The Application of Artificial Neural Networks in Software Effort Estimation

Maryam Mirzapour Moshizi, Vahid Khatibi Bardsiri

Department of Computer Science, Bardsir Branch, Islamic Azad University, Kerman, Iran

mmairr_2007@yahoo.com

ISSN: 2231-8852

ABSTRACT
Estimate the cost and time required to build the software system is one of the most important aspects of software project management. Estimation or effort required to develop a software system is one of the major concerns of the project manager. Activity and time required to complete the process and the cost of each process will be studied in the context of estimation. There have been a lot of patterns for effort and cost estimates up to now. This is beneficial and necessary that in the early stages of construction with a minimum of information of the project, estimate effort and cost of development of a software system. Learning methods, such as neural networks, is one of these models. In this paper, the software effort estimation by the neural network has been studied and their assessment criteria have been compared.

Keywords: Software, Estimation, Cost, Effort, Artificial Neural Networks

1. Introduction
In the past, the cost of software included a small percentage of the total cost of the computer system and in software cost estimation the error rate was relatively low. Today the software is the most expensive element of all computer systems and major error costs can be the difference between profit and loss. Now, the ability to more accurately estimate a critical is a factor for many of the major software successes and estimation of cost and activity of software known as a science. Many variables, such as human, technical, environmental, policy and management software can have an impact on the estimated cost and efforts for its development.

The software has a dual role: Software is a product and is a tool for the construction and delivery of the product. The software is a logical element rather than a physical system and has a characteristic that is so much different from hardware characteristics (Roger & Bruce, 2014).

The difference between software engineering and other engineerings can be seen in the following (Ian, 2010):

1- The product is intangible
2- There is no standard software process
3- Large software projects are often unique projects.

Computer technology and its applications based on three factors: hardware, software, and manpower that in a form system it has the ability to design, plan and implement. One of the factors in the development of society is access to advanced technology in the field of software and related sciences. Software and software applications are the links to use the hardware (Igor, 2000). Due to these factors, we can understand why estimating software effort in the early stages of software development is one of the most important challenges that software developers and project managers are faced with it. Software effort or cost estimation is one of the important and influencing processes in software engineering that can have an important role in the success or failure of the project. Suitable and correct software effort or cost estimations make the project manager in the software life cycle have strong support to make different decisions and project manager, analyst, designer, programmer and software development team members know how much effort and time needs to make a good product. Cost or effort estimation models in the early stages of construction, with a minimum of information on the project cost or effort estimate system, are useful and necessary. Right effort estimation method provides the possibility of controlling time and cost of the system effectively. The accuracy of effort is an important factor in the success of the project (Kim et. al, 2004). Without a proper estimate of the cost required, the project manager can determine how much time and how many people and other resources needed for the project and in the case of error, the project will move in the direction of inevitable defeat.

According to reports published by the Standish Group (chaos report) in 2014, only 16.2% of projects in the world have successfully completed, while 52.7% of them have failed and 31.1% of them have cancelled. Software failure can destroy a company's reputation and cost and it shows the importance of accurate estimates of software effort and cost. Pressman recommends that use at least two different methods to estimate (Roger & Bruce, 2014).

2. Common Problems of Software Project Cost or Effort Estimation

Despite the different techniques and tools for estimating project costs or efforts, many software project estimations have less accuracy. T. DeMarco provides 4 recommendations for such carelessness and how to overcome them:

a. Providing an estimate for a large software project is complex and time-consuming activity. Many estimates must be made quickly and before the completion of the requirements of users.

b. People who provide software development effort or cost estimation don’t have enough experience in this field, especially in big projects. The use of professionals in this field and also keep records of previous project cost or effort estimates is a good solution for this problem.

c. Humans tend to have lower estimates for this reason sometimes estimators forget some additional costs such as test and Integration costs.

d. However, administrators want to provide an estimate but in fact, they want to win a contract with an estimate. In this situation, providing time and cost of the software application is a good solution (DeMarco, 1986).
3- Software Effort Estimation Methods

Several methods are provided for estimating the effort or cost of software that each has its advantages and disadvantages and according to the conditions of the problem are selected and used. In general, the estimation methods are divided into two categories: algorithmic and non-algorithmic. In the following, each category is explained briefly and the most important of them expressed.

