International Journal of Computer Applications Technology and Research
Volume 6–Issue 7, 329-332, 2017, ISSN:-2319–8656

# Software Effort Estimation Using Adaptive Fuzzy-Neural Approach

Riyadh A.K. Mehdi
College of Information Technology
Ajman University

Abstract- Software effort estimation is an important step in software development. It predicts the amount of effort and development time required to build a software system. It is one of the most important tasks and an accurate estimate is vital to the successful completion of the project. Building software effort estimation requires developing sound computational models. This paper investigates the use of fuzzy-neural systems in estimating software effort. A comparison is made with a radial basis neural network. Results obtained based on the China dataset indicates that a hybrid model that combine fuzzy inferencing with neural networks ability to learn from examples provided more accurate results than using neural networks alone.

Index Terms - Software effort estimation; fuzzy inference; datasets; neural networks; and fuzzy-neural systems.

## 1. INTRODUCTION

Estimating software development efforts is one of the critical tasks in managing software development projects. Predicting software development effort with high accuracy is of paramount importance for project managers. However, estimating software development effort is still a challenging problem and one that attracts considerable research. Numerous software effort estimation models have been developed [1, 2, 3, 4]. The conventional models use a mathematical formula to predict project cost based on the estimates of parameters such as project size measured in lines of source code or function points, number of software engineers, and other process and product attributes [5]. Among the software cost estimation techniques, COCOMO (**Co**nstructive **Co**st M**o**del) is the most commonly used algorithmic cost modeling technique because of its simplicity for estimating the effort in person-months for a project at different stages. COCOMO uses a mathematical formula to predict project cost estimation [6]. Non-algorithmic models of software cost estimation based on soft computing approaches such as artificial neural networks (ANN) and fuzzy logic have also been used. Artificial neural networks are good at modeling complex nonlinear relationships. They are massive parallel-distributed processor made up of simple processing units, which can store experimental knowledge and making it available for use [5]. An ANN resembles the brain in two respects [5]: 1) Knowledge is acquired from its environment through a learning process, 2) Interneuron connection weights are used to store the knowledge. On the other hand, fuzzy logic is a mathematical tool for dealing with uncertainty and imprecision information. Fuzzy logic maps an input space to an output space through set of if then rules designed by a human expert in the domain [7]. Fuzzy logic models can be constructed without any data or with little data [8, 9]. This makes fuzzy logic superior over data-driven model building approaches such as neural network, regression and case based reasoning. In addition, fuzzy logic models can adapt to new environment when data become available [10]. Implementing fuzzy system requires that the distinct categories of the different inputs be represented by fuzzy sets which, in turn, are represented by membership functions. The domain of membership function is fixed, usually the set of real numbers, and whose range is the span of positive numbers in the closed interval [0, 1].

## 2. LITERATURE REVIEW

Xu and Khoshgoftaar [11] proposed a fuzzy identification cost estimation model to deal with linguistic data, and automatically generate fuzzy membership functions and rules. Azzeh et al. [12] propose an analogy-based software effort estimation using fuzzy numbers, namely Generalized Fuzzy Number Software Estimation. They compute the similarity between two generalized fuzzy numbers based on their geometric distances, center of gravities and height of the generalized fuzzy numbers, and use fuzzy c-means to cluster the existing software project data. The estimations are conducted with the use of generalized fuzzy number operations and the effort of a project is estimated as a fuzzy number which is defuzzified with the method of center of gravity. Lopez-Martin et al. [13] compare three personal fuzzy logic models to estimate the effort of small software programs, namely triangular, trapezoidal and Gaussian membership functions, with linear regression model. They develop the fuzzy logic and linear regression models using the data gathered from 105 small programs, and then the estimations generated by these models are compared with each other using 20 small programs. Wei Lin et al. [14] showed that a general neuro-fuzzy framework can function with various algorithmic models for improving the performance of software effort estimation. They used a Neuro-Fuzzy model to demonstrate that combining the neuro-fuzzy model with the SEER-SEM effort estimation model produces unique characteristics and performance improvements. They concluded that the neuro-fuzzy features of the model provided their neuro-fuzzy SEER-SEM model with the advantages of strong adaptability with the capability of learning, less sensitivity for imprecise and uncertain inputs, easy to be understood and implemented, strong knowledge integration, and high transparency. Hodgkinson and Garratt [15] introduced the neuro-fuzzy model for cost estimation as one of the important methodologies for developing non-algorithmic models. Their model did not use any of the existing prediction models, as the inputs are size and duration, and the output is the estimated project effort.

