SOFTWARE DEVELOPMENT EFFORT AND PREDICTION TECHNIQUES

Ramu Vankudoth [#1] **Dr.P.Shirisha** [#2]
[#1]Research Scholar (Ph.D),Department of Computer Science, ramuvankudoth86@gmail.com
[#2]Assistant Professor, Department of CSE, KITS, siri.niru55@gmail.com
[#1,2]Kakatiya University, Warangal

Abstract – To develop reliable software in available time and budget, software managers need to keep on a software quality and reliability and dynamically manage their resources. The resource management is highly dependent on quality and reliability assessment of software under development besides understanding the quality and cost of resources available. The later part is generally understood qualitatively and in general difficult to present in good quantitative terms. However, the quality and reliability measurement is challenging task for managers when product is under development. Indicative measures of software reliability and quality help to predict resource requirements for meeting the specified performance requirements. Software development efforts prediction is one of the important activities for the above purpose. Inaccurate prediction of efforts may lead to cost overruns or poor reliability of software or both. Overestimation of efforts leads to wastage of software development resources and underestimation of efforts causes schedule delays, poor quality of software and associated penalties.

In this work, software development efforts are predicted using some effort multipliers available during early phases of software development lifecycle. Evolutionary techniques (PCA and CFS) and some soft computing techniques (artificial neural network) are used to predict software development effort form effort multipliers.

Keyword: Software quality, ANN, PCA, CFS, Reliability, MATLAB

I. INTRODUCTION

To develop reliable software in time budget and available, software managers need to keep a on dynamically manage their resources and reliability and software quality. The resource management is highly dependent on reliability and quality assessment of software under development besides understanding the quality and cost of resources available. The later part is in general difficult to present in good quantitative terms and generally understood qualitatively. However, the reliability and quality measurement is challenging task for managers when product is under development. Indicative measures of software quality and reliability help to predict resource requirements for meeting the specified performance requirements. Software one of the important activity is development efforts prediction for the above purpose. Inaccurate prediction of poor reliability of software or efforts may lead to cost overruns or both. Overestimation of underestimation of efforts causes schedule delays and efforts leads to wastage of software development resources, associated penalties and poor quality of software. During effort and cost drivers, a number of metrics such as size software development process are independent variables and effort/cost is dependent variable. The data from similar historical driven models use data for training software projects. The been used for predicting the cumulative number of failures in certain time, trained model predicts development efforts for new projects ANN time between failures and fault-prone. It has been found predictive capability that ANNs have better most conventional methods .In the results of the ANN model trained with , present communication are reported. Principal component analysis (PCA) reduces number of neurons in hidden layer the collinearity in input features is employed to optimize. The remainder of the paper is organised better reduces by using MATLAB as follows. The ANN strategy is explained in Section. The experimental results are discussed in Section. Finally, in Section, the concluding remarks are stated.

II. LITERATURE SURVEY

Software development efforts prediction is necessary for control and effective project planning. Project managers use especially for determination of project details and, allocation of project resources, project tasks, schedule controlling and effort estimation to make better managerial decisions during project development life cycle process monitoring. There exists a number of software efforts estimation for models. ANN is software development efforts prediction for a sought after method. Compared the prediction performance concluded that ANN-based models provided improved performance over regression analysis. Reported of regression analysis procedure to predict efforts from software size and other cost and effort multipliers ANN and the predictive measures of ANN are superior to regression analysis for software efforts estimation presented a study. They compared models and performance is found that better prediction performance than. They concluded that performance of ANN results were found better than regression analysis and ANN model varied depending upon input variables .They concluded that the existing models to predict efforts during early phases of software development life cycle some presented approach had better performance than. There exist a number of regression analysis models to estimate efforts as a function of software size and research works on software efforts estimation using ANN. Used all effort multipliers and software cost along with software size to estimate efforts. They using analysis of variances of pre-processed. The stepwise multiple regression analysis models to estimate efforts and preprocessed data were taken as input of ANN. They showed that improved performance of preprocessing technique models. ANN-based models

works are used many used to train the ANN. In this work, ANN models are trained using PCA and CFS.

