Volume 3, Issue 3, March 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Prediction of Software Effort Using Artificial Neural Network and Support Vector Machine

Prabhakar,
M.E. (CSE) Student
Department of Computer Science
NITTTR, Chandigarh, India

Maitreyee Dutta
Associate Professor and Head of the Department
Department of Computer Science
NITTTR, Chandigarh, India

Abstract: ***Accurately estimating software effort is probably the biggest challenge facing software developers. Estimates done at the proposal stage has high degree of inaccuracy, where requirements for the scope are not defined to the lowest details, but as the project progresses and requirements are elaborated, accuracy and confidence on estimate increases. It is important to choose the right software effort estimation techniques for the prediction of software effort. Artificial Neural Network (ANN) and Support Vector Machine (SVM) have been used using China dataset for prediction of software effort in this work. The performance indices Sum-Square-Error (SSE), Mean-Square-Error (MSE), Root-Mean-Square-Error (RMSE), Mean-Magnitude-Relative-Error (MMRE), Relative-Absolute-Error (RAE), Relative-Root-Square-Error (RRSE), Mean-Absolute-Error (MAE), Correlation Coefficient (CC), and PRED (25) have been used to compare the results obtained from these two methods.***

Keywords: ***Software effort estimation, Machine Learning Techniques, Artificial Neural Network, and Support Vector Machine.***

1. Introduction

The Software effort estimation methods are mainly categorized in to algorithmic and non-algorithmic. The algorithmic methods are mainly COCOMO, Function Points and SLIM. Theses methods are also known as parametric methods because they predict software development effort using a formula of fixed form that is parameterized from historical data.The algorithmic methods require as input attributes such as experience of the development team, the required reliability of the software, the programming language in which the software is to be written, an estimate of the final number of delivered source line of code (SLOC), complexity and so on which are difficult to obtain during the early stage of a software development life cycle (SDLC). They have also difficulty in modelling the inherent complex relationships **[9]**.The limitations of algorithmic methods compel us to the exploitation of non-algorithmic methods which are soft computing based. These methods have advantage of

1. Ability to learn from previous data.
2. Able to model complex relationship between the dependent (effort) and independent variables.
3. Ability to generalize from the training dataset thus enabling it to produce acceptable result from previous unseen data.

2. Related Work

A lot of research has been done using machine learning techniques like Artificial Neural Networks, Decision Tree, Linear Regression, Support Vector Machine, Fuzzy Logic, Genetic Algorithm, Empirical Techniques, and Theory based techniques for predicting the software effort. The paper by **FINNIE** and **WITTIG [4]**, has examined the potential of two artificial intelligence approaches i.e. Artificial Neural Networks (ANNs) and Case Based Reasoning (CBR), for creating development effort estimation models using the dataset Australian Software Metrics Association (ASMA). Also, the potential of Artificial Neural Networks (ANNs) and Case Based Reasoning (CBR), for providing the basis for development effort estimation models in contrast to regression models is examined by the same author **[3]**. The authors concluded that Artificial Intelligence Models are capable of providing adequate estimation models. Their performance is to a large degree dependent on the data which they have trained, and the extent to which suitable project data is available will determine the extent to which adequate effort estimation models can be developed.

The paper proposed by **TOSUN, et.al. [1],** a novel method for assigning weights to features by taking their particular importance on cost in to consideration. Two weight assignment heuristics are implemented which are inspired by a widely used statistical technique called Principal Component Analysis (PCA).

The paper by **ELISH [6]**, empirically evaluates the potential and accuracy of MART as a novel software effort estimation model when compared with recently published models i.e. Radial Basis Function (RBF) neural networks, linear regression, and Support Vector regression models with linear and RBF kernels. The comparison is based on a well known NASA software project dataset.

The paper by **Martin, et. al., [10],** describes an enhanced Fuzzy Logic model for the estimation of software development effort and proposed a new approach by applying Fuzzy Logic for software effort estimates.

