International Journal of Artificial Intelligence and Neural Networks– IJAINN
Volume 5: Issue 2 [ISSN : 2250-3749]
Publication Date : 30 October, 2015

Neural Network Models for Agile Software Effort Estimation based on Story Points

Aditi Panda, Shashank Mouli Satapathy, Santanu Kumar Rath

***Abstract*— Agile software development is now accepted as a superior alternative to conventional methods of software development, because of its inherent benefits like iterative development, rapid delivery and reduced risk. Hence, the industry must be able to efficiently estimate the effort necessary to develop projects using agile methodology. For this, different techniques like expert opinion, analogy, disaggregation etc. are adopted by researchers and practitioners. But no proper mathematical model exists for this. The existing techniques are ad-hoc and are thus prone to be incorrect. One popular approach of calculating effort of agile projects mathematically is the Story Point Approach (SPA). In this study, an effort has been made to improve the prediction accuracy of estimation done using SPA. For doing this, different types of neural networks (General Regression Neural Network (GRNN), Group Method of Data Handling (GMDH) Polynomial Neural Network and Cascade-Correlation Neural Network) are used. Finally, performance of models generated using these neural networks are compared and analyzed.**

***Keywords*— Agile Software Development, General Regression Neural Network, GMDH Polynomial Neural Network, Cascade Correlation Neural Network, Software Effort Estimation, Story Point Approach.**

I. Introduction

Agile methods are used for developing software to enable organizations respond to requirements volatility. These methods provide opportunities to assess the direction of software development throughout the development life cycle [1]. By emphasizing on the repetition of work cycles along with product the teams yield an additive and iterative development. Instead of promising to market an ensemble software that hasn't been developed, agile authorizes teams to repetitively replan their releases to optimize their value throughout the development [2]. Thus, companies following agile methods give competition to others in the marketplace that don't use agile methods.

Since predictability is the primary goal of project management, we need to be able to estimate the size and complexity of the products to be built in order to determine what to do next [3], [4]. For this, requirements need to be collected. Requirements in agile development are jotted down in cards and are called user stories. These stories are estimated using story points. The team defines the relationship between story point and effort. Usually 1 story point is equal to 1 ideal working day. Total no. of story points that a team can convey in a sprint (an iteration in agile software development) is called as ``team velocity" or story points per sprint. In the SPA, aggregate number of story points alongside velocity of the project are considered to determine agile software development effort. Now for obtaining better prediction accuracy, three types of neural networks are used in this study. The results obtained by applying these networks are compared and their performance is accessed

Aditi Panda, Shashank Mouli Satapathy, Santanu Kumar Rath *(Authors)*
Department of Computer Science and Engineering,
National Institute of Technology, Rourkela, India

II. Related Work

Coelho et al. [5] described the steps followed in story point-based method for effort estimation of agile software and highlighted the areas which need to be looked into further research. Andreas Schmietendorf et al. [2] provided an investigation about estimation possibilities, especially for the extreme programming paradigm. Ziauddin et al. [6] developed an effort estimation model for agile software projects. The model was fine-tuned with the help of the empirical data acquired from twenty one software projects. The test results demonstrate that model has great estimation exactness regarding MMRE and PRED(n). Satapathy et al. [7] used SVR-Kernel methods for optimizing the effort calculated using story point approach. Out of the four kernels used, the RBF-Kernel is shown to have best accuracy. Popli et al. [8] proposed a model for effort and cost estimation in agile software development by using regression analysis. This framework is suitable to be used for project planning, execution and monitoring efficaciously. Hussain et al. [9] made an attempt to propose an approach which helps in removing problems like formalized user requirements and thus apply function points for agile software effort estimation. Hamouda et al. [10] introduced a process and methodology that guarantees relativity in software sizing while using agile story points.

P. Reddy et al. [11] proposed software effort estimation models based on Radial Basis Function Neural Network (RBFN) and GRNN. Results proved that RBFN performs better than GRNN. P. S. Rao et al. [12] proposes a GRNN to utilize improved software effort estimation for COCOMO dataset. The GRNN model is then compared with various other techniques, i.e., M5, Linear regression, SMO Polykernel and RBF kernel. Fahlman et al. [13] first proposed the cascade-correlation learning architecture for artificial neural networks. They are self-organizing networks, starting only with input and output neurons and adding hidden-layer neurons during the training process. The GMDH neural networks are illustrated by Farlow et al. in [14] but they were first developed in the Combined Control Systems (CCS) group of the Institute of Cybernetics in Kyiv (Ukraina) in 1968. They are also self-organizing, starting only with input neurons and adding hidden units during training based on some criteria.

