International Journal of Computer Applications (0975 – 8887)
Volume 52– No.6, August 2012

Performance Evaluation of Regression Techniques for Effort Estimation

Parasana Sankara Rao
Reasearch Scholor,
Dept of CSE,
JNTU KAKINADA,

Kiran Kumar Reddi
Phd, Assistant Professor,
Department of Computer Science,
Krishna University,
Machilipatnam.

ABSTRACT

Software effort estimation assesses the quantity of work required to develop a software project. It is a well known fact that the software industry is unable to give proper an estimate of effort, time and development cost and this is described in reports in various reports including those from project management consultancy companies through case studies on failed projects, and surveys. In this paper, we propose to investigate the Mean Magnitude Relative Error (MMRE) and Median Magnitude Relative Error (MdMRE) using various techniques such as M5, Linear regression, SMO Polykernel and RBF kernel. The dataset COCOMO is used for the investigations.

General Terms

Performance Evaluation, Regression analysis, Algorithms.

Keywords

Effort estimation, Mean Magnitude Relative Error (MMRE) and Median Magnitude Relative Error (MdMRE), SMO Kernels.

1. INTRODUCTION

Software effort estimation assesses the quantity of work required to develop a software project. It is a well known fact that the software industry is unable to give proper an estimate of effort, time and development cost, and this is described in reports in various reports including those from project management consultancy companies through case studies on failed projects, and surveys. Estimation accuracy results reported are sometimes biased towards high inaccuracy, e.g., consultant's studies providing estimation advice, journalists' stories on failed projects or software houses which sell estimation tools. It is hard to have a balanced view on this industry's estimation performance without unbiased information from representative projects and organizations. Scientific surveys in journals and conferences might be sources for unbiased information [1].

In software engineering, the typical break up of effort distribution is shown in Figure 1 [2]. That conventional estimation techniques focus only on actual development instead of including the additional activities involved, like software testing further complicates achievement of accurate effort estimates. This is because when techniques to estimate development effort evolved, the notion of estimating test-engineering time was overlooked. Certain preconditions have to be met independent of the software size to be tested to design test cases to reach (guaranteed) code coverage level.

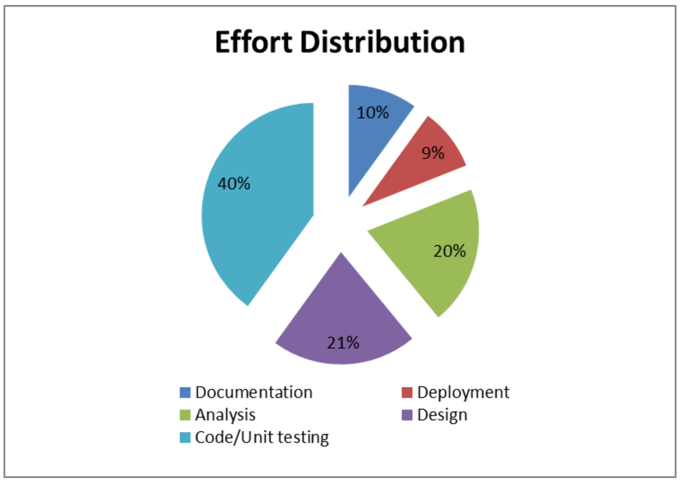


Figure 1: Effort Distribution in Software Projects.

The most common approaches for effort estimation are expert judgment, algorithmic models and analogy. Expert judgment is widely used for small projects, and sometime more than one expert's opinion is pooled for estimation. Algorithmic models such as COCOMO [3], SLIM [4], function points [5] are popular in the literature. Most of the algorithmic models are derived from:

$$effort = \alpha \ . \ size^{\beta}$$

Where α is a productivity coefficient and β refers to the economies of scale coefficient. The size is measured in estimated line of code (LOC). In analogy estimation, similar completed projects with known effort value are used for predicting the effort for the project.

