Analysis of Developers Prediction Performance on Software Models Using Back Propagation Algorithm

Sudhir Kumar Singh
Research Scholar, Veer Bahadur Singh Purvanchal University, Jaunpur,
sudhirb26@gmail.com

Abstract: That's what System Effort Estimation (SEE) is all about. Cost and schedule overruns are common in early phases of software development when the erroneous estimate is made. To build software development models, precise time and resource estimates are essential. Software models are typically built using a backpropagation neural network, that is the most prevalent architecture. Elman neural networks can be used in the same way as Back propagation networks. The difference between the predicted and actual effort in a successful prediction system should be as small as possible. Future training process will make use of historical NASA data. Back propagation has been found to be more effective in experiments than in Lstm neural networks. As the selections are made, the inputs' significance shifts. Comparing model test results with the model's predicted effort gives us an idea of the error rate in a model's predictions. With its help, a software project prediction model has been proposed.

Keywords: Elman neural network , Levenberg Marquardt algorithm, feed forward, Input layer, Hidden layer, Undertake layer, and Output layer

Introduction:

Many people find it challenging to estimate how much time it takes to build software. Estimating effort, price, and duration in the early stages of software development helps project managers plan, analyze, and control all resources given. The effectiveness of software project management depends on the ability to accurately assess the time and resources required to finish a project. A project's resources can be overestimated if they are allotted in excess. An underestimation happens when a project is given too few resources. How long it'll take to develop software depends on numerous variables, such as the type of project, how it will be implemented, and how well-versed developers are in their respective fields. As a result, a paradigm for soft computing that can deal with uncertainty can be used. With a good estimation tool, you can get an accurate estimate of the project completion time. With realistic estimations of resources, the manager could plan and manage all tasks more successfully. Most organisations put major resources into developing software, and even a little improvement in prediction performance is probably worth it. An overestimation of costs could have devastating consequences if initiatives are accepted that do not pay off and go over schedule. In the event of an overestimation of costs, it is possible that good proposals would be rejected and projects will be left unfinished. Because of the high stakes of competition, this time wasted can be expensive. A corporation that creates computer software can undoubtedly profit much from better forecasting. Before an estimation model is used effectively, it's important to understand its limitations, independent of the methodology used to develop and implement it.

Neural networks, a sort of computer representation, have their roots in biological study. Basically, they're mathematical idealizations of how we currently think about these fundamental neuronal systems. Neuronal networks have a few defining traits.
(a) Each neuron in the brain does a small computation on the information it receives, then sends the results back to the rest of the system as an output.
(b) Many weighted neuron interconnections are employed to encode the knowledge of the network. For each input/output pair, a weight is assigned.
(c) The network contains a leaning mechanism that generates internal representations automatically. The Levenberg-Marquardt algorithm is one instance of this sort of algorithm..
It is possible to use a sigmoid or cubic operational amplifier.
It is possible to use feed-forward or recurrent networks in prediction applications
(d). A special sort of recurrent neural network known as Levenberg networks and feed-forward networks are the primary topics of this paper. They were more effective than other network models when it came to predicting reliability.

Back Propagation Model:
The backpropagation learning algorithm is a popular practise in neural network architecture. An algorithm for back-propagation learning uses networks called back-propagation networks. Using this method, a neural network can be trained to accurately identify both entry and exit pairs provided during training a network by altering the weights in the neural network. Gradient descent is used to update the weights.

