SISY 2021 Plenary talk

Artificial Neural Network Architectures and Orthogonal Arrays in Estimation of Software Projects Efforts Estimation

Prof. dr Ljubomir LAZIC

Full Professor, The Computing School, UNION University Belgrade, Serbia

In the software engineering, estimation of the effort, time and cost required for the development of software projects is an important issue. It is a very difficult task for project managers to predict the cost and effort needed in the premature stages of planning. Software estimation ahead of development can reduce the risk and increase the success rate of the project. Many traditional and machine learning methods are used for software effort estimation by researchers, but always it has been a challenge to predict the effort accurately. In our study, different Artificial Neural Network (ANN) used for effort estimation is discussed. It is observed that the prediction of software effort by using ANN is more precise and better compared to traditional methods such as Function point, Use-case methods and COCOMO etc. Models based on neural networks are competitive in nature as compared to statistical and traditional regression methods. This talk explains the overview of various ANN architectures, and deep learning networks used by the researchers for software effort estimation (SEE). In addition, we have set out own criteria, which has been used to compare all the different selected methods. We have also given a score for each evaluation criteria, so that we can compare the different methods numerically for cost estimation. Our observations have shown that it is best to use a number of different estimating techniques or cost models, and then compare the results before determining the reasons for any of the large variations. None of the methods are necessarily better or worse than the others. We found, in fact, that their strengths and weaknesses often complement each other. Therefore, the main conclusion is that there is no one single technique that is best for every situation, and the results of a number of different approaches need to be carefully considered to discover what is the most likely to produce estimates that are realistic. Metrics to monitor progress, such as the mean absolute error (MAE), and mean magnitude relative error (MMRE), were used. Besides reducing the values, we compare the obtained MMRE results with two different activation functions like the sigmoid function and hyperbolic tangent function. Finally, to confirm our experiment's and functions' obtained values and efficiency, five different ANN architectures (ANN-9, ANN12, ANN18, ANN-27 and ANN-36) based on Taguchi's orthogonal vector plans and activation functions are compared.

The motivation of our research is to achieve:

- A simple ANN architecture that requires a short training time (small number of iterations, i.e., epoch, which is less than 10). Acceptable error for application in other areas with a critical mission, where a rapid response time to the studied problem is required;
- A stable training process of ANN architecture, no oscillations, peaks, etc.;
- A reliable ANN architecture training process that has been validated on a large number of datasets from practice;

- The minimization of the required number of observations in order to quickly and effectively train the ANN architecture;
- The prediction accuracy required to be the same or better than other approaches (COCOMO, function point analysis, use case point analysis, etc.), which can be compared to estimating the magnitude of software development efforts.

The outcome of our research is as follows:

- The method of clustering achieved homogenization of the heterogeneous nature of the projects of each used dataset;
- The MMRE value is in the range of 30.1% to 48.2% for the ANN-L27 architecture, depending on the nature of the projects of the dataset used;
- The best results are achieved with the sigmoid activation function of the hidden layer and the output layer;
- The number of iterations performed is in the range of 5 to 9, depending on the given dataset and the cluster within the dataset;
- The model's efficiency, reliability, and accuracy were confirmed through two correlation coefficients (Pearson's and Spearman's);

Some important software cost estimation methods have been studied in this research work.

Approach 1: Constructive Cost Model (COCOMO)

