ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 4, Issue 9, March 2015

The Effect of Dimensionality Reduction on the Performance of Software Cost Estimation Models

Riyadh A.K. Mehdi
College of Information Technology
Ajman University of Science & Technology

Abstract- Estimating software efforts is very important to a project manager as the cost and schedule for completing the software can be determined. Building these models requires developing a sound computational model as well as identifying the appropriate set of project features that accurately predict the software effort. Many data sets have been experimented with in building these models. These data sets differ in the features set used. One problem which has been cited with these data sets is that they contain significantly correlated or statistically not significant predictor variables. Choosing the right set of features can have a strong impact on the overall predictive power of the model. This paper investigates the effectiveness of using principle component analysis as a mechanism to eliminate correlated or not significant parameters on the accuracy of estimating software effort. The set of experiments conducted in this work on software cost estimation do not provide conclusive evidence that dimensionality reduction of the publically available data sets. The results obtained also indicate that the accuracy of the models are directly related to the attributes that constitute the data set.Another finding of this work is that the China dataset provided the most accurate predictions using a radial basis function neural network based software effort estimation model.

***Index Terms*-Software effort estimation, principle component analysis, datasets, neural networks, and radial basis functions.**

I. INTRODUCTION

Estimating software development efforts is one of the critical tasks in managing software development projects. Predicting software development effort with high accuracy is of paramount importance for project managers. However, estimating software development effort is still a challenging problem and one that attracts considerable research. There are two requirements for building an accurate software effort estimation model. First, a robust computational model must be developed, and secondly a relevant set of software project features set must be identified in addition to the quality of data being collected[1]. With regard to the first requirement numerous software effort estimation models have been developed [2, 3, 4, 5]. The conventional models use a mathematical formula to predict project cost based on the estimates of parameters such as project size measured in lines of source code or function points, number of software engineers, and other process and product attributes [6]. Among the software cost estimation techniques, COCOMO (**Co**nstructive **Co**st **Mo**del) is the most commonly used algorithmic cost modeling technique because of its simplicity for estimating the effort in person-months for a project at different stages. COCOMO uses a mathematical formula to predict project cost estimation [7].Non algorithmic models of cost estimation such as artificial neural networks (ANN) have also been used. Artificial neural networks are good at modeling complex nonlinear relationships. They are massive parallel distributed processor made up of simple processing units, which are capable of storing experimental knowledge and making it available for use[6]. An ANN resembles the brain in two respects [6]: 1) Knowledge is acquired from its environment through a learning process,2) Interneuron connection weights are used to store the knowledge.

As far as the second requirement of building a sound software effort estimation model, many datasets have been used for this purpose. These data sets differ in the parameters and attributes they use in describing the software project features that may be used in building a cost estimation model. One problem which has been cited with these data sets is that they contain significantly correlated or statistically not significant predictor variables [8]. Best estimation models are those in which the predictor variables each correlate highly with the dependent variable but correlate at most only minimally with each other, otherwise the redundancy will result in overfitting of the prediction model [9]. In statistics and machine learning, overfitting generally occurs when a model is excessively complex, such as having too many parameters relative to the number of observations. A model that has been overfit begins to "memorize" training data rather than "learning" to generalize from trend [10].Thus, choosing the right set of features can have a strong impact on the overall predictive power of a software effort estimation model.

This paper investigates the effectiveness of using principle component analysis to eliminate correlated or not significantparameters on the accuracy of estimating software effort using a radial basis function neural network.

II. RESEARCH METHODOLOGY

Neural networks have been shown to outperform other approached in predicting software effort [1, 2, and 12]. The most commonly used neural network architecture is the back propagation trained multilayered feed forward networks with sigmoidal activation function [11]. Each neuron in a MLP takes the weighted some of its input values. That is, each input value is multiplied by a coefficient, and the results are all summed together. A single MLP neuron is a simple linear classifier, but

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 4, Issue 9, March 2015

complex non-linear classifiers can be built by combining these neurons into a network.

