Efficient Software Cost Estimation using Neuro-Fuzzy Technique

Mr. Ihtiram Raza Khan*, Prof Afshar Alam, Ms. Huma Anwar**
*Department Of Computer Science, Jamia Hamdard, New Delhi – 62, India
**Institute of Productivity & ManagementGhaziabad, India

Abstract
Software estimation accuracy is among the greatest challenges for software developers. As Neuro-fuzzy based system is able to approximate the non-linear function with more precision so it is used as a soft computing approach to generate model by formulating the relationship based on its training. The approach presented in this paper is independent of the nature and type of estimation. In this paper, COCOMO and function point is used as algorithmic model and an attempt is being made to validate the soundness of Neuro fuzzy technique using NASA and ISBSG project data.

Keywords: Soft Computing, Neuro-fuzzy, Effort estimation, COCOMO, Mean Magnitude Relative Error, LOC, Function point.

1. Introduction
Accurate software estimation such as size estimation, effort estimation, cost estimation, quality estimation and risk analysis is a major issue in software project management. If the estimation is not properly done, it may result in the failure of software project. Accurate software estimation can provide powerful assistance for software management decisions. The principal challenges are 1) the relationships between software output metrics and contributing factors exhibit strong complex nonlinear characteristics; 2) measurements of software metrics are often imprecise and uncertain; 3) difficulty in utilizing both expert knowledge and numerical project data in one model. To solve software estimation problems, soft computing framework is based on the “divide and conquer” approach[2, 3]. Literature is available in Haykin S, Neural Networks: A Comprehensive Foundation[4] and Zadeh L A, Fuzzy Logic[5]. Huang X, Ho D, Ren J, Capretz L have given an insight into A Soft Computing Framework for Software Effort Estimation”[6]. Ali Idri and M.Khoshgoftaar have published a paper on “Can Neural nets be easily interpret in Software Cost Estimation”[7].

The soft computing framework, or NF Model as presented in Figure 1, consists of the following components[6]:

- Pre-Processing Neuro-Fuzzy Inference System (PNFIS) used to resolve the effect of dependencies among contributing factors of the estimation problem, and to produce adjusted rating values for these factors.
- Neuro-Fuzzy Bank (NFB) used to calibrate the contributing factors by mapping the adjusted rating values for these factors to generate their corresponding numerical parameter values.
- Module that applies an algorithmic model relevant to the nature of the estimation problem to produce one or more output metrics.
Where:  
- $N$ is the number of contributing factors,
- $M$ is the number of other variables in the Algorithmic Model,
- $RF$ is Factor Rating,
- $ARF$ is Adjusted Factor Rating,
- $NFB$ is the Neuro-Fuzzy Bank,
- $FM$ is Numerical Factor/Multiplier for input to the Algorithmic Model,
- $V$ is input to the Algorithmic Model,
- and $Mo$ is Output Metric.

The Effort metric is given by the following equation:

$$Effort = A \times (\text{Size}) \times \frac{B+0.01}{\sum_{i=1}^{n} SF_i} \prod EM_i$$

The Neuro Fuzzy Model provides a novel and inventive method for estimation that makes improved use of both historical project data and available expert knowledge, by uniquely combining certain aspects of relatively newer estimation techniques (e.g., neural networks and fuzzy logic) with certain aspects of more conventional estimation models (e.g., algorithmic models such as COCOMO and Function Point Analysis), to produce more accurate estimation results. One big advantage is that the architecture is inherently independent of the choice of algorithmic models and nature of the estimation problems.

This model has learning and adaptation ability, integrates the capability of expert knowledge, project data and parametric algorithmic models, and provides robustness to imprecise and uncertain inputs. It also has good interpretability and high accuracy. This promises to be a good and latest research trend in software engineering i.e. to perform estimation by combining Neuro-fuzzy technique with an algorithmic model.

3.2. Applying It To Function Point

The concepts of Function Point is being discussed here, whose aims is to fit specific software application, to reflect software industry trend and to improve cost estimation. Neuro-Fuzzy is a technique which incorporates the learning ability from neural network and the ability to capture human knowledge from fuzzy logic. Function Points (FP) is an ideal software size metric to estimate cost since it can be obtained in the early development phase, such as requirement, measures the software functional size from user’s view, and is programming language independent.

