Improve software effort estimation using information entropy

Mohammad Javad Madari [1], Mehrnaz Niazi[2*]

Dept. Electrical and Computer Engineering, Pishtazan Institute of Higher Education

Shiraz, Iran

Abstract

Effort estimation is so important and determinative in management and development of software projects. Most of the approaches that have been proposed in software estimation are suffered from low accuracy according to limitation of dataset's samples. So, in this paper a hybrid method has been proposed in order to increase the accuracy and decrease the time complexity. In this way, a feature selection has been used before the main methods to improve the accuracy and reduce feature space complexity. Finally, hybrid method performances are promising better result than other algorithms.

Keywords: *Software Effort Estimation, Feature Selection, Statistical Algorithms, Functional Algorithms*

1. Introduction

Although many techniques have been used for software effort estimation, none of them given an accurate prediction, since there is not a sufficient data. prediction of accurate effort estimation is a serious challenge in many algorithms. So, project managers are looking for a proper method to increase their prediction accuracy and reduce their costs. As well reducing time can be one of the main reasons for directing project managers to more precise estimation[1]. As it mentioned before many algorithms have been proposed in this way, which we can categorize them into two main approaches. These approaches have been categorized based on functional analysis and statistical analysis. Constructive Cost Model (COCOMO) method is a procedural software cost estimation model firstly developed by Barry W. Boehm in 1970s[2]. This method used a predefined formula to compute software effort estimation and is good for quick, estimates of software costs, but its accuracy is limited due to its lack of attributes. Referenced to basic COCOMO, in 2000, COCOMO *II* have been developed[3]. This method claimed to be more accurate for estimating software development projects according to take more attributes into account which named cost driver. Product attributes, Hardware attributes, Personnel attributes are stated as samples of cost driver attributes. Despite all the above mentioned, COCOMO families are not generalized and limited to specific datasets. So, it cannot be useful for any kind of dataset. In the other hand, many approaches have been introduced for statistical analysis of software effort estimation which can be used and trained for any kind of datasets. So, this paper, concentrated on common statistical methods for estimates of software costs, and try to improve the precision. XGBoost, is one of the statistical methods[4], which has been proposed by Tianqi Chen as a research project. This method is very flexible and comprehensive tool that can work through regression, classification, ranking of problems as well as user-generated performance[5]. Random forest is the other statistical approach which have been firstly created by Tin Kam Ho in 1995 and developed by Leo Breiman in 2001 [6] and Adele Cutler in 2012[7], It is used as an ensemble learning method for regression and classification, and constructed with multiple decision trees. Deep learning is another statistical method, which have been noted a lot recently. In this method an architecture of deep neural network has been defined and weight and bias of layers have been trained based on train data, in order to predict the software effort[8]. K nearest neighbors (KNN) is also used as a regressor to estimate software effort; this method is a simple algorithm that work based on a similarity measure of data[9]. Another non-parametric algorithm named K-means which can be used with different distance criterion and used as a regressor to predict software effort[10][11]. In continue, a brief review of some basic concepts is given in section2. In section3 our hybrid method has been proposed. In the next section experimental results are discussed and finally conclusion is given in the last section.

2. Basic concepts

In this part, we first introduce a feature selection method named information entropy and review some common statistical methods in the field of software cost estimation.

2.1. Feature selection method

Feature selection is a process of selecting a subset of features that are more relevant in order to utilized in model construction. These techniques are usually used in order to reduce the feature space, simplification and enhancement. In this part, a feature selection introduced to enhance the results as well as feature space reduction.

2.1.1. Information entropy

Information entropy shows the average rate of information is produced by a stochastic data. According to entropy definition, if a data has lower probability value, it may have more information and have better discriminatory power and if it has higher probability it may carry out less information[12][13]. So, information entropy can be used as a feature selection in order to select more informative and discriminative features and can be calculated as equation(1):

$$E = -\sum_{I} P_i log P_i \qquad (1)$$

2.2. Statistical methods

2.2.1. XGBoost

The xgboost method was presented by Tiangi Chen and Carlos Guestrin in 2016. XGBoost is an applied tree system that proposed in order to gain better results in many regression and classification machine learning challenges; the two important factors in xgboost can be mentioned are: Scalable learning system and Usage of statistical models. a highly scalable tree boosting system have been designed and a weighted quantile sketch has been used as a split finding algorithm for efficient calculation, in approximate tree learning and a novel sparsity-aware algorithm use for parallel tree learning. Finally, these techniques have been combined to make a system that scales to larger data with the least number of clusters[4].

