing techniques for software project effort estimation. Advanced Computer Mathematical Sci., 2: 160-167.

Shepper, M. and C. Schoeld, 1997. Estimating software project effort using analogies. IEEE Trans. Software Eng., 23: 736-743.

Sheta, A., H. Faris and A.K. Baareh, 2015. Predicting stock market exchange prices for the reserve Bank of Australia using auto-regressive feed-forward neural network model. Int. Rev. Computers Software. DOI: 10.15866/irecos.v10i7.6222

Sheta, A.F. and S. Aljahdali, 2013. Software effort estimation inspired by COCOMO and FP models: A fuzzy logic approach. Int. J. Advanced Computer Applications, 4: 192-197. DOI: 10.14569/issn.2156-5570

Sheta, F., 2006. Estimation of the COCOMO model parameters using genetic algorithms for NASA software projects. J. Computer Sci., 2: 118-123.

Sheta, F., A. Katangur and A.K. Baareh, 2018. Temperature forecasting: A comparison between parametric and non-parametric models. Appl. Math. Inf. Sci., 12: 1099-1108. DOI: 10.18576/amis/120604

Singh, J. and B. Sahoo, 2011. Software effort estimation with different artificial neural network. Proceedings of the IJCA Special Issue on 2nd National Conference- Computing, Communication and Sensor Network CCSN, Foundation of Computer Science, New York, USA, pp: 13-17.

Zadeh, L.A., 1994. Soft computing and fuzzy logic. J. IEEE Software, 11: 48-56. DOI: 10.1109/52.329401