World Applied Sciences Journal 14 (9): 1265-1275, 2011
ISSN 1818-4952
© IDOSI Publications, 2011

A New Fuzzy Clustering Based Method to Increase the Accuracy of Software Development Effort Estimation

[1]Vahid Khatibi B., [2]Dayang N.A. Jawawi, [3]Siti Zaiton Mohd Hashim and [4]Elham Khatibi

[1,2,3]Department of Software Engineering, Faculty of Computer Science and Information System, Universiti Teknologi Malaysia (UTM) 81300, Johor Bahru, Malaysia
[4]Bardsir Branch, Islamic Azad University, Kerman, Iran

Abstract: Project planning plays a significant role in software projects so that imprecise estimations often lead to the project faults or dramatic outcomes for the project team. In recent years, various methods have been proposed to estimate the software development effort accurately. Among all proposed methods the non algorithmic methods by using soft computing techniques have presented considerable results. Complexity and uncertain behavior of software projects are the main reasons for going toward the soft computing techniques. In this paper a hybrid system based on combining C-Means clustering, neural network and analogy method is proposed. Since, there are complicated and non linear relations among software project features, the proposed method can be useful to interpret such relations and to present more accurate estimations. The obtained results showed that fuzzy clustering could decrease the negative effect of irrelevant projects on accuracy of estimations. In addition, evaluation of proposed hybrid method showed the significant improvement of accuracy as compared to the neural network the analogy method and statistical methods.

Key words: Effort Estimation・Irrelevant Data・Analogy・Fuzzy Clustering・Neural Network

INTRODUCTION

Since, the role of software in today's business market is undeniable, the accurate estimating of the software development cost and effort is very important. Planning, developing, constructing and all aspects of the software projects are affected by the relevant estimations. During the recent decades, many methods for the software efffort estimation have been presented. Selecting a method as the best one seems to be impossible because the performance of each method depends on the various factors such as available information, used development techniques, project features and so on. But the main aim of all methods is presenting the accurate results. Since in the first stages of the project understanding of project features is incomplete, the predictions may be inaccurate. So many researchers try to present more reliable and more flexible techniques to perform the prediction at the early steps of the project. In addition, as the software projects requirements are not stable and the relations among features are hard to interpret, estimating of software project metrics is more complicated as compared to the other projects. Change of customer requirements, rapid hardware progress, change of software development frameworks and lack of standards make the prediction so difficult in this area. The first idea for software effort estimation returns to 1950 by presenting the manual rule of thumb [1]. By increasing the number of software projects and need of user society to earn high quality software, some models based on the linear equations and regression techniques were presented as the software effort techniques in 1965 [2]. Name of Larry Putnam, Barry Bohem and Joe Aron can be mentioned as the pioneers of software estimation methods [1]. Afterward in 1973, the IBM researchers presented the first automated tool, Interactive productivity and Quality (IPQ) [1]. Barry Boehm proposed a new method called COCOMO that utilized some experimental equations to estimate software development effort [3]. In addition Boehm explained several algorithms in his book "Software Engineering Economics" [3] that still are used by researchers. Other models such as Putnam Lifecycle Management (SLIM) [4] and Software Evaluation and Estimation of Resources - Software Estimating Model (SEER-SEM) continued the principles of COCOMO [2]. Introducing the Function Point (FP) as a metric for software size estimation by

Corresponding Author: Vahid Khatibi B., Department of Software Engineering, Faculty of Computer Science and Information System, Universiti Teknologi Malaysia (UTM) 81300, Johor Bahru, Malaysia.

Albrecht [5] was the other important event in that decade. Analogy Based Estimation(ABE) was proposed as a comparative method in 1997 [6]. This method predicts the software project metrics by comparing the target project features with past completed projects. Simplicity and capability of ABE in prediction increase its usage so that ABE estimates were comparable with most mathematic models. Change of the software development methods and rapid progress of software methodologies lead to present the new version of COCOMO called COCOMO II in 2000 to cover some new features and requirements of software projects.

In recent years researchers found that former estimation methods cannot response to dynamic behavior of software projects. Particularly mathematical equations and fixed relations are unable to present accurate estimation for today's software project. Therefore, soft computing techniques have been widely used to predict the software development effort because these techniques can perform accurately in changeable and unstable environments. Artificial Neural Network (ANN) has been considered as the main idea behind most researches in term of software effort estimations because it can interpret the high complicated relations among software project features. Furthermore, flexibility of neural networks can be useful to endure the non linear level of software projects. Indeed, most studies in this field tried to design a neural network which is trained by software project independent attributes and then predicts the software development effort as dependent attribute accurately. One of the most important factors for improving the neural network performance is the quality of data used for training. Outliers and irrelevant data in training set can lead to the imprecise estimations. Basically software project datasets are so complicated and the relation among parameters are non linear and hard to understand. This problem arises because software projects are naturally uncertain and unstable. In most software project datasets this problem is seen obviously so that by analyzing the data seems there is no strong correlation between them. Neural networks suffer from the mentioned problem at software project datasets and usually there are some inconsistent cases in training data which reduce the accuracy of neural network predictions. In this paper improving the performance of neural network is performed employing the clustering and analogy methods as the most important aim.