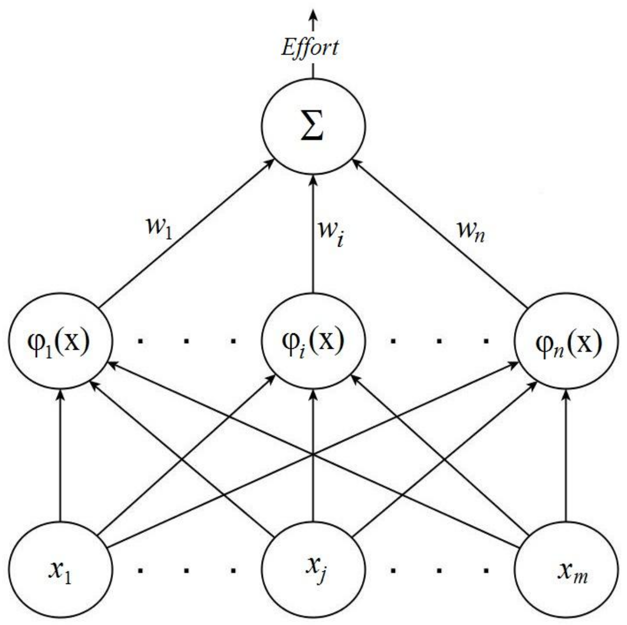

Fig 1. Radial Basis Function Neural Network

Another type of ANN that has been used in the literature is the radial basis functions neural network (RBFNN) [12]. A RBFNN can approximate any function which makes it suitable to model the relationship between inputs (the various cost drivers) and output (effort required). RBFNN performs classification by measuring the input's similarity to examples from the training set. Each RBFNN neuron stores a "prototype", which is just one of the examples from the training set. To classify a new input, each neuron computes the Euclidean distance between the input and its prototype. In other words, if the input more closely resembles class A prototypes than the class B prototypes, it is classified as class A.It has been show that RBFNN perform better than other types of neural networks based models [13]. In this paper we will use RBFNN to examine the effect of preprocessing datasets with PCA on the accuracy of software effort estimation models.

RBFNN Neural Network Implementation

The following generic description of a RBF neural network is based on a tutorial given in [12].Figure 1 describes a typical architecture of an RBFNN. It consists of an input vector, a layer of RBF neurons, and an output layer with one node per category or class of data. The input vector is the *m*-dimensional vector to be classified. The hidden layer consists of neurons where each one stores a "prototype" vector which is just one of the vectors from the training set. Each RBFNN neuron compares the input vector to its prototype, and outputs a value between 0 and 1 which is a measure of similarity. If the input is equal to the prototype, then the output of that RBFNN neuron will be 1. As the distance between the input and prototype grows, the response falls off exponentially towards zero.

The neuron's response value is also called its "activation" value. The prototype vector is also often called the neuron's "center", since it's the value at the center of the bell curve. The output of the network consists of a set of nodes, one per category to be classified or a value to be computed. Each output node computes a sort of score for the associated category. Typically, a classification decision is made by assigning the input to the category with the highest score. The score is computed by taking a weighted sum of the activation values from every RBF neuron. Because each output node is computing the score for a different category, every output node has its own set of weights. The output node will typically give a positive weight to the RBF neurons that belong to its category, and a negative weight to the others.

Each RBFNN neuron computes a measure of the similarity between the input and its prototype vector (taken from the training set). Input vectors which are more similar to the prototype return a result closer to 1. There are different possible choices of similarity functions, but the most popular is based on the Gaussian function. Equation 1 describe the formula for a Gaussian with a one-dimensional input.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad \text{---------} \ (1)$$

Where x is the input, μ is the mean, and $\sigma$is the standard deviation. This produces the familiar bell curve shown below in Figure 2, which is centered at the mean,μ.
In the Gaussian distribution, μ refers to the mean of the distribution. Here, it is the prototype vector which is at the center of the bell curve.