Huang et al. [16, 17] proposed a software effort estimation model that combines a neuro-fuzzy framework with COCOMO II. The

parameter values of COCOMO II were calibrated by the neuro-fuzzy technique in order to improve its prediction accuracy. This study demonstrated that the neuro-fuzzy technique was capable of integrating numerical data and expert knowledge.

Xia et al. [18] developed a Function Point (FP) calibration model with the neuro-fuzzy technique, which is known as the Neuro-Fuzzy Function Point (NFFP) model. The objectives of this model are to improve the FP complexity weight systems by fuzzy logic, to calibrate the weight values of the unadjusted FP through the neural network, and to produce a calibrated FP count for more accurate measurements.

Wong et al. [19] introduced a combination of neural networks and fuzzy logic to improve the accuracy of backfiring size estimates. In this case, the neuro-fuzzy approach was used to calibrate the conversion ratios with the objective of reducing the margin of error. The study compared the calibrated prediction model against the default conversion ratios. As a result, the calibrated ratios still presented the inverse curve relationship between the programming languages level and the number of function points, and the accuracy of the size estimation experienced a small degree of improvement. A survey on software effort estimation techniques is given in [20].

The Adaptive network based fuzzy inference system (ANFIS) is a hybrid of a feed forward neural network and a fuzzy inference system. The neural network uses either a pure back propagation gradient descent learning rule, or a hybrid learning rule that uses back propagation and a least squares method [21]. The fuzzy logic component takes into account the imprecision and uncertainty of the system that is being modelled while the neural network component apply its learning algorithm to tune the membership functions of the fuzzy inference system generated [22]. Using this hybrid method, at first an initial fuzzy model along with its input variables are derived with the help of the rules extracted from the input output data of the system that is being modeled. Next the neural network is used to fine tune the rules of the initial fuzzy model to produce the final ANFIS model of the system [22]. In ANFIS the parameters can be estimated in such a way that both the Sugeno and Tsukamoto fuzzy models [23] are represented by the ANFIS architecture.

This paper investigates the effectiveness of using a neuro-fuzzy approach to software effort estimate and how it compares to other approaches.

## 3. RESEARCH METHODOLOGY

In this work, an adaptive neuro-fuzzy inference system based on the Sugeno fuzzy model is used. The following exposition is adapted from [22].

### 3.1 ANFIS Architecture

A typical architecture of ANFIS is depicted in Figure 1. A circle indicates a fixed node, whereas a square indicates an adaptive node [22].

For a first-order Sugeno fuzzy model, a two rules rule base can be expressed as follows:

1. If $x$ is $A_1$ and $y$ is $B_1$, then $f_1 = p_1x + q_1y + r_1$
2. If $x$ is $A_2$ and $y$ is $B_2$, then $f_2 = p_2x + q_2y + r_2$

Let the membership functions of fuzzy sets $A_i$, $B_i$ for $i$=1,2 be $\mu_{A_i}, \mu_{B_j}$. In this work, Gaussian membership functions are used,

$$\mu_{A_i}(x) = \frac{1}{1 + \left( \frac{x - c_i}{a_i} \right)^{2b_i}}$$

In evaluating the rules, a product T-norm (logical and) is chosen. Evaluating the rule premises results in,

