Software Estimation Using Deep Learning

Mr. Manohar K. Kodmelwar, Shashank D. Joshi, V. Khanna

Abstract: ***For any effective project management, estimation stands as a key piece of project methodology. It plays a noteworthy part in project management to execute the orders required. The assessed parameter helps in sharing the resources requisite to finish the project deliverables effectively. The significant parameters that control the software projects are time, prerequisites, individuals, infrastructure/materials and cash, and dangers. This is one cause why making great appraisals of these elements like time and also resources required for a project stands exceptionally basic. The assessed parameter helps in sharing the resources requisite to finish the project deliverables effectively. The significant parameters that control the software projects are time, prerequisites, individuals, infrastructure/materials and cash, and dangers. This is one cause why making great appraisals of these elements like time and also resources required for a project stands exceptionally basic. If the estimation is lesser compared to required then the with the progress of project it lacks the cash & time as well. If over estimated then the chances of wastage of resources. Therefore it is required to estimate correctly to avoid problem in future.***

Index Terms— Differential Evolution (DE), Pareto-Based Differential Evolution (PBDE), MMRE, COCOMO,MRE

I. INTRODUCTION

Development of Software involves a few factors that have an important influence on productivity. Effort estimation considered as important part in software development phase. The correct progress of development is dependent on the correct estimation. The different techniques are used to estimate the development of software. It is most important to in today's highly competitive world the correctness of estimation.

II. LITRETURE SURVEY

Shailendra and Anoj proposed another meta-heuristic algorithm centered upon Differential Evolution (DE) by Homeostasis mutation administrator. Pareto-Based Differential Evolution (PBDE) was one of the adaptations of DE. Although that algorithm's performance was great, its convergence rate can be additionally enhanced by limiting the time complexity of non dominated sorting, and by improving the solutions diversity. That had been executed by utilizing effective non dominated algorithm whose time intricacy was superior to previous one and another mutation scheme was implemented in DE which can give greater diversity among solutions. The proposed variation increased the Homeostasis value with one more vector, termed the Homeostasis mutation vector. The suggested approach gave all the more encouraging answers for direct the evolution and helped DE flee from the circumstance of stagnation. The outcome checked that our proposed Homeostasis mutation system performed superior to other best in class algorithms.

Taeho Lee et. Al proposed a software cost estimation modeling method and led an experiential study in the Korean defense domain. The proposed software cost estimation modeling method, recognized as MND-SCEMP was appropriate aimed at the Korean defense domain. The recommended modeling process incorporated the extraction of cost factors in the Korean defense domain by domain analysis, developed a software cost estimation model to be utilized in the Korean defense domain, confirmed the model, and modified the model. To approve the proposed modeling process, they played out an experiential study of 113 software development projects on weapon frameworks in Korea. A software cost estimation model was created by implementing the suggested modeling process. The MMRE value of that model was 0.566 while the accuracy was proper for utilizing. They reasoned that the modeling procedure and software cost estimation model created in that review was appropriate for evaluating resource necessities amid weapon system development in South Korea's national defense domain. That modeling procedure and model encouraged more exact resource estimation by project planners, which prompt more fruitful project execution.

Jovan Popovic et. al [7] recognized the task types that were most correlated to the UCP size. The authors have created and cross-compared prediction models for estimating task-type efforts by means of UCP size using an Online analytical processing model and R packages on a set of 32 real-world projects, with the goal of facilitating analysis of the correlation between project sizes and effort required to complete task types. Requirements, scoping, functional specification, and functional testing task types have up to two times better estimation accuracies than project effort. Implementation had slightly better accuracy than the project effort, while the other task types were not correlated to the UCP size. Using estimates of the most correlated task types and other techniques, such as expert judgment for others, they improved the overall project effort prediction accuracy and decreased the error from 26 to 16%.

Revised Manuscript Received on December 22, 2018.

Mr. Manohar K. Kodmelwar[1], Research Scholar, Bharath University, Chennai, India be university), Guntur, AP.

Dr. Shashank D. Joshi[2], Faculty of Engineering and Technology Bhartividyapth, Pune, India .

Dr. V. Khanna Department of Information Technology, Bharath University, Chennai, India

H. Azath and R.S.D. Wahidabanu [4] proposed a method for effectively estimating the software effort using Function Points. That study was a basis for the improvement of software effort estimation research through a series of quality attributes along with constructive cost model (COCOMO). The classification of software system for which the effort estimation was to be calculated based on COCOMO classes. For that quality assurance ISO 9126 quality factors were used and for the weighing factors the function point metric was used as an estimation approach. Effort was estimated for MS word 2007 using the following models: Albrecht and Gaffney model, Kemerer model, SMPEEM model (Software Maintenance Project Effort Estimation Model and FP Matson, Barnett and Mellichamp model. The advantage of the proposed effort estimation system was to handle correctly the imprecision and the uncertainty when describing the software project. From the implementation results, they observed that the proposed method was effectively estimated the effort of the software project models.