Motivation: Rate of software project failures has been an important motivation to work on in this area. Software projects usually do not fail during the implementation and most project fails are related to the planning and estimation steps. Despite going to over time and cost, approximately between 30% and 40% of the software projects are completed and the others fail [7]. The Standish group announced the rate of 70% for the software project fails in 1994 [8]. Also the cost overrun has been indicated 189% in that year. According to the last report of Standish group just 32% of software projects are on time and on cost in 2009, 44% of the projects are in challenged and 24% of projects have been cancelled. Although may be the mentioned statistics are pessimistic, they show the deep crisis related to the future of the software projects [8-9]. During the last decade several studies have been done in term of finding the reason of the software projects failures.

Galorath and Evans [10] performed an intensive search among 2100 internet sites and found 5000 reasons for the software project failures. Among the found reasons, insufficient requirements engineering, poor planning the project, suddenly decisions at the early stages of the project and inaccurate estimations were the most important reasons. The other researches regarding the reason of project fails show that inaccurate estimation is the root factor of fail in the most software projects [1, 7, 11-12]. Therefore inaccurate estimating is a real problem in software projects and it should be considered in research works. Proposing the efficient and reliable techniques seems to be useful in this field.

Related Works: All software development estimation methods can be divided into two groups: algorithmic and no algorithmic methods. Algorithmic methods rely on statistical analysis of software project features and mathematical equations. Regression methods such as stepwise regression (SWR), multiple linear regressions (MLR) are the most common algorithmic methods. On the other hand, non algorithmic methods estimate the software development effort using analogy methods, Expert judgment and soft computing techniques. Flexibility and adaptability of neural network has made it the most common method among all soft computing techniques.

There are several studies that have compared algorithmic and non algorithmic methods [13-16]. In these studies, regression methods as most important algorithmic method and neural network as most important non algorithmic method have been compared. The results of mentioned comparison show the supremacy of neural network. Therefore, in this paper we focused on improving the performance of neural network.

Neural networks can be useful to interpret the relation between software project attributes and the final effort. Indeed, most studies in this field tried to design a neural network which is trained by software attributes and predicted the effort accurately. Several types of neural network with various structures have been proposed to estimate. Usually number of layers, number of neurons in each layer and selection of transfer function are the main subjects in this area. Among all types of neural network, back propagation has been widely used for software effort estimations [17-21]. Also we can see using of RBF and Perceptron in a few previous research works [22-23] . Here some related works are described briefly.

Witting [18] used the back propagation neural network to estimate the software effort. In this research the Function Point metric has been used as the size of projects and the amount of effort has been determined based on person-hours. The performance of the proposed network has been measured by using two datasets so that one of them is an actual dataset and another is a simulated dataset.

Samson and Ellison [22] proposed a new method to estimate the software effort by using the code size as input. The mentioned method was called Cerebellar Model Arithmetic Computer (CMAC) in which a perceptron neural network was employed. The performance of CMAC was evaluated compared with the COCOMO method. The COCOMO81 dataset was used for training and evaluating of the CMAC and obtained results compared with the COCOMO and regression methods showed the improvement.

Kumar and Ravi [24] suggested using of Wavelet Neural Network (WNN) for estimating of software project effort. Gaussian and Morlet were used as the transfer functions. Also in this study a new learning algorithm was proposed which was called threshold accepting based wavelet Neural Network (TAWNN). Canadian Financial (CF) and IBM data processing services (IBMDPS) were used to evaluate the performance of the proposed method as compared to the similar methods. All the comparisons were done based on the MMRE metric and in most of which the proposed method showed improvement.

Rao [19] used Functional Link Artificial Neural Network (FLANN) to estimate the software project effort. The architecture of network was too simple and there was no hidden layer and the learning process was very fast. This network was established based on the COCOMO method and as the input 17 cost drivers and 5 scale factors were applied to the network. A functional expansion was done on the inputs by the especial formulas. Three functional expansion methods (Chebyshev, Legendre and Power Series) were used in this study and the NASA dataset was selected to evaluate the performance of the network. Obtained results from each functional expansion method were compared by measuring the mean square error. Also, the results were compared with the ANN method.

Kaur *et al.* [20] suggested a back propagation neural network with two neurons in input layer and two neurons in hidden layer and one neuron in output layer to effort estimation. In addition for the purpose of comparing the performance of proposed network, four estimation models (Halstead, Walston,Felix Bailey,Basili, Doty) also were used to estimate . Nasa dataset was used for presenting the results with two attributes (KDLOC, Methodology), MMRE and RMSEE were used as the performance criteria and the final results showed that neural network outperforms compared to the mentioned models.

Reddy and Raju [21] presented a feed forward neural network with 22 neurons in input layer, two hidden layers and one node in output layer. This architecture was established according to the COCOMO Effort Multipliers (17 EM) and Scale Factors (5 SF). The COCOMO equation was converted to a linear equation so the linear transfer function was chosen for neural network. COCOMO81 dataset was used to evaluation the performance of neural network based on MMRE. Fifty projects were selected randomly as train set and the rest projects as the test set. The results were compared with the COCOMO results and the ability of proposed neural network to present more accurate estimates was proved.

Park and Baek [25] Investigated three common estimation technique (Regression, Expert Judgment, Neural Network) by using a dataset included 148 IT projects of Korean IT service vendors so that each project was described by 39 features. In this study the number of features was decreased to the 18 features by using expert judgment method (asking from 300 professionals) and afterward selected features were decreased to 7 features by applying the regression technique. The neural network was used three times so that the first time just function point feature was used as the input, second running was done with six features(FP removed) and the third time was performed with seven features(FP included). Also expert judgment and two regression methods (Albrecht and Matson) were used in parallel with the neural network. Finally the results based on MMRE showed that the neural network with seven inputs presents more accurate estimates.