$$w_i = \mu_{A_i}(x)\mu_{B_i}(y), \text{ i} = 1,2.$$

Evaluating the implication and the rule consequences gives,

$$f(x,y) = \frac{w_1(x,y) f_1(x,y) + w_2(x,y) f_2(x,y)}{w_1(x,y) + w_2(x,y)}.$$

Leaving the arguments out,

$$f = \frac{w_1 f_1 + w_2 f_2}{w_1 + w_2}$$

The above equation can be rewritten as,

$$f = \overline{w_1} f_1 + \overline{w_2} f_2, \text{ where}$$

$$\overline{w_i} = \frac{w_i}{w_1 + w_2}$$

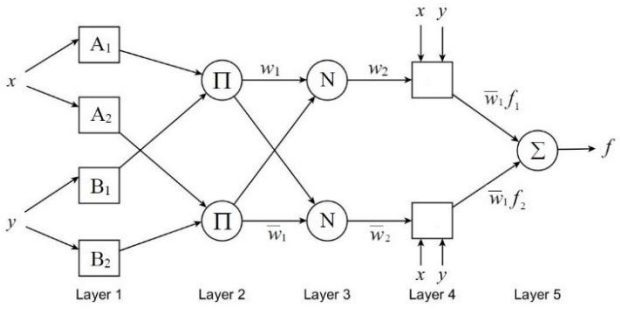


Figure 1. Structure of adaptive neuro-fuzzy inference system.

### 3.2 MATLAB Implementation

1. *genfis2*(*Xin*,*Xout*,*radii*), gen*fis2* generates a Sugeno-type FIS structure using subtractive clustering and requires separate sets of input and output data as input arguments. When there is only one output, *genfis2* may be used to generate an initial FIS for *anfis* training. *genfis2* accomplishes this by extracting a set of rules that models the data behavior. The rule extraction method first uses the subclust function to determine the number of rules and antecedent membership functions and then uses linear least squares estimation to determine each rule's consequent equations. This function returns an FIS structure that contains a set of fuzzy rules to cover the feature space. The arguments for genfis2 are as follows:
   a. *Xin* is a matrix in which each row contains the input values of a data point.
   b. *Xout* is a matrix in which each row contains the output values of a data point.
   c. *radii* is a vector that specifies a cluster center's range of influence in each of the data dimensions, assuming the data falls within a unit hyper box.
2. *anfis*(*trainingData*, *options*)

This function generates a single-output Sugeno fuzzy inference

system (FIS) structure using grid partitioning and tunes the system parameters using the specified input/output training data to adjust the membership functions parameters. This adjustment is made using a backpropagation algorithm either alone, or in combination with a least squares type of method. This allows the fuzzy systems to learn from the data they are modeling. The MATLAB statement: [*fis*,*trainError*,*stepSize*,*chkFIS*,*chkError*] = *anfis*(*trainingData*,*options*) returns the validation data error for each training epoch, *chkError*, and the tuned FIS structure for which the validation error is minimum, *chkFIS*. Using validation data prevents overfitting to training data. To use this syntax, we must specify validation data using *options.ValidationData*.

## 3.3 Software Effort Estimation Datasets

The China dataset (19 attribute, 499 projects, effort in person-hours) is used in this work. The number of records used for training, checking, and testing were 349, 100, and 50 respectively. Because the number of records is inadequate to estimate the parameters of the neuro-fuzzy estimation model, It was not possible to use the datasets below [24]:

- Desharnais (11 attribute, 77 project, effort in person-hours)
- Cocomo81 (18 attribute, 61 project, effort in person-month)
- Maxwell (27 attribute, 62 project, effort in function points)
- Albrecht (8 attribute, 24 project, effort in person-hours)

The evaluation criterion used to assess the estimation accuracy are root mean square error (RMSE), and the mean magnitude (absolute) error (MME) [25]:

$$RMSE = sqrt(\sum_{i=1}^{i=n}(actualEffort(p_i) - predictedEffort(p_i))^2 / n)$$

$$MME = \frac{1}{n}\sum_{i=1}^{i=n}|actualEffort(p_i) - predictedEffort(p_i)|$$

# 4. RESULTS

Figure 2 depicts the performance of the neuro-fuzzy model when run on the testing data (50 records). In Figure 2, the line represent actual effort and the circles represent estimated efforts.

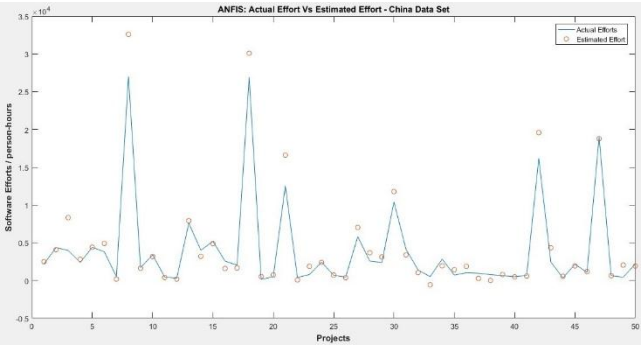


Figure 2. Actual efforts and estimated efforts using ANFIS.

