SOFTWARE EFFORT ESTIMATION USING NEURO-FUZZY APPROACH BASED ON LEVEN- MARQUARDT ALGORITHM

Arvind Sharma

M.Tech(CSE),

Lovely Professional University.

Dalwinder Singh Salaria

Assistant Professor,

Lovely Professional University.

Abstract: Software effort estimation is the method of estimating the effort and expense required to build computer. Efficient software development is focused on an accurate estimate of effort. When an effective and appropriate approach is not used for estimation, it may lead to a failure of the project. There are a number of methods available for estimating the effort. This paper discusses the Leven-Marquardt algorithm used to train neural networks. The suggested methodology used the combination of neural networks and fuzzy logic as a neuro-fuzzy approach to the calculation of machine effort.

Keywords: Software Effort Estimation, Neural Networks, Feed Forward, COCOMO, Leven-marquardt, BPNN

I. INTRODUCTION

One of the greatest problems the software industry facing is precisely measuring software effort. Software effort estimation (Minku et al., 2013) is an essential phase of system life cycle growth, as it may affect the performance of software projects if project designers estimate projects incorrectly. Estimation is the method of making an estimation or approximation, which is a value that can be used for any reason even though the input data is incomplete, uncertain or unreliable. The estimate is critical as it gives the project team some confidence in the effort and time required to prepare ahead for the project. In software engineering (Hamza et al., 2013), effort estimation is the process of estimating the most reasonable amount of effort required to create or sustain software based on incomplete, unpredictable and noisy feedback in terms of money and human-hours. The estimation of the effort required for a software development project is extremely important for the success of the overall solution delivery. Effort estimates may be used as inputs to project plans, iteration plans, budgets, investment analysis, pricing processes and bidding rounds. The four key stages in the estimation of software projects are (Saroha and Sahu, 2015):

- Assess the size of the product to be developed.
- Estimate the effort by person-months or person-hours.
- Estimate the timetable in the calendar month.
- Estimate the cost of the project in the agreed currency.

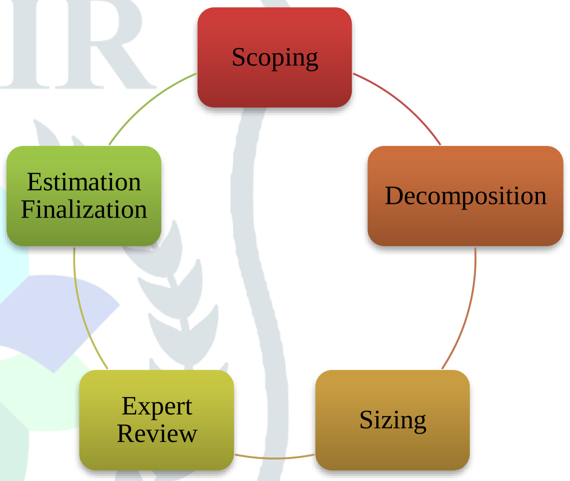


Figure 1: An Estimation Process

II. LITERATURE REVIEW

(Lee et al . 2013) presented the MUSIC algorithm for a simultaneous calculation of an azimuth and elevation in a strict Levenberg-Marquardt (LM) edition. Using the LM algorithm a multivariable nonlinear optimization is performed with the MUSIC algorithm to estimate an azimuth / AOA elevation (angle-of-arrival). In terms of accuracy and the computing complexity, the efficiency of the LM implementation of the MUSIC algorithm is compared with MUSIC's regular algorithm.

The empirical analysis of GCC controls, including optimum and sub-optimal controls, is discussed in **(Fu et al., 2015)**. It looks at how FATT can be used in the LM algorithm. The data show that RNNs are better than traditional backpropagation using a time algorithm when the LM and the FATT algorithms combined. A new RNN controller with improved input structures is proposed to approximate the ideal optimal controller in order to resolve the inapplicability of the optimum GCC controller on realistic conditions.

(Sachan et al. 2016) suggested a software effort estimation (SEE) model with a simplified genetic algorithm. The simplified GA is used to optimize the fundamental

COCOMO model parameters. A well-known NASA dataset analyzes the performance analysis of the proposed model. The simplified genetic algorithm is used to build a model for measuring machine effort. The proposed simplified GA in Manhattan is 6.7125, stronger than fundamental COCOMO. COCOMO provides an better estimate compared to core COCOMO with simpler GA tuning parameters.