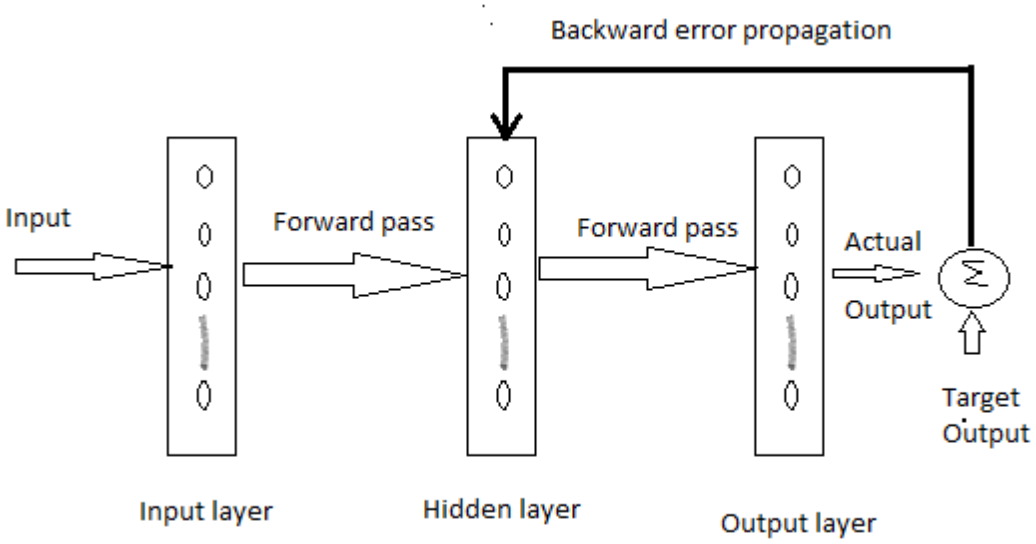

Figure: Back propagation learning algorithm model

An artificial neural network's reactivity and its capacity to produce believable answers in response to input that is similar but not identical is a primary goal while mastering the skill of building an artificial neural network. The backpropagation neural algorithm is used to calculate weights during the learning process, as opposed to other algorithms. The complexity

of the network increases as the deep learning model expands, which is a drawback of a Back propagation method.

We can now develop and test the full backpropagation neural technique using induction to show that it operates in any feedback control networks having discrete kernel function at nodes. The assumption here is that we are dealing with a network that has a single input and a single output. Only one actual input (x) one and network functional (F) are required to form a network. In order to calculate the derivative F(x), you must go through two steps:

x is computed as follows as a "feed back." At each node, primitive functions as well as the dependents of those primitive functions are weighed in. The derivatives are maintained in a safe place.

The output device is fed with constant 1 as well as the network is s new in the opposite way. This is called back propagation. Back propagation. A node's entering data is split by the value stored just on left side of the unit, as well as the result is combined. A cable transmits the result to a unit's left side. It is shown there. The input unit collects the x-dependent derivative of a centralised network.

The procedure is used to determine the derivative of the network function F with respect to the input y. A series, a parallel, and a weighted edge are all supported by the same mechanism. The algorithm assumes a feed forward network with n or less nodes. Look at the diagram depicted in Figure. Functionality of the network The feed-forward step is followed by a single output unit when a form is evaluated at x. Suppose the output unit is connected to m units, each of which outputs F1(x),..., Fm (x). Because the primary purpose of the output unit is to generate output,