P. Morrow [2] examined the trade-off between the utility of outputs from simplified functional sizing approaches, and the effort required by those sizing approaches, through a pilot study. The goal of that pilot study was to evaluate the quality of sizing output provided by NESMA's simplified size estimation methods, adapt their general principles to enhance their accuracy and extent of relevance, and empirically validate such an adapted approach using commercial software projects. The performances of those adaptations were evaluated against the NESMA approaches in three ways: (1) effort to perform; (2) the accuracy of the total function counts produced; and (3) the accuracy of the profiles of the function counts for each of the base functional component types. The adapted approach outperformed the Indicative NESMA in terms of sizing accuracy and generally performed as well as the Estimated NESMA across both datasets, and required only approximately 50 % of the effort incurred by the Estimated NESMA.

Karel Dejaeger et. al tended to the concern of univocal conclusion by giving an account of the aftereffects of an expansive scale benchmarking study. Distinctive kinds of systems were under consideration, counting strategies instigating tree/rule-based models like M5 and also CART, linear models, for example, different sorts of linear regression, nonlinear models (MARS, multilayered perceptron neural networks, radial basis function networks, least squares) together with the estimation methods that don't expressly incite a model (for instance, a case-based reasoning approach). Moreover, the part of feature subset choice by utilizing a generic backward input selection wrapper was explored. The outcomes were subjected to rigorous statistical testing as well demonstrated that conventional least squares regression in the mix with a logarithmic transformation performed best. Another key finding was that by choosing a subset of exceedingly prescient characteristics, for instance, project size, development, and environment-related qualities, commonly a critical increment in estimation precision can be acquired.

III. PROPOSED METHOD

Effort estimation is considered as the important to both the developer as well as customer point of view. The management also takes the consideration about the estimation for approving the budget and time for completion. The estimation should be the careful consideration for the development. In the proposed system neural network is used by applying the weights in each layer. The optimization technique is used to better classification of outcome. The output of deep learning depends on the previous layer of the network structure . The deep NN classifier consist of different sort of layers like convolution, pooling as well as fully connected layer. In convolution the multiple layer contains the weight matrices called as filters. These filters are also called kernels. They slide across the input features. In every convolutional layer, initially, the outcome of preceding layers is convolved with multiple learned weight matrices termed filter masks or learned kernels. Then, the outcome is processed through a 'non-linear operation' to make the layer output. A kernel stands as the matrix to be convolved with the inputs features and also stride regulates how much the filter convolve over the input features. This layer carries out the convolution on the input data with the kernel utilizing the layers. The fig.1.shows the diagrammatic representation of the steps involved in hybrid particle swarm optimization.

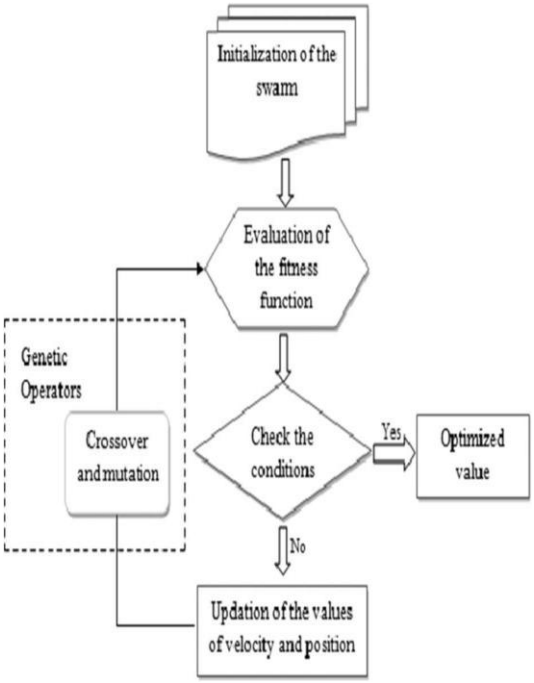


Fig-1 Diagrammatic representation of the steps involved in hybrid particle swarm optimization.

Dataset Description:

The dataset utilized as input to the proposed system is the COCOMO dataset. This is a publically available repository dataset. This dataset is made openly available to inspire the refutable, repeatable, verifiable, and also enhanceable predictive designs of software engineering. The dataset COCOMO was frequently used for validating various effort estimation methods. It includes 60 software projects that are described

by 17 attributes in conjunction with an actual effort [11]. The actual effort in the COCOMO dataset is measured by person-month which represents the number of months that one person needs to develop a given project. Despite the fact that the COCOMO dataset are now over 25 years old, it is still commonly used to assess the accuracy of new techniques [12]. The COCOMO software cost design gauges effort on 152 hours (comprises development as well as management hours). COCOMO presumes that the effort develops over linearly in software size.