3.1. Algorithmic methods

Algorithmic techniques use mathematical models to estimate the cost or effort of the project. Each algorithmic model is defined as a function of cost factors. Algorithmic methods are different in two aspects, one is the selection of cost factors and the other is defining of the cost function. First, we consider the cost factors and then describe the methods.

3.1.1. Cost factors

- **Product Factors**: having reliability, complexity, database size, reusability, consistency of documentation with the requirement of the project life cycle.
- **Computer Factors**: limitation of the runtime system, limitation of storage space, limitation of restarting the computer and variety of platforms.
- **Personnel Factors**: skills of analysis team, skills of programmers, dominance on the platform, dominance on programming language and its tools, coordination of the team.
- **Project Factors**: multisite development, using software tools.

3.1.2. Examples of algorithmic methods:

- **SLOC\(^1\):**

  One way to estimate the size of a project is based on the number of lines and a comparison with other programs that have already been calculated SLOC. However, this estimate is simple but it is difficult at the beginning of the project because as long as the requirements are not fully predicted, the number of lines of the program not a precise prediction. Meanwhile, the number of lines of the program varies according to the programming language.

- **FP\(^2\):**

  This method is based on the premise that the team members in any software project and its cost, depending on the scale of the project. Functional Points is calculated as follows:

  \[
  \text{Formula1: } \text{FP} = \text{UFC} \times \text{TC}
  \]

  UFC is the number of points the net. That is obtained from the total number of inputs, outputs, logical files, Interface and query. TCF is the technical complexity factor, which is estimated between 0.65 and 1.35.

---

\(^1\) Source Line Of Code

\(^2\) Function Point
• **COCOMO³:**

This method was presented in 1981 by B.W. Bohem. Bohem in your model, consider the following factors in the cost of a software project:


In this way, the effectiveness of each of factors on the projects will be ranked low (LOW) up to very high (EXTRA HIGH) and they are given weight. In this way, a matrix obtained by the rows of factors and the columns of the degree of impact of each factor on the project. This estimate considers many factors, and therefore the probability of error is high.

• **FPROM:**

This method is an estimating stage model and for each stage, when we can estimate human resources that the previous stage is performed. The difference between this method and FP is in estimates model (stage of it).

• **Seer-Sem:**

This method was introduced in 1980 by Galorath and more used to estimate commercial projects. The size of the project is the main factor in this estimate.

• **Linear Models:**

In this way, the simple structure is made which is calculated according to the formula:

\[ \text{Effort} = a_0 + \sum_{i=1}^{n} x_i a_i \]

Here, \( a_1 \) and \( a_2, \ldots, a_n \), based on the project are determined.

• **Multiplicative models:**

In this method, the following formula is used:

\[ \text{Effort} = a_0 \prod_{i=1}^{n} a_i^{x_i} \]

Here, \( a_1 \) and \( a_2, \ldots, a_n \), based on the project are determined and the values of \( x_i \) are just - 1, 0 and 1.

• **Putman’s model:**

This method was developed by Putman and is used for many projects. Equation method is presented below:

\[ S = E \times (\text{Effort})^{4/5} t_d^{4/5} \]

\( E \) is characteristic of the environment and the response time is \( T_d \).

3 CONSTRUCTION COST MODEL
3.2. Non-algorithmic models

• Analogy Costing:

In this way, based on a previous project, the new project cost estimate is done. This method can be used in the entire project or used in subsystems. In the first case, all the components of costs checked and in the second case, additional assessment of the similarities and differences between the current system and the previous system are done so it would be a more accurate estimate.

The advantage of this method is that it is based on real experience and disadvantage is that the former systems are not consistent with the current system and compare them are ignorant and wrong and can distort estimates.

• Expert Judgment:

In this method, costs estimate is based on personal experience of experts in the field of software development. To solve the possible inconsistencies in the estimates provided by different people, the techniques that the consensus estimates, such as Delphi and PERT will be used.

• Machine learning Models:

Most methods of software estimation used the methods and techniques of this model. Estimation Accuracy is increasing because of the learning ability and the ability to run multiple times of these models. This method can be divided into two main categories:
   a) Neural networks
   b) Fuzzy Method

These procedures are consistent with most projects and widely used (Khatibi & Jawawi, 2010).