As a comparison, Figure 3 shows the performance of a RBFN network using the same set of testing records.

Table 1 shows the mean absolute error, and root mean square root for the China dataset using the adaptive neuro-fuzzy model and a radial basis function neural network (RBFN) [26].

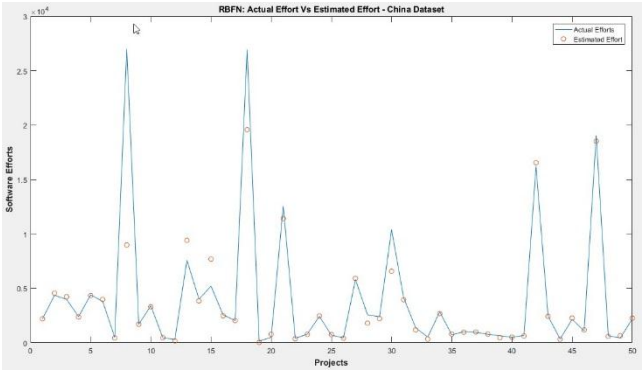


Figure 3. Actual efforts and estimated efforts using RBFN.

**Table 1. Errors obtained using ANFIS and RBFN on the China dataset.**

| Error Type | ANFIS | RBFN |
|---|---|---|
| Root mean square error | 1483 | 2850 |
| Mean magnitude error | 874 | 802 |

The results indicate that although the mean magnitude error for the ANFIS model is slightly higher than that for the RBFN model, they are comparable. However, the root mean square error for the RBFN is almost twice that of the ANFIS model. This indicate while the average error is almost the same for the two models, the estimation of the ANFIS model has much less deviations from the actual values compared with the RBFN model.

# 5. CONCLUSIONS

In this paper, the adaptive neuro fuzzy model was applied to the problem of estimating software developments efforts using the China data set. The results were compared with using a radial basis function neural network on the same data set and on the same testing records. The results indicate while both models are comparable with regard to the mean magnitude error, the ANFIS model has a better performance in the sense that the estimates have a far less deviation that those of the RBFN model.

Future work will investigate the relevance of the various attributes in determining the size of efforts required in developing software so that standardized set of attributes can be used in collecting data sets.

# 6. REFERENCES

[1]. S. A. Abbas, A. R. Liao, A. Azam, 2012. "Cost Estimation: A Survey of Well-Known Historic Cost Estimation Techniques," *Journal of Emerging Trends in Computing and Information Sciences*, vol. 4, no. 1, pp 612-636.

[2]. R. Malhotra, A. Jain, 2011. "Software Effort Prediction using Statistical and Machine Learning Methods," *International Journal of Advanced Computer Science and Applications*, vol. 2, no. 1.

[3]. Abbas Heiat, 2002. "Comparison of Artificial Neural Network and Regression Models for Estimating Software Development Effort," *Information and Software Technology*, vol. 44, pp. 911-922.

[4]. R. Sharma, 2013. "Survey: Non Algorithmic Models for Estimating Effort," *European International Journal of Science and Technology*, vol. 2, no. 3.

[5]. Kaushik, A. Chauhan, D. Mittal, and S. Gupta, 2012. "COCOMO Estimates Using Neural Networks," International *Journal of Intelligent Systems and Applications*, vol. 9, pp. 22-28.

[6]. Reddy, and K. Raju, "An Optimal Neural Network Model for Software Effort Estimation," *International Journal of Software Engineering*, vol.12 no.1, pp. 66-78.

[7]. Zadeh, L.A., 2002. From Computing with numbers to computing with words-from manipulation of measurements to manipulation of perceptions. International Journal of Applied Mathematics and Computer Science 12(3), 307-324.

[8]. MacDonell, S.G., Gray, A.R., Calvert, J.M. 1999. Fuzzy Logic for Software Metric Practitioners and Researchers.In : The Proceedings of the 6th International Conference on Neural Information Processing ICONIP, Perth, pp. 308-313.