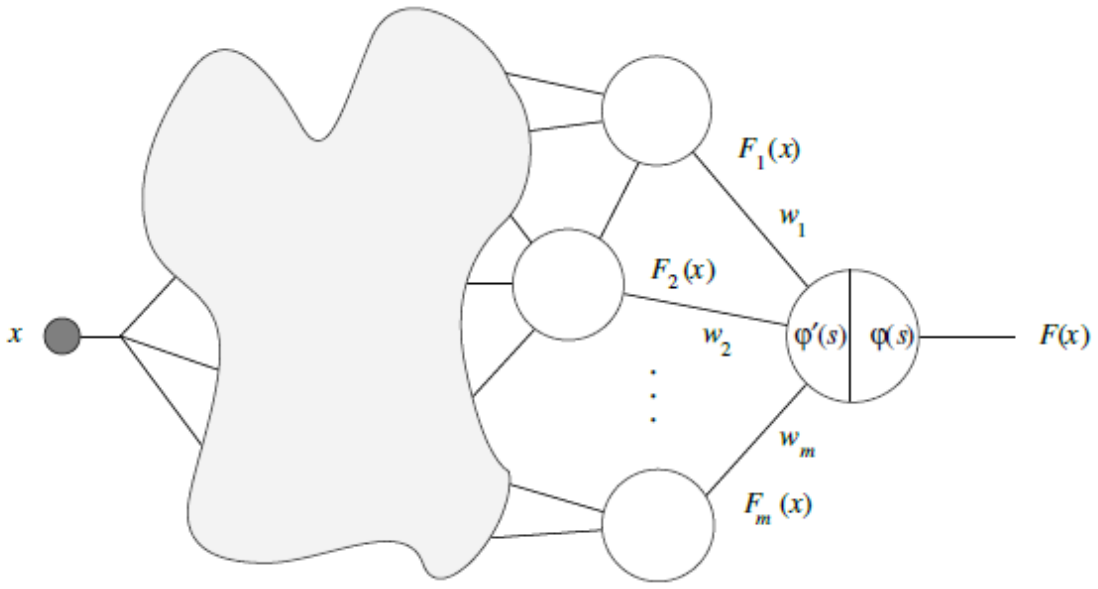

Figure: B diagram

Back propagation at the last node

$$F(x) = \varphi(w_1 F_1(x) + w_2 F_2(x) + \cdots + w_m F_m(x)).$$

The derivative of F at x is thus

$$F'(x) = \varphi'(s)(w_1 F_1'(x) + w_2 F_2'(x) + \cdots + w_m F_m'(x)),$$

The back propagation strategy works as expected in networks with more than one input unit.. When x1 and x2 are sent into the internet, the result of the internet backbone can be alluded to as F. (x1, x2). So because making a connection already has two parameters, we can compute the derivative of F with respect to either x1 or x2. The feed back step has not changed; the left-side slots of the units are still utilised as is normal. Both input units 1 and 2 are connected to each other in the back propagation stage, but only one is connected to each of the output units. In the first subnetwork, backpropagation neural stage is used to derive all partial derivatives of F w with respect to x1. The second subnetwork's backpropagation neural step provides additional second derivative of F w with respect to x2 just at second unit of input. Think about how we can combine these calculations in one step to perform a single back - propagation neural step so over entire network. The results are the same as they always have been.

Elman Neural Network:

Power systems with moment features in the mechanisms can adapt to dynamic systems' moment recognition or process forecasting issues by using a Lstm neural network. Because of its inherent dynamic and memory features, this model has been shown through simulation results to improve prediction reliability. Forecasting abilities are greater in Elman neural network. In dynamic systems, it is easier to fine-tune forecasting and regression algorithms in real time.

The Elman neural network integrates a time delay operators into the hidden units of circuit configuration so that the system could be characterised by temporal variation and much more global stability based on the research done in 1990 by Elman on rear neural networks. Some of the disadvantages of backpropagation neural neural networks are their tendency to fall into local minima and their fixed rate of learning. The Lstm neural network probably inherited these problems. Many sectors have used the Lstm neural network since its debut more than two decades ago. For all their ease in addressing complex issues, neural networks aren't perfect due to a variety of factors, including the sample and algorithm or structure of the network. When working with actual data, gathering as much feature data as possible is just as critical as it is when working with samples. Numerous details are available, but this also increases the complexity for neural networks to deal with this inputs; in other words, crucial information is continually disguised in redundant data.... It is possible that the samples hold evidence that is associated or even repeated, necessitating a significant quantity of storage space and computer time. The physical configuration of the Elman neural network consists of four layers: input, hidden units, undertaking layer, the output vector. The output of a hidden layer is stored in memory using the undertake layer. Since it is constructed on a back - propagation algorithm, the undertake layer's latency and memory relate the hidden gradient output with its input.