• Parkinson:

In this method, software costs are not estimated, but according to available resources (regardless of project goals) are determined. Although this method in some cases provides a reasonable estimate the technique is not appropriate for the estimation of project costs.

• Price-to-Win:

In this method, instead of the software, features and applications, the cost is estimated based on the client's budget.

• Button-up:

In this way, each of the system components is estimated separately and then sum these estimates, the overall cost estimate of the project will be considered and the collection of them will be considered as total project costs estimation. To use this method, first, it is necessary to do a preliminary design of the system to get the structural components.

• Top-Down:

In this method, the total project costs are estimated based on general criteria. In the next step, the cost can be distributed between the different system components.

Accurate estimates of a project at least use two or three techniques (Roger & Bruce, 2014).
4. Artificial Neural Network

An artificial neural network is inspired by the biological nervous system and such as the brain, process information. The system consists of many processing elements called neurones that work together to solve a problem. ANNs, such as humans, learn by example and by processing the experimental data, pass the knowledge to the network structure. Thus, these systems are called smart because by calculating on numerical data or examples, learns the general rules. Figure 1 shows the general structure of an artificial neural network.

![Diagram of an artificial neural network](image)

Figure1: Structure of an artificial neural network.

4.1. Why ANNs are worth reading?

- Neural networks, due to parallel processing, have high speed.
- Neural networks have the potential to solve the problems that are difficult or impossible to simulate by logic or other methods.
- Neural networks, such as the human brain is continuously learning and adapting to the environment. This means that if the network was trained for a situation and a small change in environment occurred, by a little training, are also used for the new situation.
- In a neural network, the wrong performance of part of the brain's neurones may not be a complete failure and there is also the possibility of making the correct decision.
- This method can provide a logical answer for data in uncertainty conditions.

4.2. The Reasons for using a Neural Network in Software Effort Estimation

There are many different methods for software effort estimation and Artificial Intelligence techniques have been used in this field to enhance the accuracy and reliability. One of the best artificial intelligence models that are used to estimate the software effort is an artificial neural network. These networks by using the technique of training assess data with minimal error. Evaluation criteria for estimating the effort/cost of software are errors. Previous studies have shown the error of neural networks is lower than the algorithmic effort estimations. In addition, in the process of software development, software factors and information about effort estimates are low and there is no possibility of using the algorithm with good approximation. Also, because of parallel processing of neural networks, the processing speed is high. Therefore, the use of neural networks to estimate the
effort in the early stages of software development is useful. The neural network is made of two main computational parts. Part I: neurones, which are the nodes of ANN and part II: synapses, which are the weights and connections of ANN (Dave & Dutta, 2014).

5. Assessment Articles

Many articles about software effort estimation by using an artificial neural network were performed. In this paper 35 articles of the 2004 and 2015 selected. Among the articles were studied, 21 of them are journals and 14 of them are conferences which are listed in the following tables and charts.

