Increasing the Accuracy of Analogy Based Software Development Effort Estimation Using Neural Networks

Vahid Khatibi. B, Dayang N. A. Jawawi, and Elham Khatibi

***Abstract*—Estimation of development effort in software projects has been a challenging issue since many years ago. Uncertain behavior of software projects makes the estimating process difficult at early stage of project so that achieving to accurate estimations seems to be impossible in software projects. Neural Networks(NN) and Analogy Based Estimation (ABE) methods have been widely used in this field because the nature of which is adaptable with dynamic environment of software projects. Since most software project datasets include some irrelevant and inconsistent projects, the quality of neural network training and also quality of ABE estimations have been significant problems in all prior studies. In this paper, to overcome the problem of inconsistence projects, fuzzy clustering has been used for placing the similar projects in several clusters. A new framework was proposed to combine ABE and NN using C-Means clustering. The results showed that the proposed method improved the performance of NN and ABE noticeably.**

***Index Terms*—Development effort estimation, inconsistent projects, analogy based estimation, fuzzy clustering, neural network**

I. Introduction

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 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. Neural Network (NN) 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.

Neural networks can be useful to interpret the relation between software project attributes and final effort. 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. Several types of neural network with various structures have been proposed to estimate the effort of software development. 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 networks, feed forward has been widely used for software effort estimations [7-9] this type of neural network is useful to solve the estimation problems. Most common structure regarding the software estimations is two layers feed forward. The other types of neural network such as RBF and Perceptron can be seen in a few previous research works [10]-[11] and also Wavelet neural network has been employed in this field [12]. Due to variety of estimation methods in term of software project metrics, comparison framework is various and different in previous works but most of them tried to show the difference of algorithmic and non algorithmic methods by comparing regression techniques and neural network [13]-[15].

Manuscript received July 24, 2012; revised September 19, 2012.

B. V. Khatibi is with the Bardsir Branch, Islamic Azad University, Kerman, Iran.

D. N. A. Jawawi and E. Khatibi are with the Department of Software Engineering,Faculty of Computer Science and Information System, Universiti Teknologi Malaysia (UTM), 81310 Skudai, Johor, Malaysia (e-mail: dayang.n.a.jawawi@gmail.com).

II. Methodology

As mentioned before, neural networks have been widely used for software development effort estimation. One of the most important factors for improving the performance of

neural network is 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 relations among features are non linear and hard to understand. This problem arises due to uncertain and unstable nature of software projects. In most software project datasets by analyzing the project features it seems that there is no strong correlation between dependent and independent features. Therefore, this characteristic of software project datasets leads to appear high number of irrelevant projects which do not follow the overall behavior of dataset projects. Neural networks are confronted with profound difficulties when they are used on a software project datasets with high level of non normality. In such situation, estimations are not reliable because the process of training has been done with inconsistent and contradictory projects. The main idea behind the proposed method in this paper was splitting projects into several groups based on the level of similarity and using these groups to train the neural network separately. Since software projects are described by large number of features, C-Means clustering was used to categorize the projects. This method can perform accurately on datasets with many features. After performing the clustering on all projects it is possible that some clusters are appeared with a few projects that are not suitable for training the neural network. ABE method was applied to cover such low population clusters beside the neural network. Actually, C-Means clustering and ABE method were applied to proposed method as two complementary techniques to decrease the effect of outliers and irrelevant projects on estimations. The proposed method was organized in two main stages called training and testing stages. In training stage the structure of proposed hybrid method is configured and in testing stage the software development effort is estimated.

In training stage 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. If the number of projects in a cluster is less than 10 then this cluster is considered as an analogy cluster. There are three reasons to select a filter based on 10 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 10). Secondly, using of C-means clustering on several software datasets showed that clusters with less than 10 projects 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 datasets while low population clusters are not proper for training the NN. After clustering the most proper weights and biases of NN for each cluster with more than 10 projects are computed by several reconstructing the NN. Subsequently the testing stage is presented based on the results of training stage. The number of clusters depends on dataset characteristics and especially variance of data in dataset. If the dispersion level and variance are high in a dataset, therefore 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 computed to specify which cluster the project belongs to. The cluster with minimum distance is selected for project under estimating. 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 NN is employed for prediction the effort. These steps are repeated until all test projects are applied to the hybrid system. Final result is computed based on evaluation metrics.

A. Evaluation Procedure

In this section the overall evaluation trend for proposed method is described. Three fold cross validation and leave-one-out methods are explained, some information about used dataset is mentioned and performance metrics are described.