2.2.2. Random forest

Random forest was first introduced by Tin Kam Ho in 1995[14], and extended by Leo Breiman. The random forest is an ensemble method and starts with a technique called a decision tree. Ensembles are used to improve performance. The main idea of ensemble methods is to construct strong learner via a group of weak learners. The random forest combining trees with the notion of an ensemble. So, the trees are known as weak learners and the random forest is known as a strong learner[15]. a random forest has been shown in figure1.

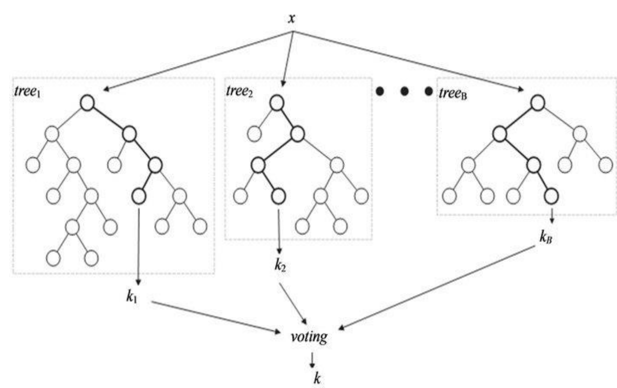

Fig 1: shows a random forest

2.2.3. Multilayer Perceptron

A multilayer perceptron (MLP) is a class of feedforward artificial neural network. MLP use a supervised learning technique called backpropagation for training the network. In this part, An MLP (Artificial Neural Network - ANN) with a four hidden layer, ReLU activations functions to eliminate vanishing during the network training and a sigmoid activation function of output layer represented and viewed as a logistic regression classifier[12][16]. a multilayer perceptron with one hidden layer has been shown in figure2.

IJCSI International Journal of Computer Science Issues, Volume 16, Issue 2, March 2019
ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784
www.IJCSI.org https://doi.org/10.5281/zenodo.3234115

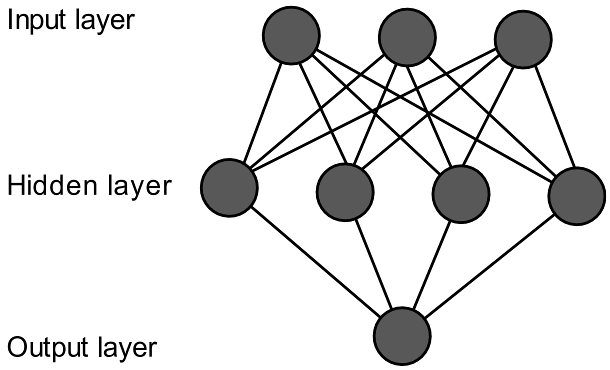

Fig 2: shows multilayer perceptron with one hidden layer

2.2.4. Kmeans and hamming distance

k-means clustering aims to discriminate data into k clusters in which each data belongs to the cluster with the nearest mean. In this part, a kmeans clustering first used in order to cluster train data into k cluster and labels of test data have been defined, according to cluster centers[17][18]. Finally, the hamming distance of test data with all data in the same cluster have been calculated in order to predict software effort estimation. a Kmeans regressor has been shown in figure3.

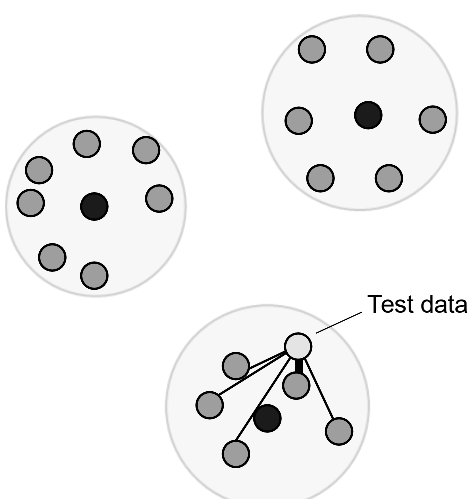

Fig 3: shows the kmeans regressor with hamming distance

3. proposed method

In this paper a hybrid method has been proposed which have been constructed from two terms; a feature selection and a statistical regressors. Regarding to this, information entropy, have been used in order to select efficient features[16]. In the next step, three common statistical regressor named, XGBoost, random forest, multilayer perceptron and a kmeans with hamming distance regressor have been applied on data with selected features. This combination will be used, not only for reducing the feature space but also for development of software effort estimation accuracy. Our proposed method steps can be seen in figure 4.

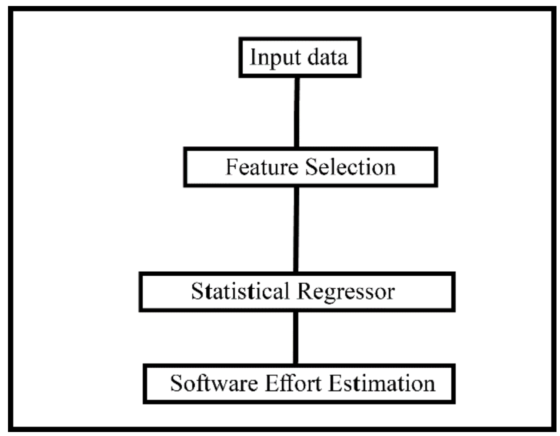

Fig 4: shows proposed method steps.

4. Experimental results

In this paper, a famous dataset named NASA with 17 features used to evaluate our proposed method. Features of NASA dataset have been introduced in table 1.

Table 1: shows features of NASA dataset.

	Feature	description
1	ACAP	analyst's capability
2	PCAP	programmer's capability
3	AEXP	application experience
4	MODP	Modern programing practices
5	TOOL	use of software tools
6	VEXP	virtual machine experience
7	LEXP	language experience
8	SCED	schedule constraint
9	STOR	main memory constraint
10	DATA	data base size
11	TIME	time constraint for CPU
12	TURN	Turnaround time
13	VIRT	machine volatility
14	CPLX	process complexity
15	RELY	Required software reliability
16	LOC	line of code
17	ACTUAL EFFORT	actual effort

IJCSI International Journal of Computer Science Issues, Volume 16, Issue 2, March 2019
ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784
www.IJCSI.org https://doi.org/10.5281/zenodo.3234115

The standard numeric values of the effort multipliers are mention in table 2.

Table 2: shows numeric values of the effort multipliers.

Features	Very low	Low	nominal	high	Very high	Extra high	Productivity range
CAP	**1.46**	**1.19**	**1.00**	**0.86**	**0.71**		**2.06**
PCAP	**1.42**	**1.17**	**1.00**	**0.86**	**0.70**		**1.67**
AEXP	**1.29**	**1.13**	**1.00**	**0.91**	**0.82**		**1.57**
MODP	**1.24**	**1.10**	**1.00**	**0.91**	**0.82**		**1.34**
TOOL	**1.24**	**1.10**	**1.00**	**0.91**	**0.83**		**1.49**
VEXP	**1.21**	**1.10**	**1.00**	**0.90**			**1.34**
LEXP	**1.14**	**1.07**	**1.00**	**0.95**			**1.20**
SCED	**1.23**	**1.08**	**1.00**	**1.04**	**1.10**		
STOR			**1.00**	**1.06**	**1.21**	**1.56**	**-1.21**
DATA		**0.94**	**1.00**	**1.08**	**1.16**		**-1.23**
TIME			**1.00**	**1.11**	**1.30**	**1.66**	**-1.3**
TURN		**0.87**	**1.00**	**1.07**	**1.15**		**-1.32**
VIRT		**0.87**	**1.00**	**1.15**	**1.30**		**-1.49**
CPLX	**0.70**	**0.85**	**1.00**	**1.15**	**1.30**	**1.65**	**-1.86**
RELY	**0.75**	**0.88**	**1.00**	**1.15**	**.40**		**-1.87**

The dataset is split into two subsets; a training set and a test set. The training set is used for learning; whereas the test set is used for evaluating the accuracy. In this paper, 80% of data are utilized as training and 20% are used as test data.
In the first term, a feature selection named information entropy has been used in order to select efficient features. Information entropy is utilized to distinguish fewer effective features. As it mentioned before, the features with high entropy has less information and we can omit features with less information. Figure 5 and table 3 shows the entropy of 17 features.