(Rizwabi and Jain, 2016) proposed usage of a technologically advanced model based on ANN (Artificial Neural Network) using the Multi Layered Feed Forward Neural Network, which provides training with the Back Propagation Training Method. COCOMO data set is used to test and train the network. Mean-Square-Error (MSE) and Mean Magnitude Relative-Error (MMRE) are used as performance measurement indices.

The objective of **(Pandey et al., 2019)** was to present a practical approach to the determination of the most appropriate model for the quickest estimation of SAMOA dataset applications using four popular estimation techniques (MLR, MLP-NN, GA and Naïve). The paper assessed the popular techniques for estimating comfort empirically. The SAMOA data set consisting of data on mobile app development of 19 projects has been used to meet this objective. Genetic Algorithm has achieved the best result of all four methods, both PRED (25) and MRE. Genetic Algorithm based technique is found to be suitable for mobile app efort estimation using the SAMOA dataset.

(Behera et al., 2020) proposed an efficient approach to fault localization using a backpropagation neural network and the authors used the actual number of times the statement is executed to train the network. The Siemens Suite approach is investigated in which the results show that there is an average increase in efficiency of 35 per cent over the existing BPNN.

III. SOFTWARE ESTIMATION MODELS

The Estimate is a prediction or a rough idea of how much effort would be needed to complete a defined task. The effort could have been time or cost. Various models (Laqrichi et al., 2015) are defined to determine the time / cost effort, which is discussed below:

- **Leven-Marquardt Scheme**

The Levenberg-Marquardt (LM) training method is the most efficient method for the transmission of neural networks in terms of training accuracy. This method (Khan et al., 2013) is well known and popularly described in the literature on neural networks. However, its implementation presents some difficulties due to the specific shape of the cost function and the large number of variables. The Levenberg – Marquardt algorithm developed by Kenneth Levenberg and Donald Marquardt provides a numerical solution to the problem of minimizing non-linear function. It's fast and it has a stable convergence. In the field of artificial neural networks, this algorithm is suitable for the training of small and medium-sized problems. The fundamental idea of the Levenberg-Marquardt algorithm is that a combined workout is carried out: the Levenberg-Marquardt algorithm changes around the area with complex curvature to the steeper descent, until the local curvature can make a quadratic approximation and approximate Gauss-Newton algorithm, which can significantly speed up the convergence.

- **Back Propagation Neural Network (BPNN)**

The core of neural network training is back-propagation. It is the way weight of a neural net is modified based on the error rate in the previous cycle. A neuron, which stores and processes information is the basic component of BPNN. This is a modern supervised learning approach used for both linear and non-linear classifications. BPNN (Wang et al., 2019) is a controlled algorithm in which the error difference is propagated from the desired result to the measured output. Under apprenticeship, the process is repeated to reduce the bug by changing the weights due to back spread errors. Due to weight changes, secret units identify their weight as key characteristics of the mission domain. There are three layers of BPNN:

  - Input Layer
  - Hidden Layer
  - Output Layer

  The difficulty of the problem depends on the number of hidden layers and on the number of hidden units on each hidden layer. The advantages of BPNN are based on previous studies:

  - BPNN supports high speed classification.
  - BPNN can be used for linear as well as non linear classification.
  - BPNN supports multi class classification.

- **Constructive Cost Model (COCOMO) Model**

COCOMO was developed by software engineer Barry Boehm in 1981 and is one of the most commonly used cost estimating software model. Concerning efforts (resources needed to complete project work) and timetable (time taken to complete project work), the COCOMO (Sachan et al. ,2020) estimates the cost of software product creation based on the sized software product. It estimates the number of MMs required for full software product development. The models of COCOMO are as follows:

Model 1: Basic COCOMO Model

The Basic COCOMO is a static, single-view model that measures the effort (and cost) in software development based on the system size of projected code lines (LOC).

Model 2: Intermediate COCOMO Model

The Intermediate COCOMO model calculates software development effort as a function of program size and a set of ‘cost drivers’ that include subjective assessment of product, hardware, personnel and project attributes.

Model 3: Advanced COCOMO Model

The Advanced COCOMO model incorporates all the features of the intermediate version with an assessment of the impact of the cost driver on each step of the softc i6t5ware engineering process, e.g. analysis, design, etc.

IV. SIMULATION RESULTS

One of the biggest challenges facing the software industry is precisely predicting software effort. If the estimate is not accurate, it may lead to a failure of the project. Effort estimation is a very challenging task for the organization of software development, because if estimation of effort is not correct, the organization must suffer. This paper uses a neural feed-forward network that will be trained using the Leven-Marquardt algorithm.

