A PRELIMINARY PERFORMANCE EVALUATION OF MACHINE LEARNING ALGORITHMS FOR SOFTWARE EFFORT ESTIMATION

POONAM RIJWANI[a1] AND SONAL JAIN[b]

ABSTRACT

Accurate Software Effort Estimation is vital to the areas of Software Project Management. It is a process to predict the Effort in terms of cost and time, required to develop a software product. Traditionally, researchers have used the off the shelf empirical models like COCOMO or developed various methods using statistical approaches like regression and analogy based methods but these methods exhibit a number of shortfalls. To predict the effort at early stages is really difficult as very less information is available. To improve the effort estimation accuracy, an alternative is to use machine learning (ML) techniques and many researchers have proposed plethora of such machine learning based models. This paper aims to systematically analyze various machine learning models considering the traits like type of machine learning method used, estimation accuracy gained with that method, dataset used and its comparison with empirical model. Although researchers have started exploring Machine learning from past two decades, this paper analyses comparison on studies being used in recent years. Subsequently exploring various studies, we found that the estimation accuracy of these ML models is near to the satisfactory level and gives enhanced results than that of non-Machine Learning based models.

KEYWORDS: Estimation, Machine Learning, Neural Network, Software Effort Model, Systematic Review.

Software development Effort Estimation is the process of estimating the cost and time to develop the software. Collectively we call it Software Effort Estimation. Estimations done at early stages of software development play a vital role in effective software project management. There are numerous algorithmic and non-algorithmic models exists to estimate the software effort but still the research advocates that on an average, the overrun of the software projects appears to be nearly 30 percent. [Torleif and Jorgensen, 2012]

A detailed review was ushered by Jorgensen and Shepperd, 2007 which ascertains nearly 10 estimation approaches for software effort estimation. Amongst those methods, the dominating ones were regression based methods and also the usage of expert judgment and analogy based methods are growing. Myriad of software effort estimation techniques exists from expert judgement to analogy, empirical models to statistical techniques.

Instead of using expert judgment to decide the minimum and maximum range of effort, software specialists better focus to use historical data about former estimation error to set realistic minimum–maximum effort intervals [Jorgensen and Sjoeberg, 2003].

Though expert judgment can be very precise, it can also be simply misled. If the experts, who are involved with estimating the effort, are made aware of the budget, expectations of the clients, time availability or other parameters that govern the estimation, the estimations can be misled.

One chronicle way to improve the precision of effort estimates is using historical data and estimation checklists consisting of various estimation parameters. When relevant past data and parameter checklist are included in the process, actions are less probable to be overlooked, and it's more likely to produce realistic estimates. Many software organizations use tools for this so that to improve software effort estimations.

Too-low estimates can lead to lower quality of product developed, possible rework in later phases, and greater risks of project failure; whereas higher estimates can diminish productivity in accordance with Parkinson's law, which states that work expands to fill the time available for its completion [Jorgensen, 2014].

Several studies corresponding to estimation of effort analyzes and compares the precision of such models and approaches. The study shows that There Is No "Best" Effort Estimation Model or Technique. One of the foremost reasons for this instability in results is essential correlation between various parameters governing the software effort, such as project size, type of project, development environment etc. [Javier, 2001] .In addition to this, the parameters which have prevalent impact on the development effort seems to fluctuate, signifying that estimation models should be personalized to the environments in which they're used.

In past few years, machine learning centered methods have been getting growing consideration in software development effort estimation research. Amongst various popular estimation models like algorithmic model and expert judgement, Machine learning based models are also considered as an

[1]Corresponding author

important category of effort estimation [Mendes et. al., 2003 & Elish, 2009].

Zhang and Tsai, 2003 summate the uses of many Machine Learning techniques in software development domain, including support vector machines, case-based reasoning, decision trees, artificial neural networks, and genetic algorithms.

Though the study on Machine Learning models is growing in academia, latest investigations [Jorgensen and Shepperd, 2007, Moløkken-Østvold et. al., 2004 & Jorgensen, 2004] have shown that expert judgment which a non-machine learning based model is still the prevailing technique for software effort estimation in industry.