Genetic Algorithms (GA) are also widely used for accurate effort estimation. The paper by **BURGESS** and **LEFLEY [2],** evaluates the potential of Genetic Programming (GP) in software effort estimation and comparison is made with the Linear LSR, ANNs etc. The comparison is made on the Desharnais dataset of 81 software projects.

In **Ref. [13],** comparative research has been done by using three machine learning methods such as Artificial Neural Networks (ANNs), Case-Based Reasoning (CBR) and Rule Induction (RI) to build software effort prediction systems. The paper compares the software effort prediction systems in terms of three factors: accuracy, explanatory value and configurability.

The paper by **Bibi Stamatia, et. al., [14],** suggests the several estimation guidelines for the choice of a suitable machine learning techniques for software development effort estimation.

In **Ref. [9],** the author's presents that the one of the greatest challenges for software developers is predicting the development effort for a software system based on developer abilities, size, complexity and other metrics for the last decades. The ability to give a good estimation on software development efforts is required by the project managers.

The paper by **Parag C. Pendharkar [15],** propose a Probabilistic Neural Networks (PNN) approach for simultaneously estimating values of software development parameters (either software size or software effort) and probability that the actual value of the parameter will be less than its estimated value.

The paper by **L. Radlinki** and **W. Hoffmann [16],** analyses the accuracy of predictions for software development effort using various machine learning techniques. The main aim is to investigate the stability of these predictions by analyzing if particular techniques achieve a similar level of accuracy for different datasets. The results show that the accuracy of predictions for each technique varies depending on the dataset used.

3. Research Methodology

The Artificial Neural Network (ANN) and Support Vector Machine (SVM) learning techniques have been used for predicting the software effort using China dataset of software projects in order to compare the performance results obtained from these models.

(A) Empirical Data Collection

The data we have used is China Dataset. This data is obtained from PROMISE (PROMISE = PRedictOr Models In Software Engineering) Data Repository **[7]**. The mostly used software data sets for software Effort Predictions are China, Maxwell, NASA, Finnish, Telecom, Kemerer and Desharnais.

The China Dataset consists of 19 features, 18 independent variable and 1 dependent variables. It has 499 instances correspond to 499 projects. The descriptive statistics of China data set is appended at Table 1.

Table 1 China Data Set Statistics

S N	Variables	Min	Max	Mean	Standard Deviation
1	ID	1	499	250	144
2	AFP	9	17518	487	1059
3	Input	0	9404	167	486
4	Output	0	2455	114	221
5	Enquiry	0	952	62	105
6	File	0	2955	91	210
7	Interface	0	1572	24	85
8	Added	0	13580	360	830
9	Changed	0	5193	85	291
10	Deleted	0	2657	12	124
11	PDR_AFP	0.3	83.8	12	12
12	PDR_UFP	0.3	96.6	12	13
13	NPDR_AFP	0.4	101	13	14
14	NPDU_UFP	0.4	108	14	15
15	Resource	1	4	1	1
16	Dev. Type	0	0	0	0
17	Duration	1	84	9	7
18	N_effort	31	54620	4278	7071
19	**Effort**	**26**	**54620**	**3921**	**6481**

Set of independent variables decides the value of the dependent variable. The dependent variable is effort in this work. Some of the independent variables may be removed, if they are not much important to predict the effort, thus making the model much simpler and efficient. It has been observed from the China data set that independent variables ID and Dev. Type does not play any role in deciding the value of effort. Hence, independent variables ID and Dev. Type have been removed. The China data set was divided into two parts, i.e. training and testing set in a ratio of 4:1. Thus, 80% of the data was used for the purpose of training the model and remaining 20% was used for testing purpose.

MATLAB programs were developed for training and testing of various models with 16 independent variables- AFP, Input, Output, Enquiry, File, Interface, Added, Changed, Deleted, PDR_AFP, PDR_UFP, NPDR_AFP, NPDR_UFP, Resource, Duration, and N_effort for computation of dependent variable effort.