Table1: Assessment Article and their Properties

<table>
<thead>
<tr>
<th>Purposes and Results</th>
<th>Algorithm to compare</th>
<th>Parameters for assessment</th>
<th>Database used</th>
<th>The neural network used</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>The third model has a better performance at estimating software effort</td>
<td>Three neural networks were compared</td>
<td>MRE</td>
<td>148 (IT Project)</td>
<td>1: Neural network model (FP only). 2: Neural network model (six variables only). 3: Neural network model (FP + six variables).</td>
<td>Park &amp; Baek, 2008</td>
</tr>
<tr>
<td>The third model has a better performance at estimating software effort</td>
<td>Three neural networks were compared</td>
<td>MRE, BRE, MMRE, Pred</td>
<td>41 Lopez-Martin</td>
<td>1: Cascaded Feed Forward Back Propagation Neural Network model, Elman 2: Back Propagation Neural Network model, Layer Recurrent 3: Neural Network model &amp; Generalised Regression Neural</td>
<td>Ghose et al, 2011</td>
</tr>
<tr>
<td>Neural networks have provided a better performance in estimating software effort</td>
<td>Regression Analysis</td>
<td>41 Lopez-Martin</td>
<td>Feed Forward-Back Propagation Neural Network</td>
<td>Bhatnagar et al, 2010</td>
<td></td>
</tr>
<tr>
<td>Improvements of 3.27% in software effort estimation</td>
<td>COCOMO and typical neural network</td>
<td>MRE, MMRE, Pred</td>
<td>COCOMO With 63 project, COCOMO With 93 NASA 93 project</td>
<td>PSO-ANN-COCOMO II</td>
<td>Dan, 2013</td>
</tr>
<tr>
<td>Improvements of 8.36% in software effort estimation</td>
<td>COCOMO II</td>
<td>MRE, MMRE, Pred</td>
<td>COCOMO With 63 project, COCOMO With 93 NASA 93 project</td>
<td>ANN-COCOMO II</td>
<td>Attarzadeh et al, 2012</td>
</tr>
<tr>
<td>According to MMRE parameter, FFNN performance is better than RBFNN model but based on RSD parameters RBFNN model provides a more accurate estimate</td>
<td>Regression Analysis Model And with each other</td>
<td>MMRE, RSD</td>
<td>60 projects from COCOMO NASA</td>
<td>1: Feed-Forward Neural Network (FFNN) 2: Radial Basis Functional Neural Network (RBFNN)</td>
<td>Dave &amp; Dutta, 2011</td>
</tr>
<tr>
<td>Improvements in software effort estimation</td>
<td>Compare with databases</td>
<td>MMRE, PRED</td>
<td>240</td>
<td>Cascade Correlation Neural Network (CCNN)</td>
<td>Nassif, et al, 2012 (a)</td>
</tr>
<tr>
<td>Improve efforts</td>
<td>Use Case Point</td>
<td>MMRE</td>
<td>240</td>
<td>Feed-Forward Neural Network</td>
<td>Nassif,</td>
</tr>
<tr>
<td>Compared to UCP estimates model</td>
<td>(UCP)</td>
<td>PRED</td>
<td>(FFNN)</td>
<td>et al, 2012 (b)</td>
<td></td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------</td>
<td>------</td>
<td>--------</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td>Reducing the time and effort required to estimate the cost of the software in the early stages of the project</td>
<td>Two neural networks were compare</td>
<td>MMRE</td>
<td>530</td>
<td>BPN with GA BPN with trial and error</td>
<td></td>
</tr>
<tr>
<td>Improvements in software effort estimation</td>
<td>FFNN</td>
<td>MMRE MdMRE PRED</td>
<td>NASA Maxwell COCOMO81</td>
<td>PSO - Feed forward Link Artificial Neural Networks (PSO-FLANN)</td>
<td></td>
</tr>
<tr>
<td>For more accurate estimates</td>
<td>COCOMO</td>
<td>MRE</td>
<td>COCOMO81 With 63 project</td>
<td>Feed-Forward Neural Network (FFNN)</td>
<td></td>
</tr>
<tr>
<td>Improvements in software estimation</td>
<td>Compare with data bases</td>
<td>MRE</td>
<td>NASA</td>
<td>Feed Forward-Back Propagation Neural Network</td>
<td></td>
</tr>
<tr>
<td>Improvements in software estimation</td>
<td>Compare with data bases</td>
<td>MMRE MRE PRED</td>
<td>COCOMO81</td>
<td>Feed Forward Neural Network (FFNN)</td>
<td></td>
</tr>
<tr>
<td>Neural network model has a better performance at estimating software effort</td>
<td>Regression Analysis</td>
<td>MMRE</td>
<td>COCOMO With 63 project</td>
<td>Feed Forward Neural Network (FFNN)</td>
<td></td>
</tr>
<tr>
<td>Fuzzy Neural Network Model has better performance than neural network model and COCOMO</td>
<td>COCOMO and neural network</td>
<td>MMRE MRE PRED</td>
<td>COCOMO With 63 project</td>
<td>Fuzzy Neural Network</td>
<td></td>
</tr>
<tr>
<td>ANN model has a better performance at estimating software effort</td>
<td>APF, SLIM, COCOMO, Regression Analysis</td>
<td>MMRE PRED</td>
<td>COCOMO With 63 project</td>
<td>Feed Forward Neural Network (FFNN)</td>
<td></td>
</tr>
<tr>
<td>Objective: To solve the problem of neural network learning The result: better estimates</td>
<td>FFNN</td>
<td>MMRE MdMRE PRED</td>
<td>COCOMO NASA With 60 project NASA93 With 93 project USC With 63 project</td>
<td>Multilayer Perceptrons (MLP)</td>
<td></td>
</tr>
<tr>
<td>Objective: To determine the scope and boundaries of the neural network</td>
<td>FFNN</td>
<td>MMRE PRED</td>
<td>COCOMO’81 With 63 project</td>
<td>FFNN+ COCOMO + K-mean</td>
<td></td>
</tr>
<tr>
<td>RBFNN model has a better performance at estimating software effort</td>
<td>three neural networks were compared</td>
<td>BRE MRE MIBRE With ISBSG 5052 project</td>