III. Evaluation Criteria

The performance of the various models is accessed by employing criteria outlined below [15]:

- The **Mean Square Error (MSE)** is calculated as:

$$MSE = \frac{\sum_{i=1}^{TD}(AE_i - PE_i)^2}{TD} \quad (1)$$

Where AE_i = Actual Effort of ith test data, PE_i = Predicted Effort of ith test data and TD = Total number of data in the test set.

- The **Squared Correlation Coefficient (R^2)**, otherwise called as the *coefficient of determination* is calculated as:

$$R^2 = 1 - \frac{\sum_{i=1}^{TD}(AE_i - PE_i)^2}{\sum_{i=1}^{TD}(AE_i - \overline{AE})^2} \quad (2)$$

where AE = Mean of Actual Effort Value.

- The **Mean Magnitude of Relative Error (MMRE)** is calculated by summing MRE values over N observations as shown below:

$$MMRE = \sum_{1}^{TD} \frac{|AE_i - PE_i|}{AE_i} \quad (3)$$

- The **Prediction Accuracy (PRED)** is calculated as:

$$PRED = \left(1 - \left(\frac{\sum_{i=1}^{TD}|AE_i - PE_i|^2}{TD}\right)\right) * 100 \quad (4)$$

IV. Proposed Work

The proposed approach is implemented using the twenty one project data set developed by six software houses [6]. The data set is three-dimensional. The first dimension indicates the number story points required to complete the project, the second represents the velocity of the project, and the third represents the actual effort required to complete that project. This data is used by [6] for predicting effort using regression. In this study, in order to enhance the effort estimation accuracy, neural networks are employed. The block diagram, demonstrated in figure 1, shows the proposed steps used to compute the predicted effort using various neural networks.

The steps taken to determine the effort of a software product are described below:

1. **Collection of Total Number of Story Points, Project Velocity and Actual Effort**: The total number of story points, project velocity values and actual effort are collected from [6].

2. **Normalization of Data Set**: In this step, the total number of story points and project velocity values are

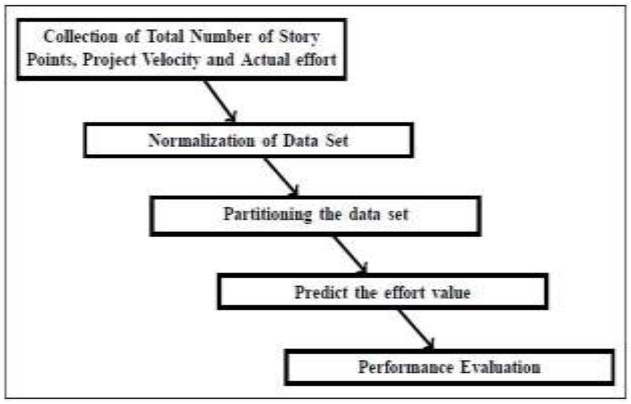
Collection of Total Number of Story Points, Project Velocity and Actual effort
Normalization of Data Set
Partitioning the data set
Predict the effort value
Performance Evaluation

Figure 1. Proposed Steps to Apply Neural Network Models for Agile Software Effrot Estimation

normalized to the range [0,1]. Let Y be the data set and y is an element of the data set, then the normalized value of y can be calculated as:

$$y' = \frac{y - \min(Y)}{\max(Y) - \min(Y)} \quad (5)$$

where y' = Normalized value of y in the range [0,1],
min(Y)$ = min. value of Y and
max(Y)$ = max. value of Y.
When max(Y) = min(Y), y' = 0.5

3. **Partitioning the Data Set**: The data set is partitioned into training set, validation set and test set.

4. **Predict the effort value**: The GRNN algorithm is run to get effort values. No network building is involved as everything is decided before making the effort prediction. In case of self-organizing networks, the values of learning parameters are assigned and the algorithm is run until an optimum network configuration is obtained.