A varied range of metrics was proposed for early software project effort estimation. Many authors suggested that standard sets had too many parameters, and suggested a reduction in the number of sets (large metric sets had high collection costs, and risked generation of over-fitted models). Reductions relied on linear methods for metrics elimination, and linear models to estimate metric sets size and effort. However, there is a risk that some dependencies might be non-linear. Researchers have started investigating alternative methods to develop predictive models, including fuzzy logic, evolutionary approaches, neural networks, and regression trees.

In this paper, it is proposed to investigate the Mean Magnitude Relative Error (MMRE) and Median Magnitude Relative Error (MdMRE) using various regression techniques including M5, Linear regression, SMO Polykernel and RBF kernel. COCOMO dataset is used for the investigations. The paper is organized as follows: section II deals with related work, section III details the materials and methods used. Section IV gives the results of the experiments and discussion of the same and section V concludes the paper.

2. RELATED WORKS

In a meaningful manner the main requirement to collect and analyze data is to design, manage and evaluate the software development process. To analyze software engineering data, the traditional methods for analysis are not always suitable. Hence, Lionel C. Briand et al., [7] illustrated a pattern recognition method to analyze the software engineering data, which is named as optimized set reduction (OSR). Most of the problems related to the usual methods are considered by the OSR. In order to use the method for prediction, risk management and quality evaluation, the techniques are discussed in this paper. The efficiency/effectiveness of the proposed OSR method for a specific application of software cost estimation is illustrated through the experimental results obtained. The proposed OSR is able tackle the problems giving simple interpretable patterns in management decisions/corrective actions that are done in the software development on the basis of empirical quantitative prototypes. A large variety of modeling problems and evaluating the other data sets is done efficiently by OSR.

Kjetil Moløkken et al., [8] reviewed surveys on software effort estimation and summarized estimation knowledge. The results concluded are: 1) There was 60-80% encounter effort and/or schedule overruns existing in most of the projects. But the overruns were observed to be lower than that reported by few consultancy companies. For instance, 30-40% was the average overruns observed in most of the surveys whereas the Standish Group's 'Chaos Report' reported an average cost overrun of 89% higher to the others. 2) The methods used to estimate are mostly based on expert judgment. The main cause for using expert judgment is that the basic estimation model has no confirmation to direct to an accurate estimate. The motive for effort and schedule overruns along with extensive analyses, is not described properly in the software estimation surveys, hence there is a lack of surveys.

To estimate the staff resources or effort necessary for a software project in advance is complicated. Recently, the major works did emphasis on the algorithmic cost models like COCOMO and Function Points. This causes the limitation to experience the necessity of calibrating the models with every individual measurement environment united with alterable accuracy levels subsequent to the calibration also. Hence another method for effort estimation using analogy is required. Martin Shepperd et al., [9] demonstrated this alternative method and revealed its performance in six different datasets to prove that it outperforms over traditional algorithmic techniques. The only limitation of this proposed method is that it requires more amount of computation. Martin Shepperd et al., also proposed ANGEL, an automated environment that maintains the collection, storage and recognition of the majority of analogous projects for the purpose to estimate the effort for a new project. On the basis of the minimization of Euclidean distance in *n* dimensional space is ANGEL. With differing datasets, in terms of both the number of observations (projects) and in the variables collected, this software is more flexible and is able to address it easily. Evaluated with six distinct datasets obtained from a variety of different environments, the performance of the proposed method is estimated and it outperforms other techniques. The estimation by analog is a candidate method that is also extremely practical method with the use of an automated environment.

Accurate software development effort estimates are not produced by the existing accessed algorithmic prototypes. Tridas Mukhopadhyay et al., [10] developed a case-based reasoning model, Estor, to deal this issue. Estor was modeled on the basis of verbal methods of a human expert, which resolved many problems in estimation. While comparing this method to the expert along with the function point and COCOMO estimations of the projects, the estimates of Estor and human experts estimate produced consistent and more accurate estimate than that of the function point and COCOMO protocols. The plausibility of case-based reasoning as a solution to solve a problem in this domain and the potential for increasing the accuracy of software cost estimates through this form of deliberation is demonstrated. Hence, the case-based reasoning method for software effort estimation is a promising and additional research provides better performance than the traditional methods.