(B) Machine Learning Techniques

Machine Learning is considered as a subfield of Artificial Intelligence and it is concerned with the development of techniques and methods which enable the computer to learn. In simple terms development of algorithms which enable the machine to learn and perform tasks and activities. Over the period of time many techniques and methodologies were developed for machine learning tasks.

(1) Artificial Neural Networks (ANN)

A neural network is a simplified model of the biological neuron system. It is a massively parallel distributed processing system made up of highly interconnected neural computing elements that have ability to learn and thereby acquire knowledge and make it available for use.

McCulloch and Pitts (1943) proposed the first computational model of a neuron, namely the binary threshold unit, whose output was either 0 or 1 depending on whether its net input exceeded a given threshold. The most common algorithm for training or learning is known as error back-propagation algorithm. The error back propagation learning consists of two phases: a forward pass and a backward pass, an input is presented to the neural network, and its effect is propagated through the network layer by layer. This is also called Testing Phase. During the forward pass the weights of the network are all fixed. During the backward pass or "Training Phase", the weights are all updated and adjusted according to the error computed. Anerror is composed from the difference between the desired response and the system output. This error information is feedback to the system and adjusts the system parameters in a learning rule. The process is repeated until the performance is acceptable **[8]**.

(2) Support Vector Machine (SVM)

Support Vector Machine (SVM) is a concept in computer science for a set of related supervised learning methods that analyze data and recognize patterns, used for classification and regression analysis. The standard Support Vector Machine takes a set of input data and predicts, for each given input, which of two possible classes the input is a member of, which makes the support vector machine a non-probabilistic binary linear classifier.

Support Vector Machine (SVM) has been found to be successful when used for pattern classification problems. Applying the Support Vector approach to a particular practical problem involves resolving a number of questions based on the problem definition and the design involved with it.

Support Vector Machine (SVM) is a classification and regression prediction tool that uses machine learning theory to maximize predictive accuracy while automatically avoiding over-fit to the data. It is also being used for many applications, such as hand writing analysis, face analysis and so forth, especially for pattern classification and regression based applications.

(a) Kernel

If data is linear, a separating hyper plane may be used to divide the data. However it is often the case that the data is far from linear and the datasets are inseparable. To allow for this kernels are used to non-linearly map the input data to a high-dimensional space. The new mapping is then linearly separable.

(b) Feature Space

Transforming the data into feature space makes it possible to define a similarity measure on the basis of the dot product. If the feature space is chosen suitably, pattern recognition can be easy.

$$\langle x_1, x_2 \rangle \leftarrow K(x_1, x_2) = \langle \Phi(x_1).\Phi(x_2) \rangle$$

(2.1) Linear Kernel

The Linear kernel is the simplest kernel function. It is given by the inner product $x^T y$ plus an optional constant **c**. It is given by the following equation:

$$k(x, y) = x^T y + c$$

(2.2) ANOVA Kernel

The ANOVA kernel is a radial basis function kernel. It is said to perform well in multidimensional regression problems. It is given by the following equation:

$$k(x, y) = \sum_{k=1}^{n} \exp(-\sigma_{(} x^k - y^k)^2)^d$$

(C) Evaluating the Performance of the Models

The main measures used for evaluating the performance of machine learning techniques for predicting the software effort are as follows:-

1. Sum Squared Error (SSE)

The sum squared error is defined as

$$E = \left(\sum_{i=1}^{n} (Pi - Ai)^2 \right)$$

Where Pi = Predicted value for data point i;

Ai =Actual value for the data point i;

n = Total number of data points.

2. Mean Squared Error (MSE)

The mean squared error is defined as

$$\mathrm{E} = \left(\frac{1}{n} \sum_{i=1}^{n} (Pi - Ai)^2 \right)$$

Where Pi = Predicted value for data point i;

Ai =Actual value for the data point i;

n = Total number of data points.