The purpose of this paper is to present a systematic review of machine learning techniques mainly artificial neural networks and its comparison with existing empirical models. One of the most popular empirical models used in the industry is COCOMO for estimating the software effort. Although the research of amalgamating machine learning has started from past two decades, our paper mainly focuses on the latest machine learning procedures being proposed and implemented.

MACHINE LEARNING

Machine learning solely focuses on writing softwares that can learn from past experience. A computer program is said to learn from experience 'E' with respect to some class of task 'T' and performance measure 'P', if its performance at tasks in 'T', as measured by 'P', improves with experience 'E' [Wang, 2003]. It is an extraction of knowledge from data. Machine learning can be categorized into three types: Supervised Learning, unsupervised Learning and Reinforcement Learning. Supervised learning is where we teach; train the machine using data already available with the correct outcome. The more the dataset, the better the machine will learn about that subject. After the machine is trained, it will be given unseen data and based on the past experience it will give the outcome. Unsupervised learning is where the machine is trained using a dataset without labels. The learning algorithm is never told what the data represents and it infers a function to define hidden structure from unlabeled data. Reinforcement learning is the one in which training data is available but unlike supervised one, correct input/output pairs are never presented. Once the unlabeled data has been processed it only takes one example of labeled data to make the learning algorithm fully effective. A good example is in playing games. When a machine wins a game, then the result is trickled back along with all the moves to reinforce the validity of those moves.

We are focusing on the problem of software effort estimation and our goal is to create a machine which can mimic a human mind and to do that it needs learning capabilities. Once a machine is trained based on the above category of learning, the effort can then be predicted. The machine learning particularly neural network approaches give estimations close to human level estimations.

METHODOLOGY

Here in this paper, Constructive Cost Model (COCOMO) is being used for investigation purposes. This regression based method to estimate effort has given by Sir Barry Boehm in 1981 and then to adapt to new software development environment, its new revised version COCOMOII was published. Its various parameters are from data of various historical projects. The procedure of effort estimation is performed in following steps:

A. Preprocessing of Data
B. Procedure Setup
C. Selection of Input used
D. Experimentation
E. Evaluation Criteria
F. Testing and Validations

All the models were implemented using standard datasets available and trained with 70 percent inputs as training data and rest used for testing and validation purposes. In the papers that areexplored, COCOMO 81, NASA (63), NASA (93), IBMDPS, Kemerer, Hallmark and Maxwell datasets are used for the Software development Effort Estimation.

NEURAL NETWORK TECHNIQUES FOR EFFORT ESTIMATION

Software development Effort estimation is a challenging task for people associated with software project management. Here, in this section, we present a review of various neural network models for effort estimation proposed and implemented by many researchers.

Researcher Venkatachalam, 1993 presented simplified feed-forward neural network (FFNN) for software development effort estimation. Venkatachalam used back propagation neural network for estimating effort exhausting 22 independent variables which were COCOMO's cost drivers. Evaluation criteria were not specified with his study.

Researcher Finnie et al., 1997 presented a comparison of statistical regression based model with other artificial Intelligence based estimation models for estimation of software development effort. The Researchers found out that statistical regression model underperformed for intricate and complex software projects while the Artificial Intelligence based models provide agreeable estimation results. They considered dataset among Projects from 17 organization and Desharnais. MMRE was used as an evaluation criterion.

Another researcher Heiat, in 2002 investigated Feed Forward Neural Networks with function point and Radial Basis Neural Network with Source Lines of Codes for diverse datasets encompassing projects of varied generation languages. For every dataset separately, Heiat has given evaluation with regression model. He utilized Kemerer dataset of 15 projects and IBM DP service organization dataset of 24 projects for first investigation, and for second trial, utilized Hallmark dataset of 28 projects. The IBM and Kemerer projects are developed using third generation languages while Hallmark projects are developed with fourth generation languages. The results have shown that artificial neural network method is modest with regression when a third generation language data set is used. But in case of fourth generation languages data set or mixed dataset were used, neural network methodology works expressively precise for software effort estimation.