5. **Performance Evaluation**: In this step, the models are accessed in order to evaluate their performance using MSE, R^2, MMRE, and Prediction Accuracy (PRED) values obtained using test samples. The model giving lower estimations of MSE, R^2, MMRE and higher estimations of PRED is considered as best model.

After the neural network implementation is completely done, the results obtained are compared to see which network performs better. A comparison with the results of models existing in literature is also done to see if the proposed work improved the prediction accuracy.

V. Experimental Details

For implementing the proposed approaches, the data set given in [6] is used. The detailed description about the data set has as of now been given in the proposed approach section. The inputs to the neural network models are total number of story points and project final velocity and the output is the effort i.e., the completion time. All the models are tested and validated using leave-one-out validation for achieving better accuracy.

International Journal of Artificial Intelligence and Neural Networks– IJAINN
Volume 5: Issue 2 *[ISSN : 2250-3749]*
Publication Date : 30 October, 2015

A. *Model Design Using General Regression Neural Networks*

General Regression Neural Networks (GRNN) were first proposed by Donald F. Specht [16] in 1991. It consists of four layers. Input layer comprises of one neuron for every input variable. The yields of this layer are sustained to all the hidden layer neurons. The hidden layer has the same number of neurons as there are lines in the training set. Hidden neurons find the Euclidean distance of the input from the center of the neuron and afterward apply the RBF kernel function. Then, the result obtained from this is given to the pattern layer, which has two neurons, namely the numerator summation unit and the denominator summation unit. Results obtained from all the hidden layer neurons are summed up and given to the denominator summation unit. For the numerator summation unit, the yields from hidden layer neurons are multiplied by the actual effort and after that summed up. The last layer (Decision layer) divides the estimation of the numerator summation unit by the estimation of the denominator summation unit. This value is the estimated effort value for that input. For improving the accuracy, unnecessary neurons are removed from the network and then the network is retrained (called model optimization and simplification). Three different criteria can be used for carrying out this i.e., Minimize error, Minimize number of neurons, Fix number of neurons.

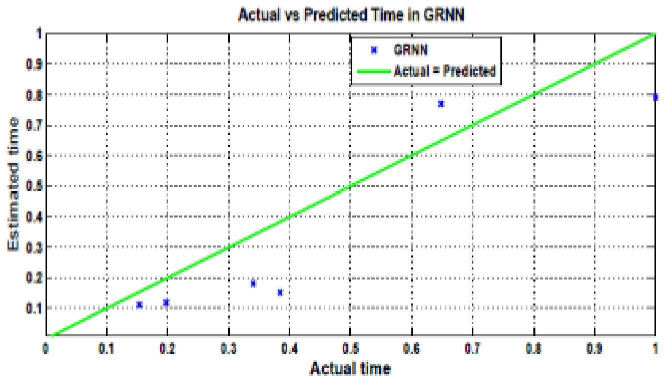

Figure 2. Effort Estimation Model using GRNN

Figure 2 shows the deviation of predicted effort value from actual effort for General Regression Neural Networks. Nearer the points to the actual = predicted line in the figure, better is the prediction accuracy. In this study, the following values are used:

- Type of Kernel function: Reciprocal
- Minimum Sigma: 0.001
- Maximum Sigma: 0.5
- Model optimization and simplification Criteria: Minimize the error

The minimum and maximum values of sigma were chosen by taking the suitable combinations of values to generate best possible result. The values which gave the best results were chosen for testing. The network is trained using different sigma values (from 0.001 to 0.5) and the value of sigma which gives the best accuracy and least error is chosen as the final sigma value.

B. *Model Design Using GMDH Polynomial Neural Networks*

GMDH networks date back to 1968 (first originated by Prof Alexey G. Ivakhnenko [17]). They are self-organizing networks i.e., the network's connections are not defined initially; instead, they are determined when the network is trained, with the objective of optimizing the network. Maximal accuracy is obtained by restricting the number of layers to be added to the network. This also helps prevent over-fitting. Initially a GMDH network starts with input layer only. This layer contains one unit for each input variable. The successive layer neurons derive their inputs from any two units of the previous layer. The final layer i.e., the output layer derives two of its inputs from the previous layer and gives rise to one value (output of the network). Neurons in any layer can take input(s) from any of the previous layers.

Two variable-quadratic polynomials are used as transfer functions in the neurons. These polynomials have the form:

$$y = q0 + q1 \times r1 + q2 \times r2 + q3 \times r1^2 + q4 \times r2^2 + q5 \times r1 \times r2 \qquad (6)$$

The training data is divided into two parts: training data and control data. The latter is used to halt the training process as soon as over-fitting starts.

The algorithm adopted in this network works as follows: The transfer functions are used for defining all possible functions by combining inputs from the previous layer. Then least squares regression is applied for calculating the optimal parameters for the transfer function in each neuron of the candidate list so that it best fits the training data. Then for each neuron, the mean squared error(MSE) value is calculated by applying it to the control data. Finally the candidate neurons are sorted in the increasing order of error and the neurons with least error are picked up to be used in the next layer. In this work, following values are used for network parameters:

- Maximum no. of layers = 5
- Number of neurons per layer = 10 (fixed)
- Control data amount = 20\% of training data
- Network layer connections: Connections from the former layer as well as the input layer are allowed.

The following functions are used:

- Linear:
  $$y = q1 + q2 \times r1 + q3 \times r2 + q4 \times r3 \qquad (7)$$
- Quadratic:
  $$y = q1 + q2 \times r1 + q3 \times r1^2 + q4 \times r2 + q5 \times r2^2 + q6 \times r1 \times r2 \qquad (8)$$

The parameters are self-explanatory. Values for parameters in this as well as the next model are fixed by choosing the suitable combinations of values to generate best possible result, i.e. values which yield best results are chosen for testing.

International Journal of Artificial Intelligence and Neural Networks– IJAINN
Volume 5: Issue 2 [ISSN : 2250-3749]
Publication Date : 30 October, 2015

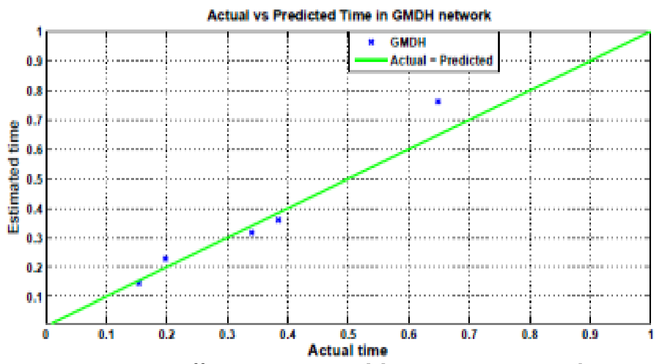

Figure 3. Effort Estimation Model using GMDH Network

Figure 3 shows the deviation of predicted effort value from actual effort for GMDH polynomial neural network. Nearer the points to the actual = predicted line in the figure, better is the prediction accuracy.

C. *Model Design Using Cascade Correlation Neural Networks*

Cascade correlation neural networks [13] are also “self-organizing”. These networks initially have only an input layer and an output layer. While training the network, neurons are chosen from a list of neurons and included in the hidden layer. The new neurons get their inputs from all the existing neurons of the network, hence it is called cascade. The training algorithm attempts to increase the amount of correlation between the newly added neurons and the network's residual error(which has to be minimized).

In this study, Gaussian kernel function is used. The following network training parameters are used:

- Minimum neurons = 1
- Maximum neurons = 8
- Candidate Neurons = 8
- Maximum steps without improvement = 10
- Over-fitting protection control = 3-fold cross validation

Candidate neurons are the list of neurons from which hidden units are successively chosen and added to the network. Minimum number of neurons show how many hidden neurons must be added to the network. Maximum number of neurons shows the maximum number of hidden neurons that can be added to the network. It is same as the number of candidate neurons, we can add one or more (even all) the candidate neurons to the network. The training process is continued till the results don't improve for a certain number of times, denoted by the maximum steps without improvement parameter. Cascade-correlation networks have this disadvantage of over-fitting the training data. Due to this, the accuracy values are quite high in case of training data, but low in testing data. So, for preventing this, the model is validated as it grows (3-fold cross validation is used). As stated earlier, the values which yield the best results are taken for testing.