Balich and Lopez [26] compared the ANN and multi regression methods by using two own datasets. The first dataset comprises 132 projects developed by 40 programmers and the second one including 77 projects developed by 24 programmers. New and changed codes as well as reused codes were considered as the independent project attributes and the actual effort was the dependent attribute. The proposed neural network had 2 layers that there were two neurons in input layer and 10 neurons in hidden layer. According to the actual effort a multi linear regression equation was determined to estimate. The results were presented by MER and it was seen that there was no significant differences between ANN and MLR methods for estimating the actual effort.

Idri *et al.* [23] tried to find a suitable structure for Radial Basis Functional (RBF) neural network particularly the number of neurons in hidden layer. This study focused on the effect of Gaussian function widths on accuracy level of prediction in software projects. Two models were proposed to determine the widths using the k-means clustering. An artificial dataset based on COCOMO81 has been generated with 252 projects as well as Tulutuku dataset with 53 web projects have been used to evaluate the network. Two configurations of network were evaluated with the various numbers of neurons in hidden layer and the high effect of adjusting the widths on results was proved.

According to the related works it can be concluded that most previous research works have improved the performance of neural networks by trial and error and reconstructing the network. Previous works in this area suffer from inconsistent and irrelevant projects that decrease the accuracy of estimations. Almost all existing methods use simple data preprocessing for outlier detection and data preparation. There is no previous experience on using fuzzy clustering for data preprocessing. In this paper the performance of analogy method and neural network has been improved significantly using fuzzy clustering based on proposing a new hybrid method.

Fuzzy Clustering: Data Clustering is defined as a technique to divide all data into several clusters so that similar data is located in the same cluster. Furthermore, dissimilarity of data located in separate clusters should be high as possible. Determining the level of similarity depends on the data characteristics. Several parameters can be chosen to measure the level of similarity such as connectivity, intensively and distance.

A point can belong to more than one cluster; this is the main idea behind the fuzzy clustering technique. Therefore, a membership level is defined to determine how a point is associated to a cluster. The value of membership level varies from 0 to 1. C-Means clustering is one of the most important fuzzy clustering techniques which has been proposed in 1981 [27]. In this method the final aim is minimizing a target function as Equation 1.

$$l_m = \sum_{i=1}^{N} \sum_{j=1}^{G} U_{ij}^{in} \left\| X_i - C_j \right\|^2 \quad 1 < \text{m} < \infty \tag{1}$$

Where, U_{ij} is the membership degree of X_i in cluster j and belongs to [0, 1]. C_j is the center of cluster j and ||*|| is a similarity measurement between X_i and the center. C-Means clustering algorithm is presented as following.

C-means Clustering Algorithm: C-means Clustering Algorithm Is Organized in Four Steps as Following:
Input: A vector X of n data points as below:

$$\text{X} = \{x_1, x_2 ....,x_n\} \tag{2}$$

Output: Matrix U which is c×n, c is number of clusters.

Step 1: Initialize U=[u_{ij}] matrix, U(0),Where U_{ij} is the value of membership for point i in cluster j.

Step 2: At k-step: calculate the centers vectors C(k)=[c_j] with U(k),

$$cj = \frac{\sum_{L=1}^{N} U_{if}{}^{m}{}_{Xi}}{\sum_{L=1}^{N} U_{if} m} \tag{3}$$

Step 3: Update U(k), U(k+1)

$$U_{if} = \frac{1}{\sum_{k=1}^{C} \left(\frac{|x_L - c_f|}{|x_L - c_k|} \right)} \tag{4}$$

Step 4: If || U(k+1) - U(k)||< threshold then STOP; otherwise return to Step 2.

Analogy Based Estimation (ABE): ABE method was produced by Shepperd in 1997 [6] as a substitute for algorithmic methods. In this method estimation of software project metrics is performed by comparing target project with previous done projects and finding most similar projects to the target project. Due to simplicity, ABE has been widely used in term of software projects that there are a lot of previous similar projects for them. Basically, ABE includes four parts:

- Historical dataset
- Similarity function
- Solution function
- The associated retrieval rules

Each Part Can Be Described as Following:

- Gathering the previous projects data and producing the historical dataset.
- Choosing new proper features of the project such as FP and LOC.
- Retrieving the previous projects and calculating the similarities between the target project and the previous projects. Usually the weighted Euclidean distance and the weighted Manhattan distance are used at this stage.
- Estimating the target project effort.

Similarity Function: ABE uses a similarity function which compares the features of two projects. There are two popular similarity functions, Euclidean Similarity (ES) and Manhattan Similarity (MS)[6]. The Equation 5 shows the Euclidean Similarity function.

$$\mathrm{Sim}(p,p') = 1/\left[\sqrt{\sum_{i=1}^{n} w_i \mathrm{Dis}(f_{i,}f_i')} + \delta\right] \quad \delta = 0.0001$$

$$\mathrm{Dis}(f_{i,}f_i') = \begin{cases} (f_{i,}f_i')^2, & \text{if } f_i \text{ and } f_i' \text{ are numeric or ordinal} \\ 1 & \text{if } f_i \text{ and } f_i' \text{ are nominal and } f_{i,} = f' \\ 0 & \text{if } f_i \text{ and } f_i' \text{ are nominal and } f_{i,} \neq f' \end{cases} \tag{5}$$