Mohammad Azzeh et al., [11] proposed a novel basic EA prototype on the basis of integration of Fuzzy set theory with Grey Relational Analysis (GRA). To decrease the uncertainty in distance measure between two tuples at the k^{th} continuous feature ($\left|x_0(k)-x_i(k)\right|$) is performed by the Fuzzy set theory. A method solving problems, applied to measure the association between two tuples with *M* features is a GRA. The uncertainty in the similarity degree is maximized, as these characteristics are not necessarily to be continuous and also have ordinal and nominal scale type, aggregating various forms of similarity measures. Between two software projects for both continuous and categorical characteristics, the major application of GRA is to decrease uncertainty in the distance measure. While the relationship between effort and other effort drivers is complicated, both methods are appropriately used. When comparing with the results obtained using other well-known estimation models like Case-Based Reasoning, Artificial Neural Networks methods and Multiple Linear Regression, the experimental results attained from the integration of GRA with FL yielded better credible estimates.

3. MATERIALS AND METHODS

3.1 COCOMO Dataset

The COCOMO dataset [6] contains details of 63 software project. Each project is described by 16 cost derivers or effort multipliers. Of the 16 attributes, 15 are measured on the scale of six categories: *very low, low, nominal high, very high, and extra high*. The categories are represented by a numeric value. Kilo Delivered Source Instructions (KDSI) is the only numeric attribute. COCOMO dataset is generally used to evaluate the comparative accuracy of proposed new techniques. The effort histogram for COCOMO dataset is shown in Figure 2.

3.2 Regression analysis with MMRE (Mean Magnitude Relative Error) and MdMRE (Median Magnitude Relative Error)

To determine the accuracy of the software estimates and also for evaluating and validating the estimates, MMRE (Mean Magnitude Relative Error) and MdMMRE (Median Magnitude Relative Error) are used [12].

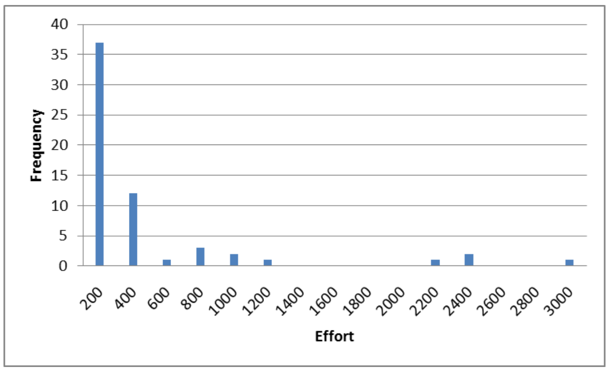


Figure 2: Effort histogram of COCOMO

Magnitude Relative Error (*MRE*) calculates the absolute percentage of error between actual and predicted effort for each reference project.

$$MRE_i = \frac{|actual_i - estimated_i|}{actual_i}$$

Mean Magnitude Relative Error (*MMRE*) computes the average of *MREs* over all reference projects. As the *MMRE* is susceptible to an individual outlying prediction, when a large number of observations is available, *MdMRE* is adopted. The median of *MREs* for the *n* projects is the *MdMRE* which is less sensitive to the extreme values of *MRE* is adopted. In spite of it the *MMRE* is widely used in estimation accuracy. *MMRE* has been criticized that it is unbalanced for many validation procedures and often leads to overestimation [12].

