A Neuro Fuzzy Algorithm to Compute Software Effort Estimation

By N. Shivakumar, N. Balaji & K. Ananthakumar
Thiagarajar College of Engineering, India

Abstract- Software Effort Estimation is highly important and considered to be a primary activity in software project management. The accurate estimates are conducted in the development of business case in the earlier stages of project management. This accurate prediction helps the investors and customers to identify the total investment and schedule of the project. The project developers define process to estimate the effort more accurately with the available mythologies using the attributes of the project. The algorithmic estimation models are very simple and reliable but not so accurate. The categorical datasets cannot be estimated using the existing techniques. Also the attributes of effort estimation are measured in linguistic values which may leads to confusion. This paper looks in to the accuracy and reliability of a non-algorithmic approach based on adaptive neuro fuzzy logic in the problem of effort estimation. The performance of the proposed method demonstrates that there is a accurate substantiation of the outcomes with the dataset collected from various projects. The results were compared for its accuracy using MRE and MMRE as the metrics. The research idea in the proposed model for effort estimation is based on project domain and attribute which incorporates the model with more competence in augmenting the crux of neural network to exhibit the advances in software estimation.

Keywords: ANFIS, effort estimation, MRE, MMRE.

GJCST-C Classification : I.5.1 I.2.3

© 2016. N. Shivakumar, N. Balaji & K. Ananthakumar. This is a research/review paper, distributed under the terms of the Creative Commons Attribution-Noncommercial 3.0 Unported License http://creativecommons.org/licenses/by-nc/3.0/), permitting all non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.
A Neuro Fuzzy Algorithm to Compute Software Effort Estimation

N. Shivakumar, N. Balaji & K. Ananthakumar

Abstract- Software Effort Estimation is highly important and considered to be a primary activity in software project management. The accurate estimates are conducted in the development of business case in the earlier stages of project management. This accurate prediction helps the investors and customers to identify the total investment and schedule of the project. The project developers define process to estimate the effort more accurately with the available mythologies using the attributes of the project. The algorithmic estimation models are very simple and reliable but not so accurate. The categorical datasets cannot be estimated using the existing techniques. Also the attributes of effort estimation are measured in linguistic values which may lead to confusion. This paper looks in to the accuracy and reliability of a non-algorithmic approach based on adaptive neuro fuzzy logic in the problem of effort estimation. The performance of the proposed method demonstrates that there is a accurate substantiation of the outcomes with the dataset collected from various projects. The results were compared for its accuracy using MRE and MMRE as the metrics. The research idea in the proposed model for effort estimation is based on project domain and attribute which incorporates the model with more competence in augmenting the crux of neural network to exhibit the advances in software estimation.

Keywords: ANFIS, effort estimation, MRE, MMRE.

I. Introduction

Achieving software economics in large-scale software development projects are very important today. Software effort estimation is the process of determining the accurate effort required to maintain or develop a software. It is always an important practical problem in software engineering which is still unsolved. Effort estimates are done in initial stages of software engineering to calculate the effort in person-months required for the software development. Accurate effort estimation helps in planning design construction and transition phases of development and prioritize the components in business case. Unreliable estimates is the main important reason for project failure, which is expressed in 2007 Comp TIA survey of thousand IT professionals, finding that three of the four most-cited causes of IT project failure are due to poor estimation.

II. Related Work

Cuauhtemoc[1] provides justification that Fuzzy logic can be used to predict the effort of the small programs based on lines of code obtained from new and changed (N&C) and reused code from small programs developed by 74 programmers. This was used as the input for the fuzzy model for estimating effort and the accuracy of output was compared with the accuracy of Statistical regression model using the comparison criterion Mean Magnitude Error Relative to the estimate.