3. Root Mean Squared Error (RMSE)

The root mean squared error is defined as

$$\mathrm{E} = \left(\sqrt{\frac{1}{n} \sum_{i=1}^{n} (Pi - Ai)^2} \right)$$

Where Pi = Predicted value for data point i;

Ai =Actual value for the data point i;

n = Total number of data points.

4. Mean Magnitude of Relative Error (MMRE) [3, 5]

$$\mathrm{MMRE} = \frac{1}{n} \sum_{i=1}^{n} \frac{\left| (Pi - Ai) \right|}{Ai}$$

Where Pi = Predicted value for data point i

Ai = Actual value for data point i

n = Total number of data points.

5. Relative Absolute Error (RAE)

The Relative absolute error is defined as the summation of the difference between predictive value and given value for the sample case j to that divide it by the summation of the difference between the given value and average of the given value.

The relative absolute error of individual data set j is defined as

$$\mathrm{Ej} = \frac{\sum_{i=1}^{n} \left| Pij - Ai \right|}{\sum_{i=1}^{n} \left| Ai - Am \right|}$$

Where Pij = Predicted value by the individual data set j for data point i.

Ai = Actual value for data point;

n = Total number of data points;

Am = Mean of all Ai

6. Root Relative Squared Error (RRSE)

The root relative squared error of individual data set j is defined as

$$\mathrm{Root\ Relative\ Squared\ Error} = \left(\sqrt{\frac{\sum_{i=1}^{n} (Pij - Ai)^2}{\sum_{i=1}^{n} (Ai - Am)^2}} \right)$$

Where Pij = Predicted value by the individual dataset j for data point in i;

Ai = Actual value for the data point i ;

n = Total number of data points;

Am =Mean of all Ai;

7. Mean Absolute Error (MAE)

The mean absolute error measures of how far the estimates are from actual values. It could be applied to any two pairs of numbers, where one set is "actual" and the other is an estimate prediction.

$$\mathrm{MAE} = \frac{1}{n} \sum_{i=1}^{n} \left| Pi - Ai \right|$$

Where Pi = Predicted value for data point i

Ai = Actual value for data point i

n = Total number of data points.

8. Correlation Coefficient

Correlation measures of the strength of the relationship between two variables. The strength of the relationship is indicated by the correlation coefficient. The larger the value of correlation coefficient, the stronger the relationship.

9. PRED (A)

It is calculated from the relative error. It is defined as the ratio of data points with error less than equal to A to the total number of data points. Thus, higher the value of PRED (A), the better it is considered.

$$\text{PRED (A)} = \frac{d}{n}$$

d = value of MRE where data points have less than or equal to A error.

4. Result Analysis

The Artificial Neural Network (ANN) and Support Vector Machine (SVM) machine learning techniques have been used for predicting the software efforts using China dataset. Nine performance indices have been used in order to compare the results obtained from these models. These indices are Sum-Square-Error (SSE), Mean-Square-Error (MSE), Root-Mean-Square-Error (RMSE), Mean-Magnitude-Relative-Error (MMRE), Relative-Absolute-Error (RAE), Relative-Root-Square-Error (RRSE), Mean-Absolute-Error (MAE), Correlation Coefficient (CC), and PRED(25). The model possessing the lower values of SSE, MSE, MMRE, RMSE, RAE, MAE, and RRSE and the higher values of correlation coefficient and PRED (25) is considered to be the best among others.