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE_i$$

$$MdMRE = \underset{i}{median}(MRE_i)$$

3.3 Linear Regression

Consider the problem of approximating the set of data [13],

$$D = \{(x^1, y^1), \ldots, (x^l, y^l)\}, \quad x \in \Re^n, y \in \Re$$

with a linear function,

$$f(x) = \langle w, x \rangle + b$$

the optimal regression function is given by the minimum of the functional,

$$\Phi(w, \xi) = \frac{1}{2}\|w\|^2 + C\sum_i (\xi_i^- + \xi_i^+)$$

where C is a pre-specified value, and ξ^-, ξ^+ are slack variables representing upper and lower constraints on the outputs of the system.

3.4 SMO Polykernel and RBF kernel

SVMs based methods are used widely for classification tasks [14]. For a given training data (x_i, y_i), $i = 1,..,n$, where $x_i \in \Re^d$ is a feature vector and $y_i \in \{+1, -1\}$ indicates the class value of x_i solve the following optimization problem:

$$\min_{w,b,\xi} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi_i$$

subject to

$$y_i\left(w^T \Phi(x_i) + b\right) \geq 1 - \xi_i \text{ for i=1...n}$$

$$\xi_i \geq 0$$

where $\Phi : \Re \to \mathrm{H}$, H being the high dimensional space $w \in \mathrm{H}$, and $b \in \Re$. $C \geq 0$ is a parameter which controls minimization of the margin errors and maximization of the margins. Φ is chosen so that an efficient kernel function K exists. In practice, Lagrange Multiplier methods are used to solve the above optimization problem. Sequential Minimal Optimization (SMO) [14] is a simple algorithm which is used for solving SVM QP problem. The advantage of SMO is its capability to solve the Lagrange multipliers without using numerical QP optimization. The following is the Lagrangian form:

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) - \sum_{i=1}^{n} \alpha_i$$

subject to

$$0 \leq \alpha_i \leq C \text{ for i=1...n}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

On solving the optimization problem, *w* is computed as follows:

$$w = \sum_{i=1}^{n} \alpha_i y_i \Phi(x_i)$$

x_i is a support vector if $\alpha_i \neq 0$. New instance x is computed by the following function:

$$f(x) = \sum_{i=1}^{n_S} \alpha_i y_i K(s_i, x) + b$$

Where s_i are support vectors and n_S is the number of vectors.

The polynomial kernel function is given by:

$$K(x_i, x_j) = \left(\gamma x_i^T x_j + r\right)^d, \text{ where } \gamma > 0$$

And the Radial basis function (RBF) kernel:

$$K(x_i, x_j) = \exp\left(-\gamma\|x_i - x_j\|^2\right), \text{ where } \gamma > 0$$

4. RESULTS AND DISCUSSIONS

The experimental setup consisted of using the attributes of the cocomo dataset as it is and feature transformation of the attributes using Principal Component Analysis (PCA). In the first experiment without data transformation Sequential Minimal Optimization (SMO) with RBF kernel produced the lowest relative error between the estimated and the actual value as seen in Table I. After attribute transformation using PCA, SMO with polykernel showed a decrease in MMRE by 17.6%. However with PCA linear regression technique did not decrease the MMRE. Figure 3, Figure 4 and Figure 5 show the instance wise MMRE for the three techniques under study.

Table 1: Average MMRE and MdMRE for various techniques

Technique Used	MMRE	MdMRE
M5 + PCA	6.875919	157.6677
Linear regression +PCA	5.21539	219.7753
SMO polykernel + PCA	4.833664	93.8034
SMO RBF kernel + PCA	2.662263	85.94772
M5	4.674381	185.6412
Linear regression	5.215322	219.7713
SMO polykernel	5.67898	91.95121
SMO RBF kernel	2.675412	84.55606

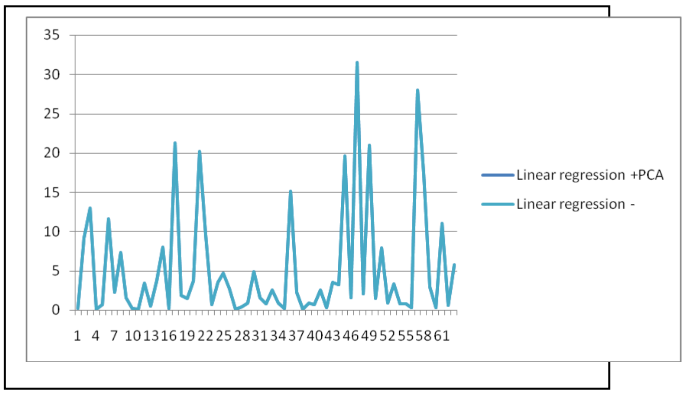
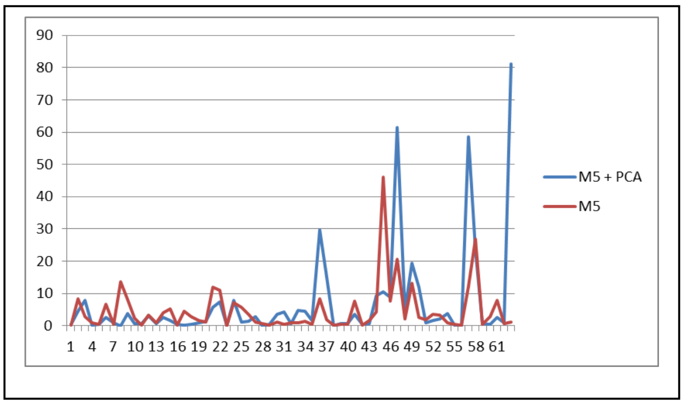


Figure 3: Comparison of MMRE for linear regression with and without PCA



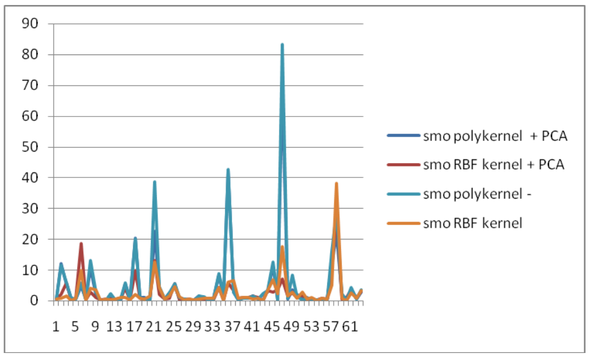


Figure 5: Comparison of MMRE SMO Polykernel and RBF Kernel with and without PCA

Figure 6 shows the comparative graph of the MMRE from six algorithms. Figure 7 shows the MdMRE and it can be seen that Sequential Minimal Optimization with PCA for feature transformation has the lowest error