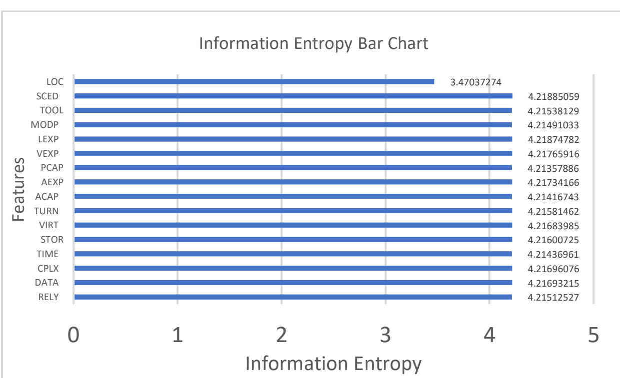

Fig 5: shows information entropy of 17 features.

Table 3: shows information entropy of 17 features.

Feature	Information Entropy
RELY	4.21512527
DATA	4.21693215
CPLX	4.21696076
TIME	4.21436961
STOR	4.21600725
VIRT	4.21683985
TURN	4.21581462
ACAP	4.21416743
AEXP	4.21734166
PCAP	4.21357886
VEXP	4.21765916
LEXP	4.21874782
MODP	4.21491033
TOOL	4.21538129
SCED	4.21885059
LOC	3.47037274

As it is obvious, two features of LEXP, SCED have the most values of entropy and so the less information. Considering the result of information entropy, we omit two features named LEXP, SCED. In the next step, four regressors named Xgboost, random forest, multilayer and kmeans with hamming distance have been applied on data with the selected features. finally, the hybrid methods have been utilized on test data to evaluate the results. In order to have more reliable results, the results have been extracted from the average of five iteration with different test data and all data used at least one time as a test data. The result of these method without feature selection and the results after feature selection have been compared with mean magnitude of the relative error (MMRE), which is the percentage of the absolute values of the relative errors, averaged over the T items in the test data and calculated as mentioned in equation (2)-(4):

$$RE.i = \frac{(estimate.i - actual.i)}{actual.i} \quad (2)$$

$$MRE.i = abs(RE.i) \quad (3)$$

$$MMRE.i = \frac{100}{T * (MRE.1 + MRE.2 + \cdots .MRE.T)} \quad (4)$$

Result of MMRE of four methods have been indicated in table 4. For better comparison the results have been presented in bar chart in figure 6.

Table 4: shows the result of MMRE on RF, MLP and kmeans without feature selection in the second column and the result of these three algorithms after feature selection in third column.

algorithms	All features	Selected features
XGBoost	79.22	71.66
Random Forest	54.8	52.83
MLP	98.21	78.41
Kmeans	75.13	71.98

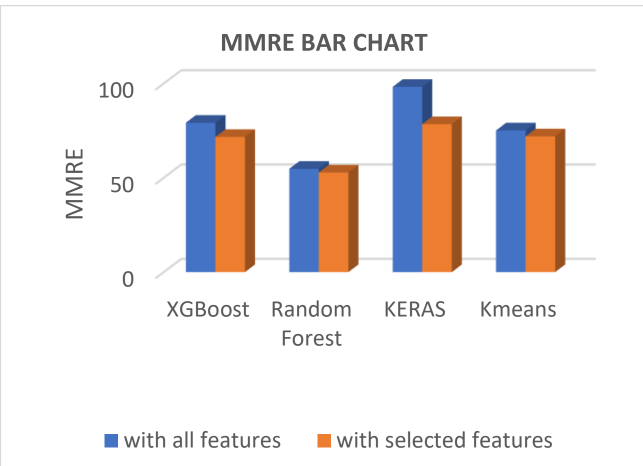

Fig 6: shows the result of regressors with all features and with selected features.

as it has been shown in table 4 and figure 6, the MMRE of all the algorithms have decrease and multilayer perceptron has the most improvement and random forest has the best and the less MMRE after using the information entropy as a feature selection.