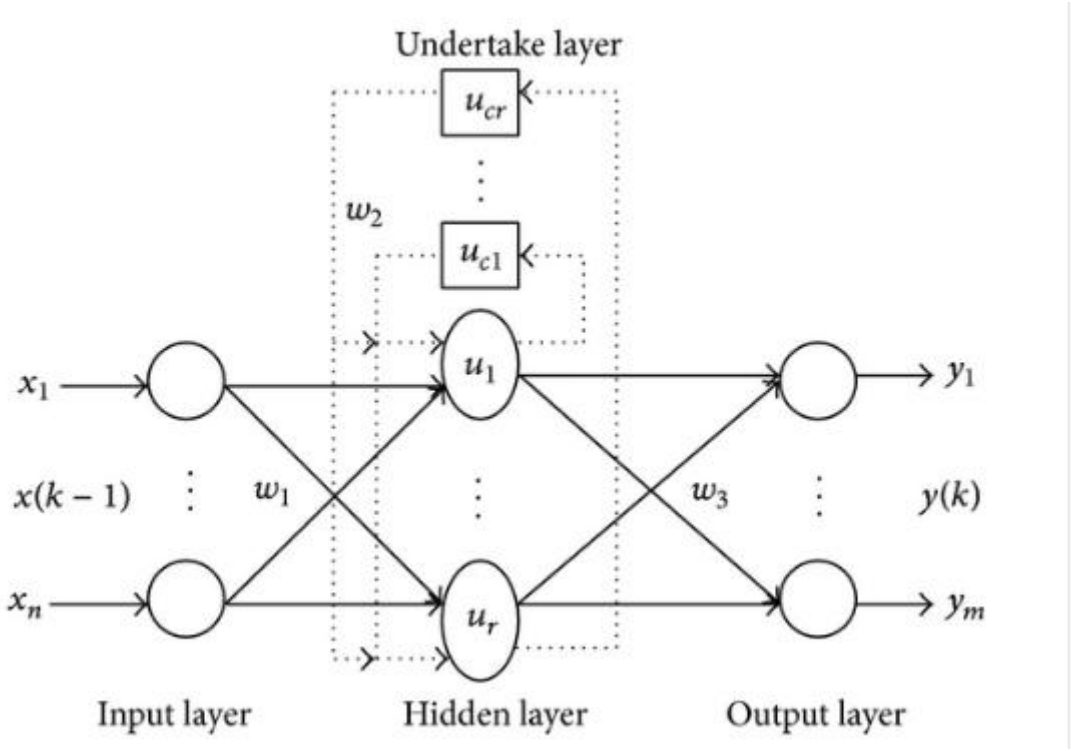

Figure: Topology Structure of Elman Neural Network

It is important to remember that the topology consists of four layers: input, hidden, active, and output layers. A step-delayed operator, the Undertake layer, receives the output of the concealed layer. The delay and storage supplied by the bottom layer link the hidden output to its input using a back - propagation neural network. Internal current control networks can improve an organization's capacity to keep up with the constant flow of new data. It is possible for the system to react to changing features over time if the internal information is remembered.

References:

[1] D. Eck, B. Brundick, T. Fettig, J. Dechoretz and J. Ugljesa, "Parametric estimating handbook," The International Society of Parametric Analysis (ISPA), Fourth Edition. 2009.
[2] Baker, D. R. (2007). A hybrid approach to expert and model based effort estimation: Citeseer.Baughman, D. R., & Liu, Y. A. (2014). Neural networks in bioprocessing and chemical engineering: Academic press.
[3] Boehm, B. (1981).Software engineering economics: Prentice-Hall, Englewood Cliffs, NJ.
[4] J. Lynch. Chaos manifesto. The Standish Group. Boston. 2009[Online]. Available: http://www.standishgroup.com/newsroom/chaos_2009.php.
[5] M. Jorgensen and M. Shepperd, "A Systematic Review of Software Development Cost Estimation Studies,"IEEE Transactions on Software Engineering, vol. 33(1), pp. 33-53, 2007.
[6] E. Mendes, N. Mosley and I. Watson, "A comparison of case-based reasoning approaches," in Proceedings of the 11th International Conference on World Wide Web, Honolulu, Hawaii, USA, 2002, pp. 272-280.