III. PROPOSED METHODOLOGY:

Two different models such as ANN-PCA and ANN-CFS are proposed in this paper to predict software development efforts. A new ANN architecture is proposed in this paper which contains an input encoding layer with additional. This additional input encoding layer is added between the input hidden layers and traditional. This additional input layer contains neurons with used for scaling the input values of ANN single input and single output. This new ANN architecture is trained using PCA technique. In the ANN-PCA (size) model, only size is considered as input in the ANN-CFS (all effort multipliers) model, size including all effort variables are taken as input of ANN to predict development efforts. In the ANN-PCA and ANN-CFS model, PCA and CFS applied for after applying of result data we take has input feature space to reduce dimension of input space and then the reduced data are taken as input of ANN to predict development efforts. Finally, in the ANN-PCA model, PCA is applied to reduce dimension of input space and then is applied to optimize ANN architecture of ANN using the reduced input .The proposed additional input encoding layer ANN architecture. The proposed ANN-PCA model is proposed ANN-CFS-based model with PCA (ANN-PCA) .The proposed ANN architecture optimization (ANN-PCA).

ANN is used to predict software development efforts based on different effort multipliers such as reliability, complexity, software size, analyst capability, programmer capability and many more. These development efforts sand effort multipliers are considered as the independent and dependent variable respectively. The organization of neurons in ANN is called the topology of ANN. The non-linear relationship between output and input of ANN is obtained by training ANN. Feed forward network is the simplest type of ANN used for prediction and classification problems. An additional input encoding layer ANN architecture is proposed in this work. It consists of an additional input encoding layer, a hidden layer and an input layer, an output layer. The different variables (effort multipliers and size) present in the software project such as taken as input of ANN. Effort is considered as output of ANN.

Artificial neural network (ANN) Architecture:

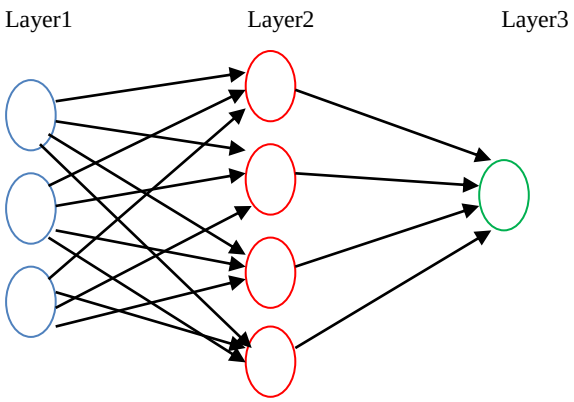


Fig:1 ANN architecture

The proposed ANN architecture consists of an additional input encoding layer which is added between hidden layer and input layer. The number of input layer neurons is equal to the number of input features used for predicting efforts. The additional input encoding layer has the same number of neurons as input layer. Some of the effort multipliers are of smaller value and some are of higher value. The learning process of ANN becomes complex using the original input variables. Scaling function is used to capture the of software failure process in some models for better ANN training. The ANN inputs are scaled into the range [0, 1] in different models for better learning in ANN .In the proposed ANN architecture, the same logarithmic function used in is used as activation function in the additional input encoding layer. The authors have used this activation function and the scaling parameter is obtained automatically during ANN training to scale inputs in the range [0, 1]. The multipliers used for datasets are of variable ranges. In this work, a logarithmic scaling function is used to equalize the importance of all the effort multipliers used as input of ANN in the proposed ANN architecture. The following logarithmic function is used as activation function in the proposed additional input encoding layer to scale ANN inputs in the range [0, 1].The search space biases are considered as [0, 1]. The minimum scaled value is obtained automatically in this search space in the proposed approach.