The significant relationship between the software size and cost has been recognized for a long time. In the classical view of cost estimation process (Figure ), the outputs of effort and duration are estimated from software size as the primary input and a number of cost factors as the secondary inputs. There are mainly two types of software size metrics: Source Lines of Code (SLOC) and Function Points. SLOC is a natural artifact that measures software physical size but it is usually not available until the coding phase and difficult to have the same definition across different programming languages. Function Points is an ideal software size metric to estimate
cost since it can be obtained in the early development phase, such as requirement, measures the software functional size, and is programming language independent. Calibrating Function Points incorporates the historical information and gives a more accurate view of software size. Hence more accurate cost estimation comes with a better software size metric.

The paper proposes an approach to estimate cost using Function Points with Neuro-Fuzzy technique. The model overview and two parts of the model: fuzzy logic part and neural network part are described here. The block diagram shown in Figure 4 gives an overview of this approach. The project data provided by ISBSG is imported to extract an estimation equation and to train the neural network. An estimation equation is extracted from the data set by statistical regression analysis. Fuzzy logic is used to calibrate Function Points complexity degree to fit specific application. Neural network calibrates UFP weight values to reflect the current software industry trend by learning from ISBSG data.

3.2.1 Fuzzy Logic Part

The fuzzy logic part calibrates the Function Points complexity degree to fit the specific application. A fuzzy logic system is constructed based on the fuzzy set, fuzzy rules and fuzzy inference. The input fuzzy sets are to fuzzify the component associated file numbers and the output fuzzy set are to fuzzify the complexity classification. The fuzzy rules are defined in accordance with the original complexity weight matrices. The fuzzy inference process using the Mamdani approach is applied based on the fuzzy sets and fuzzy rules.

3.2.2 Neural Network Part

The neural network part is aiming at calibrating Function Points to reflect the current software industry trend. By learning from ISBSG data repository, this part is believed to achieve the calibration goal. The neural network is constructed to receive 15 UFP breakdowns as inputs to give the work effort as the desired output. A back-propagation learning algorithm is conducted in order to minimize the prediction difference between the estimated and actual efforts (We will be checking the performance using RBF network also). An effort estimation equation is extracted based on the data subset using statistical regression analysis. The equation in the form of $\text{Effort} = A \cdot \text{UFP}^B$ is achieved with the help of regression techniques.

4. Validation Of Both Approaches (Cocomo And Function Point)

The validation results of the experiments are to be assessed by Mean Magnitude Relative Error (MMRE) for estimation accuracy. MMRE is defined as: for $n$ projects,

$$\text{MMRE} = \frac{1}{n} \sum_{i=1}^{n} (\frac{|\text{Estimated}_i - \text{Actual}_i|}{\text{Actual}_i})$$

RMSE is another frequently used measure of differences between values predicted by a model and the values actually observed.

PRED(N) is the third criteria used for comparison and this reports the average percentage of estimates that were within N% of the actual values.

The COCOMO '81 project data, ISBSG project data and NASA project data will be used to validate these approaches.
5. Proposed Neuro Fuzzy Tool

It is proposed to build prototypes on the NF COCOMO and NF Function Point products as part of our initial research effort. The tools or final products will adopt the following generic architecture, as depicted in Fig 5.

There will be a common front-end GUI, back-end database, and training algorithm. Various types of estimation will be simple plug-in of algorithmic models to the NFA engine, and the effort for additional extension will be small.

6. Conclusion

This paper has talked about a general soft computing technique for software estimation. The technique has been checked against the algorithmic models like COCOMO and function point. On paper the results promise to be better than previous approaches mathematically with MMRE less than 15% in the case of Function point and improvement of accuracy in larger proportion in the case of COCOMO. For better and accurate results, the process of developing a prototype is on and it should be able to calculate the accuracy and MMRE more effectively. We should be able to experiment with Effort equation \((E = a \cdot (LOC)^b + c \cdot ME^d + e)\) and simulate the results with even better results.

7. Future Scope

Developing prototype is underway. My work will be to apply the above concept to NF COCOMO, NF Function point, NF SLIM, NF Size, NF Defect and so forth. The research will be extended to important estimates which are required in Software project management and might apply to Object oriented systems too, a domain which no one has ventured so far.

8. References