<td>1: General Regression Neural Network (GRNN) 2: Radial Basis Function Neural Networks (RBFNN) 3: Multilayer Perceptrons (MLP)</td>
<td>Lopez-Martin, 2015</td>
<td></td>
</tr>
<tr>
<td>The preferred method is determined</td>
<td>SLIM, COCOMO And with each other</td>
<td>MMRE PRED</td>
<td>NASA COCOMO</td>
<td>1: Feed-forward Neural Network 2: Recurrent Neural Networks 3: Radial Basis Function (RBF)</td>
<td>Hmaza &amp; Kamel,</td>
</tr>
<tr>
<td>according to the terms of issue</td>
<td>Compared with databases</td>
<td>MMRE</td>
<td>MRE</td>
<td>5 project</td>
<td>Feed-forward Neural Network + COCOMO II</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>---------------------------</td>
<td>-------</td>
<td>-----</td>
<td>-----------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>For small database, using MLP and for large database using the linear regression model offers a better estimate</td>
<td>ISBSG With 223 project Western University Canada With 65 project CompuTop With 45 project</td>
<td>Multilayer Perceptrons (MLP)</td>
<td>Nassif et al, 2013</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reduce the difference between actual costs and estimated amounts</td>
<td>MRE</td>
<td>Feed-forward Neural Network + GA</td>
<td>Li, 2010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Improving accuracy in estimating software effort</td>
<td>COCOMO 81 With 63 project</td>
<td>Feed Forward-Back Propagation Neural Network</td>
<td>Kaur et al, 2010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANN model has a better performance at estimating software effort</td>
<td>With 18 NASA project</td>
<td>Multilayer Perceptrons (MLP)</td>
<td>Mittas et al, 2015</td>
<td></td>
<td></td>
</tr>
<tr>
<td>More accurate estimates</td>
<td>ISBSG With 4106 project NASA93 With 93 project</td>
<td>COCOMO With 63 project</td>
<td>Feed Forward-Back Propagation Neural Network</td>
<td>Soleimanian, 2011</td>
<td></td>
</tr>
<tr>
<td>Improvements in software effort estimation</td>
<td>Compare with databases</td>
<td>COCOMO81 With 63 project NASA60 With 60 project NASA93 With 93 project Albrec With 24 project CF With 21 project Desharnais With 77 project</td>
<td>Feed Forward Neural Network</td>
<td>Jodpimai et al, 2010</td>
<td></td>
</tr>
<tr>
<td>Improvements in software effort estimation</td>
<td>MMRE</td>
<td>COCOMO With 63 project</td>
<td>Clustering the project and choose the best method for each cluster</td>
<td>Khatibi et al, 2013</td>
<td></td>
</tr>
<tr>
<td>Improvements in software cost</td>
<td>MMRE</td>
<td>COCOMO With 63 project Maxwell With 62 project ISBSG With 5052 project</td>
<td>Feed Forward Neural Network</td>
<td>Ivica &amp; Lopez-Martin, 2010</td>
<td></td>
</tr>
<tr>
<td>In projects with small size, offers more accurate estimates</td>
<td>MMRE</td>
<td>132 project</td>
<td>Feed Forward Neural Network</td>
<td>2013</td>
<td></td>
</tr>
<tr>
<td>Improvements in software cost</td>
<td>COCOMO</td>
<td>COCOMO II</td>
<td>Feed Forward-Back Propagation Neural Network</td>
<td>Tadayon 2005</td>
<td></td>
</tr>
<tr>
<td>Improvements in software cost and effort estimation</td>
<td>Compare with databases</td>
<td>COCOMO 81 With 63 project Kermerer With 15 project IBM DPS With 24 project Hallmark With 28 project NASA 96 With 93 project NASA 63 With 63 project Maxwell With 100 project</td>
<td>1- Multi-Layer Perceptron (MLP) 2- Radial Basis Function (RBF) 3- Wavelet Neural Network (WNN) 4- Functional Link Artificial Neural Network (FLANN) 1- Generalised Regression Neural Network (GRNN)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The best values in the China database is made when the learning rate is 0.9, and the number of neurons in the hidden layer is 15 and in the Maxwell database is made when the learning rate is 0.9, and the number of neurons in the hidden layer is 17</td>
<td>Apply different learning rates and compare with each other</td>
<td>MMRE, MRE, MdMRE, PRE, D, CC, MAE, RMSE China With 499 project Maxwell With 62 project</td>
<td>Error Back Propagation Network (EBPN) Hota et al, 2015</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Improvements in software cost estimation</td>
<td>COCOMO</td>
<td>MMRE MRE PRED 69 project</td>
<td>Neuro-Fuzzy- COCOMO Huang et al, 2007</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.54% improvement by applying PCA technique in GRNN</td>
<td>- M5 - Linear regression - SMO polykernel - SMO RBF kernel - GRNN - M5 + PCA - Linear regression +PCA - SMO polykernel + PCA - SMO RBF kernel + PCA - GRNN + PCA</td>
<td>MMRE MdMRE COCOMO With 63 project</td>
<td>General Regression Neural Network (GRNN) Sankara &amp; Kumar, 2015</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The proposed method has less error than the other two methods</td>
<td>COCOMO and typical neural network</td>
<td>MMRE MRE PRED NASA with 60 project</td>
<td>MLP + ICA Soleimani &amp; Maroufi, 2014</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In Table 1 the literature review and the neural networks and the databases for each article are shown. Also, the evaluation parameters and algorithms which were compared with the model and the advantages and purposes of the proposed model are expressed.