Shiyna[2] compares the frameworks designed by using fuzzy logic and Neural Networks based on the accuracy of effort estimation. COCOMO NASA dataset had been used as the input for both the frameworks. These frameworks are validated using the parameters MMRE (Mean Magnitude of Relative Error) and Pred (Prediction Accuracy). The results show that Fuzzy Logic based framework works better when compared to the Neural Network framework.
Ochodek [3] proposed the usage of Use Case Points (UCP) method to estimate the effort based on the use case model and two adjustment factors (With or Without Unadjusted Actor Weights). The cross-validation procedure has been used to compare the variants of adjustment factors. A group of 14 projects is considered as input which are used to arrive at a conclusion that the UCP method can be simplified without the use of adjustment factors.

Iman [4] compares the software effort estimation computed by the conventional methods like function points, regression models and COCOMO with the model designed using fuzzy logic. The parameter Mean Magnitude of Relative Error (MMRE) is used to compute the accuracy of the considered methods.

Anjana Bawa [5] explains the usage of Artificial Neural Networks to estimate the project effort as it is capable of learning from the previous data. The machine learning algorithms, Back-Propagation and Cascade Correlation are used to learn and classify the dataset and hence estimate the effort using the Neural Networks.

By analyzing the previous work, it is evident that fuzzy logic is better than the conventional methods of effort estimation. By using the package points the complexity of estimating the lines of code for the considered software is reduced. By using the factor refinement, the time taken to compute the effort is less compared to the previous method where 15 attributes where obtained from the programmer to compute the effort.

### III. Proposed Approach

We have considered 93 instances of NASA historical project data and also investigated and gathered thirty projects from many case studies and experiments [11][12][14][15] with consists of 15 attributes and actual effort along with domain, area of work, Size. The fifteen attributes art converted to three index valued labelled as Human Perception and Performance Index (HPPPI), Machine Requirement and Performance Index (MRPI), Process Requirement and Performance Index (PRPI). Adaptive Neuro-Fuzzy model (The Figure 1) for software development effort estimation is perfect in the learning and good interpretability. Artificial neural networks are made up of processing units in a parallel manner called as neurons these neurons are inter linked by connections. The input for this model is six grouped attributes. Each attribute represents one factor which leads to the development effort. Table 1 describes the refinement of attributes in such a way that 15 effort multipliers are grouped in 3 clusters of refined attributes whose values are obtained from the software project developer.

<table>
<thead>
<tr>
<th>REFINED ATTRIBUTES</th>
<th>EFFORT MULTIPLIERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human Perception and Performance Index (HPPI)</td>
<td>1. Analyst Efficiency</td>
</tr>
<tr>
<td>2. Programmer Efficiency</td>
<td></td>
</tr>
<tr>
<td>3. Application Maturity</td>
<td></td>
</tr>
<tr>
<td>4. Modern Programming Practices</td>
<td></td>
</tr>
<tr>
<td>5. Use of Software tools</td>
<td></td>
</tr>
<tr>
<td>6. Virtual Machine Experience</td>
<td></td>
</tr>
<tr>
<td>7. Language Experience</td>
<td></td>
</tr>
<tr>
<td>Machine Requirement and Performance Index (MRPI)</td>
<td>8. Time Constraint for CPU</td>
</tr>
<tr>
<td>9. Turnaround Time</td>
<td></td>
</tr>
<tr>
<td>10. Machine Volatility</td>
<td></td>
</tr>
<tr>
<td>Process Requirement and Performance Index (PRPI)</td>
<td>11. Process Complexity</td>
</tr>
<tr>
<td>12. Storage Space Requirement</td>
<td></td>
</tr>
<tr>
<td>13. Schedule Constraint</td>
<td></td>
</tr>
<tr>
<td>14. Database Size</td>
<td></td>
</tr>
<tr>
<td>15. Required Software Reliability</td>
<td></td>
</tr>
</tbody>
</table>

This proposed model consolidates neural networks and fuzzy-logic principles in a combined ANFIS framework. This inference system correlates to the learning capability of fuzzy IF THEN rules to approximate the nonlinear functions.