Another Researcher Ideri et. al., 2006 applied clustering algorithms with Radial Basis Feed Forward Networks. For clustering the training sets, clustering algorithms were used and evidenced that C-means with Radial Basis Feed Forward Networks achieves improved results with APC III algorithm with Radial Basis Feed Forward Networks for software effort estimation.

In 2007, Tronto et. al., made comparisons of conventional linear regression model and simplified Feed Forward Neural Networks for Boehm's COCOMO dataset. The experimentations were showed that the Neural Network based method accomplishes enhanced results as that of with linear regression model. The reason for improved results is due to adaptable and non-parametric nature of neural networks.

In 2009, Reddy and Raju suggested a multilayer feed-forward neural network to accommodate the Boehm's COCOMO and its parameters to estimate effort. Reddy, Raju shared the complete dataset into training and validation set. The ratio for division of dataset is kept to be 80 %: 20 % respectively of total 63 projects. The various input parameters of the COCOMO are accommodated with natural logarithmic order in feed-forward neural network, which was a decent try to place together expert knowledge, project data and the traditional algorithmic approach into one single framework which is appropriate to predict effort.

Wong et. al. in 2008 presented a blend of neural nets and fuzzy logics to expand the precision of backfiring size estimations. The neuro-fuzzy method was used to attune the conversion ratios with the goal of minimizing the margin of error.

Wei et. al. in 2010 are to assess the estimate performance of the neuro-fuzzy model with System Evaluation and Estimation of Resource Software Estimation Model (SEERSEM) in software estimation practices and to apply the architecture that combines the neuro-fuzzy method with diverse algorithmic model. The results of this research also demonstrate that the general neuro-fuzzy structure can perform well with many algorithmic models for refining the performance of software development effort estimation

Another researcher used the amalgamation of Functional Link Artificial Neural Network (FLANN) and Particle swarm Optimization (PSO) algorithm for Software Effort Estimation [Benala et. al., 2013]. Hybrid PSO-FLANN architecture is a type of three-layer Feed Forward neural network. PSO algorithm is used to train the weight of FLANN vector. Calculation has been done on three datasets COCOMO 81, NASA63 and Maxwell. Hybrid algorithm increases the accuracy of the input vector parameters.

Another hybrid approach by combining Functional Link Artificial Neural Network (FLANN) and Genetic Algorithms (GA) for effort estimations were proposed by Benala, Dehuri in 2012. The Genetic Algorithm fitness function will be selected to minimize the error find out by evaluation criteria MMRE as shown in equation:

$$f = \frac{1}{MMRE}$$

Kalichanin-Balich, 2010 relates linear regression, and Logarithmic regression with Feed Forward Neural Network. According to the test results, it has been witnessed that software estimate is more precise and genuine using FFNN rather than regression and logarithmic models. MMRE is used as an evaluation criterion.

Kumar V. et. al. 2008 used wavelet neural network (WNN) with four approaches, i.e., WNN-morelet, WNN-guassian, TAWNN-guassian, and TAWNN-morelet. A Threshold acceptance training

algorithm is used for wavelet neural network, i.e., TAWNN. WNN-Morelet and WNN-Guassianovertookvarious techniques. Results were efficiently improved.

Rao B.T. et. al., 2009 suggested a FLANN for software effort estimation. It generates effort and then processed final output layer. Its one shortcoming is that in this relation between inputs and outputs is not reasonable.

Kaur J. et-al. 2010 instigated a back propagation Artificial Neural Network of 2-2-1 design on NASA dataset comprises of 18 projects. Input was KDLOC and development methodology and effort was the output. MMRE was found to be 11.78 with his applied approach.

Attarzadeh and Ow, 2010 proposed a new model to accommodate COCOMO II. 5 Scale factors and 17 Effort multipliers were used as input. A sigmoid activation function is used to create network in order to accomplish post architecture COCOMOII model. Results shown in terms of MMRE, and Pred (0.25) to compare it with algorihmic COCOMO.

Attarzadeh et. al., 2012 projected a novel software development effort estimation model exhausting neural networks. In this, the Initial weights of the network were set in such a way that it lead to COCOMOII model. The proposed neural network model provides improved result as related to COCOMO model after appropriate training.

