Software Effort Estimation Using Machine Learning Methods

Bilge Başkeleş, Burak Turhan, Ayşe Bener
Department of Computer Engineering, Boğaziçi University
bilge.baskeles@boun.edu.tr, turhanb@boun.edu.tr, bener@boun.edu.tr

Abstract

In software engineering, the main aim is to develop projects that produce the desired results within limited schedule and budget. The most important factor affecting the budget of a project is the effort. Therefore, estimating effort is crucial because hiring people more than needed leads to a loss of income and hiring people less than needed leads to an extension of schedule. The main objective of this research is making an analysis of software effort estimation to overcome problems related to it: budget and schedule extension. To accomplish this, we propose a model that uses machine learning methods. We evaluate these models on public datasets and data gathered from software organizations in Turkey. It is found out in the experiments that the best method for a dataset may change and this proves the point that the usage of one model cannot always produce the best results.

1. Introduction

Software cost estimation which implicitly includes software effort estimation has been the center of attention since the late 1970s [1]. Many software cost estimation models have been proposed such as SLIM, Checkpoint, Price-S, Seer and COCOMO [1]. The developers of these models have all faced the same problem: as software grew in size and importance it also grew in complexity, making it very difficult to accurately predict the cost of software development [1]. Many methods have been proposed to accurately estimate cost as a function of a large number of cost factors. These methods include not only parametric models such as SLIM, COCOMO but also methods such as regression analysis and expertise-based analysis.

In this research, we planned to collect software development related data including COCOMO related metrics in companies ranged from small to medium size. The collection of the data is carried out using questionnaires and metric-collection programs. The major difficulty in collecting the data is the scarcity of data due to the unwillingness or lack of data collection at the companies' side. We have also used datasets from previous research such as USC and NASA. After the data is collected, we applied machine learning methods for effort estimation, such as back-propagation neural networks, regression trees, radial basis functions and support vector regression methods and we compared the results. These methods are used with historical data as input to learn from previous experiences and then they could be used for estimation. The data has been multi-organizational rather than company specific for the sake of a general estimation. This also brings the possibility of constructing a general model for effort estimation

The major contribution of this research is the development of a hybrid effort estimation model that uses model-related data with machine learning methods. We then compared the different machine learning methods as to how they perform when our proposed model is used.

In the second part of this paper, related work about the former cost estimation models related to our estimation model is given. In the third section the problem is stated. In the fourth section, our methodology and machine learning methods used in our model are explained. In the fifth section experimental results and their comparison is given. In the last section, conclusions and future work are stated.

2. Related Work

Reliable and accurate estimates of software development costs are needed throughout the software development cycle to determine feasibility of software projects and to provide for appropriate allocation of resources. In the literature, many methods have been proposed for predicting the cost of software [3]. One of the most popular software estimation methods is estimation by expert knowledge. In this method, the reliability of estimates based on expert judgement depends on the degree in which a new project concurs with the experience and the ability of the expert to

1-4244-1364-8/07/$25.00 2007 IEEE

remember facts of historical projects [2, 3]. An important problem in using this method is that it is difficult for someone else to reproduce and use the knowledge and experience of an expert [2, 3].

Other methods in literature include parametric models (COCOMO, COCOMO II) and learning-oriented techniques [4, 5, 6]. In parametric models the estimation of development time and effort is a function of a number of variables [3]. The nucleus of such an estimation model is a number of algorithms and parameters (variables). The values of the parameters and the kind of algorithms are based on the content of a database of completed projects and research.

There have been two main approaches that are referred to as learning-oriented techniques. One of them is learning by analogy and the other one is neural networks [1].The analogy cost estimation technique involves comparing one or more completed projects in a similar domain as a means of producing new estimates. This can be achieved by analyzing the data collected from these completed projects against the new project, and by assessing similarities. However, there are some problems with this model such as the restriction of project features to information that is available at the start of the estimation process and the number of projects that should be measured against [1]. A previous study compares the use of analogy with prediction models based upon stepwise regression analysis and it yields higher accuracies in estimation by analogy [7]. The usage of neural networks is the alternative to mean least squares regression [1]. These regression methods are based on mean least square error. The simple idea behind this method is using historical data to train the network and produce better estimates by adjusting the parameters of the network. It has been used in software engineering from defect prediction to schedule estimation [9].

3. Problem Statement

Software quality is driven by many aspects in software development. In software development organizations, the main aim is to start a project and finish it within acceptable schedule and budget. The main drivers affecting the budget are the number of employees and time. These two come together to form a measure called effort [1, 2]. Effort is the most important factor in a software project determining the cost. Estimating effort is a very important factor affecting the quality of software. At early stages of software development, effort must be estimated to come up with a planned schedule and budget. Software processes constantly evolve as new and different technologies and applications are developed and used. Especially, in current software industry, where changes are arbitrary, an evolving system is required for the problem of effort estimation.

In this research, the main aim is to improve software effort estimation by constructing a learning system that makes use of several machine learning methods on software effort estimation datasets. The main reason for using such a learning system for this problem is to keep the estimation process up-to-date by incorporating up-to-date project data. If the effort estimation process evolves with new projects, the estimation model is kept up-to-date. This overcomes the main problem in parametric estimation models. The learning system here evolves with new projects to come and create better estimations for effort while depending on the efficiency of metrics formation invented earlier such as COCOMO and COCOMO II.

4. Data and Models

For the particular problem of effort estimation, we follow from the fact that software engineering research is the glue between theory and practice [12]. We have used data from public data repositories and we also collected data from software organizations in Turkey. We try to construct models for estimating effort by using machine learning approaches.

We use publicly available data that is collected by other software researchers around the world. Two different datasets USC and NASA have been obtained this way [11]. The main characteristic of these datasets is that they have been used by many researchers before and they are proven to be trusted to use in any research [8, 11]. USC data consists of data from 63 projects and 17 metrics including effort value in person-month. NASA dataset consists of data from 60 projects and 17 metrics which are same as in the USC dataset.

The dataset collected from software organizations in Turkey, SoftLab Data Repository (SDR), consists of 23 projects ranging from various automation projects to online-banking applications. We have used a questionnaire to be filled up and we have prepared a detailed tutorial on how to collect data [13]. While the other datasets are fairly old ones, SDR dataset is both a recent one and reflects the properties of Turkish software industry.

4.1 Methods Used in the Implementation

The machine learning methods are generally used for classification, learning and estimation problems where the usage of historical data is stressed upon. The

real world application areas include not only speech and pattern recognition but also financial predictions and analysis [10].

In this research, we use several machine learning methods as tools to estimate effort using past data gathered by experience using COCOMO guidelines based on COCOMO and COCOMO II models (See Figure 1). The methods we use for effort estimation are back propagation multilayer perceptrons, regression trees, radial basis function (RBF) and support vector regression (SVR). Principal components analysis (PCA) has also been used as a method to reduce dimensionality.

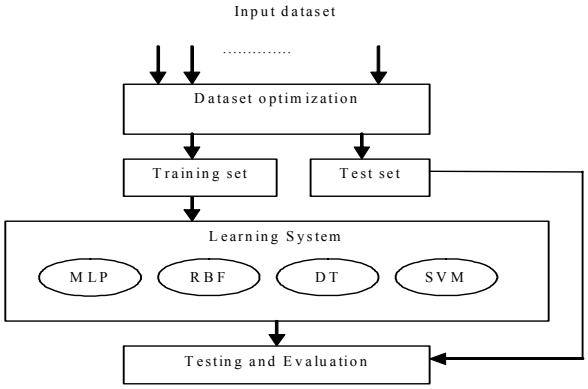

Figure 1. Learning Model for Effort Estimation

4.2 Evaluation of Experimental Results

The performance of effort estimation datasets and models can be evaluated using a number of different approaches such as PRED(L), MMRE (mean magnitude of relative error) and correlation which are measures that are widely known and almost always used in research on effort estimation for comparison purposes [2]. PRED, mdMRE and MMRE are the de facto standard evaluation criterions to assess the accuracy of software project prediction models [15, 16]. PRED(L) is calculated form the relative error which states the difference between the actual and estimated effort. The equation for PRED(L) is below:

$$PRED\quad(L) = \frac{k}{N} \qquad (1)$$

Here, the prediction accuracy is at the level of L, where k is the number of observations where MRE is less than or equal to L. L is determined according to the accuracy expectations. Commonly 25 or 30 are used as values of L [2]. In the experiments, L is taken to be 30. It is suggested that a model is acceptable if PRED value is greater than or equal to 75 percent of the predicted values [14].

The implicit rationale for using relative error measures such as MRE, rather than an absolute one, is to have a measure that is independent of the output value. The concept of relative error makes intuitive sense to software researchers and practitioners because it is reasonable to predict effort of small and large systems with the competing relative error rather than the competing absolute error [15, 16]. This need is also supported by the use of PRED where it is reasonable to choose models with high PRED meaning high percentage of accuracy. MMRE is used to select between competing models where models with low MMRE is preferred. It is also used to provide a quantitative measure of the uncertainty of a prediction where a low MMRE is taken to mean low uncertainty or inaccuracy.

5. Experimental Results

5.1 Results for NASA dataset

In these experiments, there are two types of experiments according to the size of the training and test sets. The machine learning methods are applied firstly with 50 projects in the training set and 10 projects in the test set and secondly with 40 projects in the training set and 20 projects in the test set. All experiments are carried out by cross validation.

The results for the 16 different experiments are given in Table 1. From the table, it is observable that RT is the best performer. One common behavior of all the algorithms is that the error increases when the size of the training set decreases. Another common behavior for MLP, RBF and SVR is the decrease in error when PCA is applied as an initial method to decrease number of dimensions. This behavior may suggest the existence of unnecessary attributes in COCOMO model, especially considering the difference in MMRE values.

When COCOMO parametric model is used, the mdMRE is 18.63, MMRE is 29.50 and PRED(30) is 73.33. Considering this, the results of RBF, RT and SVR are better than or equal to the results for COCOMO parametric model. This shows that the learning system is comparable with the COCOMO model in estimating effort. RT method is satisfying as an estimator since PRED(30) is greater than 75, whereas in the parametric model PRED(30) is less than 75.

Table 1. Summary of experimental results for NASA dataset

NASA (# of training)	Without PCA		With PCA	
	MMRE (%) mdMRE(%)	PRED (%)	MMRE (%) mdMRE(%)	PRED (%)
MLP (50)	110.94 68.01	33	109.96 65.75	34
MLP (40)	166.57 76.89	23	130.62 74.58	24.50
RBF (50)	22.11 14.44	73.25	20.19 11.78	75.25
RBF (40)	22.39 15.47	73	20.27 11.99	75
DT (50)	12.85 5.31	83.75	22.51 11.03	72
DT (40)	13.10 6.29	83.50	23.22 12.58	67.75
SVM (50)	25.72 9.39	75	23.11 7.53	77
SVM (40)	47.56 14.95	69.50	23.43 8.71	75.75

5.2 Experimental Results for USC dataset

In these experiments, there are two types of experiments according to the size of the training and test sets. The machine learning methods are applied firstly with 53 projects in the training set and 10 projects in the test set and secondly with 42 projects in the training set and 21 projects in the test set. All experiments are carried out by cross validation.

The results for the 16 different experiments are given in Table 2. For this dataset, the general observations are the almost the same as stated for NASA dataset. Here, MMRE increases when the training set's size decreases. Again, the decrease is low for methods such as RBF and DT. For this dataset, for MLP, RBF and SVR methods, the initial application of PCA causes a decrease in MMRE. However, it causes an increase in error for RT method. But, unlike NASA dataset, PCA is considerable here, since the best performing combination of methods is RBF and PCA. These results show the existence of unnecessary attributes in the dataset probably caused by low deviation.

When COCOMO parametric model is used, the mdMRE is 28.47, MMRE is 49.16 and PRED(30) is 50.79. Considering this, the results of RBF, RT and SVR are better than or equal to the results for COCOMO parametric model. This shows that the learning system is highly compatible with the COCOMO model in estimating effort.

Table 2. Summary of experimental results for USC dataset

USC (# of training)	Without PCA		With PCA	
	MMRE (%) mdMRE(%)	PRED (%)	MMRE (%) mdMRE(%)	PRED (%)
MLP (53)	116.56 82.31	11.49	91.73 75.37	14.62
MLP (42)	126.18 82.80	8.12	114.99 78.13	8.50
RBF (53)	13.38 8.42	84.76	12.15 7.44	88.33
RBF (42)	14.56 9.61	84.50	12.46 7.46	86
DT (53)	13.57 3.98	85.24	33.12 9.15	72.14
DT (42)	14.25 5.18	85	37.97 17.19	66
SVM (53)	30.94 14.83	75.5	33.28 13.77	82
SVM (42)	31.02 16.45	70	33.39 15.66	73.81

5.3 Experimental Results for SDR dataset

In these experiments, there are two types of experiments according to the size of the training and test sets. The machine learning methods are applied firstly with 20 projects in the training set and 3 projects in the test set and secondly with 16 projects in the training set and 7 projects in the test set. All experiments are carried out by cross validation.

The results for the 16 different experiments are given in Table 3. For this dataset, the general

observations are almost the same as stated for NASA and USC datasets. Here, MMRE increases when the training set's size decreases. Again, the decrease is low for methods such as RBF and RT.

Table 3. Summary of experimental results for SDR dataset

SDR (# of training)	Without PCA		With PCA	
	MMRE (%) mdMRE(%)	PRED (%)	MMRE (%) mdMRE(%)	PRED (%)
MLP (20)	103.59 64.89	31.50	82.82 66.60	32.00
MLP (16)	199.41 69.64	25.00	141.88 89.51	29.50
RBF (20)	19.38 9.57	79.00	18.40 8.31	79.50
RBF (16)	22.81 15.46	74.00	20.69 10.10	75.00
DT (20)	13.14 4.36	87.50	18.83 7.55	77.50
DT (16)	12.50 5.02	87.00	24.35 11.54	73.00
SVM (20)	25.96 7.72	74.50	19.63 5.17	80.50
SVM (16)	49.41 8.45	70.50	22.59 7.02	79.25

For this dataset, in MLP, RBF and SVR methods, the initial application of PCA causes a decrease in MMRE. However, it causes an increase in error for RT method. The case here is alike to the case for NASA dataset where RT outperforms the rest of the methods. RT method is satisfying as an estimator since PRED is greater than 75.

When COCOMO parametric model is used, the mdMRE is 1860.32, MMRE is 3368.15 and PRED(30) is 8.7. Considering this, the results of all the methods are better than the results for COCOMO parametric model. This shows that the learning system outperforms the parametric model. This result might be due to the fact that current software practices include tools that automatically produce most of the code itself. This approach makes the size bigger when the effort is low considering size. Although there is a metric called TOOL in the COCOMO model, it fails to adopt this practice and does not reflect the consequences to the estimations. Thus, the strength of our model can be seen more apparently in this case.

6. Conclusions and Future Work

In this research, we have conducted experiments on software effort estimation using various machine learning methods on three different datasets one of which is a native dataset. We have obtained acceptable results in the context of software effort estimation, especially when considering the results of the parametric models that are previously used. As a conclusion, we have seen that parametric models are insufficient for software effort estimation and the problem must be handled using an evolving system rather than a static one. The metrics based on those models might be changed or calibrated or new metrics can be added to improve the model.

Models based on limited data sets such as COCOMO and COCOMO II tend to incorporate the particular characteristics of the data. This results in a high degree of accuracy for similar projects, but it restricts the application of the model. As we have seen in the experiments, COCOMO model is good at estimating effort for the two datasets USC and NASA. It is not surprising that for these two datasets the results of the model is good. However, COCOMO II model fails to perform well on the SDR dataset which we have collected. We think that this result is due to the fact that most of the software have been developed using some readily available software tools such as Visual Studio or Java plug-ins which provides automatically created code. This way a considerable percentage of the code is created by the program rather than the coders. Although there is a related metric called TOOL in both models COCOMO and COCOMO II, the effect of this metric does not suffice. This is why considering the huge values of size, the results for effort is also very high causing a huge inaccuracy in the SDR dataset because most of the projects in that dataset are developed within 2-3 years and they include heavy usage of tools. This clearly shows that using a learning system instead of a parametric model is highly preferable since the parametric model may fail to give good estimates because it is based on restricted data in this domain.

As a future direction, the learning system might be calibrated or different methods can be incorporated to test the system further. Another future work can be developing the dataset such that it reflects the software development characteristics of the software development organizations in Turkey and come up

with a better model to fulfill the needs of the industry in this context. This is necessary, because there is an obvious lack of work and knowledge in industry about this particular problem.

7. Acknowledgements

This research is supported in part by Bogazici University research fund under grant number BAP-06HA104.

8. References

[1] Boehm B., Abts C., "Software *Development Cost Estimation Approaches – A Survey*", University of Southern California, 1998.

[2] Briand L. C., Emam K. E., Surmann D., Wieczorek I., Maxwell K. D., "*An Assessment and Comparison of Common Software Estimation Modeling Techniques*", International Software Engineering Research Network Technical Report, 1998.

[3] Heemstra, F. J.,"*Software Cost Estimation Models*", IEEE Software, 1990.

[4] Boehm, B. W., "*Software Engineering Economics*", Prentice Hall, 1981.

[5] Rubin, H. , "*ESTIMACS*", IEEE, 1983.

[6] University of Southern California, *USC COCOMO Reference Manual*, 1994.

[7] Shepperd, M., Schofield, M., *"Estimating Software Project Effort Using Analogies"*, IEEE Transactions on Software Engineering, Nov 1997.

[8] Cowderoy, A.J.C. and J.O Jenkins, *"Cost estimation by analogy as a good management practice"*, *Proceeding.Software Engineering*, 1988.

[9] Evren Ceylan, F. Onur Kutlubay, Ayşe B. Bener, "*Defect Identification Using Machine Learning Techniques*", in Proceedings of 32nd Euromicro Conference on Software Engineering and Advanced Applications, 2006

[10] Mitchell, T. M., *Machine Learning*, McGraw-Hill and MIT Press, 1997.

[11] Basili, V., McGarry, F., Pajerski, R., Zelkowitz, M., "Lessons learned from 25 years of process improvement : The rise and fall of the nasa software engineering laboratory", Proceedings of the 24th International Conference on Software Engineering (ICSE) 2002, Orlando, Florida, 2002.

[12] Colin Potts, *Software Engineering Research Revisited*, IEEE Software, September 1993. COCOMO II Cost Estimation Questionnaire, http://sunset.usc.edu/research/COCOMOII/Docs/cform22.pdf, 2006.

[13] Software Cost Estimation : Metrics and Models, *http://sern.ucalgary.ca/courses/seng/621/W98/johnsonk/cost.htm#Original%20COCOMO, 2006.*

[14] Fenton, N. E., Pfleeger, S.L., *Software Metrics: A Rigorous and Practical Approach,* International Thomson Computer Press, 1997.

[15] Conte, S. D., Dunsmore, H. E., Shen, V. Y., *"Software Engineering Metrics and Models"*, Benjamin-Cummings, Menlo Park CA, 1986.

[16] University of South California Software Engineering Research, *http://sunset.usc.edu/research/, 2006.*