B. 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] .

$$\text{RE} = \frac{(Estimate - Actual)}{Actual} \quad (1)$$

$$\text{MRE} = \frac{|Estimated - Actual|}{Actual} \quad (2)$$

$$\text{MMRE} = \frac{\sum_{i=1}^{N} MRE}{N} \quad (3)$$

$$\text{PRED(X)} = \frac{A}{N} \quad (4)$$

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.

C. Dataset Description

For the purpose of evaluating the proposed method, Maxwell dataset [16] is used because this dataset is relatively new and comprises of 62 software projects(enough large). Each project in this dataset is described by 26 features and this high number of features is useful to show the performance of proposed method. All features excluding features 1, 24, 25, 26 are categorical and the mentioned features are numerical. High number of categorical features usually decreases the accuracy of analogy method. Therefore, we selected this dataset with 22 categorical features to appear the real improvement of analogy method by using proposed method. Table I gives some statistical information about Maxwell dataset. 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, Maxwell dataset with 62 projects can be a suitable choice to show the power of proposed method. The supplementary information about this dataset is presented in Table I.

TABLE I: MAXWELL CHARACTERISTICS

Feature	Description	Mean	Std Dev	Min	Max
Time	Time	5.58	2.13	1	9
App	Application type	2.35	0.99	1	5
Har	Hardware platform	2.61	1	1	5
Dba	Database	1.03	0.44	0	4
Ifc	User interface	1.94	0.25	1	2
Source	Where developed	1.87	0.34	1	2
Telonuse	Telon use	2.55	1.02	1	4
Nlan	Number of different development languages used	0.24	0.43	0	1
T01	Customer participation	3.05	1	1	5
T02	Development environment adequacy	3.05	0.71	1	5
T03	Staff availability	3.03	0.89	2	5
T04	Standards use	3.19	0.70	2	5
T05	Methods use	3.05	0.71	1	5
T06	Tools use	2.90	0.69	1	4
T07	Software's logical complexity	3.24	0.90	1	5
T08	Requirements volatility	3.81	0.96	2	5
T09	Quality requirements	4.06	0.74	2	5
T10	Efficiency requirements	3.61	0.89	2	5
T11	Installation requirements	3.42	0.98	2	5
T12	Staff analysis skills	3.82	0.69	2	5
T13	Staff application knowledge	3.06	0.96	1	5
T14	Staff tool skills	3.26	1.01	1	5
T15	Staff team skills	3.34	0.75	1	5
Duration	Duration	17.21	10.65	4	54
Size	Application size	673.31	784.08	48	3,643
Effort	Effort	8,223.21	10,499.90	583	63,69

III. NUMERICAL RESULTS

In this paper, four clusters are considered to perform the training stage. Applying the C-means clustering algorithm on Maxwell dataset with 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, splitting projects into less than four clusters leads to have some clusters with outlier data that decreases the performance of NN training. Evaluation of proposed method has been done based on two main steps as following.

A. Results on all Data

At first step all 62 projects are grouped into four clusters by C-Means clustering algorithm then ABE or NN are chosen for each cluster based on the number of projects. Table II shows the clusters obtained from C-Means algorithm and 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 as ABE. Maximum and minimum number of projects in a cluster is 28 and 2 respectively.

TABLE II: SUMMARY OF CLUSTERING ON MAXWELL

Cluster #	Projects	Selected Method
1	26	NN
2	28	NN
3	6	ABE
4	2	ABE

Before using the proposed method, ABE and NN are applied to all dataset separately and performance parameters are presented in Table II. Three fold cross validation has been performed for evaluation of NN 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 III: USING NN AND ABE SEPARATELY

Method	Projects	MMRE	PRED(0.25)
ABE	62	0.72	0.35
NN	62	0.48	0.51

B. Results on Clusters

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

TABLE IV: USING NN AND ABE ON CLUSTERS

Cluster #	Method	MMRE	PRED(0.25)
1	NN	0.13	0.69
2	NN	0.50	0.55
3	ABE	0.14	0.82
4	ABE	0.10	1
	Average	0.22	0.76

Fig. 1 and Fig. 2 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.