- *Mean Magnitude of Relative Error (MMRE)*

The mean magnitude of relative error, MMRE, is probably the most widely used assessment criterion for assessing the performance of competing software prediction models. The performance of the different algorithms will be compared on the basis of the Mean magnitude of the Relative Error (MMRE) calculated as:

$$\text{MMRE} = \frac{(Actual\ Effort) - (Estimated\ Effort)}{Actual\ Effort}$$

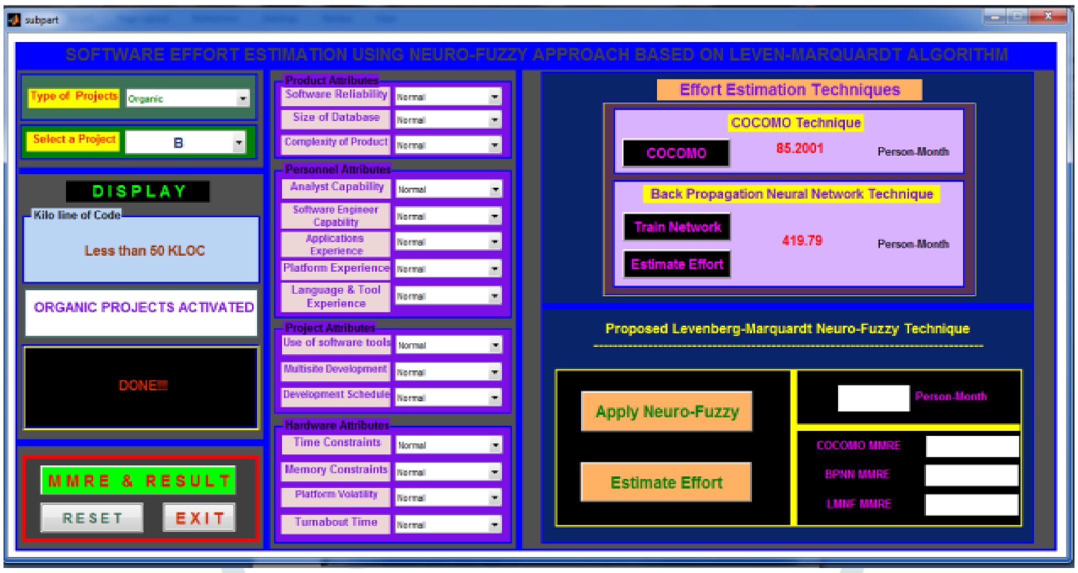


Figure 2: Effort estimation using Back Propagation Neural Network Technique