It is considered that COCOMO is a very important model that can calculate a software cost estimate.This uses an algorithmic formula in order to estimate the software's cost.. Therefore, this model is based on both mathematics and a number of experimental equations. Barry Bohem proposed it in 1981 for software cost estimation. It is considered to be the most complete approach and is better documented than the other cost estimation model. In addition, many of researchers in the software engineering field are now trying to increase the efficiency by keeping the COCOMO model's base. We used Four (4) Input metric from COCOMO2000 which are: E, PEMi, KSLOC, i SFj to construct different ANN architectures to predict SEE. The search/optimization embraced here is motivated by the Taguchi method based on Orthogonal Arrays (an extraordinary set of Latin Squares), which demonstrated to be an effective apparatus in a robust design. This work aims to minimize the magnitude relative error (MRE) in effort estimation by using Taguchi's Orthogonal Arrays, as well as to find the simplest possible architecture of an artificial Neural Network for optimized learning. A descending gradient (GA) criterion has also been introduced to know when to stop performing iterations. Given the importance of estimating software projects, our work aims to cover as many different values of actual efficiency of a wide range of projects as possible by division into clusters and a certain coding method, in addition to the mentioned tools. In this way, the risk of error estimation can be reduced, to increase the rate of completed software projects.

Approach 2: Function Point Analysis (FPA) - The COSMIC FFP

The COSMIC FFP is more effective than traditional FPA for Web effort estimation using an industrial dataset. The results showed that the new prediction model based on estimated COSMIC FFP sizes was more precise. Using popular prediction models based on various artificial intelligence tools and the COSMIC FFP method, it is possible to improve existing estimation processes further. The parameters in our approach represent the input values of the ANN architecture based on the Taguchi Orthogonal Array and are defined as follows:

A. Entry - represent messages that a user sends to the system or that one module sends to another to send data to him. These messages do not have to be system entries.
B. Exit - represent messages that the system or module returns in response. Data can be read from files, be the results of arithmetic operations, and more.
C. Read - represent messages that update the data in the system (i.e. files, tables, etc.).
D. Write - represent messages that read data from systems, tables and files.

Approach 3: Use Case Point Analysis (UCP)

The estimated value is calculated based on G. Karner formulas. UAW (Unadjusted Actor Weight) - this input value is a functional point that can determine the level of complexity of system users. UUCW (Unadjusted Use Case Weight) - this input value is a functional point that can determine the

level of complexity of use cases. TCF (Technical Complexity Factor) is an estimate of the technical complexity of the system and ECF (Environmental Complexity Factor) is one of the factors affecting the size of the project expressed in Use Case points. Finally, AUCP (Adjusted Use Case Point) is the final size of the system expressed in Use Case points and is calculated as follows:
AUCP = UUCPxTCFxECF
Use Case Point Analysis (UCP) is the latest and most accurate method for estimating the effort and cost of realizing software products. This paper will present a new, improved UCP model constructed based on two different artificial neural network (ANN) architectures based on Taguchi Orthogonal Vector Plans. ANNs are an exceptional artificial intelligence tool that has proven to be reliable and stable in this area of software engineering.

In addition, we have set out own criteria, which has been used to compare all the different selected methods. We have also given a score for each evaluation criteria, so that we can compare the different methods numerically for cost estimation. Our observations have shown that it is best to use a number of different estimating techniques or cost models, and then compare the results before determining the reasons for any of the large variations. None of the methods are necessarily better or worse than the others. We found, in fact, that their strengths and weaknesses often complement each other. Therefore, the main conclusion is that there is no one single technique that is best for every situation, and the results of a number of different approaches need to be carefully considered to discover what is the most likely to produce estimates that are realistic.

Ljubomir Lazic received BS, MS, and PhD degrees in Electrical and Computer Engineering from the University of Belgrade, Serbia. He is a professor at the Union University, School of Computing, Belgrade, Serbia. Before he joined the School of Computing, he was an associate professor at the METROPOLITAN University of Belgrade. His research interests include software engineering, software project management, software testing, human computer interaction, and component-based engineering. Current research interests, as a project leader, are in two projects supported in part by the Ministry of Science and Technological Development of the Republic of Serbia under Grant No. TR-1318 (2008–2011) and TR-35026 (2011–2020) involving optimal software project management, software metrics, effort estimation modeling, etc. He is the author of about 140 papers published in international journals, book chapters, and conference proceedings, and invited speaker (keynote speaker at three International Conference on Software QA and Testing on Embedded Systems).