Where, p and p' are the projects, w_i is the weight assigned to each feature and varies between 0 and 1. Fi and f_i' display the ith feature of each project and n demonstrates the number of features. ä is used for obtaining the none zero results. The MS formula is very similar to the ES but it computes the absolute difference between the features. The Equation 6 shows the Manhattan similarity function.

$$\mathrm{Sim}(p,p') = 1/\left[\sqrt{\sum_{i=1}^{n} w_i \mathrm{Dis}(f_{i,}f_i')} + \delta\right] \quad \delta = 0.0001$$

$$\mathrm{Dis}(f_{i,}f_i') = \begin{cases} |f_{i,}f_i'|, & \text{if } f_i \text{ and } f_i' \text{ are numeric or ordinal} \\ 1 & \text{if } f_i \text{ and } f_i' \text{ are nominal and } f_{i,} = f' \\ 0 & \text{if } f_i \text{ and } f_i' \text{ are nominal and } f_{i,} \neq f' \end{cases} \tag{6}$$

Solution Functions: After choosing the K most similar projects, it is possible to estimate the effort of new project according to the selected features. The common solution functions are: the closest analogy as most similar project [28], average of most similar projects [6], median of most similar projects [29] and inverse distance weighted mean [30].

The average describes the average effort of K most similar projects, where K> 1.

The median describes the median effort of K most similar projects, where K > 2.

The inverse distance weighted mean adjusts the portion of each project in estimation by using Equation 7.

$$\hat{C}_p = \sum_{K=1}^{K} \frac{\mathrm{Sim}(p,p_k)}{\sum_{i=1}^{n} \mathrm{Sim}(p,p_k)} \mathrm{C}_{\mathrm{pk}} \tag{7}$$

Where p shows the new project, p_k illustrates the kth most similar project, C_{pk} is the effort value of the kth most similar project p_k, Sim(p,p_k) is the similarity between projects p_k and p and K is the total number of most similar projects.

Proposed Hybrid Method: In this paper a feed forward neural network is employed to estimate the software development effort. This type of neural networks with two layers is so common to solve the estimation problems. Our proposed structure for feed forward network is seen in Figure 1. According to Figure 1, ten neurons are used in hidden layer. Number of neurons has been computed by an exhaustive search for the best network structure. Tansigmoid and Pureline transfer functions are employed in the hidden layer and output layer respectively.

C-Means technique (described before) is used to find the most similar projects of dataset and to cluster them into several groups. The projects are located in clusters based on their highest membership amount. Each group is treated as a training set separately. The training stage for proposed method is depicted in Figure 2.

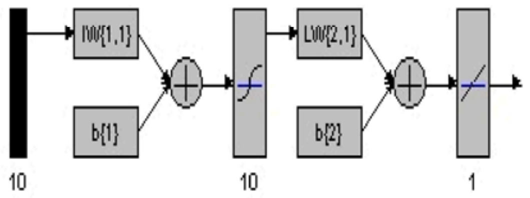

Fig. 1: Proposed NN structure

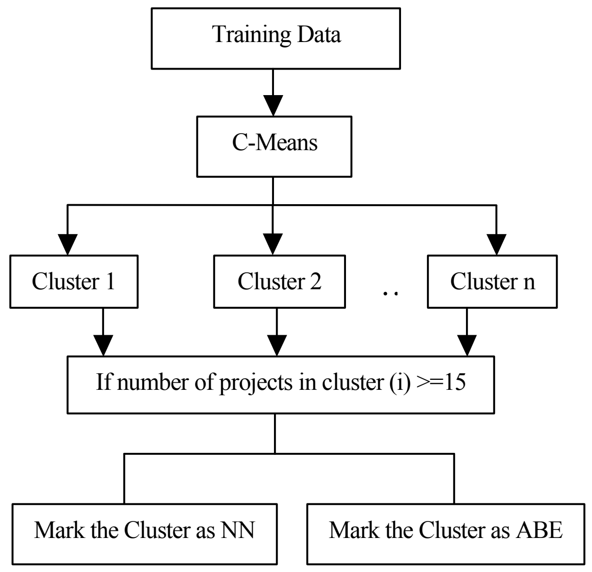

Fig. 2: Training stage of proposed method

If the number of projects in a cluster is less than fifteen then this cluster is considered as an analogy cluster. There are three reasons to select a filter based on fifteen projects. Firstly, each project in software project datasets usually has at least 10 features so the NN cannot train properly by a few numbers of projects in the training set (less than fifteen). Secondly, using of C-means clustering on several software datasets showed that clusters with less than fifteen projects usually consists of irrelevant data and these types of clusters are not useful for training the NN. Thirdly, ABE method usually presents acceptable results in low population data sets. After marking the clusters, most proper weights and biases of NN are computed by several reconstructing the NN. Training the network is done for clusters (with NN mark) separately. Subsequently, the testing stage is presented based on the results of training stage. The number of clusters depends on the dataset characteristics and especially variance of data in the dataset. If the dispersion level and variance are high in a dataset so the number of clusters are increased otherwise a few clusters are enough.

Testing stage consists of three main steps. At first, a project from test data is selected afterward Euclidean distance between selected project and clusters centers is omputed to specify which cluster the project belongs to. The cluster with minimum distance is selected for under estimating project. If related cluster has been marked as ABE in the training stage therefore the ABE method is used to estimate the effort of project and otherwise related

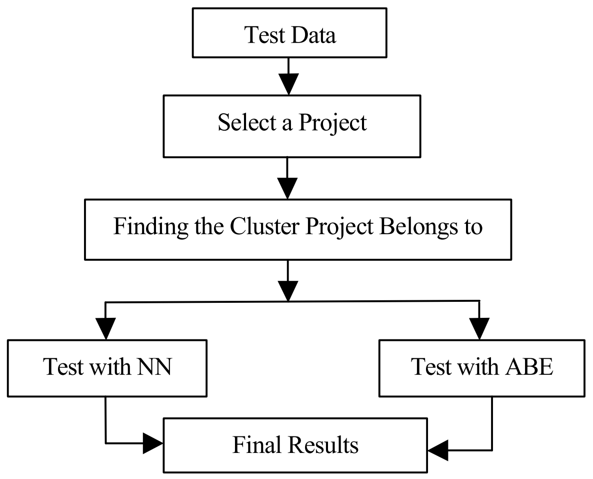

Fig. 3: Testing stage of proposed method

NN is employed for prediction the effort. These steps are repeated until all test projects are applied to the hybrid system. For each test project the MRE is calculated and the final result is computed based on MMRE and PRED (0.25).

Performance Evaluation: Performance of estimation methods is evaluated using several metrics including Relative Error (RE), Magnitude of Relative Error (MRE), Mean Magnitude of Relative Error (MMRE) and Percentage of the Prediction (PRED) which are computed as following [6] .

$$\mathrm{RE} = \frac{(\mathrm{Estitnate\text{-}Actual})}{\mathrm{Actual}} \tag{8}$$

$$\mathrm{MRE} = \frac{|\mathrm{Estitnate\text{-}Actual}|}{\mathrm{Actual}} \tag{9}$$

$$\mathrm{MMRE} = \frac{\sum_{i=1}^{N} \mathrm{MRE}}{N} \tag{10}$$

$$\mathrm{PRED}(X) = \frac{A}{N} \tag{11}$$

Where, A is the number of projects with MRE less than or equal to X and N is the number of projects in test set. Usually the ideal amount of X in software effort estimation methods is 0.25 and the various methods are compared based on this level.

Leave-One-Out: In this method a project is removed from the dataset as the target project. The ABE method is employed to estimate the effort of the target project by using all the remained projects in dataset.

Table 1: Desharnais Characteristics

Feature	Name	Type	Statement
TeamExp	Team experience	Numerical	Measured in years
ManagerExp	Manager's Experience	Numerical	Measured in years
YearEnd	Year project ended	Categorical	Determined by year
Length	Length of project	Numerical	Measured in Month
Transactions	Transactions	Numerical	Number of transactions
Entities	Entities	Numerical	Number of entities
PointsNonAdjust	Non Adjusted function points	Numerical	Number of Non adjusted function points
AdjustFctor	Sum of complexity factors	Numerical	Sum of complexity factors
PointsAdjust	Adjusted function points	Numerical	Number of adjusted function points
Language	Programming language	Categorical	1 = 1st generation
			2 = 2nd generation
			3 = 3rd generation
Effort	Development effort	Numerical	Measured in Person-Hours

Then the target project joins to the dataset and another project is selected as the target project. The mentioned process is repeated until all projects in dataset are estimated. The MRE is computed for each project after estimating and finally the MMRE is calculated. The mentioned technique is so common to evaluate the performance of ABE methods.

Three Fold Cross Validation: Since regarding the evaluation of ANN performance, three fold cross validation technique is so common, it was used to evaluate the performance of proposed method in this study. In this technique all projects are divided into three groups with the same size (or approximately same size) randomly. A group is used as the testing set and two other groups are considered as the training sets and afterward the result for testing set is computed. This process is repeated three times; therefore all projects will be assessed. Indeed all projects will be considered as test case once out of all iterations. The mean amount of obtained results from three iterations is treated as the final result and final MMRE and PRED are determined by computing the mean obtained amount of them from three iterations.

Dataset Description: Dasharnais [31] is one of the most common dataset in field of software effort estimation. In this paper Desharnais is used to evaluate the proposed method. Although Desharnais dataset is relatively old, it has been employed by many recent research works [32-34]. There are 81 projects of a Canadian company in the mentioned dataset which four out of them are missing therefore 77 projects are considered for evaluation process. Each project is described by eleven features. Two features out of eleven features are categorical and the rest features are numerical. In addition the first ten features are considered as the independent features and the final feature (effort) is treated as the dependent feature.

For the purpose of clustering, selection of high population dataset is compulsory. In other word, selecting a dataset with large number of projects leads to appear the supremacy of hybrid method compared with the other methods. Therefore, Desharnais dataset with seventy seven projects can be a suitable choice to show the power of proposed method. The supplementary information about this dataset is presented in Table 1.

Numerical Results: In this paper four clusters are considered to perform the training stage. Executing the C-means clustering algorithm on Desharnais dataset by different number of clusters showed that the best number of clusters is four. If the number of clusters is more than four then some clusters with one or two projects are appeared which are not suitable for estimating. On the other hand, considering less than four clusters leads to have some clusters with outlier data that decreases the performance of NN training. Evaluation of the proposed method is performed in three steps which are described as follows.

Results on All Data: At first step, all 77 projects are grouped into four clusters by means of C-Means clustering algorithm then ABE or NN are chosen for each cluster based on the number of projects. Table 2 shows the clusters obtained from C-Means algorithm and also it shows the selected method for each cluster.

Selecting a cluster for a project has been done according to the maximum degree of membership function. First and second clusters are marked by NN and two other clusters are marked by ABE. Maximum and minimum number of projects in a cluster are thirty two and six respectively.

Table 2: Summary of clustering on Desharnais

Cluster #	Projects	Selected Method
1	28	NN
2	32	NN
3	6	ABE
4	11	ABE

Table 3: Using NN and ABE separately

Method	Projects	MMRE	PRED (0.25)
ABE	77	0.80	0.26
NN	77	0.49	0.38

Table 4: Using NN and ABE on Clusters

Cluster #	Method	MMRE	PRED (0.25)
1	NN	0.17	0.79
2	NN	0.46	0.47
3	ABE	0.20	0.73
4	ABE	0.08	1
	Average	22.5	0.73

Before using the proposed method, ABE and NN are applied to all dataset separately and the performance parameters are presented in Table 3. For evaluation of NN the three fold cross validation has been performed and Leave-One-Out technique has been used to evaluate the ABE performance (described in previous sections). As it is seen, the performance of ABE is not satisfying by employing all data.

Table 4 illustrates the performance of ABE and NN in each cluster based on the performance parameters. By using three fold cross validation in cluster one, the amount of MMRE and PRED showed the significant improvement as compared to using all data (Table 3). Also by using Leave-One-Out technique the performance of ABE on cluster three and cluster four showed the noticeable precise increasing. But in cluster two the percentage of improvement is not considerable. Nevertheless average of all obtained results states that dividing all data into several clusters decreases the level of error significantly.

Figure 4 and Figure 5 depict the comparison of using NN and ABE on each cluster and on all data according to the MMRE and PRED(0.25) respectively. In both figures it is seen that that ABE method can present very accurate results in low population clusters compared with all data.

Results on Test Cases: Since similar projects have been grouped in four clusters, improvement of performance is predictable and it is not enough to conclude that the proposed method outperforms in all criteria. Therefore, two random test cases are determined to evaluate the proposed method as shown in Table 5.

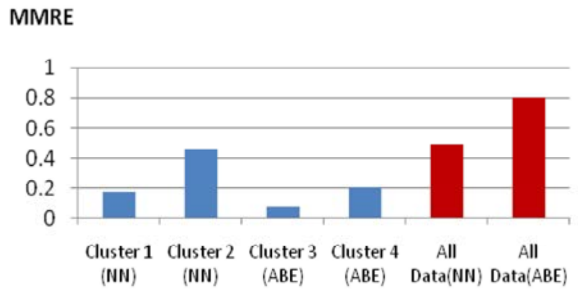

Fig. 4: MMRE in clusters versus all data

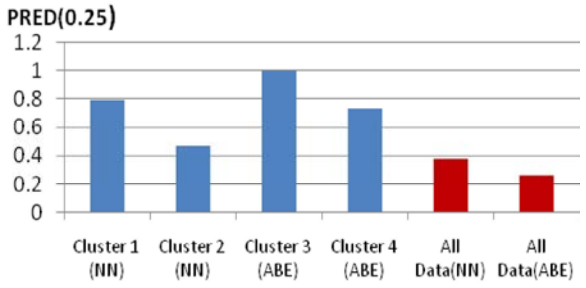

Fig. 5: PRED (0.25) in clusters versus all data

Table 5: Test Cases characteristics

Test Case	Projects	Description
1	11	Multiples of Seven (P_7, $P_{14, \ldots}$)
2	15	Multiples of Five (P_5, $P_{10\ldots}$)

The first test case includes eleven projects which their indices are multiples of seven therefore the training set includes sixty six remained projects. The second test case comprises of fifteen projects which their indices are multiples of five therefore the sixty two remained projects are considered as the training set. Figure 6 and Figure 7 show the actual and estimated effort obtained from applying the proposed method to test case one and test case two respectively.

The hybrid proposed method presents reasonable estimates for all eleven projects of test case one excluding projects three and eleven.

According to the Figure 7 the hybrid method presents accurate estimates in all fifteen projects excluding projects ten, fourteen and fifteen. In next step a comparison between NN, ABE, MLR, SWR and proposed method is presented based on two test cases.

Comparison of Results on Test Cases: In this step the results obtained from proposed method have been compared with NN, ABE, MLR and SWR separately. The MMRE and PRED (0.25) parameters are considered for evaluation of each method performance. Each method is applied to each test case (determined in Table 5) separately and the results are shown in Figure 8 and Figure 9.

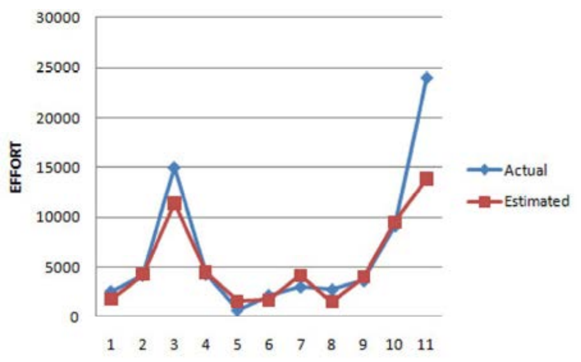

Fig. 6: Actual and estimated effort on first Test Case

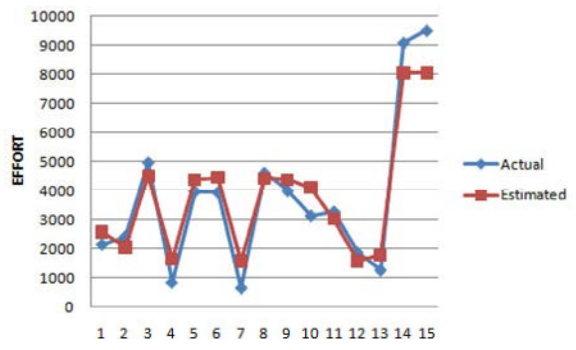

Fig. 7: Actual and estimated effort on second Test Case

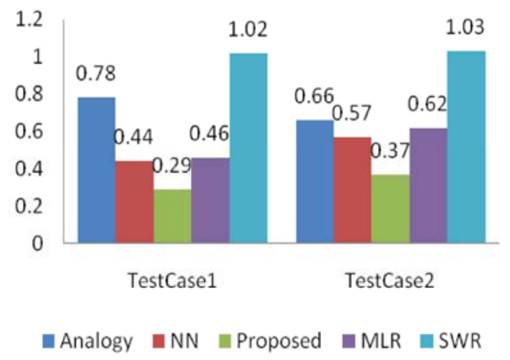

Fig. 8: Methods comparison based on MMRE

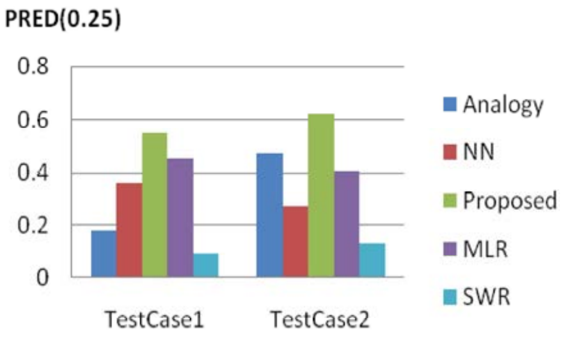

Fig. 9: Methods comparison based on PRED (0.25)

According to the Figure 8 in both test cases the proposed method presents more accurate results based on the MMRE. Also it should be mentioned that SWR in both test cases has the highest level of MMRE.

Figure 9 depicts comparison of PRED (0.25) among three methods by applying two test cases. The proposed method presents the more accurate results based on PRED(0.25) in both test cases compared with NN, ABE, MLR and SWR.

DISCUSSION

It was expectable that the performance of proposed method in each cluster will be acceptable so two test cases were determined randomly to evaluate the real performance of proposed method. But there are some key points about two selected test cases. In this study all dataset is considered to train and test the proposed method and there is no elimination. As seen in Figure 6 in the first test case, nine projects out of eleven projects have been estimated accurately but about project three and project eleven the estimated effort is far from the actual effort so the result is not satisfying for these two cases. Here it should be mentioned that the non normality level of Desharnais dataset is high as compare to the other datasets so there are several outlier projects in this dataset which are not similar to the other projects. Proposed method decreases the effect of these types of projects but still some outliers like project eleven influences on estimations. Despite there are two inaccurate forecasts, the overall estimations are acceptable. Regarding the second test case as seen in Figure 7 the overall level of accuracy is proper excluding the projects fourteen and fifteen which have been estimated slightly inaccurate. To evaluate the effect of combination of ABE, NN and clustering method, the results of applying combined method to the mentioned test cases have been shown in Figure 8 and Figure 9. According to the amount of MMRE and PRED the proposed hybrid method outperforms the NN, ABE, MLR and SWR. Due to using clustering technique the performance of ABE and NN in hybrid method increases significantly.

CONCLUSION

Irrelevant projects in software project datasets usually lead to the inaccurate estimations and make the process of training so difficult. Neural networks suffer from the high number of outliers in datasets especially in term of software project datasets. In this paper a new method was proposed to decrease the negative effect of irrelevant project on estimations. Instead of using all data for training the NN, several clusters were produced by means of fuzzy clustering to learn the NN separately. Also in low population clusters, ABE method was employed for predicting. Comparing of the MMRE and PRED computed for each cluster and all dataset showed that the level of accuracy in clusters is significantly higher than all dataset. In addition, two random test cases were applied to evaluate the proposed method and it was proved that NN and ABE could not present the accurate estimations separately. But in proposed hybrid method by combination of NN and ABE, the level of accuracy in both test cases was improved noticeably. According to the obtained results, it seems that the fuzzy clustering can be used widely in term of software effort estimation to increase the accuracy level of predictions. Therefore, we are going to apply this technique to other estimation methods as our future work.

ACKNOWLEDGEMENT

Special thanks to the Universiti Teknologi Malaysia for financing and funding this research through Research University Grant and also to our Embedded & Real-Time Software Engineering Laboratory (EReTSEL) members for their continuous support.