[9]. Ryder, J. 1998. Fuzzy Modeling of Software Effort Prediction. In: Proceeding of IEEE Information Technology Conference, Syracuse, New York, pp. 53-56.

[10]. Sailu, M.O., Ahmed, M., AlGhamdi, J. 2004. Towards Adaptive Soft Computing Based Software Effort Prediction. In: Fuzzy Information, Processing NAFIPS, pp, 16-21.

[11]. Xu, Z., Khoshgoftaar, T.M., 2004. "Identification of fuzzy models of software cost estimation," Fuzzy Sets and Systems, vol. 145 (1), pp. 141-163.

[12]. Azzeh, M., Neagu, D. Cowling, P.I., 2011. "Analogy-based software effort estimation using Fuzzy numbers," Journal of Systems and Software, vol. 84 (2), pp. 270-284.

[13]. Lopez-Martin, C., Yanez-Marquez, C., Gutierrez-Tornes, A., 2008. "Predictive accuracy comarison of fuzzy models for software development effort of small programs," Journal of Systems and Software, vol. 81 (6), pp. 949-960.

[14]. Wei Lin Du, Danny Ho, Luiz Fernando Capretz, 2010. Improving Software Effort Estimation Using Neuro-Fuzzy Model with SEER-SEM, Global Journal of Computer Science and Technology, Vol. 10 Issue 12, pp. 51-63.

[15]. Hodgkinson, A. C. and Garratt, P. W. 1999. A NeuroFuzzy Cost Estimator. Proc. 3rd Int Conf Software Engineering and Applications (SAE): 401–406

[16]. Huang, X., Ho, D., Ren, J., and Capretz, L. F. 2005. A Soft Computing Framework for Software Effort Estimation. Soft Computing: 170–177

[17]. Huang, X., Ho, D., Ren, J., and Capretz, L. F. 2006. Improving the COCOMO Model Using A Neuro-Fuzzy Approach. Applied Soft Computing: 29–40

[18]. Xia, W., Capretz, L. F., Ho, D., and Ahmed, F. 2008. A New Calibration for Function Point Complexity Weights. International and Software Technology, Vol. 50, Issue 7-8: 670–683

[19]. Wong, J., Ho, D., and Capretz, L. F. 2008. Calibrating Functional Point Backfiring Conversion Ratios Using Neuro-Fuzzy Technique. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, Vol. 16, No. 6: 847 – 862

[20]. Karunakaran, and Sreenath, 2015. Survey on Software Effort Estimation Technique – A Review International Journal of Scientific & Engineering Research, Volume 6, Issue 12.

[21]. J.S. Jang, 1999. "Anfis: Adaptive-network-based fuzzy inference system," IEEE Trans. Syst., Man, Cybern., vol. 23, pp. 665--685, Mar. 1993 [2] W. H. Wolberg and O.L. Mangasarian: "Multisurface method of pattern separation for medical diagnosis applied to breast cytology", Proceedings of the National Academy of Sciences, U.S.A., Volume 87, pp 9193-9196.

[22]. M. Buragohain, 2008. Adaptive Network based Fuzzy Inference System (ANFIS) as a Tool for System Identification with Special Emphasis on Training Data Minimization, PhD Thesis, Indian Institute of Technology Guwahati, India.

[23]. Y. Tsukamoto, 1979. M. M. Gupta, R. K. Ragade, and R. R. Yager, "An approach to fuzzy reasoning method," in Advances in Fuzzy Set Theory and Application, M. M. Gupta, R. K. Ragade, and R. R. Yager, Eds., North-Holland, Amsterdam, pp. 137–149.

[24]. S. J. Shirabad, and T. J. Menzies, 2005. "The PROMISE Repository of Software Engineering Databases," School of Information Technology and Engineering, University of Ottawa, Canada. Available: http://promise.site.uottawa.ca/SERepository.

[25]. S. Conte, 1986. H. Dunsmore, and V. Shen, Software Engineering, Metrics and Models. Benjamin/Cummings.

[26]. Riyadh A.K. Mehdi, 2015. "Software Defect Prediction Using Radial Basis and Probabilistic Neural Networks", International Journal of Computer Applications and Research, Volume 5, Issue 5, pp. 260-265.