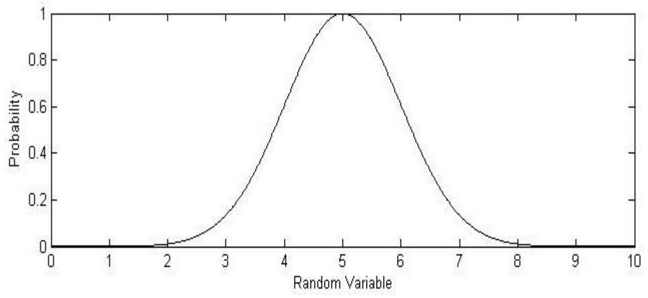

Fig 2. A Gaussian function with μ=5, and σ=1.

For the activation function, $\varphi(\ )$, the standard deviation, σ, is not important and the following two simplifying modifications can be made.The first modification is to remove the outer coefficient, 1 / (σ * sqrt (2 * π)). This term normally controls the height of the Gaussian curve. Here, though, it is redundant with the weights applied by the output nodes. During training, the output nodes will *learn* the correct coefficient or "weight" to apply to the neuron's response.The second change is to replace the inner coefficient, 1 / (2 * σ^2), with a single parameter β which controls the width of the bell curve. Again, in this context, the value of σ is not in itself important and what is

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 4, Issue 9, March 2015

needed is some coefficient that controls the width of the bell curve. Thus, after making these two modifications, the RBFNN neuron activation function can be written as in equation 2 [12]:

$$\varphi(x) = e^{-\beta\|x-\mu\|^2} \quad \text{---------} \quad (2)$$

There is also a slight change in notation here when we apply the equation to n-dimensional vectors. The double bar notation in the activation equation indicates that we are taking the Euclidean distance between x and mu, and squaring the result. For the 1-dimensional Gaussian, this simplifies to just $(x – \mu)^2$.

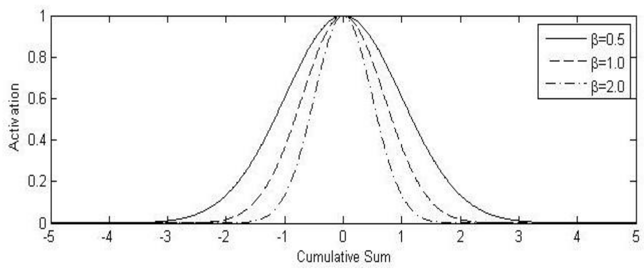

Fig 3. Activation Function for different values of beta.

It's important to note that the underlying metric here for evaluating the similarity between an input vector and a prototype is the Euclidean distance between the two vectors. Also, each RBF neuron will produce its largest response when the input is equal to the prototype vector. This allows to take it as a measure of similarity, and sum the results from all of the RBFNN neurons. As we move out from the prototype vector, the response falls off exponentially. Every RBF neuron in the network will have some influence over the classification decision. The exponential fall off of the activation function, however, means that the neurons whose prototypes are far from the input vector will actually contribute very little to the result [12].

Principle Component Analysis

Principal components analysis (PCA) is normally used for two objectives: reducing the number of features used in a dataset while retaining the variability in the data, and identifying hidden patterns in the data, and classifying them according to how much of the information, stored in the data, they account for [14]. In this work PCA has been used to eliminate correlated attributes and examine the effect on estimation model accuracy.

Software Effort Estimation Datasets

The following datasets are used in this research:

- Desharnais (11 attribute, 77 project, effort in person-hours)
- Cocomo81 (18 attribute, 61 project, effort in person-month)
- China (19 attribute, 499 projects, effort in person-hours)
- Maxwell (27 attribute, 62 project, effort in function points)
- Albrecht (8 attribute, 24 project, effort in person-hours)

Attributes that represent identification numbers had been removed from the datasets in this work. For more details on these datasets, see [15].