Effort Multipliers:

The seventeen Post-Architecture effort multipliers (EM) are used in the COCOMO model to adjust the nominal effort, Person-Months, to reflect the software product under development. Each multiplicative cost driver is defined below by a set of rating levels and a corresponding set of effort multipliers [13]. The Nominal level always has an effort multiplier of 1.00, which does not change the estimated effort.

Off-nominal ratings generally do change the estimated effort. For example, a high rating of Required Software Reliability (RELY) will add 10% to the estimated effort, as determined by the COCOMO II.2000 data calibration. A Very High RELY rating will add 26%. It is possible to assign intermediate rating levels and corresponding effort multipliers for your project. The size and cost driver ratings can be different for each module, with the exception of the Required Development Schedule (SCED) cost driver and the scale factors. The COCOMO II model can be used to estimate effort and schedule for the whole project or for a project that consists of multiple modules [14]. The effort multiplier terms are show in a table 3.1

acap	analysts capability
pcap	programmers capability
aexp	application experience
modp	modern programing practices
tool	use of software tools
vexp	Virtual machine experience
lexp	Language experience
sced	Schedule constraint
stor	main memory constraint
data	data base size
time	time constraint for cpu
turn	turnaround time
virt	machine volatility
cplx	process complexity
rely	required software reliability

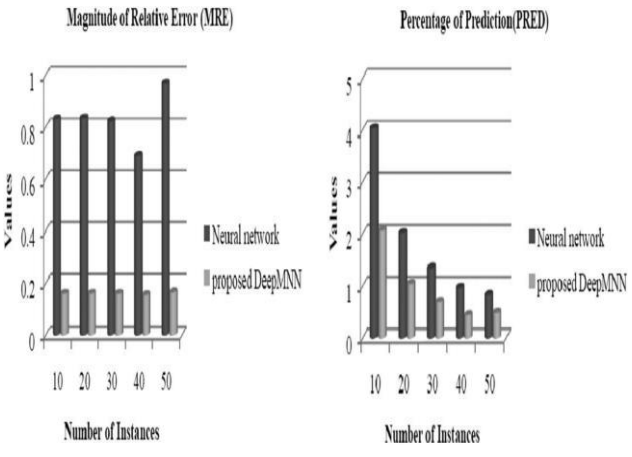
Table 3.1 Effort multiplier terms

IV. EXPERIMENTAL RESULT

The proposed system is implemented by using the data set's values as the input values. The input dataset con-tains various attributes. The proposed system is accessed via determining the performance parameters. It was deter-mined for the existing and the proposed one. A clear evaluation was performed to analyze the existing with the proposed system. A total of 50 instances were taken for evaluation. The results were performed after the addition of every ten instances.

Instances	Parameter	Existing system	Proposed system
10	RE	0.4153	0.4029
	MRE	0.8306	0.1611
	MMRE	2.4178	1.2460
	MARE	3.2238	1.6613
	PRED	4.0297	2.0767
20	RE	0.4168	0.4032
	MRE	0.8336	0.1613
	MMRE	1.2097	0.6252
	MARE	1.6130	0.8336
	PRED	2.0162	1.0421
30	RE	0.4116	0.4022
	MRE	0.8233	0.1609
	MMRE	0.8045	0.4116
	MARE	1.0727	0.5489
	PRED	1.3409	0.6861
40	RE	0.3877	0.3454
	MRE	0.6908	0.1550
	MMRE	0.5816	0.2590
	MARE	0.7754	0.3454
	PRED	0.9693	0.4317
50	RE	0.4836	0.4143
	MRE	0.9673	0.1657
	MMRE	0.4972	0.2902
	MARE	0.6629	0.3869
	PRED	0.8286	0.4836

For MMRE is determined for the NN as well as the pro-posed Deep MNN. MMRE stands as a common gauge of the 'average estimation accuracy.' It is a de facto assessment criterion to evaluate the software project's accuracy in prediction models. The values are obtained form different evaluation.



V. CONCLUSION

Software development effort estimation can be performed in an effective method by using the proposed algorithm. The effort estimation aimed at software development is not anymore a monotonous task with the implementation of the proposed system.

This system enables professionals and industrialists to select the most profitable jobs in respect of the effort involved in it. In the given paper, the performance parameters for different instances were evaluated and studied. The experimental outcome illustrates that the proposed work outweighs the existing works in all features. This lets to the conclusion that the Deep Learning MNN has better competence than the existing systems.