Table 2 Comparison of Performance indices with Artificial Neural Network and Support Vector Machine

S N	Performance Measures	Artificial Neural Network (ANN)		Support Vector Machine (SVM)	
		One Hidden Layer	Two Hidden Layer	Linear Kernel	ANOVA Kernel
1	**Sum Square Error (SSE)**	**0.04490**	0.06440	0.0183	**0.0187**
2	**Mean Square Error (MSE)**	**0.00045**	0.00064	0.0002	**0.0002**
3	**Root Mean Square Error (RMSE)**	**0.02120**	0.02540	0.0135	**0.0137**
4	**Mean Magnitude Relative Error (MMRE)**	**0.07630**	0.11120	0.2023	**0.1879**
5	**Relative Absolute Error (RAE)**	**0.05650**	0.08710	0.0843	**0.0842**
6	**Root Relative Squared Error (RRSE)**	**0.02180**	0.03120	0.0089	**0.0090**
7	**Mean Absolute Error (MAE)**	**0.00460**	0.00710	0.0069	**0.0069**
8	**Pred(25)**	**0.92000**	0.90000	0.7500	**0.7800**
9	**Correlation Coefficient**	**0.99350**	0.99370	0.9960	**0.9959**

MATLAB programs were developed for training and testing of various models and also for computation of performance indices. The results are tabulated in Table 2 and plotted in Figures 4.1-4.2. In these plots, the blue curve represents the curve for the actual value and red curve represents the curve for the predicted values. The more the closeness between the curves for actual and predicted output values, the lesser is the error and hence better is the model.

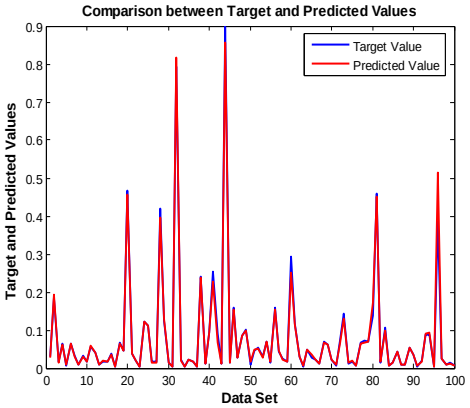

Figure 4.1(A): Target and Predicted values using Artificial Neural Networks with One hidden layers

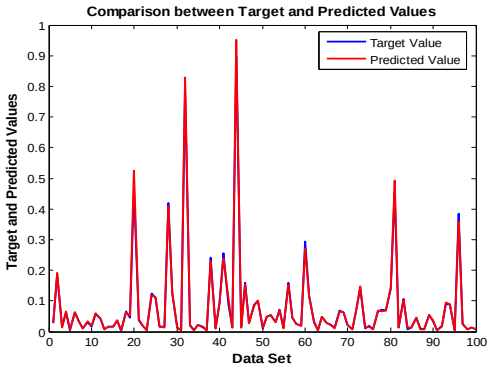

Figure 4.1(B): Target and Predicted values using Artificial Neural Networks with Two Hidden layers

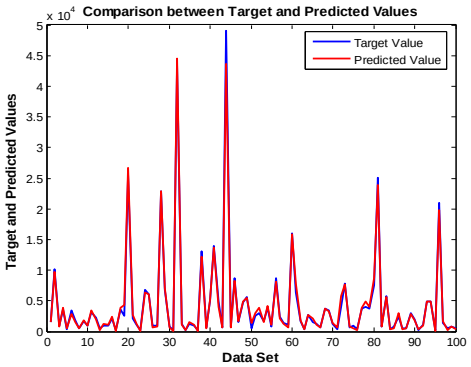

Figure 4.2 (A): Target and Predicted values using Support Vector Machine with Linear Kernel

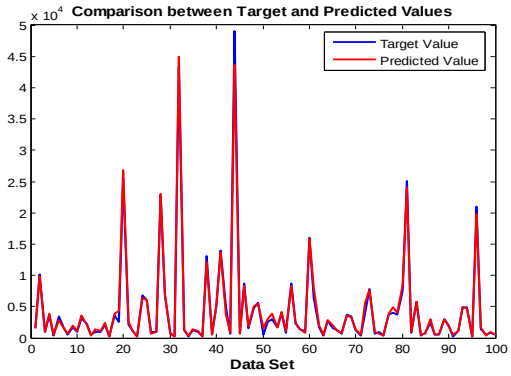

Figure 4.2(B): Target and Predicted values using Support Vector Machine with ANOVA Kernel

As shown in Table 2, the Artificial Neural Network with one hidden layer and Support Vector Machine with ANOVA kernel show the best results, and the former is better than the latter in accordance with most of the performance indices.