A. ANN-PCA Model:

PCA is used as the dimension reduction technique to reduce the dimension of input features for the proposed approaches. PCA involves a mathematical procedure that transforms a number of correlated variables into a (smaller) number of uncorrelated variables called principal components (PCs). It may happen that the transformed data may contain negative values which are not handled by the logarithmic function during scaling. After applying PCA on each dataset, the original dataset is transformed into reduced dataset by considering bias which helps to scale the input of ANN properly using logarithmic function. The Final Data set obtained after applying PCA may consists of negative values for some points. To eliminate negative values, Bias is calculated along each dimension of Final Data set. Bias is considered as minimum negative value along each dimension and subtracted along each dimension of the Final Data set. Then, the resultant Final Data set is divided into learning and testing sets. Then ANN is trained using PSO as discussed. The ANN architecture considered consists of an input layer, an additional input layer, a hidden neuron and an output layer. The number of input neurons in the input layer is same as the number of PCs selected after applying PCA discussed above. The additional input layer also contains the same nu Layer2 (Hidden Layer) neurons. The output layer contains a single neuron and uses linear activation function. Bias is added to every neuron in the additional input layer, hidden layer and output layer. The ANN is optimized by using PSO method.

B. Correlation Feature Selection:

Feature selection is a pre-processing step to machine learning which is effective in reducing dimensionality, removing irrelevant data, increasing learning accuracy, and improving result comprehensibility. It is a measure which evaluates the subset of features on the basis of the hypothesis i.e. good feature subsets contain features highly correlated with the prediction of the class, yet uncorrelated with (non-predictive of) each other. A feature of a subset is good if it is highly correlated with the class but not much correlated with other features of the class.

Let, S is a subset of features which contains k features, and then the merit of S is written as:

$$Merit_{sk} = \frac{\overrightarrow{kr}_{cf}}{\sqrt{k} + k(k-1)r_{ff}}$$

Where, r_{cf} is the average value of all feature-classification correlations and r_{ff} is the average value of all feature-feature correlations. The correlation feature selection criteria(CFS) is defined as:

$$CFS = max_{sk}\left[\frac{r_{cf1} + r_{cf2} + r_{cfk}}{\sqrt{k + 2(r_{f1f2} + r_{fif_j} + r_{fif1})}}\right]$$

Where, r_cf_i and r_{fifj} are referred as correlations. This technique is applied on software metrics to select the important sub set of metrics for fault-prone module prediction in software.

IV. OBJECTIVES

Software plays a major role starting from a simple system like washing machine to a highly complex and safety system like air traffic control system and nuclear system. Failures in the software lead to customer dissatisfaction, economic loss and loss of life. Therefore, software professionals are trying to deliver software which are not only functional attractive but also safe and reliable. Software Reliability is defined as the probability of failure free operation of software for a specified period of time in a specified environment. Software development effort is one of the software reliability indicators in software industry.

V. RESULTS AND DISCUSSIONS:

In this paper, one software effort datasets viz. Desharnais are used for predictive performance of the ANN models. Desharnais datasets consist of a large number of projects. . In this section, the results of ANN-PSO, ANN-PCA and ANN-CFS models are discussed and compared with existing models. The results are reported for all models in this work with best generalization performance on the test datasets. For the ANN and ANN-PCA approach, for each test set, 17 simulations are carried out and the average value of these simulations is reported for each of the datasets considered. In the proposed ANN-PCA-CFS approach, for each test set,17 simulations are carried out as results . The averaged results of ten simulations are reported in this work. function = mean magnitude relative error (MMRE) acceleration , MMRE as the fitness function ,the results are reported for the above combination of parameters in this work . To test if there is a significant difference between the proposed approaches, statistical test (t-test) with 95% confidence level was performed on one datasets. t-test is a statistical test which is used to determine .whether the mean of a population significantly different from the mean of another population. The prediction capability of different models on testing set is evaluated using MMRE.

$$MMRE = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{Actual - Predicted}{Actual}\right]$$

Where Actual and Predicted are the actual and predicted efforts and N is the number of projects for which effort is predicted. ANN is the percentage of predictions that fall within 25% of Actual value. In the following subsections, experimental results of proposed approaches

A. MATLAB in ANN

Workflow for Neural Network Design The work flow for the neural network design process has seven primary steps.