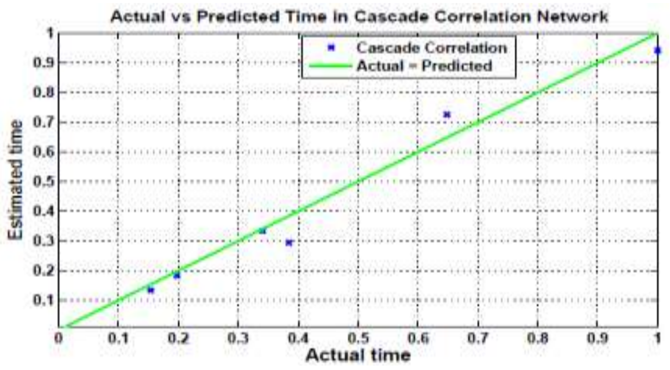

Figure 4. Effort Estimation Model using Cascade Correlation Network

Figure 4 shows the deviation of predicted effort value from actual effort for cascade correlation neural network. Nearer the points to the actual = predicted line in the figure, better is the prediction accuracy.

D. *Comparison*

On comparing the results, it is observed that cascade correlation network performs better (gives better values of MSE, R^2, MMRE and PRED) than other networks.

TABLE I. COMPARISON OF PROPOSED RESULTS WITH EXISTING WORK

	Proposed Work (Best)	Proposed Work (Worst)	Regression [6]
PRED (%)	94.76	85.92	57.14

In table I, our results have been compared with the existing work in literature. It can be seen that the neural networks used in this study performed better than the existing method [6].

Table II demonstrates the comparison of MSE, R^2, MMRE and PRED values for three different types of neural networks. GRNN performs the prediction in an off-line manner i.e., there is no real training of the network. The network parameters don't get optimal values, so the accuracy of prediction is lower than others. The self-organizing networks perform well because proper training is done for obtaining optimal network configuration with the objective of achieving maximum accuracy. Thus, the network built at the end of training best fits the data set and yields high values of accuracy when used on testing. Among the two self-organizing networks, Cascade-Correlation network performs better. The learning process in Cascade-Correlation network is quick. The size and topology of the network are decided by the network itself. Moreover, error signals need not be back-propagated through the network connections and the training is quite robust. The main advantage of GMDH networks is that they find weights using least squares fitting, which guarantees locally good weights. Back-propagation may be applied for obtaining globally-optimal weights. GMDH networks have this demerit of searching a polynomial network structure by only exploiting small neighbourhoods. Alternative nodes are discarded when growing the network and not considered later in the learning process, which leaves chance for sub-optimal network structure, and hence the lesser prediction accuracy. But this particular problem doesn't exist in cascade-correlation networks.

TABLE II. COMPARISON OF RESULTS OBTAINED USING THREE DIFFERENT TYPES OF NEURAL NETWORKS

	MSE	R^2	MMRE	PRED (%)
General Regression Neural Network	0.0244	0.7125	0.3581	85.9128
GMDH Polynomial Neural Network	0.0317	0.6259	0.1563	89.6689
Cascade Correlation Neural Network	0.0059	0.9303	0.1486	94.7649

VI. Threats to Validity

There are some areas which still remain un-addressed. This section aims to highlight those.

- All the effort estimation models proposed in this work have been developed by assuming that the initial project velocity value is given. This value is taken from the past projects developed by the same team in similar working conditions. But when a team is new, the company won't be having any past record for it. In that case, no clear assignment to initial project velocity can be done. However, the average velocity value of all the teams working in similar conditions and with nearly same size can be taken and assigned to initial project velocity and different neural network models can be used to estimate the effort.
- In this study, data from [6] is used for implementation. It has records of twenty-one projects developed by six software houses. One shortcoming is there is no information about the type of projects taken for study. If it contains data of only one type of software, then the results of this work also apply to that particular type of software. For our results to be valid for the general software engineering paradigm, the work should be based on data which covers all categories of software developed by agile methods.
- Due to small size of dataset, the testing data is very small in size. The data set size is 21 × 3, hence not much data is left for testing. Only six data points are used for testing. Thus optimal accuracy of the model's performance cannot be guaranteed..