The evaluation criterion used to assess the prediction accuracy obtained from the different datasets is the mean magnitude error (MME) and the mean magnitude relative error (MMRE) computed from the formulae 3 and 4:

$$MME = \frac{1}{n}\sum_{i=1}^{i=n}\left|actualEffort(p_i) - predictedEffort(p_i)\right| \quad \text{---}(3)$$

$$MMRE = \frac{1}{n}\sum_{i=1}^{i=n}\frac{\left|actualEffort(p_i) - predictedEffort(p_i)\right|}{actualEffort(p_i)} \quad \text{---}(4)$$

The measure prediction level PRED(λ) is also used as complementary criterion to count the percentage of projects that have MMRE of λor less.A value 0.25 for λis commonly used to compare software cost estimation models[16].

III. RESULTS AND TABLES

MATLAB neural network toolbox was used to implement the RBFNN using *newrb* function:

Net=newrb(*P*, *T*, *goal*, *spread*, *MN*,*DF*), where

- *P* is a*m*×*n* matrix of input vector, *m* is the number of cost drivers for each project, and*n* is the number of projects used in the training phase.
- *T* is 1×*n* vector of actual efforts for each project used in the training phase.
- *Goal: mean squared errors*,0.0001 is used.
- *Spread*: A value between 1.0 and 3.0 is used.
- *MN*: maximum number of neurons (default *n*)
- *DF*: number of neurons to add between displays, 2 is used.

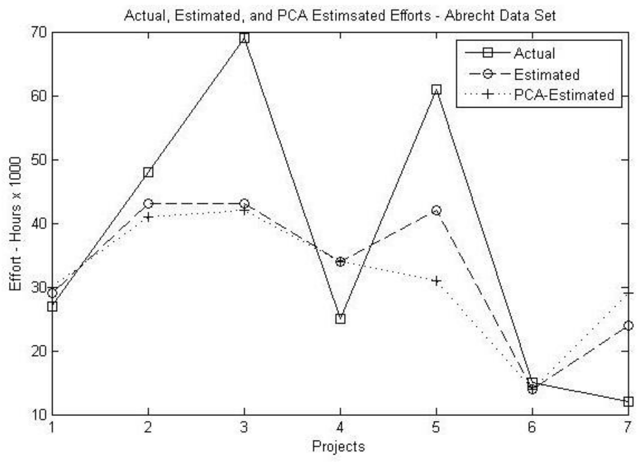

Fig 4. Actual, estimated, and PCA estimated efforts for the Albrecht data set.

The MATLAB function *processpca*( ) has been used to preprocess datasets using PCA. RBFNN was applied to the datasets listed above with and without applying principle component analysis as a mechanism to reduce the dimensionality of data and reduce correlation. Figures 4 to 8 shows the predictions obtained with and without applying PCA and the actual effort for the Albrecht, Desharnais, Cocomo81, Maxwell, and China datasets respectively.

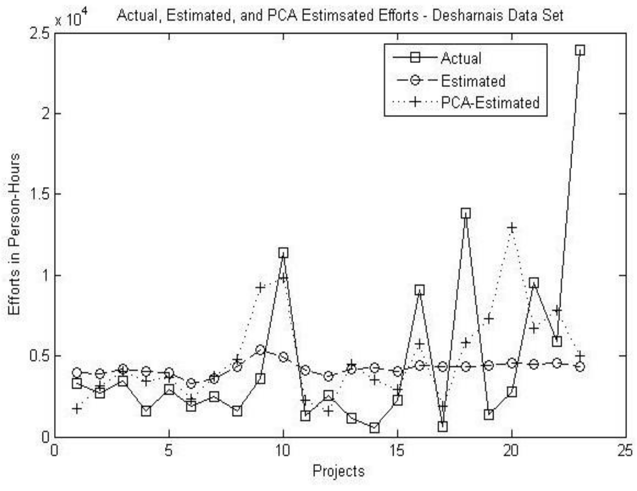

Fig 5. Actual, estimated, and PCA estimated efforts for the Desharnais data set.