[7] Costagliola, G., Ferrucci, F., Tortora, G., & Vitiello, G. (2005). Class point: an approach for the size estimation of object-oriented systems. IEEE Transactions on software engineering,31(1), 52-74.
[8] M. Jørgensen,"Forecasting of software development work effort: Evidence on expert judgment and formal models," International Journal of Forecasting,vol.23(3),pp.449-462, 2007.
[9] B. W. Boehm, Software Engineering Economics. Prentice-Hall, 1981.
[10] D. D. Galorath and M. W. Evans, Software Sizing, Estimation, and Risk Management. Boston, MA, USA: Auerbach Publications, 2006.
[11] L. H. Putnam, "A General Empirical Solution to the Macro Software Sizing and Estimating Problem,"IEEE Transactions on Software Engineering, vol.4(4),pp.345-361, 1978.
[12] Heiat, A. (2002). Comparison of artificial neural network and regression models for estimating software development effort. Information and Software Technology, 44(15), 911-922.
[13] Higham, D. J., & Higham, N. J. (2000). MATLAB guide: Philadelphia: SIAM.
[14] C. Lopez-Martin, "A fuzzy logic model for predicting the development effort of short scale programs based upon two independent variables," Applied Soft Computing, vol. 11(1), pp. 724-732, 1, 2011.
[15] A. B. Nassif, D. Ho and L. F. Capretz, "Towards an Early Software Estimation Using Log-linear Regression and a Multilayer Perceptron Model," Journal of Systems and Software, vol. 86(1), pp. 144-160, 2013.
[16] A. B. Nassif, L. F. Capretz and D. Ho, "Estimating software effort using an ANN model based on use case points," in 11th International Conference on Machine Learning and Applications (ICMLA 2012), Boca Raton, Florida, USA, 2012, pp. 42-47.
[17] A. B. Nassif, L. F. Capretz and D. Ho, "Estimating software effort based on use case point model using sugeno fuzzy inference system," in 23rd IEEE International Conference on Tools with Artificial Intelligence, Florida, USA, 2011, pp. 393-398.
[18] Lopez-Martin, C., Isaza, C., & Chavoya, A. (2012). Software development effort prediction of industrial projects applying a general regression neural network. Empirical SoftwareEngineering, 17(6), 738-756.
[19] A. B. Nassif, L. F. Capretz and D. Ho, "A regression model with mamdani fuzzy inference system for early software effort estimation based on use case diagrams," in Third International Conference on Intelligent Computing and Intelligent Systems, Guangzhou, Guangdong, 2011, pp. 615-620.
[20] A. Idri and A. Abran, "COCOMO cost model using fuzzy logic," in 7th International Conference on Fuzzy Theory and Technology, 2000, pp. 1-4.
[21] Boehm, B., Abts, C., Brown, A., Chulani, S., Clark, B., Horowitz, E., . . . Steece, B. (2000). Software cost estimation with COCOMO II. Prentice Hall PTR. Upper Saddle River, NJ.
[22] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," Advances in Neural Information Processing Systems, vol. 2pp. 524-532, 1990.

[23] S. Chen, X. Hong and C. J. Harris, "Orthogonal forward selection for constructing the radial basis function network with tunable nodes," in IEEE International Conference on Intelligent Computing, 2005, pp. 777-786.
[24] X. Huang, D. Ho, J. Ren and L. F. Capretz, "Improving the COCOMO model using a neuro-fuzzy approach," Applied Soft Computing, vol. 7(1), pp. 29-40, 2007.
[25] W. L. Du, L. F. Capretz, A. B. Nassif and D. Ho, "A Hybrid Intelligent Model for Software Cost Estimation," Journal of Computer Science, vol. 9(11), pp. 1506-1513, 2013.
[26] MacDonell, S. G., & Shepperd, M. J. (2003). Combining techniques to optimize effort predictions in software project management.Journal of Systems and Software,66(2),91-98.
[27] Mashrei, M. A., Abdulrazzaq, N., Abdalla, T. Y., & Rahman, M. (2010). Neural networks model and adaptive neuro-fuzzy inference system for predicting the moment capacity of ferrocement members. Engineering Structures, 32(6), 1723-1734.
[28] Nassif, A. B., Azzeh, M., Capretz, L. F., & Ho, D. (2016). Neural network models for software development effort estimation: a comparative study. Neural Computing and Applications, 27(8), 2369-2381.
[29] Papatheocharous, E., & Andreou, A. S. (2010). On the problem of attribute selection for software cost estimation: Input backward elimination using artificial neural networks. Paper presented at the IFIP International Conference on Artificial Intelligence Applications and Innovations.
[30] Shepperd, M., & Schofield, C. (1997). Estimating software project effort using analogies. IEEE Transactions on software engineering, 23(11), 736-743.
[31] Mittas, N., & Angelis, L. (2008). Combining regression and estimation by analogy in a semiparametric model for software cost estimation. Paper presented at the Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement.
[33] R. Rojas: Neural Networks, Springer-Verlag, Berlin, 1996
[34] Weikuan Jia, Dean Zhao, Tian Shen, Yuyang Tang, and Yuyan Zhao, Study on Optimized Elman Neural Network Classification Algorithm Based on PLS and CA, Research Article, Open Access Volume 2014, Article ID 724317