Dave and Dutta, 2011 suggested a Adjusted MMRE. They used NASA dataset comprises of 60 projects. Experiments were conducted with three different assessment methods, i.e., Mean Magnitude Relative Error, Modified Mean Magnitude Relative Error, and Relative Standard Deviation. Three estimation modes are used for this purpose, i.e., Regression analaysis, FFNN, and RBFNN. According to authors, RBFNN is found to be a superior technique for effort estimation, on the basis of RSD and Modified MMRE.

COCOMO II

Originally, the COCOMO model is given by Boehm in 1981. It is implemented after various investigation on 63 software projects [Boehm, 2000]. This empirical model provides effort in terms of cost and schedule for a development of a software project. In late 1990's, Boehm proposed COCOMO II [Rao et. al., 2009] to accommodate the environmental changes in software industry. The purpose of COCOMO model is to express effort with software size and a series of cost and scale factors, as given in the equation below:

$$PM = A.(SIZE)^{1.01+\sum_{i=1}^{5} SFi} . \prod_{j=1}^{17} EMj$$

Where A is a multiplicative constant, and the set of SF (Scale Factor) and EM (Effort Multiplier) parameters which have a strong impact on calculated effort.

Moreover Size can be calculated by various methods like Kilo Source Lines of Code (KSLOC), Function Points, Extended Function Points and adaptation adjustment factors.

Maximum work has focused based on algorithmic cost models such as COCOMO and Function Points. These might undergo from the shortcoming such as the necessity to adjust the model to each individual measurement environment coupled with very variable accuracy levels even after calibration.

DISCUSSION

Table 1: Summarized methods of few Researchers

Researcher (year)	Method deployed	Dataset (no. of projects	Evaluation criteria
Vinay Kumar et al (2008)	Wavelet Neural Networks	IBMDPS (24),CF	MMRE, Pred(0.25), MdMRE
B. Tirimula Rao (2009)	C-FLANN,P-FLANN,LFLANN	NASA(60)	RMSE
Sriman Srichandan (2010)	Radial Basis Functional Neural Networks	COCOMO81 (252), Tukutuku (53)	MMRE, Pred(0.25)
Jaswinder Kaur(2010)	Back propagation artificial neural network	NASA	MMRE, RMSSE
Iman Attarzadeh (2010)	Back Propogation ANN	COCOMO (63)	MMRE, Pred (0.25)
Vachik S. Dave(2011)	RBFNN, FFNN, Regression Analysis	cocomonasa_v1(60)	MMRE, Modified MMRE, RSD
Iman Attarzadeh (2012)	ANN-COCOMOII	COCOMO-1(63), NASA93 (93)	MMRE, Pred(0.25)
Jagannath Singh(2012	Cascade Forward ANN, Elman ANN, Feed Forward ANN, Recurrent ANN	NASA(60)	) MMRE, RMSE, Means BRE, Pred(0.25)
SrimanSrichandan(2012)	RBFNN	COCOMO 81, Tukutuku	MMRE, Pred(0.25)

Various Performance Evaluation Criteria for Effort Estimation

The purpose of Performance evaluation criteria is to identify the accurate and truthful implementation of the effort estimation algorithms. The most significant evaluation measures used in software effort estimation is presented in table 2.

Table 2: Significant Performance Evaluation criteria in effort estimation

Evaluation Criteria	Explanation
$\mathrm{RE_j} = \frac{\lvert \mathrm{actual_j} - \mathrm{estimated_j} \rvert}{\mathrm{actual_j}}$	Relative Error
$\mathrm{MRE_j} = \mathrm{RE_j} * 100$	Magnitude of Relative Error
$\mathrm{MMRE} = \frac{1}{\mathrm{N}} \sum_{\mathrm{j=1}}^{\mathrm{N}} \mathrm{MRE_j}$	Mean Magnitude of Relative Error
$\mathrm{MdMRE} = \mathrm{Median(MRE)}$	MdMRE is Median (MRE). It is measure for mean MRE error
$\mathrm{MER_j} = \frac{\lvert \mathrm{actual_j} - \mathrm{estimated_j} \rvert}{\mathrm{estimated_j}} * 100$	Magnitude Error Relative is the error relative to the estimate.
$\mathrm{MMER} = \frac{1}{\mathrm{N}} \sum_{\mathrm{j=1}}^{\mathrm{N}} \mathrm{MER_j}$	Mean of all observations of MER
$\mathrm{MAE} = \frac{1}{\mathrm{n}} \sum_{\mathrm{j=1}}^{\mathrm{n}} \lvert \mathrm{actual_j} - \mathrm{estimated_j} \rvert$	Mean of Absolute Errors
$\mathrm{MAPE} = \sum_{\mathrm{j=1}}^{\mathrm{n}} \frac{\left(\frac{\lvert \mathrm{actual_j} - \mathrm{estimated_j} \rvert}{\mathrm{actual_j}} \right)}{\mathrm{n}} * 100$	Mean Absolute Percentage Error
$\mathrm{MSE} = \frac{1}{\mathrm{n}} \sum_{\mathrm{j=1}}^{\mathrm{n}} (\mathrm{actual_j} - \mathrm{estimated_j})^2$	Mean Squared Error
$\mathrm{RMSE} = \sqrt{\frac{1}{\mathrm{n}} \sum_{\mathrm{j=1}}^{\mathrm{n}} (\mathrm{actual_j} - \mathrm{estimated_j})^2}$	Root Mean Square Error

CONCLUSION

The paper presented a number of Software effort Estimation models based on machine learning techniques for the choice of suitable Artificial Neural Network techniques for calculating crucial effort for new projects. The techniques considered are MLFF, RBFFN, wavelet neural networks, Cascade Forward ANN, Elman ANN, Feed Forward ANN, Recurrent ANN etc. That trained and tested occurrences are considered with these approaches. Purpose of this entire thing is evaluating and comparing ANN methods with Post Architectural COCOMO in prediction accuracy. Studies conducted on Machine Learning techniques indicate that the estimated cost of the software with these models has more rapidity and precision of algorithmic models such as COCOMO II, which is a widely used empirical model in software industry. Further, effective results show that ANN models in the local data are improved responses in comparison with algorithmic models. The exploitation of machine learning techniques like genetic algorithms, fuzzy decision trees, case based reasoning, etc can also be applied along with these approaches for topology optimization and structural optimization.

REFERENCES

Torleif H. and Jørgensen M., 2012. "From origami to software development: A review of studies on judgment-based predictions of performance time." Psychological bulletin, **138**(2):238.

Jorgensen M. and Shepperd M., 2007. "A systematic review of software development cost estimation studies." IEEE Transactions on software engineering, **33**(1):33-53.

Jørgensen M. and Sjoeberg D.I.K., 2003. "An effort prediction interval approach based on the empirical distribution of previous estimation accuracy." Information and Software Technology, **45**(3):123-136.

Jorgensen M., 2014. "What We Do and Don't Know about Software Development Effort Estimation." IEEE software, **31**(2).

Javier D.J., 2001. "On the problem of the software cost function." Information and Software Technology, **43**(1):61-72.

Mendes E., Watson I., Triggs C., Mosley N. and Counsell S., 2003. A comparative study of cost estimation models for web hypermedia applications, Empirical Software Engineering, **8**(2):163–196.

Tronto I.F.B., Silva J.D.S. and Anna N.S., 2008. An investigation of artificial neural networks based prediction systems in software project

management, Journal of Systems and Software, **81**(3):356–367.

Elish M.O., 2009. Improved estimation of software project effort using multiple additive regression trees, Expert Systems with Applications, **36**(7):10774–10778.

Zhang D. and Tsai J.J.P., 2003. Machine learning and software engineering, Software Quality Journal, **11**(2):87–119.

Moløkken-Østvold K., Jørgensen M., Tanilkan S.S., Gallis H., Lien A.C. and Hove S.E., 2004. A survey on software estimation in the norwegian industry, in: Proceedings of the 10th International Symposium on Software Metrics, Chicago, Illinois, USA, pp. 208–219.

Jørgensen M., 2004. A review of studies on expert estimation of software development effort, Journal of Systems and Software, **70**(1–2):37–60.