Looking into the results, it is clear that there is no conclusive evidence that applying PCA has a positive impact on the prediction accuracy of neural networks based software efforts estimation models. Figures 4 to 8 also indicate that prediction accuracy is dependent to a large extent on the dataset being used and consequently the set of features used to describe software projects.

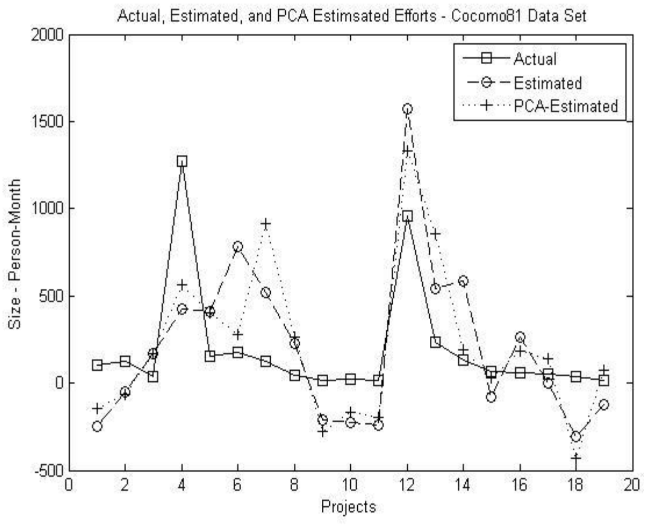

Fig 6. Actual, estimated, and PCA estimated efforts for the Cocomo81 data set.

The computed mean absolute error, mean relative absolute error, and PRED (0.25) are shown in Table 1. It is evident from the prediction results and the statistics of Table 1 that the China dataset provided the most accurate predications compared to the other data sets. This accuracy may also be attributed to the large number of projects (499 project) contained in the data set.

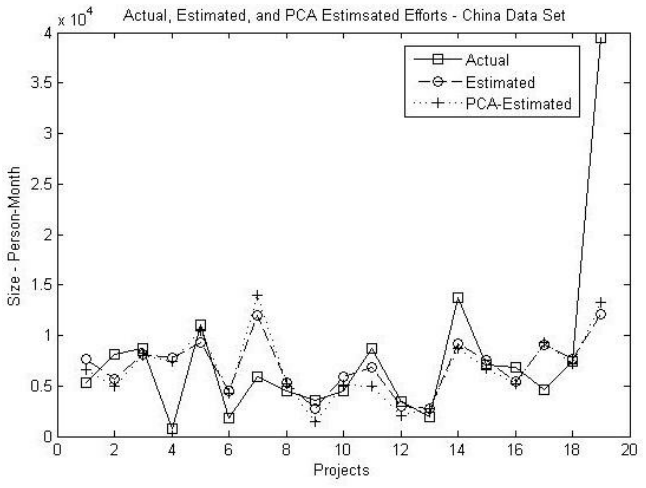

Fig 7. Actual, estimated, and PCA estimated efforts for the Maxwell data set.

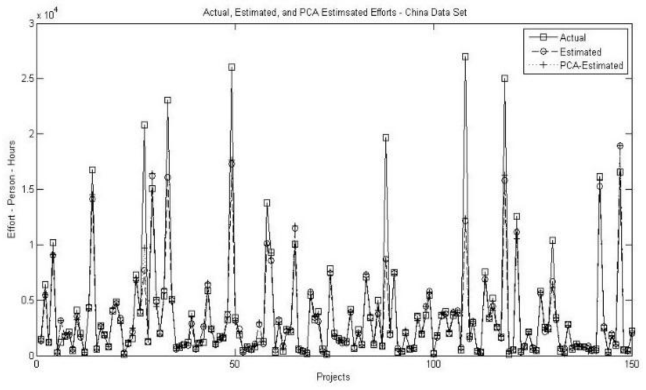

Fig 8. Actual, estimated, and PCA estimated efforts for the China data set.