5. Conclusion

In this paper, a hybrid method proposed with a feature selection algorithm, and two features named, LEXP, SCED have been removed according to the result of information entropy. Finally, the four regressor named Xgboost, random forest, multilayer perceptron and Kmeans with hamming distance is applied on data with selected features. regarding to this the feature space and the MMRE of the methods have been decrease.

References

[1] A. Idri, A. Zakrani, and A. Zahi, “Design of Radial Basis Function Neural Networks for Software Effort Estimation,” *IJCSI Int. J. Comput. Sci. Issues*, vol. 7, no. 4, pp. 11–17, 2010.

[2] S. Chulani and B. Boehm, “Software engineering economics,” in *Software Management, Seventh Edition*, Prentice-Hall, 2007, pp. 203–225.

[3] and B. S. Barry Boehm, Chris Abts, A. Winsor Brown, Sunita Chulani, Bradford K. Clark, Ellis Horowitz, Ray Madachy, Donald J. Reifer, “Software Cost Estimation with Cocomo {II},” *Englewood Cliffs, NJ:Prentice-Hall*, p. 502, 2000.

[4] C. G. Tianqi Chen, “XGBoost: A Scalable Tree Boosting System,” in *Proceeding KDD ’16 Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, no. 8, pp. 785–794.

[5] S. M. Satapathy, B. P. Acharya, and S. K. Rath, “Class point approach for software effort estimation using stochastic gradient boosting technique,” *ACM SIGSOFT Softw. Eng. Notes*, vol. 39, no. 3, pp. 1–6, Jun. 2014.

[6] L. Breiman, “Random Forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[7] A. Cutler and D. R. Cutler, *Ensemble machine learning*, no. January. Springer, Boston, MA, 2012.

[8] A. Marblestone, G. Wayne, and K. Kording, “Towards an integration of deep learning and neuroscience,” *Front. Comput. Neurosci.*, vol. 10, p. 94, Sep. 2016.

[9] Y. Song, J. Liang, J. Lu, and X. Zhao, "An efficient instance selection algorithm for k nearest neighbor regression," *Neurocomputing*, vol. 251, pp. 26–34, Aug. 2017.

[10] G. Hamerly and C. Elkan, "Alternatives to the k-means algorithm that find better clusterings," in *Proceedings of the eleventh international conference on Information and knowledge management - CIKM '02*, 2002, p. 600.

[11] D. S. Brian S. Everitt, Sabine Landau, Morven Leese, *Cluster analysis*. Wiley, 2011.

[12] R. de A. Araújo, A. L. I. Oliveira, and S. Meira, "A class of hybrid multilayer perceptrons for software development effort estimation problems," *Expert Syst. Appl.*, vol. 90, pp. 1–12, Dec. 2017.

[13] C. Largeron *et al.*, "Entropy based feature selection for text categorization To cite this version : HAL Id : hal-00617969 Entropy based feature selection for text categorization," in *ACM Symposium on Applied Computing*, 2011, pp. 924–928.

[14] Tin Kam Ho, "Random decision forests," in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, 1995, vol. 1, pp. 278–282.

[15] J. Moeyersoms, E. Junqué De Fortuny, K. Dejaeger, B. Baesens, and D. Martens, "Comprehensible software fault and effort prediction: A data mining approach," *J. Syst. Softw.*, vol. 100, pp. 80–90, Feb. 2015.

[16] P. Rijwani and S. Jain, "Enhanced Software Effort Estimation Using Multi Layered Feed Forward Artificial Neural Network Technique," in *Procedia Computer Science*, 2016, vol. 89, pp. 307–312.

[17] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, Jun. 2010.

[18] G. Hamerly and C. Elkan, "Alternatives to the k-means algorithm that find better clusterings," in *Proceedings of the eleventh international conference on Information and knowledge management - CIKM '02*, 2002, p. 600.