On the Problem of Attribute Selection for Software Cost Estimation: Input Backward Elimination Using Artificial Neural Networks

Efi Papatheocharous[1] and Andreas S. Andreou[2]

[1] University of Cyprus, Department of Computer Science. 75 Kallipoleos Street, P.O. Box 2053, CY1678 Nicosia, Cyprus
efi.papatheocharous@cs.ucy.ac.cy
[2] Cyprus University of Technology, Department of Electrical Engineering and Information Technology, 31 Archbishop Kyprianos Street, 3036 Lemesos, Cyprus
andreas.andreou@cut.ac.cy

Abstract. Many parameters affect the cost evolution of software projects. In the area of software cost estimation and project management the main challenge is to understand and quantify the effect of these parameters, or 'cost drivers', on the effort expended to develop software systems. This paper aims at investigating the effect of cost attributes on software development effort using empirical databases of completed projects and building Artificial Neural Network (ANN) models to predict effort. Prediction performance of various ANN models with different combinations of inputs is assessed in an attempt to reduce the models' input dimensions. The latter is performed by using one of the most popular saliency measures of network weights, namely Garson's Algorithm. The proposed methodology provides an insight on the interpretation of ANN which may be used for capturing nonlinear interactions between variables in complex software engineering environments.

Keywords: Software Cost Estimation, Artificial Neural Networks, Connection Weights, Garson's Algorithm.

1 Introduction

Software effort estimation is the process of predicting the required effort to support the software development process by utilising attributes of cost, often called 'cost drivers'. This process usually involves activities after product specification and until software implementation and delivery, and is usually performed at the initiation of a project. The accurate computation of the development effort in software organisations is critical since it enables project managers to effectively deal with uncertainties and risks associated with resource planning and allocation. Specifically, cost overestimations result, in over allocation of resources and budget increase, which may cause loss of contracts and interruption of negotiations. On the other end, cost

underestimations cause loss of money upon project completion, misallocation of project resources, quality compromises or budget and schedule extensions.

Artificial Intelligence (AI) techniques are quite popular in software cost estimation and are used for building models and calculating the effort factor. Especially, Artificial Neural Networks (ANN) that have the ability to provide a non-linear mapping among the inputs and the output have been used extensively. Nevertheless, in previous related works ANN are commonly used only as predictors and very rarely quantitative analysis is conducted regarding the influence of the network inputs on the output. In this work we focus on the ability of ANN to capture interactions between the influencing cost factors and effort and, in addition, the input's degree of influence built within the network is examined using Garson's Algorithm [1]. The overall purpose is to examine the prediction accuracy of development effort through the utilisation of different models and variable contributing factors. The values of the cost drivers are located within two widely-known and public databases, namely the Desharnais and the ISBSG, which are selected for experimentation. The contribution of inputs is assessed through a random sampling approach and using the resulting values of internal weights from the ANN. Gradually the contributing weights of inputs whose values do not significantly affect the output of the ANN are removed from the initial complete set of cost factors. The experiments conducted show that in software cost estimation there are several factors which are not critically significant but are commonly used for predicting effort in related research work. The input analysis conducted using Garson's Algorithm helps in removing factors in a backward manner, starting from the least significant ones and until half of the initial cost factors are left in each dataset whereas during this process ANN's prediction performance is continuously assessed.

The rest of this paper is organised as follows: Section 2 discusses the recent work on ANN utilisation for the problem of software cost estimation and also presents common approaches used for simplification and interpretation of ANN in other problem domains. Section 3 specifies the modeling technique and theory behind Garson's Algorithm. Section 4 presents in detail the methodology proposed, accuracy measures used in the experimental process and discusses the main results obtained. Finally, Section 5 summarises the conclusions and future research steps.

2 Related Work

This section initially presents the latest applications of Artificial Neural Networks (ANN) in the software cost estimation literature and identifies that even though the approach is considered promising, one of the most important steps, the identification and inspection of the dominant cost attributes, is not given proper thought.

Recent work of Tronto et al. [2] investigates the application of ANN and stepwise regression for software effort prediction. The experiments were conducted on the COCOMO dataset employing categorical variables whose impact was identified based on the work of Angelis et al. [3] forming new categorical values. The authors identified a strong relationship between the success of each technique and the size of

the learning dataset, the nature of the cost function and dataset characteristics, such as existence of outliers, collinearity and number of attributes.

In software cost estimation the comparison of models is a common research goal. Kaur et al. [4] prove the effectiveness of ANN models for the NASA dataset compared to the Halstead, Walston-Felix, Bailey-Basili and Doty models, all of which are popular legacy models used in software cost estimation. Backpropagation ANN were used and reported as the most generalised networks currently in use that present good estimation capabilities. In addition, Reddy and Taju [5] used the popular COCOMO model in software cost estimation mapped to an ANN with minimal number of layers and nodes to increase the performance of the network. They employed a feedforward backpropagation MLP and obtained improved predictions for effort using the COCOMO dataset compared to the COCOMO model. Rao et al. [6] used a Functional Link Artificial Neural Network (FLANN) which does not contain any hidden layers so that the network architecture becomes simple and training does not involve full backpropagation, thus reducing computational complexity. Their method provides more accurate results compared to other methods for software cost estimation on the NASA dataset.

Many researchers applied ANN on software cost estimation and yielded very accurate results. However, when using ANN one crucial step is to identify the dominant cost factors, or attributes, that affect development effort [7]. A number of measures exist to determine the significance of ANN input attributes [1, 8, 9, 10] but we identified that they have never been applied for software cost drivers. For example, sensitivity analysis, fuzzy curves, MSE change, weight elimination and node pruning, and optimal brain damage (OBD) methods are measures that rank input feature importance. Some of these measures are heuristic (forward and backward selection), sensitivity index-based, are based on pseudo weights, rely on Garson's algorithm and some of its modified and extended versions that appear in the literature [11]. More specifically, in this work the concepts described in the following methodologies have been adopted for software cost drivers: Garson [1] proposed a method for partitioning the ANN connection weights to determine the relative importance of each input variable in the network (for more details see Section 3.2). Glorfeld [9] presented a methodology to simplify ANN using a backward selection process to eliminate input variables that are not contributing to the predictive power of accurate networks. According to the author this enables decision makers to understand the resulting effect of each contributing variable in producing accurate predictions. The application is on two classification examples, a commercial loan and a cheque overdraft problem.

3 Modeling Technique and Methodology for Input Elimination

3.1 Artificial Neural Networks

One of the primary applications of ANN involves models to forecast a dependent variable from a given set of independent variables. These are non-linear, model-free and alternative to traditional statistical methods. ANN consist of basic computational

elements called neurons organized in groups forming layers. Certain types of neurons organised in multiple layers form the Multi-Layer Perceptron (MLP) [12] which is one of the most popular networks. The number of neurons in the input (first) layer is equal to the number of attributes used as independent variables. The last layer is the network output. Each subsequent layer uses the weights coming from the previous layers and adjusts them so that the accuracy performance error between the actual and predicted values for the dependent variable represented by the output is diminished.

3.2 Methodology with ANN and Garson's Algorithm

There are many methods for measuring the contribution of independent variables within a neural network, but most of which are very complicated and thus are rarely used in the area of software cost estimation. Garson's algorithm [1] is considered a good trade-off example among complexity and effectiveness. It partitions the hidden layer weights into components associated with each input node. Next, the percentage of all hidden nodes weights associated with a particular input node is used to measure the relative importance of that attribute. The interested reader may refer to [13] for a step-by-step example on the algorithm.

A variety of ANN architectures were implemented, starting with a topology which contains a number of neurons in the hidden layer equal to the number of attributes used as inputs in each experiment and continuing with topologies resulting from increasing the number of hidden neurons by 1 until their number becomes twice the size of the input attributes. In addition, the 'weakest' attribute is removed from the sample until the inputs are reduced to half the initial size. Moreover, the following randomisation process was followed for each sample: The initial weights and biases of the network were randomly set and the dataset used was randomly divided into three holdout subsets, training, validation and testing, with the percentages of 60%, 20% and 20% of the total available samples respectively, where each sample participates in only one subset.

The scaled conjugate gradient training function was used which is based on the derivative functions of weights, net inputs and transfer functions. The training process is repeated ten times so that the optimal network that minimizes the prediction error is identified and the weights of each input-hidden-output path are stored for further use by Garson's algorithm. Evaluation of the networks was performed using the testing data samples based on the well known *MMRE* and *pred(.25)* accuracy measures. For each experiment ten holdout random samples were chosen so that validation on random data is performed.

After training is executed and the network is stabilized, for each input $j, j=1,2,...,i$, the Relative Importance (RI_j) is calculated using equation (1), where N_i and N_h are the number of input and hidden neurons, respectively and w is the connection weight, the superscripts 'i','h' and 'o' refer to input, hidden and output layers, respectively and subscripts 'k', 'm' and 'n' refer to input, hidden and output neurons (in our case n=1 as there is only one output neuron). According to Garson's algorithm, for each input node j the relative contribution of j to the outgoing signal of each hidden neuron is calculated and converted to a percentage, which serves as a measure of importance for each input node representing each variable. According to the proposed methodology,

each input that makes the smallest contribution to the final output of the network, as this is reflected through the weight connections, is eliminated. Thus, in each repetition the initial number of variables utilised is lowered gradually by one until the necessary number of variables are left in the dataset.

$$RI_j = \frac{\sum_{m=1}^{Nh}\left[\left[\frac{\left|w_{jm}^{ih}\right|}{\sum_{k=1}^{Ni}\left|w_{km}^{ih}\right|}\right]\times\left|w_{mn}^{ho}\right|\right]}{\sum_{k=1}^{Ni}\left[\sum_{m=1}^{Nh}\left[\left[\frac{\left|w_{km}^{ih}\right|}{\sum_{k=1}^{Ni}\left|w_{km}^{ih}\right|}\right]\times\left|w_{mn}^{ho}\right|\right]\right]} \quad (1)$$

4 Experiments and Results

4.1 Datasets Description

The Desharnais (1989) dataset [14] included 81 observations for systems developed by a Canadian Software Development House. The second dataset ISBSG R9 [15] provided by the International Software Benchmarking Standards Group contains an analysis of the cost and other measurements for a large group of software projects, approximately 3,024. The projects come from a broad cross section of industry and range in size, effort, development platform, language, etc. These projects underwent a series of quality checks and pre-processing to create filtered versions of the datasets that do not contain null values and conform to the standards we set for homogeneity and integrity before feeding them as inputs to the ANN. The filtered datasets contained 77 and 113 in the Desharnais and ISBSG datasets and the attributes selected and used in this work, along with their abbreviations are summarised in Table 1.

4.2 Results

The results reported in this section include the initial and final lowest performance values of the best obtained network architectures in terms of prediction accuracy (*MMRE* value). The results of the training and testing phases of the various network architectures created are sensitive to the initialisation of weights, bias values and random division of the data samples used for training and testing. Initially, the number of input attributes for each experiment is reduced gradually by one according to importance of the inputs suggested by Garson's algorithm; then, the network is trained again with the reduced variables and the new performance is traced. This process is repeated ten times (on random holdout samples) and Tables 2 and 3 report the *MMRE* and *Pred(.25)* ANN performance figures for the training and testing phases using the Desharnais and ISBSG datasets respectively. The 'Initial' and 'Final'

column results report accuracy having the number of inputs being equal to the initial number of attributes in each dataset *p* and being reduced to *p/2* respectively. The order in which attributes are removed in each experiment repetition (first column) is given in the second column, while the rest columns present the forecasting performance observed after removing the 'less important' attributes.

Table 1. Summary of the attributes in the datasets used

Desharnais		**ISBSG**	
Team Experience (years)	TE	Functional Size	FS
Manager Experience (years)	ME	Adjusted Function Points	AFP
Duration (months)	DU	Project Elapsed time	PET
Transactions	TR	Project Inactive time	PIT
Entities	EN	Resource Level (ordinal)	RL
Points Adjusted	PA	Maximum Team Size	MTS
Scope	SC	Input count	INC
Points Non Adjusted	PNA	Output count	OC
		Enquiry count	EC
		File count	FC
		Interface count	IFC
		Added count	AC
		Changed count	CC
		Deleted count	DC

Table 2. Random sampling and first four attributes removed from the Desharnais dataset.

#	Order of Attributes Removed	ANN Training Phase				ANN Testing Phase			
		Initial *MMRE*	Initial *Pred*	Final *MMRE*	Final *Pred*	Initial *MMRE*	Initial *Pred*	Final *MMRE*	Final *Pred*
1	TE,DU,SC,ME	0.384	0.936	0.559	0.936	0.536	0.867	0.600	0.867
2	ME,DU,TR,TE	0.280	0.979	0.343	1.000	0.409	0.933	0.487	0.933
3	SC,EN,TE,DU	0.557	0.936	0.583	0.936	0.198	1.000	0.335	1.000
4	TE,TR,PA,SC	0.474	0.915	0.485	0.894	0.364	1.000	0.387	1.000
5	ME,PA,TE,DU	0.361	0.957	0.360	0.979	1.264	0.867	1.060	0.867
6	TE,SC,DU,ME	0.512	0.915	0.784	0.915	0.386	0.933	0.346	0.933
7	ME,SC,PNA,PA	0.472	0.957	0.572	0.957	0.569	0.800	0.512	0.800
8	TE,ME,SC,EN	0.509	0.957	0.572	0.957	0.351	0.800	0.512	0.800
9	TE,ME,EN,DU	0.358	0.936	0.356	0.936	0.507	0.933	0.578	0.933
10	TE,SC,EN,DU	0.482	0.894	0.768	0.894	0.312	0.933	0.293	0.933
	Mean	**0.439**	**0.938**	**0.538**	**0.940**	**0.490**	**0.907**	**0.511**	**0.907**

The experiments indicate that quite accurate and successful predictions were obtained, as suggested by the consistently low *MMRE* values in both the Desharnais and ISBSG cases throughout the random holdout validation sampling process. Moreover, comparing the initial and final values of the accuracy measures we observe some performance degradation, something which indicates that maybe a part of useful information is lost when reducing the number of the participating attributes. One may argue that this is expected as the information contributing to the ANN learning process is truncated and hence prediction accuracy is gradually lowered as we move from the initial to the final network state. More specifically, the accuracy degree of the Desharnais dataset decreases 0-29% in the training phase depending on the

experiment repetition, while during testing accuracy increases in some cases by 20% and in others decreases by 16%. This also occurs with the ISBSG dataset, where by removing attributes the performance accuracy increases by 6% and decreases by 6% in the training phase depending on the experiment, while during the testing phase accuracy increases by 16% and decreases by 24%.

Another interesting finding is that the attributes that seem to be the 'weakest' effort contributors in the majority of the experiments are TE, ME, DU, SC for the Desharnais case and RL, FC, CC, INC, OC for the ISBSG. We should also report, though, that there were validation cases in which the general picture of input significance was a bit disrupted. This may be considered as a weakness of the method as it relies on ANN models that behave differently when the initial conditions (initialisation) and the training/testing data samples change. Therefore, the results must be interpreted cautiously and only after a satisfactory number of repetitions that will enable a statistically safe conclusion.

Table 3. Random sampling-first seven attributes removed from the ISBSG dataset.

#	Order of Attributes Removed	ANN Training Phase				ANN Testing Phase			
		Initial *MMRE*	Initial *Pred*	Final *MMRE*	Final *Pred*	Initial *MMRE*	Initial *Pred*	Final *MMRE*	Final *Pred*
1	CC,AFP,MTS,PET,OC,PIT,EC	0.223	0.957	0.329	0.928	0.358	1.000	0.578	1.000
2	DC,CC,PET,RL,PIT,INC,EC	0.280	0.986	0.352	0.971	0.418	0.955	0.575	0.955
3	INC,CC,DC,RL,OC,AC,FC	0.337	1.000	0.404	0.986	0.255	0.955	0.265	0.909
4	MTS,CC,RL,INC,FC,FS,AC	0.350	0.957	0.493	0.957	0.199	0.955	0.191	0.955
5	FS,RL,FC,EC,IC,INC,CC	0.367	0.971	0.411	0.986	0.202	0.955	0.298	0.955
6	RL,DC,CC,INC,OC,EC,FS	0.362	1.000	0.303	1.000	0.346	0.909	0.385	0.864
7	CC,OC,RL,FC,INC,AFP,IC	0.210	1.000	0.271	0.986	0.377	0.909	0.257	0.909
8	FC,OC,AC,AFP,RL,IC,FS	0.247	0.957	0.344	0.957	0.188	1.000	0.424	1.000
9	FC,IC,RL,AFP,AC,EC,PIT	0.261	0.957	0.305	0.971	0.662	1.000	0.500	1.000
10	FS,IC,PIT,OC,AFP,RL,FC	0.338	0.986	0.279	0.986	0.249	0.955	0.270	0.955
	Mean	**0.297**	**0.977**	**0.349**	**0.972**	**0.325**	**0.959**	**0.374**	**0.950**

5 Conclusions

This work investigates the ability of ANN to capture interactions between the influencing cost factors and effort within empirical software engineering project samples and attempt to provide cost predictive models. The main contribution is the understanding of the explanatory value of the inter-relationships between the input variables and the final output (effort) which is extracted from the internal network weights. Thus, it may provide an insight regarding each variable's contribution to the overall prediction.

We performed a backward elimination strategy to minimise the initial inputs and progressively evaluated the significance of connection weights and input variables. The approach was based on Garson's Algorithm which exploits the various ANN models created, trained and tested over ten random sets of training and testing distinct samples. Moreover, from the various architectures created, trained and tested the results obtained from the best networks in terms of *MMRE* of actual vs. the prediction testing samples, i.e. result of the simulation phase of the process provide quite

promising results. The approach enables decision makers to understand the resulting effect of each contributing variable in producing accurate predictions.

An interesting issue for future research is the comparison of the contributions of other saliency measures reported in the literature along with Garson's over the same or newer datasets. In addition, a cost-benefit analysis of accuracy declination and the fewer attributes leading to less cost for collecting data and faster cost estimation modeling should be carried out to prove the validity of this work.

References

1. Garson, G.D.: Interpreting Neural-Network Connection Weights. AI Expert 6, 46--51 (1991)
2. Tronto, I.F.D.B., Silva, J.D.S.D., Sant'Anna, N.: An Investigation of Artificial Neural Networks based Prediction Systems in Software Project Management. Journal of Systems and Software 81, 356--367 (2008)
3. Angelis, L., Stamelos, I., Morisio, M.: Building A Software Cost Estimation Model Based On Categorical Data. Proceedings of the 7th International Symposium on Software Metrics, IEEE Computer Society, pp. 4--15 (2001)
4. Kaur, J., Singh, S., Kahlon, K. S., Bassi, P.: Neural Network – A Novel Technique for Software Effort Estimation. International Journal of Computer Theory and Engineering 2 (1) 1793-8201, 17--19 (2010)
5. Reddy C.S., Raju, K.: A Concise Neural Network Model for Estimating Software Effort. International Journal of Recent Trends in Engineering 1 (1), 188--193 (2009).
6. Rao, B.T., Sameet, B., Swathi, G.K., Gupta, K.V., RaviTeja, C., Sumana, S.: A Novel Neural Network Approach for Software Cost Estimation using Functional Link Artificial Neural Network (FLANN). International Journal of Computer Science and Network Security 9 (6), 126--131 (2009)
7. Park, H., Baek, S.: An Empirical Validation of a Neural Network Model for Software Effort Estimation. Expert Systems with Applications 35, 929--937 (2008)
8. Belue, L.M., Bauer, K.W.: Determining Input Features for Multilayer Perceptrons. Neurocomputing 7, 111--121 (1995)
9. Glorfeld, L.W.: A Methodology for Simplification and Interpretation of Backpropagation-Based Neural Network Models. Expert Systems with Applications 10, 37--54 (1996)
10. Satizábal, H.M., Pérez-Uribe, A.: Relevance Metrics to Reduce Input Dimensions in Artificial Neural Networks, Artificial Neural Networks-ICANN, Springer Berlin / Heidelberg, pp. 39--48 (2007)
11. Zhang, G.: Neural Networks for Classification: A Survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 30, 451--462 (2000)
12. McCulloch, W.S., Pitts, W.: A Logical Calculus of the Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biology 5 (4), 115--133 (1943)
13. Olden, J.D., Jackson, D.A.: Illuminating the "Black Box": a Randomization Approach for Understanding Variable Contributions in Artificial Neural Networks. Ecological Modelling 154 135--150 (2002)
14. J.M. Desharnais, Analyse Statistique de la Productivite des Projects de Development en Informatique a Partir de la Technique de Points de Fonction, MSc. Thesis, Université du Québec, Montréal, 1989.
15. The International Software Benchmarking Standards Group, http://www.isbsg.org/ Repository Data Release 9, 2005.