Data Set	MME		MMRE		PRED(0.25)	
	No PCA	PCA	No PCA	PCA	No PCA	PCA
Albrecht	10842	13384	33.5%	41.9%	42.8%	42.8%
Desharnais	2580	2491	101%	101%	30.4%	39.1%
Cocomo81	313.2	272.3	501%	466%	0	0
Maxwell	3551	3687	85%	86%	47.4%	36.8%
China	1274	1197	21.5%	18.5%	80%	83%

Table 1. Performance statistics of datasets used.

IV. CONCLUSIONS

The conclusions to be drawn from this the set of experiments conducted in this work on software cost estimation do not provide a conclusive evidence that dimensionality reduction of the publically available data set used in building software cost estimation models can help with the accuracy of these models. The results obtained also indicate that the accuracy of the models are

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 4, Issue 9, March 2015

directly related to the attributes chosen by the data set. According to the work conducted in this research, the China data set provided the most accurate results from among the data sets used in this work. Future work will investigate the importance of the various attributes in explaining the size of efforts required so that standardized set of attributes can be used in collecting data sets.

REFERENCES

[1]. B. Kitchenham, and E. Mendes, "Why comparative effort prediction studies may be invalid," 5th International conference on software predictors PROMISE'09, Vancouver, Canada, ACM International Conference Proceedings, Series archive, Article number 4.

[2]. S. A. Abbas, A. R. Liao, A. Azam, "Cost Estimation: A Survey of Well-Known Historic Cost Estimation Techniques,"Journal of Emerging Trends in Computing and Information Sciences, vol. 4, no. 1, pp 612-636, 2012.

[3]. 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, 2011.

[4]. Abbas Heiat, "Comparison of Artificial Neural Network and Regression Models for Estimating Software Development Effort,"Information and Software Technology, vol. 44, pp. 911-922, 2002.

[5]. R. Sharma, "Survey: Non Algorithmic Models for Estimating Effort,"European International Journal of Science and Technology, vol. 2, no. 3, 2013.

[6]. Kaushik, A. Chauhan, D. Mittal, and S. Gupta, "COCOMO Estimates Using Neural Networks," International Journal of Intelligent Systems and Applications, vol. 9, pp. 22-28, 2012.

[7]. Reddy, and K. Raju, "An Optimal Neural Network Model for Software Effort Estimation," International Journal of Software Engineering, vol.12 no.1, pp. 66-78.

[8]. P. Reddy, K. R. Sudha, and C. Ramesh, "Software Effort Estimation using Radial Basis and Generalized Regression Neural Networks,"Journal of Computing, vol. 2, no. 5, 2010.

[9]. B. S. Everitt, Cambridge Dictionary of Statistics, 2nd ed., Cambridge University Press, 2002.

[10]. V. Tetko, D. J. Livingstone, A. I. Luik, "Neural network studies. 1. Comparison of Over fitting and Overtraining, "Journal of Chemical Information and Computer Sciences, vol 35, no. 5, pp. 826–833, 1995.

[11]. C. Reddy, and K. Raju, "A Concise Neural Network Model for Estimating Software Effort," International Journal of Recent Trends in Engineering, vol. 1, no. 1, pp. 188-193, 2009.

[12]. Liu, Radial Basis Function (RBF) Neural Network Control for Mechanical Systems. Springer, 2013.

[13]. E. Praynlin, and P. Latha, "Performance Analysis of Software Effort Estimation Models Using Radial Basis Function Network,"International Journal of Computer, Information, Systems and Control Engineering, vol. 8, no. 1, 2014.

[14]. T. W. Anderson, Principal Components: An Introduction to Multivariate Statistical Analysis, 3rd Edition, pp. 451-460, Wiley, 2003.

[15]. S. J. Shirabad, and T. J. Menzies, "The PROMISE Repository of Software Engineering Databases," School of Information Technology and Engineering, University of Ottawa, Canada. Available: http://promise.site.uottawa.ca/SERepository, 2005.

[16]. S. Conte, H. Dunsmore, and V. Shen, Software Engineering, Metrics and Models. Benjamin/Cummings, 1986.