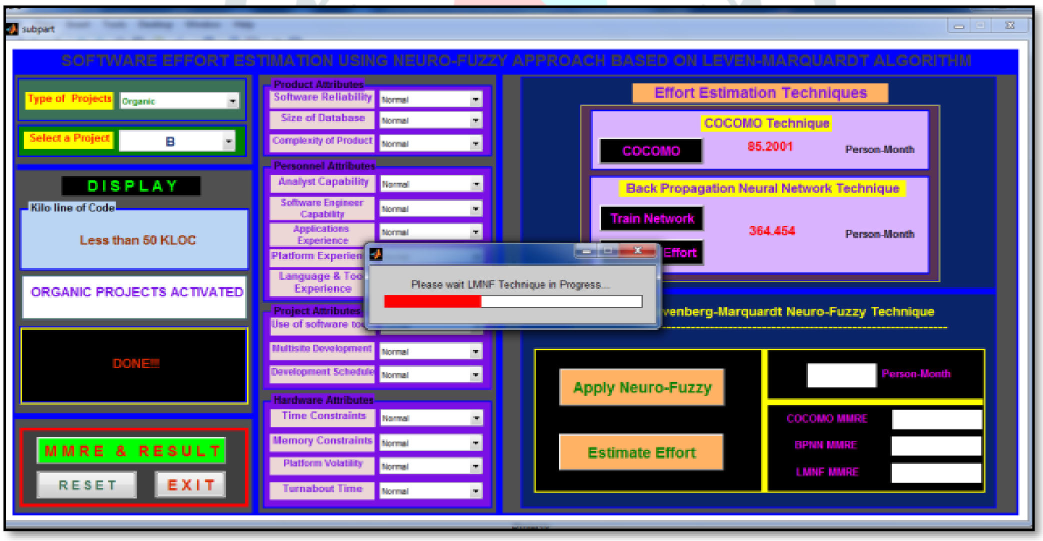
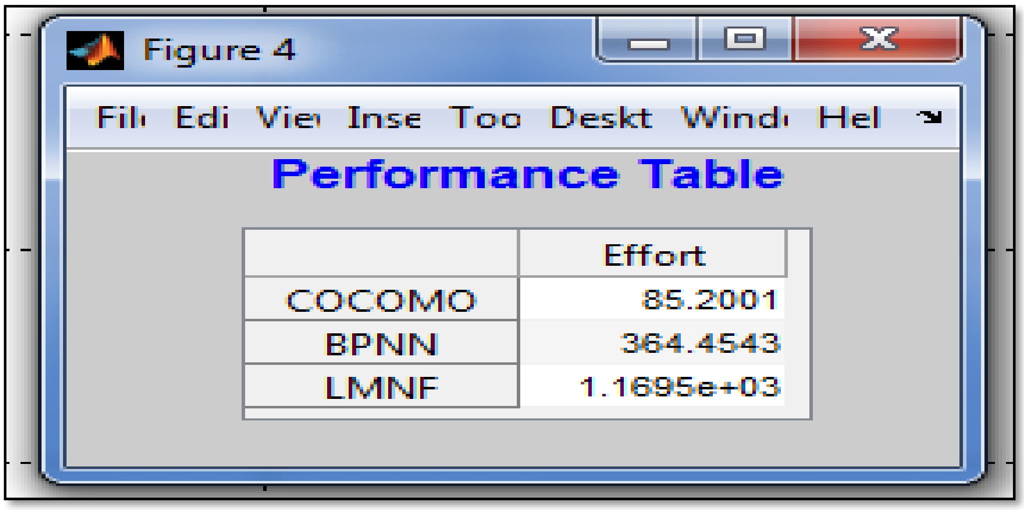


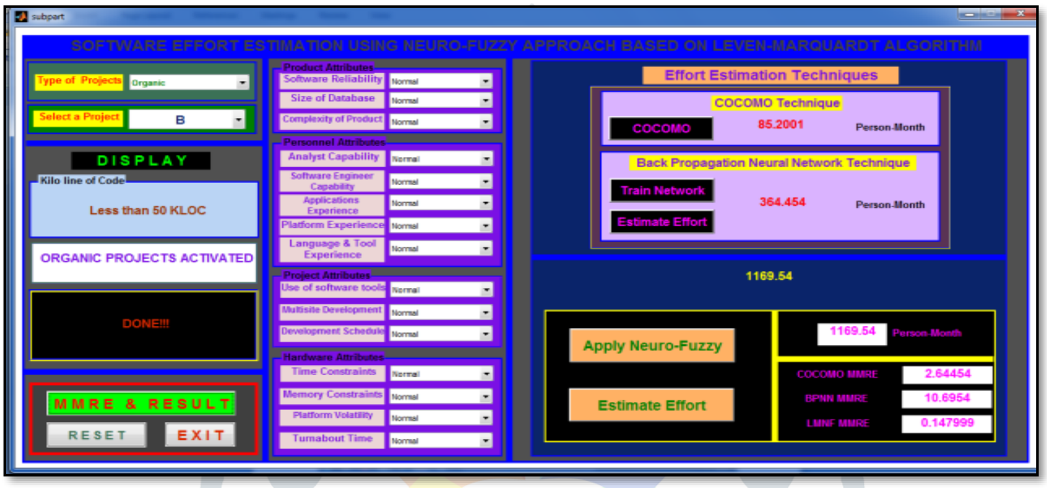
Figure 3: LMNF Algorithm to estimate effort



Performance Table

	Effort
COCOMO	85.2001
BPNN	364.4543
LMNF	1.1695e+03

Figure 4: Performance Table of Three Schemes



SOFTWARE EFFORT ESTIMATION USING NEURO-FUZZY APPROACH BASED ON LEVEN-MARQUARDT ALGORITHM

Type of Projects: Organic
Select a Project: B

Kilo line of Code: Less than 50 KLOC

ORGANIC PROJECTS ACTIVATED

DONE!!!