Data collection in step 1 generally occurs outside the framework of Neural Network Toolbox software, but it is discussed in general terms in ―Multilayer Neural Networks and Back propagation Training‖. The Neural Network Toolbox software uses the network object to store all of the information that defines a neural network. This topic describes the basic components of a neural network and shows how they are created and stored in the network object. After a neural network has been created, it needs to be configured and then trained. Configuration involves arranging the network so that it is compatible with the problem you want to solve, as defined by sample data. After the network has been configured, the adjustable network parameters (called weights and biases) need to be tuned, so that the network performance is optimized. This tuning process is referred to as training the network. Configuration and training require that the network be provided with example data. This topic shows how to format the data for presentation to the network. It also explains network configuration and the two forms of network training: incremental training and batch training. More about "Four Levels of Neural Network Design" "Neuron Model" "Neural Network Architectures" "Understanding Neural Network Toolbox Data Structures".

B. Four Levels of Neural Network Design

There are four different levels at which the Neural Network Toolbox software can be used. The first level is represented by the GUIs that are described in ―Getting Started with Neural Network Toolbox‖. These provide a quick way to access the power of the toolbox for many problems of function fitting, pattern recognition, clustering and time series analysis. The second level of toolbox use is through basic command-line operations. The command line functions use simple argument lists with intelligent default settings for function parameters. (You can override all of the default settings, for increased functionality.) This topic, and the ones that follow, concentrate on command-line operations. The GUIs described in Getting Started can automatically generate MATLAB code files with the command-line implementation of the GUI operations. This provides a nice introduction to the use of the command-line functionality. A third level of toolbox use is customization of the toolbox. This advanced capability allows you to create your own custom

neural networks, while still having access to the full functionality of the toolbox. The fourth level of toolbox usage is the ability to modify any of the code files contained in the toolbox. Every computational component is written in MATLAB code and is fully accessible. The first level of toolbox use (through the GUIs) is described in Getting Started which also introduces command-line operations. The following topics will discuss the command line operations in more detail. The customization of the toolbox is described in ―Define Neural Network Architectures.

C. Neuron Models

The fundamental building block for neural networks is the single-input neuron, such as this example. There are three distinct functional operations that take place in this example neuron. First, the scalar input p is multiplied by the scalar weight w to form the product wp, again a scalar. Second, the weighted input wp is added to the scalar bias b to form the net input n. (In this case, you can view the bias as shifting the function f to the left by an amount b. The bias is much like a weight, except that it has a constant input of 1.) Finally, the net input is passed through the transfer function f, which produces the scalar output a. The names given to these three processes are: the weight function, the net input function and the transfer function. For many types of neural networks, the weight function is a product of a weight times the input, but other weight functions (e.g., the distance between the weight and the input, |w − p|) are sometimes used. (For a list of weight functions, type help nnweight.) The most common net input function is the summation of the weighted inputs with the bias, but other operations, such as multiplication, can be used. (For a list of net input functions, type help nnnetinput.) ―Introduction to Radial Basis Neural Networks‖ .how distance can be used as the weight function and multiplication can be used as the net input function. There are also many types of transfer functions. Examples of various transfer functions are in ―Transfer Functions‖. (For a list of transfer functions, type help nntransfer.) Note that w and b are both adjustable scalar parameters of the neuron. The central idea of neural networks is that such parameters can be adjusted so that the network exhibits some desired or interesting behavior. Thus, you can train the network to do a particular job by adjusting the weight or bias parameters. All the neurons in the Neural Network Toolbox software have provision for a bias, and a bias is used in many of the examples and is assumed in most of this toolbox. However, you can omit a bias in a neuron if you want.

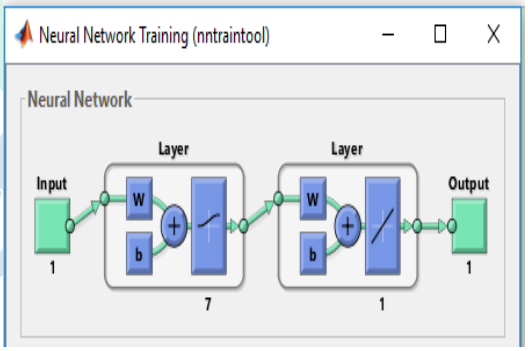
D. Transfer Functions

Many transfer functions are included in the Neural Network Toolbox software. Two of the most commonly used functions are shown below. The following figure illustrates the linear transfer function. Neurons of this type are used in the final layer of multilayer networks that are used as function approximators. This is shown in ―Multilayer Neural Networks and Back propagation Training‖. The sigmoid transfer function shown below takes the input, which can have any value between plus and minus infinity, and squashes the output into the range 0 to 1.

This transfer function is commonly used in the hidden layers of multilayer networks, in part because it is differentiable. The symbol in the square to the right of each transfer function graph shown above represents the associated transfer function. These icons replace the general f in the network diagram blocks to show the particular transfer function being used. For a complete list of transfer functions, type help nntransfer. You can also specify your own transfer functions. You can experiment with a simple neuron and various transfer functions by running the example program nnd2n1.

Table:1 Actual dataset

0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
0.11	0.12	0.13	0.14	0.15	0.16	0.17	0.18	0.19	0.2
0.21	0.22	0.23	0.24	0.25	0.26	0.27	0.28	0.29	0.3
0.31	0.32	0.33	0.34	0.35	0.36	0.37	0.38	0.39	0.4
0.41	0.42	0.43	0.44	0.45	0.46	0.47	0.48	0.49	0.5
0.51	0.52	0.53	0.54	0.55	0.56	0.57	0.58	0.59	0.6
0.61	0.62	0.63	0.64	0.65	0.66	0.67	0.68	0.69	0.7
0.71	0.72	0.73	0.74	0.75	0.76	0.77	0.78	0.79	0.8
0.81	0.82	0.83	0.84	0.85	0.86	0.87	0.88	0.89	0.9
0.91	0.92	0.93	0.94	0.95	0.96	0.97	0.98	0.99	1

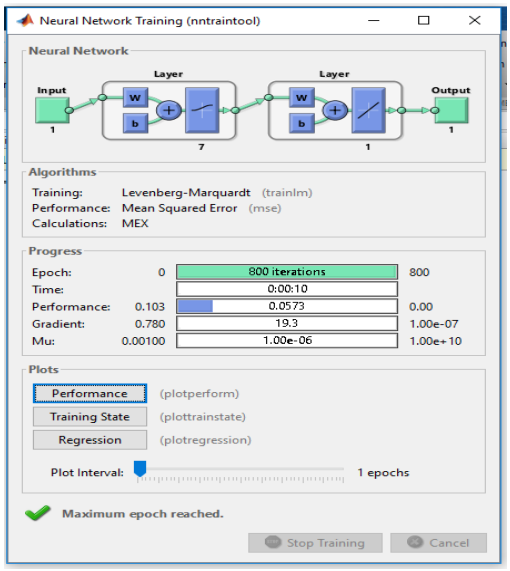


Neural Network Training ANN Architecture

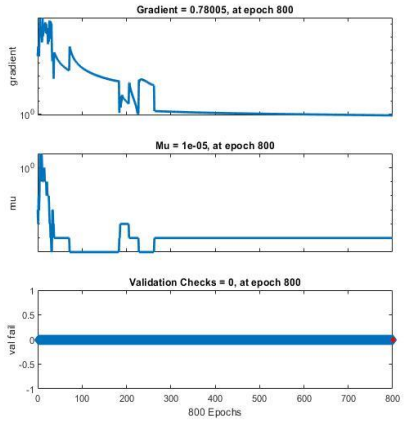
Actual data result

Original data set result we found that in this way after test the process.

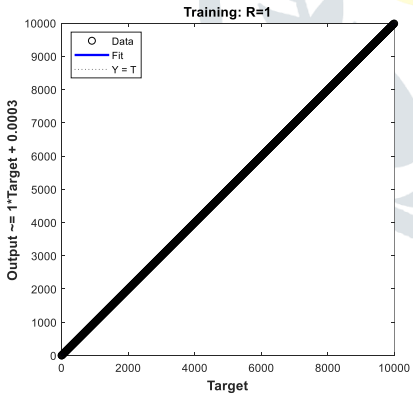
RESULT:



Neural network Performance



Neural Network Training State



Neural Network Training Regression

VI. Conclusions

An ANN architecture by adding an additional input layer with logarithmic activation function has been proposed in this paper and trained using CFS and PCA algorithm to predict software development efforts. CFS is applied to reduce dimension of input features by using MATLAB tool kit in that it will reduced data actual is 11 it will reduced in to 4 . PCA is used in matlab and it will reduced the effort of this data set and it will also give less out put but compare both result PCA is the best result and reeducation. The proposed approach provides better accuracy for the compared datasets in terms of MMRE .The proposed approach provides improvement in MMRE Desharnais dataset, respectively, than the compared models. Similarly, the proposed approach provides Desharnais dataset, respectively, than the compared models. In this paper we can also apply the genetic algorithm (GA) in that linear regression and back propagation also we can apply in this paper.

References

[1] Jorgensen, M.: Forecasting of software development work effort: evidence on expert judgment and formal models‘, Int. J. Forecast., 2007, 23, pp. 449–462

[2] Boehm, B.W.: Software engineering economics‘ (Prentice-Hall, Englewood Cliffs, NJ, 1981)

[3] Albrecht, A.J., Gaffney, J.: Software function, source lines of code, and development effort prediction: a software science validation‘, IEEE Trans. Software Engineering., 1983, 9, (6), pp. 639–648

[4] Delone, W.H.: Determinants of success for computer usage in small business‘, MIS Q., 1988, 12, (1), pp. 51–61

[5] Golden, J.R., Mueller, J.R., Anselm, B.: Software cost estimating: craft or witchcraft‘. DATABASE, 1981, pp. 12–14

[6] Putnam, L., Myers, W.: Measures for excellence‘ (Yourden Press Computing Series, 1992)

[7] Pressman, R.S.: Software engineering a practitioner‘s approach‘ (McGraw-Hill, Englewood Cliffs, NJ, 1997

[8] Jorgensen, M., Shepperd, M.: A systematic review of software development cost estimation studies‘, IEEE Trans. Software Engineering., 2007

[9] Heiat, A.: Comparison of artificial neural network and regression models for estimating software development effort‘, Inf. Software. Technol., 2002

[10] Tronto, I.F.B., Silva, J.D.S., SantAnna, N.: Comparison of artificial neural network and regression models in software effort estimations‘. IEEE Int. Joint Conf. Neural Networks, Orlando, USA, 2006

[11] Changjie, M., Guochang, G., Jing, J.: Improved neural network based on dynamic predication model of software reliability‘, J. Convergence Inf. Technol., 2011

[12] Jin, C., Jin, S.W., Ye, J.M.: Artificial neural network-based metric selection for software fault-prone prediction model‘, IET Softw., 2012

[13] Gray, A.R., MacDonell, S.G.: A comparison of techniques for developing predictive models of software metrics‘, Inf. Softw. Technol., 1997

[14] Tronto, I.F.B., Silva, J.D.S., SantAnna, N.: An investigation of artificial neural networks based prediction systems in software project management‘, J. Syst. Software., 2008.