![Figure 1: Adaptive Neuro-Fuzzy model](image1)

![Figure 2: Package Point Process](image2)

a) Package Points

The package point (figure 2) provides an alternate way to estimate the size that needs to be applied in a software project. Unlike function points and class points approach, the package points proves to be highly efficient in aiding to estimate effort and it is proven to work well with the ERP projects. In order to compute the package points, inputs are obtained
namely, scope, tasks and complexities. Package points are defined by standardizing the number of modules in the project primarily. Then the tasks to complete the modules are prioritized and defined by the client as per their requirement. Finally the complexity factors are loaded to arrive at the package point for the considered module.

b) Validation in ANFIS Model

ANFIS is a hybrid supervised method which adopts a hybrid learning algorithm to determine the parameters for fuzzy inference systems. It utilizes both least-squares method and propagation gradient descent method. This is used for training FIS membership function parameters to examine the given training data set. ANFIS can be executed using an optional argument for model validation. This is called as checking model for over fitting. The argument used for this is called as checking data set.

c) Fuzzy Rule Inference

Fuzzy rules are generated using package points, domain, type, Human Perception and Performance Index (HPPI), Machine Requirement and Performance Index (MRPI), Process Requirement and Performance Index (PRPI), Machine Requirement and Process Requirement (MRPRP) and Actual effort. A fuzzy set performance index is illustrated using a membership function that relates performance index and actual effort. A fuzzy set is defined using a triplet (α, m, β), where m denotes modal value, α and β signify the right and left boundary correspondingly and is expressed as:

\[ \mu(x) = \begin{cases} 0, & x \leq \alpha \\ \frac{x - \alpha}{m - \alpha}, & \alpha \leq x \leq m \\ \frac{\beta - x}{\beta - m}, & m \leq x \leq \beta \\ 0, & x \geq \beta \end{cases} \]  

(1)

The equivalent characteristics of the rules the most significant aspect of fuzzy logic systems. Instead of sharp swapping among modes based on breakpoints, logic flows efficiently from sections where the system’s performance is governed by one rule or another.

Defuzzification converts from fuzzy to crisp conversions. The process converts the fuzzy value to the estimated value for the single data set. This is similar to a “rounding off” method. Defuzzification converts the collection of membership function data to a single sealer quantity in corresponding membership degrees. It is characterized in structure of rules that convert variables to a fuzzy result, that is, the outcome is defined in terms of membership in fuzzy sets.

\[ R_{alv} = \begin{cases} \mu^* w_1 \quad 0 < ALV \leq 1 \\ \mu^* w_1 + (1-\mu) w_2 \quad 1 < ALV \leq 2 \\ \mu^* w_1 + (1-\mu) w_3 \quad 2 < ALV \leq 3.5 \\ \mu^* w_2 + (1-\mu) w_3 \quad 3.5 < ALV \leq 5 \\ \mu^* w_2 + (1-\mu) w_4 \quad 5 < ALV \leq 6.5 \\ \mu^* w_3 + (1-\mu) w_4 \quad 6.5 < ALV \leq 8 \\ w_4 \quad ALV > 8 \end{cases} \]  

(2)

IV. EXPERIMENTAL DESIGN

a) Evaluation Criteria

1. Mean Magnitude Relative Error (MRE), is an error ratio between the absolute deviations of prediction to the actual effort in each of the referred project

\[ MRE = \frac{1}{n} \sum |Actual_i - Estimated_i| \]  

(3)

2. Mean Magnitude Relative Error (MMRE) is the average value of MRE of all the referred projects

\[ MMRE = \frac{1}{n} \sum MRE_i \]  

(4)

The Table 2 indicates the Package point and its subsequent domain with its actual effort and the non-algorithmic estimated effort using the proposed method. The effort in the dataset is compared with this estimated effort.