REFERENCES

1. Jones, C., 2007. Estimating software costs: Bringing realism to estimating, 2nd ed. New York: NY: McGraw-Hill,
2. Boehm, B.W. and R. Valerdi, 2008. Achievements and Challenges in Cocomo-Based Software Resource Estimation, IEEE Softw. 25: 74-83.
3. Boehm, B.W., 1981. Software engineering economics. Englewood Cliffs: NJ: Prentice Hall,
4. Putnam, L.H., 1987. A general empirical solution to the macrosoftware sizing and estimating problem, IEEE Transactions on Software Engineering, 4: 345-361.
5. Albrecht, A.J. and J.A. Gaffney, 1983. Software function, source lines of codes and development effort prediction: a software science validation, IEEE Trans Software Eng. SE, pp: 639-648.
6. Shepperd, M. and C. Schofield, 1997. Estimating Software Project Effort Using Analogies, IEEE Transactions on Software Engineering, pp: 23.
7. Molokken, K. and M. Jorgensen, 2003. A review of software surveys on software effort estimation, in Proceeding of International Symposium on Empirical Software Engineering, pp: 223-230.
8. Jorgensen, M. and K. Molokken Ostvold, 1994. How large are software cost overruns? A review of the 1994 CHAOS repor, Information and Software Technol., 48: 297-301.
9. Glass, R.L., 2006. The Standish report: does it really describe a software crisis?, Commun. ACM, 49: 15-16.
10. Galorath, D.D. and M.W. Evans, 2006. Software sizing, estimation and risk management: When performance is measured performance improves. Boca Raton: FL: Auerbach,
11. Kemerer, C.F., 1987. An empirical validation of software cost estimation models, Commun. ACM, 30: 416-429.
12. Jorgensen, M. and M. Shepperd, 2007. A Systematic Review of Software Development Cost Estimation Studies, Software Engineering, IEEE Transactions on, 33: 33-53.
13. Bhatnagar, R., *et al.* 2010. Software Development Effort Estimation Neural Network Vs. Regression Modeling Approach, International J. Engineering Science and Technol., 2: 2950-2956.
14. De Barcelos Tronto, I.F., *et al.* Comparison of Artificial Neural Network and Regression Models in Software Effort Estimation, in Neural Networks, 2007. IJCNN 2007. International Joint Conference on, pp: 771-776.
15. Heiat, A., 2002. Comparison of artificial neural network and regression models for estimating software development effort, Information and Software Technol., 44: 911-922.
16. Mair, C. and M. Shepperd, 2005. The consistency of empirical comparisons of regression and analogy-based software project cost prediction, in Empirical Software Engineering, 2005. 2005 International Symposium on, pp: 10.
17. Khatibi, B.V. and D.N.A. Jawawi, 2011. Software Cost Estimation Methods: A Review. Journal of Emerging Trends in Computing and Information Sciences, 2(1): 21-29.
18. Kalichanin-Balich, I. and C. Lopez-Martin, 2010. Applying a Feedforward Neural Network for Predicting Software Development Effort of Short-Scale Projects, in Software Engineering Research, Management and Applications (SERA), 2010 Eighth ACIS International Conference on, pp: 269-275.
19. Wittig, G. E. and G. R. Finnic, 1994. Using Artificial Neural Networks and Function Points to Estimate 4GL Software Development Effort, Australasian Journal of Information Systems. 1: 87-94.

20. Rao, B.T., 2009. A Novel Neural Network Approach For Software Cost Estimation Using Functional Link Artificial Neural Network (FLANN), International Journal of Computer Science and Network Security, 9: 126-131.
21. Kaur, J., *et al.* 2010. Neural Network-A Novel Technique for Software Effort Estimation, International J. Computer Theory and Engineering, pp: 2.
22. Reddy, C.S. and K. Raju, 2009. A Concise Neural Network Model for Estimating Software Effort, International J. Recent Trends in Engineering, 1: 188-193.
23. Samson, B., *et al.* 1997. Software cost estimation using an *Albus perceptron* (CMAC), Information and Software Technol., 39: 55-60.
24. Idri, A., *et al.,* 2010. Design of Radial Basis Function Neural Networks for Software Effort Estimation, IJCSI International J. Computer Sci., 7: 11-16.
25. Vinay, Kumar K. *et al.* Software development cost estimation using wavelet neural networks, J. Systems and Software, 81: 1853-1867.
26. Park, H. and S. Baek, 2008. An empirical validation of a neural network model for software effort estimation, Expert Systems with Applications, 35: 929-937.
27. Kalichanin-Balich, I. and C. Lopez-Martin, 2010. Applying a Feedforward Neural Network for Predicting Software Development Effort of Short-Scale Projects, presented at the Eighth ACIS International Conference on Software Engineering Research, Management and Applications,
28. Bezdec, J.C., 1981. Pattern Recognition with Fuzzy Objective Function Algorithms. New York: Plenum Press.
29. Walkerden, F. and R. Jeffery, 1999. An Empirical Study of Analogy-based Software Effort Estimation, Empirical Softw. Eng., 4: 135-158.
30. Angelis, L. and I. Stamelos, 2000. A Simulation Tool for Efficient Analogy Based Cost Estimation, Empirical Software Engineering, 5: 35-68.
31. Kadoda, G., *et al.* 2000. Experiences Using Case-Based Reasoning to Predict Software Project Effort in 4th International Conference on Empirical Assessment and Evaluation in Software Engineering,
32. Desharnais, J., 1989. Analyse statistique de la productivitie des projets informatique a partie de la technique des point des foncti on, Master, University of Montreal,
33. Auer, M., *et al.* 2006. Optimal project feature weights in analogy-based cost estimation: improvement and limitations, Software Engineering, IEEE Transactions on, 32: 83-92.
34. Li, Y.F., *et al.* 2009. A study of the non-linear adjustment for analogy based software cost estimation, Empir Software Eng., 14: 603-643.
35. Jodpimai, P., *et al.* 2010. Estimating Software Effort with Minimum Features Using Neural Functional Approximation, in Computational Science and Its Applications (ICCSA), 2010 International Conference on, pp: 266-273.