![Figure 2: The Articles studied in the period 2004-2015](image)

![Figure 3: The amount and Neural Network Models used to Estimate Effort or Cost](image)

In Figure 3, a variety of neural networks used in various articles and the amount of their use in the form of a graph are shown. As can be seen, in most papers, the typical neural network is used while the other models of them have been used less. According to this chart suggests researchers who are interested in estimates by using neural networks topics use other models of neural networks.
Figure 4: The use of other techniques with neural network in estimating the effort/cost software

Figure 5: The use of different databases the articles

Figure 6: The total number of projects and the number of projects to learn and test used in each article
6. Conclusion

Creative and abstract nature of software projects makes extremely difficult to estimate software cost and time. A successful software project is a project that is done in terms of specific and pre-determined cost and time. Project estimation is a method that before the start of the project.

we realise it can be possible. This estimate can be the difference between profit and loss. One of the methods used to estimate software projects is neural networks. Neural networks because of learning ability can be provided a more accurate estimation. In this case, 35 articles of the 2004 to 2015 assessment that all of them improve the software effort or cost estimation. Finally, the drawn graphs and tables can be a good guide to future works.

REFERENCES

2. Attarzadeh, I, Mehranzadeh, A. Barati, A. “Proposing an Enhanced Artificial Neural Network Prediction Model to Improve the Accuracy in Software Effort Estimation” Fourth International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN), 2012, pp 167-172
24. Nikolaos Mittas, Efi Papatheocharous, Lefteris Angelis, Andreas S. Andreou “Integrating non-parametric models with linear components for producing software cost estimations” The Journal of
35. TR Benala, R Mall, S Dehuri “Software Effort Estimation Using Functional Link Neural Networks Optimised by Improved Particle Swarm Optimisation” Swarm, Evolutionary, and Memetic Computing Lecture Notes in Computer Science, Volume 8298, 2013, pp 205-213