<table>
<thead>
<tr>
<th>Package Point</th>
<th>Actual Effort</th>
<th>Estimated Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avionics monitoring</td>
<td>25.9</td>
<td>117.6</td>
</tr>
<tr>
<td>Mission planning</td>
<td>31.5</td>
<td>60</td>
</tr>
<tr>
<td>Simulation</td>
<td>66.6</td>
<td>352.8</td>
</tr>
<tr>
<td>Monitor Control</td>
<td>70</td>
<td>458</td>
</tr>
<tr>
<td>Real Data Processing</td>
<td>177.9</td>
<td>124</td>
</tr>
<tr>
<td>Communications</td>
<td>240</td>
<td>192</td>
</tr>
<tr>
<td>Batch Data Processing</td>
<td>25.9</td>
<td>117.6</td>
</tr>
<tr>
<td>Data Capture</td>
<td>31.5</td>
<td>60</td>
</tr>
<tr>
<td>Launch Processing</td>
<td>66.6</td>
<td>352.8</td>
</tr>
<tr>
<td>Application Ground</td>
<td>70</td>
<td>458</td>
</tr>
<tr>
<td>Utility</td>
<td>177.9</td>
<td>124</td>
</tr>
<tr>
<td>Operating Systems</td>
<td>240</td>
<td>192</td>
</tr>
</tbody>
</table>
The preceding list of algorithmic models are tested including KLOC (Kilo Lines Of Code) value from the data sets and efforts were estimated and these estimates are compared with the results obtained from Adaptive Neuro Fuzzy model,

- Halstead Model- This model developed by Halstead, concerning the supplied lines of source code from the programmer and formulates a relation,
  \[ \text{Effort} = 0.7 \times (\text{KLOC})^2 \]  
  (5)

- Bailey-Basili Model - This model developed by Bailey-Basili, between delivered lines of source code and formulates a relation,
  \[ \text{Effort} = 5.5 \times (\text{KLOC})^{1.16} \]  
  (6)

- Doty Model - This model developed by Doty, between delivered lines of source code and formulates a relation,
  \[ \text{Effort} = 5.288 \times (\text{KLOC})^{1.047} \]  
  (7)

- COCOMO - It was the first model suggested by Barry Boehm. This model has been widely accepted in practice. In the COCOMO model, the code-size $S$ is given in thousand LOC (KLOC) and Effort is in person-month.
  \[ \text{Effort} = a \times (\text{KLOC})^b \]  
  (8)
  Where $a$, $b$ are complexity factors. This model uses three sets of $a$, $b$ depending on the complexity of the software. The basic COCOMO model is simple and easy to use.

- COCOMO II - It comprises of three variants, namely, early design model, Application composition model, and Post architecture model. This is an augmentation of intermediate COCOMO model and defined as,
  \[ \text{Effort} = 2.9 \times (\text{KLOC})^{1.10} \]  
  (9)

b) **Numerical Results**

This section reports the experimental results of predictions obtained from soft computing models and other algorithmic models. The actual effort existing in the dataset is associated with the estimated effort and finally Mean Relative Error is calculated. The following graphs (figure 4) shows the Mean Relative Error variations in Doty, Bailey, COCOMO I and COCOMO II, Halstead and Neuro Fuzzy models.
The following table summarizes the MMRE value of all the algorithmic and non-algorithmic models discussed above for all the referred projects.

<table>
<thead>
<tr>
<th></th>
<th>Doty</th>
<th>Bailey</th>
<th>Halstead</th>
<th>COCOMO I</th>
<th>COCOMO II</th>
<th>Fuzzy Estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMRE</td>
<td>5.15</td>
<td>6.47</td>
<td>6.06</td>
<td>9.14</td>
<td>7.54</td>
<td>3.11</td>
</tr>
</tbody>
</table>

V. Conclusion

Early effort estimation in software development lifecycle is an important activity for project planning and resource allocation. This research work proposes an efficient model in estimating the software effort. The outcomes of the estimation obtained using the direct algorithmic methods indicates the divergence between the actual and the estimated effort. The outcome of non-algorithmic method comprising of the adaptive neuro technique based estimation decreases the Mean Magnitude of Relative Error (MMRE). Hence the examination of effort from algorithmic method and non-algorithmic method prove that adaptive neuro fuzzy based estimation is more efficient than the algorithmic methods for the estimation process. The success of estimation depends upon the accuracy and stability of the method in various measures. Future work is planned to investigate the clustering algorithms in estimation process and apply Neuro Fuzzy model on large datasets.

References Références Referencias

5. Anjana Bawa and Rama Chawla, “Experimental Analysis of Effort Estimation Using Artificial Neural