5. Conclusion

The Artificial Neural Network (ANN), and Support Vector Machine (SVM) learning techniques have been used to analyze the results using China dataset for predicting software development effort. A similar study can be carried out to predict software effort using prediction models based on other machine learning algorithms such as Genetic Algorithms (GA) and Random Forest (RF) techniques. Cost benefit analysis of models may be carried out to determine whether a given effort prediction model would be economically viable.

References:

[1] A.Tosun, B. Turhan and A.B. Bener, "Feature Weighting Heuristics for Analogy- based Effort Estimation Models," Expert Systems with Applications, vol. 36, pp.10325-10333, 2009.

[2] C.J. Burgess and M.Lefley, "Can Genetics Programming improves Software Effort Estimation? A Comparative Evaluation," Information and Software Technology, vol.43, pp.863-873, 2001.

[3] G. R. Finnie and G.E. Wittig, "A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case Based Reasoning and Regression Models," Journal of Systems and Software, vol.39, pp.281-289, 1997.

[4] G. R. Finnie and G.E. Wittig, "AI Tools for Software Development Effort Estimation," Proceedings of the International Conference on Software Engineering: Education and Practice (SEEP' 96).

[5] K. Srinivasan and D. Fisher, "Machine Learning Approaches to Estimating Software Development Effort," IEEE Transactions on Software Engineering, vol.21, Feb.1995.

[6] M. O. Elish, "Improved Estimation of Software Project Effort using Multiple Additive Regression Tree," Expert Systems with Applications, vol.36, pp. 10774-10778, 2009.

[7] *G*. Boetticher, T. Menzies and T. Ostrand , PROMISE Repository of Empirical Software Engineering data http://promisedata.org/repository, West Virginia University, Department of Computer Science, 2007.

[8] R. Malhotra, A. Jain, "Software Effort Prediction using Statistical and Machine Learning Methods," International Journal of Advanced Computer Science and Applications, vol.2, No.1, January 2011.

[9] I. Attarzadeh and Siew Hock Ow, "Software Development Effort Estimation Based on a New Fuzzy Logic Model," International Journal of Computer Theory and Engineering, Vol. 1, No. 4, pp.1793-8201, October 2009.

[10] C. L. Martin, J. L. Pasquier and Cornelio Y M and Agustin G. T., "Software Development Effort Estimation using Fuzzy Logic: A Case Study," Proceedings of the Sixth Mexican International Conference on Computer Science (ENC'05), IEEE Software, 2005.

[11] C. Mair, G.Kadoda, M. Lefley, K.P.C.Schofield, M. Shepperd and Steve Webster, "An Investigation of Machine Learning Based Prediction Systems," Empirical Software Engineering Research Group, Bournemouth University, U.K. 09 July, 1999.

[12] Bibi Stamatia and Stamelos Ioannis, "Selecting the Appropriate Machine Learning Techniques for Predicting of Software Development Costs," Artificial Intelligence Applications and Innovations, vol. 204, pp.533-540, 2006.

[13] Parag C. Pendharkar, "Probabilistic estimation of software size and effort," An International Journal of Expert Systems with Applications, vol. 37, pp.4435-4440, 2010.

[14] L. Radlinki and W. Hoffmann, "On Predicting Software Development Effort Using Machine Learning Techniques and Local Data," International Journal of Software Engineering and Computing, vol. 2, pp.123-136, 2010.