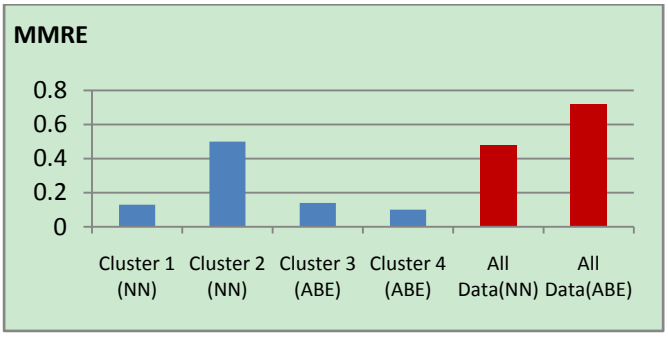

Fig. 1. MMRE (clusters versus all data)

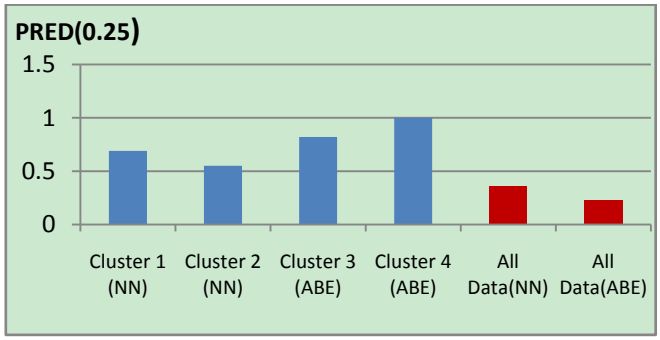

Fig. 2. PRED(0.25) (clusters versus all data).

IV. Conclusion

Existing of inconsistent and irrelevant projects in software project datasets lead to inaccurate estimation of software development effort. ABE and NN as two famous estimating methods are faced with this problem. In this paper, fuzzy clustering was used to classify software project into several groups. Indeed, it made the projects as consistent as possible. Moreover, NN and ABE were combined under a new framework to support defects of each other. Combining of these three methods, decreases the high normality level of software projects which has the negative effects on estimation process. A relatively large dataset was employed to investigate the performance of proposed method. This paper showed that the performance of ABE and NN when they used on clusters, was more accurate than using which on all projects. As future work, we are going to use other soft computing techniques to enhance the accuracy of software development effort estimation.

References

[1] C. Jones, "Estimating software costs: Bringing realism to estimating," *2nd ed*. New York: NY: McGraw-Hill, 2007.

[2] B. W. Boehm and R. Valerdi, "Achievements and challenges in cocomo-based software resource estimation," *IEEE Softw*., vol. 25, pp. 74-83, 2008.

[3] B. W. Boehm, "Software engineering economics," *Englewood Cliffs: NJ: Prentice Hall*, 1981.

[4] L. H. Putnam, "A general empirical solution to the macrosoftware sizing and estimating problem," *IEEE Transactions on Software Engineering*, vol. 4, pp. 345-361, 1987.

[5] A. J. Albrecht and J. A. Gaffney, "Software function, source lines of codes, and development effort prediction: a software science validation," *IEEE Trans Software Eng. SE*, vol. 9, pp. 639-648, 1983.

[6] M. Shepperd and C. Schofield, "Estimating software project effort using analogies," *IEEE Transaction on Software Engineering,*, vol. 23, pp. 736-743, 1997.

[7] J. Kaur, *et al*., "Neural network-a novel technique for software effort estimation," *International Journal of Computer Theory and Engineering*, vol. 2, pp. 17-19, 2010.

[8] C. S. Reddy and K. Raju, "A concise neural network model for estimating software effort," *International Journal of Recent Trends in Engineering*, vol. 1, pp. 188-193, 2009.

[9] I. Attarzadeh and S. H. Ow, "Software development cost and time forecasting using a high performance artificial neural network model," in *Intelligent Computing and Information Science*. vol. 134, R. Chen, Ed., ed: Springer Berlin Heidelberg, 2011, pp. 18-26.

[10] B. Samson, *et al*., "Software cost estimation using an Albus perceptron (CMAC)," *Information and Software Technology*, vol. 39, pp. 55-60, 1997.

[11] A. Idri, *et al*., "Design of radial basis function neural networks for software effort estimation," *International Journal of Computer Science*, vol. 7, pp. 11-16, 2010.

[12] K. V. Kumar, *et al*., "Software development cost estimation using wavelet neural networks," *Journal of Systems and Software*, vol. 81, pp. 1853-1867, 2008.

[13] I. K. Balich and C. L. Martin, "Applying a feedforward neural network for predicting software development effort of short-scale projects," in *Software Engineering Research, Management and Applications (SERA)*, , pp. 269-275, 2010.

[14] V. Khatibi. B and D. N. A. Jawawi, "Software cost estimation methods: A review," *Journal of Emerging Trends in Computing and Information Sciences*, vol. 2, pp. 21-29, 2011.

[15] V. Khatibi. B, *et al*., "Neural networks for accurate estimation of software metrics," *International Journal of Advancement in Computing Technology*, vol. 3, pp. 54-66, 2011.

[16] K. Maxwell, "Applied statistics for software managers," *Englewood Cliffs, NJ: Prentice-Hall*, 2002.