Product Attributes	
Software Reliability	Normal
Size of Database	Normal
Complexity of Product	Normal
Personnel Attributes	
Analyst Capability	Normal
Software Engineer Capability	Normal
Applications Experience	Normal
Platform Experience	Normal
Language & Tool Experience	Normal
Project Attributes	
Use of software tools	Normal
Multisite Development	Normal
Development Schedule	Normal
Hardware Attributes	
Time Constraints	Normal
Memory Constraints	Normal
Platform Volatility	Normal
Turnabout Time	Normal

Effort Estimation Techniques

COCOMO Technique
COCOMO 85.2001 Person-Month

Back Propagation Neural Network Technique
364.454 Person-Month

1169.54

1169.54 Person-Month

COCOMO MMRE	2.64454
BPNN MMRE	10.6954
LMNF MMRE	0.147999

Figure 5: Mean Magnitude Relative Error (MMRE)

V. CONCLUSION

Before any software project is introduced, software development effort estimation is a key task. The main objective of this study is to improve the accuracy of the estimation of software effort during the initial stages of the software development process. In the proposed work, Mean Magnitude of Relative Error (MMRE) is used as a performance metric to calculate the performance of the three techniques, i.e. COCOMO, BPNN, LMNF. The dataset used for this study is NASA 93 to check the improvement of algorithms. Figure 4 shows clearly that the MMRE is lower for LMNF compared to the other two techniques.

REFERENCES

Behera, H. S., Nayak, J., Naik, B., & Pelusi, D. (Eds.). (2020). *Computational Intelligence in Data Mining*. Advances in Intelligent Systems and Computing.

Fu, X., Li, S., Fairbank, M., Wunsch, D. C., & Alonso, E. (2015). Training Recurrent Neural Networks With the Levenberg–Marquardt Algorithm for Optimal Control of a Grid-Connected Converter. *IEEE Transactions on Neural Networks and Learning Systems*, 26(9), 1900–1912.

Hamza, H., Kamel, A., & Shams, K. (2013). Software Effort Estimation Using Artificial Neural Networks: A Survey of the Current Practices. *2013 10th International Conference on Information Technology: New Generations.*

Kaur, H., & Salaria, D. S. (2013). Bayesian Regularization based Neural Network Tool for Software Effort Estimation. *Global Journal of Computer Science & Technology*. 13(2), 44-50.

Khan, N., Gaurav, D., & Kandl, T. (2013). Performance Evaluation of Levenberg-Marquardt Technique in Error Reduction for Diabetes Condition Classification. *Procedia Computer Science*, 18, 2629–2637.

Laqrichi, S., Marmier, F., Gourc, D., & Nevoux, J. (2015). Integrating uncertainty in software effort estimation using Bootstrap based Neural Networks. *IFAC-PapersOnLine*, 48(3), 954–959.

Lee, J.-H., Cho, S.-W., & Jeong, S.-H. (2013). Application of the Levenberg-Marquardt Scheme to the MUSIC Algorithm for AOA Estimation. *International Journal of Antennas and Propagation*, 2013, 1–8.

Minku, L. L., & Yao, X. (2013). Ensembles and locality: Insight on improving software effort estimation. *Information and Software Technology*, 55(8), 1512–1528.

Pandey, M., Litoriya, R., & Pandey, P. (2019). Validation of Existing Software Effort Estimation Techniques in Context with Mobile Software Applications. *Wireless Personal Communications.*

Rijwani, P., & Jain, S. (2016). Enhanced Software Effort Estimation Using Multi Layered Feed Forward Artificial Neural Network Technique. *Procedia Computer Science*, 89, 307–312.

Sachan, R. K., Nigam, A., Singh, A., Singh, S., Choudhary, M., Tiwari, A., & Kushwaha, D. S. (2016). Optimizing Basic COCOMO Model Using Simplified Genetic Algorithm. *Procedia Computer Science*, 89, 492–498.

Sachan, R. K., & Kushwaha, D. S. (2020). Anti-Predatory NIA Based Approach for Optimizing Basic COCOMO Model. *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 710-715.

Saroha, M., & Sahu, S. (2015). Tools & methods for software effort estimation using use case points model — A review. *International Conference on Computing, Communication & Automation*, 874-879.

Usman, M., Britto, R., Damm, L.-O., & Börstler, J. (2018). Effort estimation in large-scale software development: An industrial case study. *Information and Software Technology*, 99, 21–40.

Wang, J., Kong, Y., & Fu, T. (2019). Expressway crash risk prediction using back propagation neural network: A brief investigation on safety resilience. *Accident Analysis & Prevention*, 124, 180–192.