Wang, John, ed. Data mining: opportunities and challenges. IGI Global, 2003.

Venkatachalam A. R., 1993. "Software cost estimation using artificial neural networks." Neural Networks, 1993. IJCNN'93-Nagoya. Proceedings of 1993 International Joint Conference on.Vol. 1.IEEE, 1993.

Finnie G.R., Wittig G.E. and Jean-Marc D., 1997. "A comparison of software effort estimation techniques: using function points with neural networks, case-based reasoning and regression models." Journal of Systems and Software, **39**(3):281-289.

Heiat A., 2002. "Comparison of artificial neural network and regression models for estimating software development effort." Information and software Technology, **44**(15):911-922.

Ideri A., Khosgoftaar T.M. and Abran A., 2002. Can neural network be easily interpreted in software cost estimation? World Congress on Computational Intelligence, Honolulu, Hawaii, pp 1162–1167.

Ideri A., Abran A. and Mbarki S., 2006. An experiment on the design of radial basis function neural networks for software cost estimation. IEEE, information and communication technologies, ICTTA, pp 1612–1617.

Tronto I.F.B., de-Silva J.D.S. and Sant'Anna N., 2007. Comparison of artificial neural network and regression models in software effort estimation. In: Proceedings of international joint conference on neural networks, Orlando, Florida.

Reddy S. and Raju K.V.S.V.N., 2009. A concise neural network model for estimating software effort. Int J Recent Trends Eng., **1**(1):188–193.

Wong J., Ho D. and Capretz L.F., 2008. Calibrating Functional Point Backfiring Conversion Ratios Using Neuro-Fuzzy Technique. International Journal of Uncertainty, Fuzziness and KnowledgeBased Systems, **16**(6):847–862.

Wei L.D., Danny H. and Luiz F.C., 2015. "Improving software effort estimation using neuro-fuzzy model with SEER-SEM." arXiv preprint arXiv:1507.06917.

Benala T.R., Chinnababu K., Mall R. and Dehuri S., 2013. "A Particle Swarm Optimized Functional Link Artificial Neural Network (PSO-FLANN) in Software Cost Estimation", Advances in Intelligent Systems and Computing Proceedings of the International Conference on Frontiers of Intelligent Computing, pp. 59-66, Springer-Verlag.

Benala T.R. and Dehuri S., 2012. "Genetic Algorithm for Optimizing Functional Link Artificial Neural Network Based Software Cost Estimation", Proceedings of the InConINDIA, pp. 75-82, Springer-Verlag.

Kalichanin-Balich I., 2010. Applying a Feedforward Neural Network for Predicting Software Development Effort of Short-Scale Projects, presented at Eighth ACIS International Conference on the Software Engineering Research, Management and Applications (SERA).

Kumar V.K. et al., 2008. Software development cost estimation using wavelet neural networks, Journal of Systems and Software, **81**:1853-1867.

Rao B. T. et al., 2009. 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**:126-131.

Kaur J. et al., 2010. Neural Network-A Novel Technique for Software Effort Estimation, International

Journal of Computer Theory and Engineering, **2**:1793-8201.

Attarzadeh I. and Ow S. H., 2010. Proposing a new software cost estimation model based on artificial neural networks, in 2nd International Conference on Computer Engineering and Technology (ICCET), pp. V3-487-V3- 491.

Attarzadeh I. et al., 2012. Proposing an Enhanced Artificial Neural Network Prediction Model to Improve the Accuracy in Software Effort Estimation, in Fourth International Conference on, Computational Intelligence, Communication Systems and Networks (CICSyN), pp. 167-172.

Dave V. S. and Dutta K., 2011. Neural network based software effort estimation & evaluation criterion MMRE, in 2nd International Conference on Computer and Communication Technology (ICCCT) pp. 347-351.

Boehm B. W., 1981. "Software engineering economics".

Boehm B. W., 2000. ”Software Cost Estimation with COCOMOII”, Prentice Hall.

Srichandan S., 2012. A new approach of Software Effort Estimation Using Radial Basis Function Neural Networks, International Journal on Advanced Computer Theory and Engineering (IJACTE) **1**:2319-2526.