VII. Conclusion

Story Point Approach (SPA) is one of the effort estimation models that can be applied to estimate the effort required to develop a software using agile methodology. In this study, first the total number of story points and velocity of the project are considered in order to estimate the effort required during development of agile softwares. The resultant values are then optimized using various neural networks such as GRNN, GMDH networks and cascade networks. Toward the end of the study, the results obtained from various neural network models are compared in order to measure their performance. It is seen that cascade network outperformed other networks. The computations for above methodologies were executed, and results were obtained using MATLAB. Extension to this procedure might be made by implementing other machine learning techniques such as Stochastic Gradient Boosting (SGB), Random Forest etc. along with SPA.

References

[1] M. Fowler and J. Highsmith, "The agile manifesto," *Software Development*, vol. 9, no. 8, pp. 28–35, 2001.

[2] A. Schmietendorf, M. Kunz, and R. Dumke, "Effort estimation for agile software development projects," in *5th Software Measurement European Forum*, 2008, pp. 113–123.

[3] S. M. Satapathy, M. Kumar, and S. K. Rath, "Fuzzy-class point approach for software effort estimation using various adaptive regression methods," *CSI Transactions on ICT*, vol. 1, no. 4, pp. 367–380, 2013.

[4] S. M. Satapathy, B. P. Acharya, and S. K. Rath, "Class point approach for software effort estimation using stochastic gradient boosting technique," *ACM SIGSOFT Softw. Eng. Notes*, vol. 39, no. 3, pp. 1–6, Jun. 2014.

[5] E. Coelho and A. Basu, "Effort estimation in agile software development using story points," *development*, vol. 3, no. 7, 2012.

[6] Z. K. Zia, S. K. Tipu, and S. K. Zia, "An effort estimation model for agile software development," *Advances in Computer Science and its Applications*, vol. 2, no. 1, pp. 314–324, 2012.

[7] S. M. Satapathy, A. Panda, and S. K. Rath, "Story point approach based agile software effort estimation using various svr kernel methods," in *2014 International Conference on Software Engineering & Knowledge Engineering, July 1-3, 2014, Hyatt Regency, Vancouver, Canada*, 2014, pp. 304-307.

[8] R. Popli and N. Chauhan, "Cost and effort estimation in agile software development," in *Optimization, Reliabilty, and Information Technology (ICROIT), 2014 International Conference on*. IEEE, 2014, pp. 57–61.

[9] I. Hussain, L. Kosseim, and O. Ormandjieva, "Approximation of cosmic functional size to support early effort estimation in agile," *Data & Knowledge Engineering*, vol. 85, pp. 2–14, 2013.

[10] A. E. D. Hamouda, "Using agile story points as an estimation technique in cmmi organization," in *Agile Conference (AGILE), 2014*. IEEE, 2014, pp. 16-23.

[11] P. Reddy, K. Sudha, P. R. Sree, and S. Ramesh, "Software effort estimation using radial basis and generalized regression neural networks," *arXiv preprint arXiv:1005.4021*, 2010.

[12] B. T. Rao, B. Sameet, G. K. Swathi, K. V. Gupta, C. RaviTeja, and S. Sumana, "A novel neural network approach fo r software cost estimation using functional link artificial neural network (flann)," *International Journal of Computer Science and Network Security*, vol. 9, no. 6, pp. 126-131, 2009.

[13] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," Computer Science Department, Carnegie Mellon University, Tech. Rep., 1989.

[14] S. J. Farlow, *Self-organizing methods in modeling: GMDH type algorithms*. CrC Press, 1984, vol. 54.

[15] T. Menzies, Z. Chen, J. Hihn, and K. Lum, "Selecting best practices for effort estimation," *Software Engineering, IEEE Transactions on*, vol. 32, no. 11, pp. 883–895, 2006.

[16] D. F. Specht, "A general regression neural network," *Neural Networks, IEEE Transactions on*, vol. 2, no. 6, pp. 568–576, 1991.

[17] A. Ivakhnenko, "Heuristic self-organization in problems of engineering cybernetics," *Automatica*, vol. 6, no. 2, pp. 207–219, 1970.