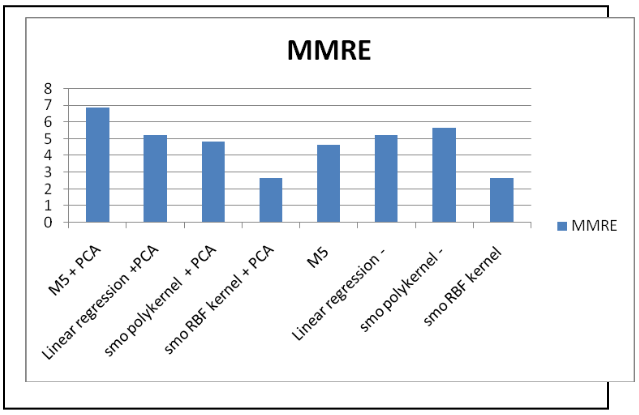


Figure 6: MMRE for different techniques used

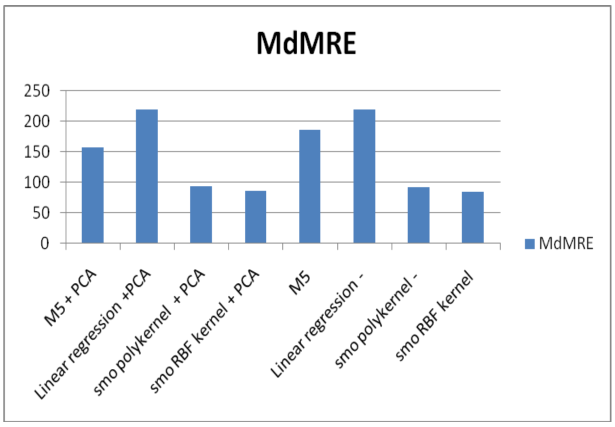


Figure 7: MdMRE for different techniques used

5. CONCLUSION

In this paper, it was proposed to investigate the performance of three regression algorithms namely Linear regression, M5 and modified Support Vector Machine (SVM) to avoid the quadratic problem. Two kernels were used in for the SVM with the first kernel being a polykernel and the second kernel using Radial Basis Function (RBF). From experimental results it is found that SVM with RBF kernel produces low MMRE. Future direction to lower the MMRE can be in the areas of soft computing.

6. REFERENCES

[1] M. Jørgensen, "A Review of Studies on Expert Estimation of Software Development Effort," J. Systems and Software, vol. 70, nos. 1-2, pp. 37-60, 2004.Suresh Nageswaran. Test effort estimation using use case points. Technology, (June), 2001

[2] Suresh Nageswaran. Test effort estimation using use case points. Technology, (June), 2001.

[3] Boehm, B. W.,Software Engineering Economics. Prentice-Hall: Englewood Cliffs, N. J., 1981.

[4] Putnam, L.H., "A General Empirical Solution to the Macro Software Sizing and Estimating Problem," IEEE Transactions on Software Engineering, vol. se-4, no. 4, pp. 345–361, July 1978.

[5] Albrecht, A.J. and J.R. Gaffney, 'Software function, source lines of code, and development effort prediction a software science validation', IEEE Trans. on Softi. Eng., 9(6), pp639-648, 1983.

[6] Boetticher G, Menzies T, Ostrand T (2007) PROMISE Repository of empirical software engineering data http://promisedata.org/ repository, West Virginia University, Department of Computer Science.

[7] L.C. Briand, V.R. Basili, and W.M. Thomas, "A Pattern Recognition Approach for Software Engineering Data Analysis," IEEE Trans. Software Eng., vol. 18, no. 11, pp. 931-942, 1992.

[8] K. Molokken and M. Joorgensen, "A Review of Software Surveys on Software Effort Estimation," Proc. Intl Symp. Empirical Software Eng., pp. 223-230, 2003.

[9] M. J. Shepperd, C. Schofield, and B. A. Kitchenham, "Effort Estimation Using Analogy," Proc. 18th Int'l Conf. Software Eng., Berlin: IEEE CS Press, 1996.

[10] T. Mukhopadhyay, S.S. Vicinanza, and M.J. Prietula, "Examining the Feasibility of a Case-Based Reasoning Model for Software Effort Estimation," MIS Quarterly, vol. 16, pp. 155-171, June, 1992.

[11] Azzeh, M., Neagu, D., Cowling, P., 2009. Fuzzy grey relational analysis for software effort estimation, Journal of Empirical software engineering.

[12] Mendes E, Mosley N, Counsell S (2003) A replicated assessment of the use of adaptation rules to improve Web cost estimation, *International Symposium on Empirical Software Engineering*, pp. 100-109.

[13] V. Vapnik, S. Golowich, and A. Smola. Support vector method for function approximation, regression estimation, and signal processing. In M. Mozer, M. Jordan, and T. Petsche, editors, Advances in Neural Information Processing Systems 9, pages 281–287, Cambridge, MA, 1997. MIT Press.

[14] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001.

[15] Platt, J. C.," Fast training of support vector machines using sequential minimal optimization". Advances in kernel methods: Support vector machines, B. Schokopf et al. (ed.), MIT Press, 1999.