VI. FUTURE WORK

The direction of the future research is also to study factors, other than the software size and team experience, such as the team size, software complexity measures and software development methodology which may have impacts on the effort estimation of software works. We are motivated to continue this research by applying the proposed model using real-live data of software projects with different scales.

In the future other algorithms, such as Bagging and Ensemble algorithms can be implemented for predicting effort in projects. Results would be observed against each in order to identify better performance keeping in view the accuracy and prediction estimates.

In future work, the effort estimated by expert judgment method has to be considered to optimize the final effort estimation. Initial value of the optimization is the effort estimated by expert judgment.

Further future work is intended to overcome the instability issues, a more generic prediction model that is not highly affected by the size and the type of data set, and preferably an enhancement to the optimization algorithm itself. Moreover, it would be important to work towards a modified hybrid approach that encompasses the best characteristics of different prediction schemes.

In future, we will use feature subset selection technique to focus on important attributes that affects the effort and extends the work in such a way so that it can work for any dataset.

REFERENCES

1. Kan Qi, and Barry W. Boehm, “A light-weight incremental effort estimation model for use case driven projects”, In Software Technology Conference (STC), 2017 IEEE 28th Annual, IEEE, 2017, PP. 1-8.
2. Philip Morrow, F. George Wilkie, and I. R. McChesney, “Function point analysis using NESMA: simplifying the sizing without simplifying the size”, Software Quality Journal, 2014, Vol. 22, No. 4, PP. 611-660.
3. Tomas Urbanek, Zdenka Prokopová, Radek Silhavy, and Stanislav Sehnálek, “Using analytical programming and UCP method for effort estimation”, In Modern Trends and Techniques in Computer Science. Springer, Cham, 2014, PP. 571-581.
4. Azath, H., and R. S. D. Wahidabanu, “Efficient effort estimation system viz. function points and quality assurance coverage”, IET Software, 2012, Vol. 6, No. 4, PP. 335-341.
5. Shashank Mouli Satapathy, Barada Prasanna Acharya, and Santanu Kumar Rath, “Early stage software effort estimation using random forest technique based on use case points”, IET Software, 2016, Vol. 10, No. 1, PP. 10-17.
6. Radek Silhavy, Petr Silhavy, and Zdenka Prokopova, “Evaluating subset selection methods for use case points estimation, Information and Software Technology, 2017.
7. Jovan Popovic, Dragan Bojic, and Nenad Korolija, “Analysis of task effort estimation accuracy based on use case point size”, IET Software, 2015, Vol. 9, No. 6, PP. 166-173.
8. Ricardo de A. Araujo, Adriano L.I. Oliveira, Silvio Meira, “A Class of Hybrid ´ Multilayer Perceptrons for Software Development Effort Estimation Problems”, Expert Systems With Applications, 2017.
9. Murillo-Morera, Juan, Christian Quesada-López, Carlos Castro-Herrera, and Marcelo Jenkins, “A genetic algorithm based framework for software effort prediction”, Journal of Software Engineering Research and Development, 2017, Vol. 5, No. 1, PP. 4.
10. Przemysław Pospieszny, Beata Czarnacka-Chrobot, Andrzej Kobyliński, “An effective approach for software project effort and duration estimation with machine learning algorithms”, The Journal of Systems & Software, 2017.
11. Mohammad Azzeh, “Dataset quality assessment: an extension for analogy based effort estimation”, 2017.
12. Sandeep Kad, and Vinay Chopra, “Software development effort estimation using soft computing”, International Journal of Machine Learning and Computing, 2012, Vol. 2, No. 5, PP. 548.
13. Majed Al Yahya, Rodina Binti Ahmad, and Sai Lee, “Impact of CMMI based software process maturity on COCOMO II's effort estimation”, Int. Arab J. Inf. Technol., 2010, Vol. 7, No. 2, PP. 129-137.
14. Nancy Merlo–Schett, Martin Glinz, and Arun Mukhija, “Seminar on software cost estimation WS 2002/2003”, Department of Computer Science, 2002, PP. 3-19.

15. Acknowledgement
16. I am thankful to the Dr. S. D. Joshi for encouraging & helping in doing Ph.D. I thankful to Dr. V. Khanna for guiding me during th software development. I am also thankful to my family for supporting. I am thankful to all friends .

AUTHOR PROFILE

Mr. Manoahr K. Kodmelwar is a research scholar at Bharat university Chennai, Tamilnadu. He completed the Engineering Degree & Master of Computer Engineering. He is interested in software engineering and operating system domain.

Dr. S. D. Joshi completed the PhD. From bharati vidyapeth. His area of intrest is software engineering. He the Head of Dept. Of Information Technology in bharati vidyapeth Pune.

Dr. V. Khanna is the dean if information technology at Bharath University Chennai, Tamilnadu. His area of interest is software engineering.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication