Faculty of Informatics and Computer Science

Software Engineering

Effort Estimation of agile software using Artificial Neural Network

By: Ramy Mamdouh ID:118371

Supervised By

DR.Abeer Hamdy

June 2016

Abstract

With the increase of novel technologies and software engineering life cycle methods, Software products have become more complex and the need for more researches on developing new and creative methods for estimation process has dramatically increases. Estimation of software development effort is the key assignment for an effective and efficient management through all software industries. Moreover in the Recent years' success or failure of a project depends heavily on the correctness of cost and effort estimations. Agile methodology is one of the new methods that have been presented lately in the software industry market. Agile development model is a type of Incremental model that provides inherent benefits as rapid delivery, Regular adaptation and reduced risk .The existence of agile methodology in the market has created several opportunities and challenges as well. One of the previously mentioned challenges is the effort and cost estimation for an agile development cycle. This research aims to construct and develop effort estimations models for agile software by the use of various types of neural networks. The types that will be used in neural networks are Cascade forward back propagation Neural Network, Feed-Forward Neural Network and recurrent neural networks. Story Point Approach will be used for agile software effort model.

Keywords: Agile software development, software effort estimation, Story point approach, feed forward neural network, Recurrent Neural Network, Cascade Forward Neural Network, Neural Networks

Turntin Report

turnitin

Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

The first page of your submissions is displayed below.

Submission author:	**Ramy Assaad**
Assignment title:	**Graduation Thesis**
Submission title:	**Effort Estimation of agile software..**
File name:	**Final_essay.docx**
File size:	**931.23K**
Page count:	**56**
Word count:	**8,077**
Character count:	**49,629**
Submission date:	**13-Jun-2016 06:29AM**
Submission ID:	**683612940**

Effort Estimation of agile software using Artificial Neural Network

ORIGINALITY REPORT

11%	5%	8%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Krenker, Andrej, Janez Bester, and Andrej Kos. "Introduction to the Artificial Neural Networks", Artificial Neural Networks - Methodological Advances and Biomedical Applications, 2011. Publication	2%
2	Panda, Aditi, Shashank Mouli Satapathy, and Santanu Kumar Rath. "Empirical Validation of Neural Network Models for Agile Software Effort Estimation based on Story Points", Procedia Computer Science, 2015. Publication	1%
3	arxiv.org Internet Source	1%
4	digitalcollections.americanancestors.org Internet Source	1%
5	Saeedi, Ehsan, Md Selim Hossain, and Yinan Kong. "Side-Channel Information Characterisation Based on Cascade-Forward Back-Propagation Neural Network", Journal of	1%

	Electronic Testing, 2016. Publication	
6	ethesis.nitrkl.ac.in Internet Source	1%
7	www.inderscience.com Internet Source	<1%
8	PlantOmics The Omics of Plant Science, 2015. Publication	<1%
9	Usman, Muhammad, Emilia Mendes, Francila Weidt, and Ricardo Britto. "Effort estimation in agile software development : a systematic literature review", Proceedings of the 10th International Conference on Predictive Models in Software Engineering - PROMISE 14, 2014. Publication	<1%
10	Kaushik, Anupama, Ashish Chauhan, Deepak Mittal, and Sachin Gupta. "COCOMO Estimates Using Neural Networks", International Journal of Intelligent Systems and Applications, 2012. Publication	<1%
11	Oliveira, A.L.I.. "GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation", Information and Software Technology, 201011 Publication	<1%

Publication

EXCLUDE QUOTES	ON	EXCLUDE MATCHES	OFF
EXCLUDE BIBLIOGRAPHY	ON		

Acknowledge

I would like to thank my guide DR. Abeer .She has always stood by my side even when I changed my thesis several times. DR Abeer is a fair teacher, who encourages you and helps every student to be on the right track. I am so thankful to have DR Abeer as my guide doctor as she always continue to support me.

Contents

1. Introduction

1.1 Overview

1.1.1 Agile Software Development Overview

Agile Software Development Methods have been introduced lately in the market due to the inefficiency and failure of other traditional software development methods (Taghi Javdani) .In June 2001, Seventeen of professional software engineers and agile leaders have been united together to introduce some ASD principles in a formal way (Taghi Javdani).Although agile concepts and theories were not new but it was the first time to capture and integrate them into one individual document that are known nowadays as "agile manifesto". Agile manifesto document was based on simple yet effective ideas such as:

- Individuals and interaction over process and tools
- Working and efficient software over a lot of useless comprehensive documentation
- Customer partnership over contract negotiation
- Responding to user requirements change over following a plan.

Agile Software developments methods such as: Scrum and extreme programming are primarily based on reducing the cost of change and therefore decreasing the overall cost. Moreover, All of Agile methods tend to achieve "short release cycles, simple design, continuous testing, refactoring, collective ownership, coding standard and continuous integration" (Schmietendorf, Kunz, & Dumke, 2008).The continuous change in requirements by customers is overwhelming and hence traditional software development methodologies such as waterfall life cycle are not suitable for the current situation as they are not flexible and do not adapt to new or modified requirements. Thus agile has come to the front page of Software development methods (Panda, Effort Estimation of Agile and Web-based Software using Artificial Neural Networks, 2015) .

1.1.2 Effort Estimation in Agile Software Development Overview:

Effort estimation in agile is a bit different than any other traditional development method estimations. Briefly, user story is a functional or nonfunctional requirement written in natural language by an end-user in order to capture a description of a software feature. Moreover story point is the unit of estimation used by agile groups to estimate user stories .Story point includes the amount of development and testing effort required to get the work or story done. Story point's estimation is done using

- Relative Sizing: "comparing Novel story with a sample set of previously sized stories". (Coelho & Basu, 2012)
- Abstract sizing: No mathematical basis For example:
  - Fibonacci sequence
  - Dog breeds
  - T-shirt sizes
  - Numeric sizing
- A team might assign the "Add plant "story 2 points according to its complexity. This exercise is executed until each story has a correspondent point.

1.2 Objectives of the Research

The main objective of the research is to construct an efficient and effective agile effort estimation models using several types of artificial neural network like Feed forward, Cascade and recurrent Networks that shows lower values of error and higher levels of prediction and Accuracy values.

1.3 Motivation

According to our literature review, Neural Networks have proven its efficiency and precise predictions. Till that day, no efficient model has been designed for an agile effort estimation. So it's my priority to present a good and efficient neural network model for agile projects

1.4 Report organization

Chapter 2: literature survey: This chapter includes the description of the existing work.

Chapter 3: Basic concepts and performance: This chapter includes a detailed description of agile development and artificial neural networks.

Chapter 4: Proposed Solution for Agile Software Effort Estimation: This chapter includes a detailed description of the model proposed for effort estimation in agile software using ANNS.

Chapter 5: Comparison and Discussion: This chapter includes a comparison between proposed models (All ANNs). It also includes a discussion about the models results.

2 Literature Survey

After stating the importance of agile development and the need to estimate its effort before doing any project, now is the time to illustrate and look into the previous work done in agile effort estimation and in Artificial Neural Network. The survey has been categorized into two parts, one for the agile software effort estimation and the other is the use of Artificial Neural network in effort estimation

2.1 A review of studies on Agile Software Effort Estimation

(Keaveney & Conboy) Have both investigated and examined how accurate would the conventional and current estimation techniques fit to the new agile development. To verify their methodologies, (Keaveney & Conboy) have moved out to different and diverse companies in order to check the applicability of using current estimates on agile pattern of work. Main estimation used were analogy and expert judgment. (Keaveney & Conboy) Concluded that the keys for a success agile effort estimation are: Company must have experience in agile, previous agile projects data should be used in estimation, fixed priced budgets for certain tasks should be specified in a document and documentation of previous data should be available. (Schmietendorf, Kunz, & Dumke, 2008) Have delivered a research about estimation possibilities, focusing on the estimation of extreme programming (Agile method) paradigm. Moreover, (Coelho & Basu, 2012) have analyzed different estimation models while focusing on estimation using story points, which is used in most

of agile development nowadays. (Coelho & Basu, 2012)Have done a great job in describing the steps needed to use story point based method for agile effort estimation. (Coelho & Basu, 2012) Have figured out that story points and velocity are the best parameters used to predict Effort estimation where story point is the size of work needed to be done while team velocity measures how fast the team rate of progress. (Ziauddin, tipu, & Zia , 2012) have developed a unique and novel model for agile project .With the aid of 21 software project , (Ziauddin, tipu, & Zia , 2012) were able to develop and construct a model that have good estimation accuracy . (Ziauddin, tipu, & Zia , 2012) Have used the story points as the basis for the agile estimation. (Hearty, Fenton, Marquez, & Neil, 2009)Have constructed a full and complete Bayesian network of an extreme programming surrounding. While (Satapathy, Panda, & Rath) have focused on estimating agile project using story points. Moreover (Litoriya & Kothari, 2013) have presented a new approach(making use of Cocomo II) to measure agile web-based software in addition to computing effort and cost using number of constraints captured from environmental characteristics, furthermore (Litoriya & Kothari, 2013) have explained the difference between web-based software and conventional software. At First (Georgsson, 2011) has described story points and the importance of them, and then recommend the use of story points in agile to increase the estimation accuracy. (Javdani, Zulzalil1 , & Ghani) Have also succeeded in gathering all measurement practices in agile software development methods with providing a review of agile version of COSMIC. (Satapathy, Panda, & Rath) Used various types of Support vector machine in order to achieve better predication accuracy and decrease the mean error. Moreover, a comparison has been done between four different types of support vector machine in order to figure out the best model design. While comparing the four different types of support vector machine results, (Satapathy, Panda, & Rath)have concluded that radial basis kernel based have outperformed all other support vector machines. (Hussain, Kosseim, & Ormandjieva, 2012)Have made a research on how to remove and avoid problems like formalized user requirements and solve the problem using function points for agile software effort development.

2.2 A review of studies on the use of ANNs for effort estimation

Through the reading of a numerous articles, it is observed that artificial neural network is the most common model used for any effort estimation. Therefore a review on the use of artificial neural

network in the effort estimation category must be arranged. (Wena, Li a, Lin b, & Huc, 2011) Have studied different types of machine learning techniques and non-machine learning techniques for effort estimation models. The study investigated that machine learning estimation models have surpassed non-machine models. (Idri, Zakrani, & Zahi, 2010) Have designed Radial basis function networks to measure software cost and effort estimations, Moreover they have focused on achieving the maximum predication through changing the number of hidden layers and neurons. Furthermore (Idri, Zakrani, & Zahi, 2010) have used two different data sets named as cocomo 81 and Tukituki .In conclusion (Idri, Zakrani, & Zahi, 2010) have discussed the impact of widths on the accuracy of the radial basis function network in addition to comparing two Radial basis function network with different formula. A functional link artificial neural network was offered by (Rao, 2009) for accurate cost and effort estimation, as well as (Rao, 2009) main purpose was to reduce the computational complexities for further use in online applications and to verify that functional link neural network can be used for effort prediction based on size of the project and effort multipliers. (K.R, Sree P , & S.N.S.V.S.C, 2010) Have presented two different model, the first one is the generalized regression and the second is radial neural network for effort estimation. Results of the models were analyzed using 5 different criteria MARE, MME, VARE, Mean BRE and prediction.IN (K.R, Sree P , & S.N.S.V.S.C, 2010) paper, the radial basis provided better results. Moreover (Oliveira, L. Braga, Lima, & L., 2010) Have illustrated the purpose of the genetic algorithm and how to use it, they suggested that genetic algorithm can help in optimizing the parameters of the machine learning methods. In conclusion (Oliveira, L. Braga, Lima, & L., 2010) have proved that genetic algorithm was able to improve performance of three machine learning techniques as well as decreasing the number of input significantly for the two data sets used

3 Basic Concepts and Performance Metrics

3.1 Characteristics of Agile:

Team organization

- Whole Teams: Team itself must have sufficient skills to tackle any problem and it shouldn't depend on any external resources.
- Stable : Changing team can lead to huge delays so it is preferable to keep the team stable
- Formed of generalizing specialists: People who has one or more technical specialties are needed in agile teams in order to fill in gaps and act as a sweet spot between two extremes of specialists

Characteristics of agile project

Evolutionary approach

As believed by (Moniruzzaman), agile methods are evolutionary, iterative and incremental delivery model. By using agile, the entire application is divided into small incremental units called iterations. Each iteration include functions and is built on top of the previous iteration (Moniruzzaman).

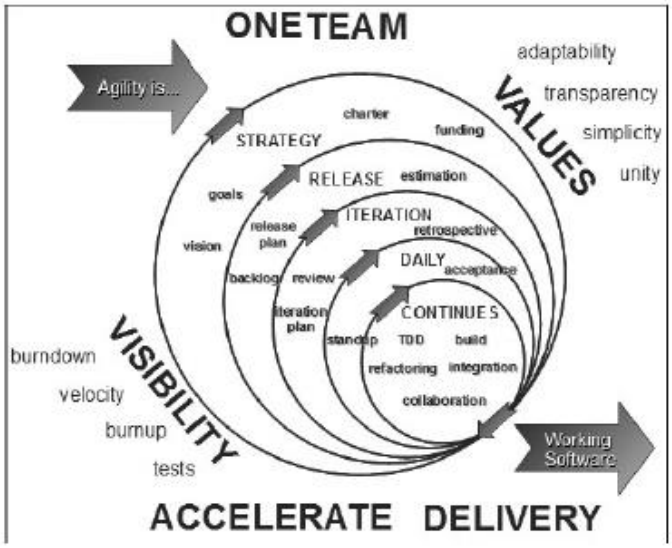
Lightweight methods:

Agile methods are called lightweight methods as they employ short iterative cycles that include user interaction as prioritize and verify requirements.

Rapid delivery of software products

Agile methods emphasize on the power of rapid delivery to the client where agile "delivers value to the customer quickly, and speeds requirements emergence" (Moniruzzaman). Thus agile

development cycle satisfies the customer through the early and continuous delivery of the software.



Highly tolerant of change requirements

The main difference between agile and traditional lifecycles is the acceptance of change (Moniruzzaman).Users change their requirements every day, therefore agile was designed to respond quickly to the changes without any loss as well as the flexibility to change requirements even in development.

Accept prioritizing requirements

In traditional software, Requirements are delivered by clients but prioritized by developer. But that is not the case in agile, Agile allow the client to prioritize task as needed by customer.

Active customer involvement & feedback

"Customers are actively involved, and get higher priority in agile approaches rather than any traditional approaches. There is face to face communication and continuous feedback from customer (product owner) always happen in agile approach." (Moniruzzaman).Moreover customer love and appreciate the real participation in project as it gives them the feeling of control over the project.

Reduce cost and time

Researches have proved that agile development life cycle decreases the cost of the project by 50% than a traditional development life cycle.

Less Documentation

Agile focuses more on development than documentation, it focuses on value to market rather than the consumption of resources (people) in useless work.

Improves software quality

Agile development increases the rate of detecting errors Due to the user involvement in iterations

Success possibility increased

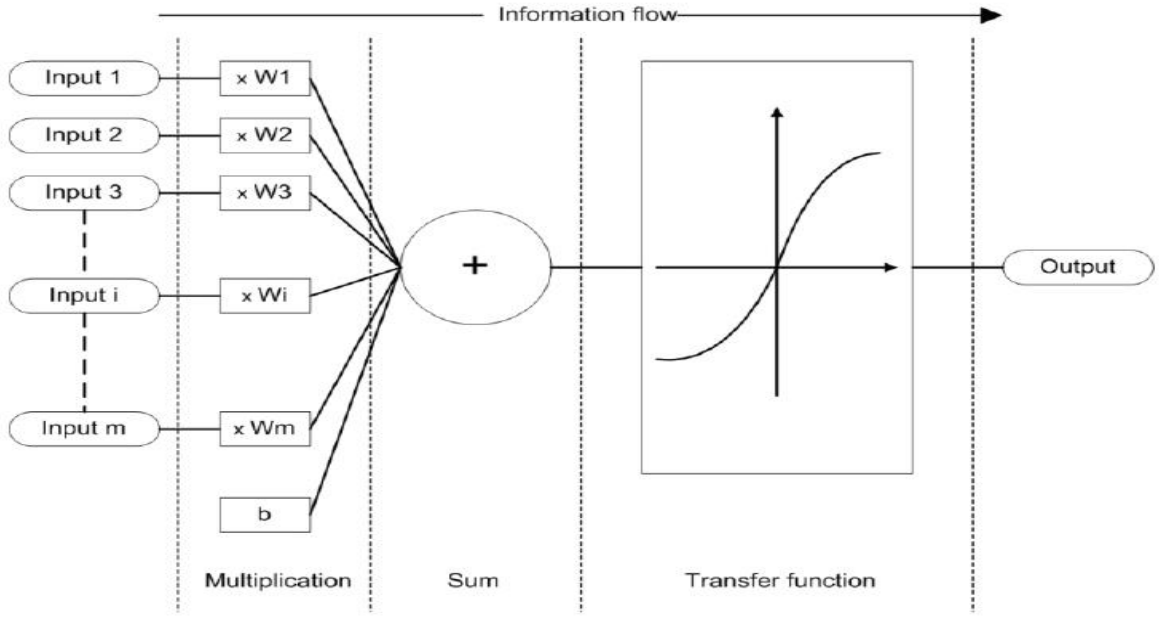
Agile process benefit the rate of project success, better risk management, delivery on time, delivery of good quality and most important is the response to changing requirements. Therefore according to researches, the rate of success of agile is higher than any other traditional life cycle.

3.2 Basic Concepts and Neural Networks

Overview:

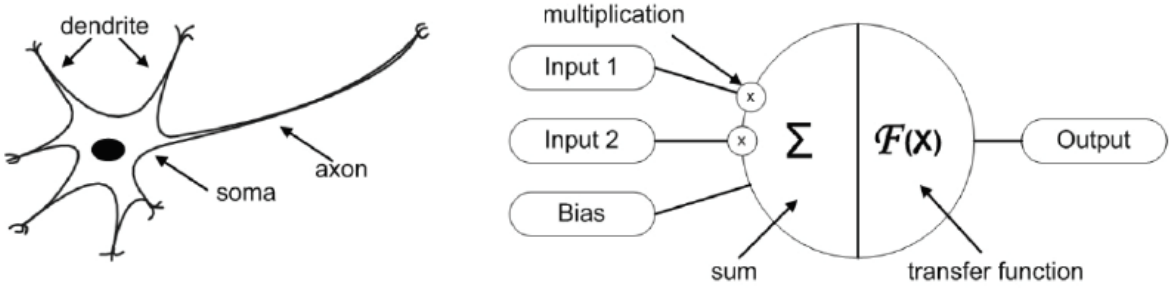
Artificial Neural Networks is a mathematical model that tries to mimic the Functions and structure of a Human brain (Krenker, Bešter, & Kos, 2011).Artificial Neural Networks is named after biological neural network that exist in the human brain. Artificial Neural Network consist of a set of artificial Neuron ,where each one present a simple mathematical function, Artificial Neural Networks is composed of three simple rules and tasks : Multiplication, Summation and Activation.

1. At the beginning of the process inputs enter the neurons where each input is weighted, it means that every value is multiplied by the weight of neuron.
2. After the previous iteration, the Sum function add all weighted inputs together.
3. At the end, the sum of all neurons pass through an activation function that is called transfer function



Artificial Neuron:

Artificial neuron is the simplest building block of artificial neural network. (Krenker, Bešter, & Kos, 2011).The Artificial Neuron functionalities and design are captured from the biological Neuron which includes the spinal cord, brain and peripheral ganglia.



Inputs that are weighted come and enter into the body of neuron as information. After that Neuron sum the inputs, bias and do a process on the sum through the transfer function. The mathematical description of the neuron can be illustrated as

$Y(k) = F\left(\sum_{i=0}^{m} w_i(k) \cdot x_i(k) + b\right)$

Where:

- W is the weight value
- X is the input value

- B is bias
- F is the transfer function
- Y(k) is the output of the neuron

The transfer function is chosen on the basis of the problem needed to be solved, so if the problem is linear then a linear function will be selected otherwise a nonlinear (sigmoid) function will be selected.

Step function (perceptron):

- It is used mainly in the classification problems, the Step function is a binary function which yields only 0 or 1 based on the Threshold.
- Commonly used in the last layer

$$y = \begin{cases} 1 \; if \; w_i x_i \geq threshold \\ 0 \; if \; w_i x_i < threshold \end{cases}$$

Linear Function:

- A linear transformation over the sum of bias and Weighted inputs
- Mostly used in the input layers

Nonlinear function (sigmoid):

- Simple derivative
- Can be used in nonlinear problems

Artificial Neural Network:

So the question that must be asked will a single artificial neuron be able to solve a real life problem or not? The answer is no .Real life problems are complex, parallel, distributed and nonlinear. In order to solve a real life problem, a combination between various artificial neurons must be integrated into one large network. So combining two or more neurons together will form an artificial neuron Network. Moreover Topology describes how artificial neurons are interconnected inside the whole network (Krenker, Bešter, & Kos, 2011). Topologies can be categorized into:

- Acyclic graph where data and information flow into one direction only
- Cyclic graph where data can flow in both directions

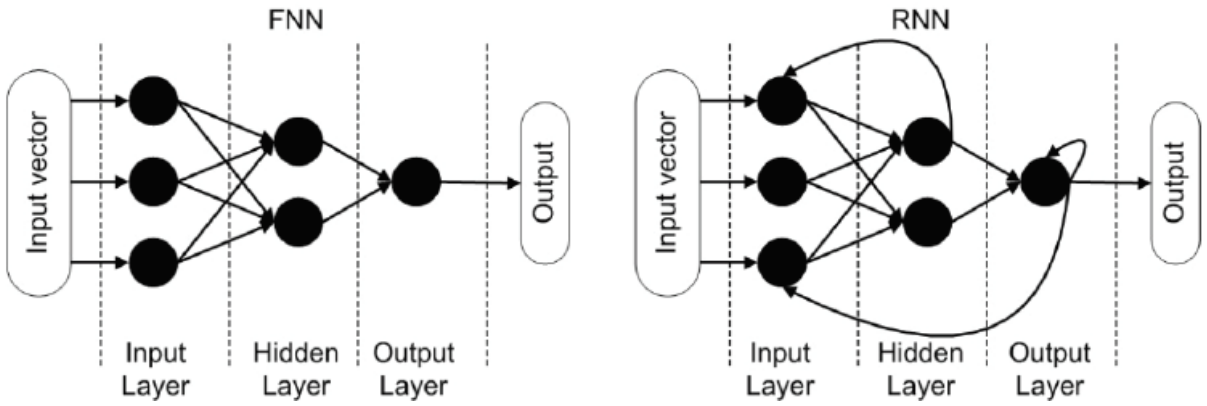


Figure 1:acyclic and cyclic topologies

Neural Network Learning:

Just as Biological neural networks, artificial neural network learn their target or behavior through the input they get from the user .There are three learning Methods: supervised learning, unsupervised learning and reinforcement learning. The purpose of the all the learning paradigms is the same, all paradigms help Artificial Neural network to achieve precise and accurate output according to the input given.

Supervised learning

In Supervised learning, the training data include both the input and its correspondent output. A group of steps must be done to train artificial neural network and the steps are:

- Gather A training set that effectively illustrate a certain problem
- Organize the data set in order to be understandable and easy for the Artificial Neural Network
- Organize the data set to contain both input and required output
- Then let the Artificial Neural Network do the learning

- In the last step , we must test the performance of the Artificial neural network through the use of validation data set

Unsupervised learning:

Unsupervised learning is based on an input and a cost function that should be minimized, unsupervised learning is mostly used in finding hidden patterns or cluster analysis. Self-organizing maps are the neural network that uses the unsupervised learning extensively.

Reinforcement learning

Reinforcement learning is a machine learning technique that its main priority is to maximize a numerical reward signal. The learner is not told which actions to take, but instead the learner must try numerous options to discover which actions produce the most reward. Reinforcement learning uses several different algorithms to yield the maximum return. Brute force algorithm is the algorithm used to try every possible option, however brute force algorithm is considered to be a weakness in large and huge data sets.

Description of the neural network used

Feed Forward Neural Network (FFNN)

The Feed Forward Neural Network is composed of an input layer, output layer and some hidden layers in between (Panda, Effort Estimation of Agile and Web-based Software using Artificial Neural Networks, 2015).The algorithm behind the Feed forward neural network is the Back propagation algorithm .Back propagation algorithm main purpose is to find a correct and accurate weights parameters which closely related to the original problem real weights parameters. During the training phase, the Artificial Neural Network performance is evaluated by calculating the

difference between the Artificial Neural networks outputs and the expected output .The difference between the two outputs is known as the error and stated as

$$E = \frac{1}{2} \sum_{k \in T_r} \sum_{j=1}^{m} (y_j(x_k, w) - d_{jk})^2$$

Where

D_{jk} is the desired output

X_k is the input

W is the weight

Y_j is the activation function

Steps in Error Back propagation /Training Process:

1. First step is to Initialize weights randomly
2. During the training ,weights are updated
3. By the help of back propagation(example : gradient descent) techniques ,we can figure out which way that will cause the cost function to decrease
4. If the gradient descent is negative then another computation must be done in order to reduce the error.
5. When the gradient descent value tends to be constant ,then no more learning can be done and we have to stop learning.

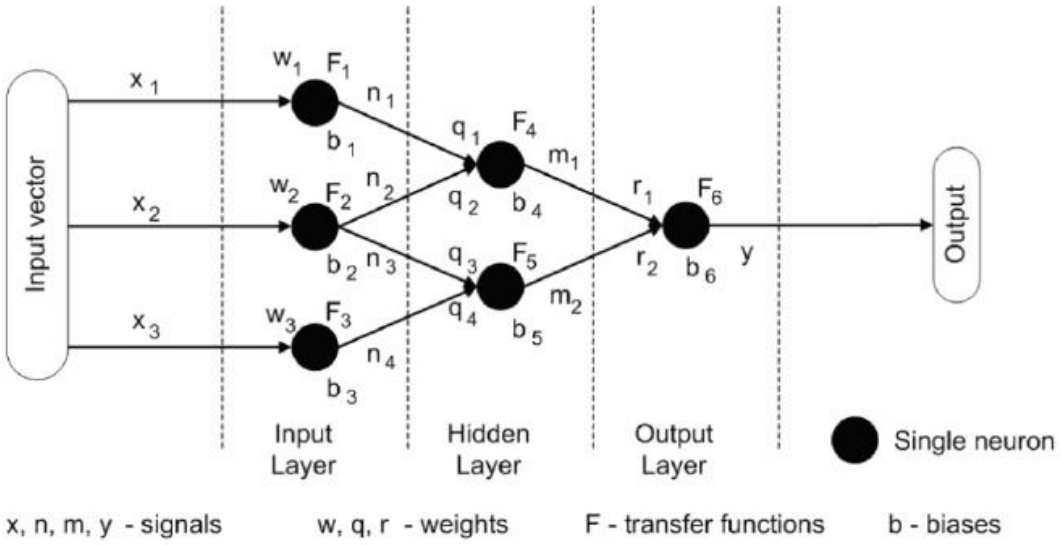


Figure 2: Feed Forward

Feed forward Artificial Neural network data must flow in one direction where the direction is from the input to the output with no loops. There is flexibility in the number of layers, types of functions, connections used between neurons. The simplest form of FFNN is a single Neuron using a linear function that can solve only linear simple problems

Cascade Forward back propagation network:

Feed forward and cascade forward are quite similar, but the difference here lies in the topology of each neural network and how neurons are connected. Cascade Forward back propagation network include a weight connection from the input to each layer and from each layer to the successive layer

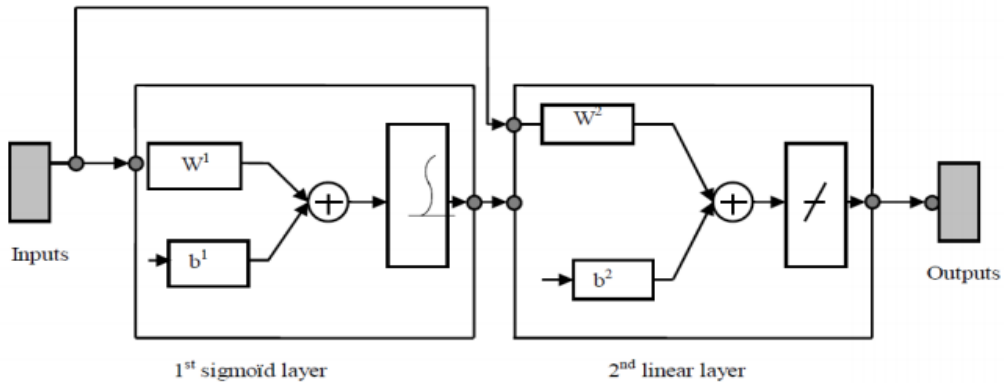


Figure 3:Cascade forward

Recurrent Neural network:

The power within recurrent neural network lies in sequential information, most implemented neural network assume that inputs are not dependent on each other .But that is not the case in modern technology .Nowadays Inputs are highly dependable on each other , So it is highly recommended to use the recurrent neural network . People thought that recurrent neural network has memory due to its knowledge of the previous steps. But in reality Recurrent neural network can only look few steps back. Recurrent neural network allow us to model sequence of data

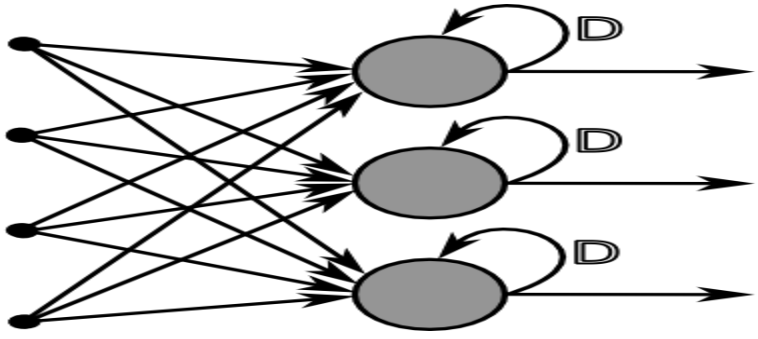


Figure 4:Recurrent neural network

Training Functions:

Levenberg-Marquardt back propagation:

It is also known as the damped least squares method, its main purpose is to solve nonlinear equations, Kenneth levenberg was the first to publish the algorithm. Levenberg-Marquardt back propagation solves equation by minimizing sum of square error. In the Levenberg-Marquardt algorithm, the error function is minimized, while the step size is kept minimal in order to ensure the verification of the linear approximation. Levenberg-Marquardt algorithm is one of the functions used to train neural network, Training will stop when the mean square error start to increase again.

Gradient descent

Gradient descent is another algorithm that is used to train neural network, Gradient descent is a supervised learning pattern that uses an error correcting term. Gradient descent main purpose is to find the local minimum. In order to find the local minimum (minimum error) some steps should be considered

- If the result is negative, then we should complete our testing as the minimum error haven't been found yet
- Should do the previous iteration until gradient descent start to be positive or constant
- Then we should stop as the minimum error is found successfully

Conjugate gradient back propagation

The basic back propagation algorithm adjusts the weights in the steepest descent direction (negative of the gradient), Conjugate gradient back propagation is mostly used for linear equations.

Why use ANNs for Software Effort Estimation?

The following facts have contributed in choosing neural network over any other machine learning:

- due to their essential learning ability and good interpretability
- Can manage Complex nonlinear equations
- Have proven its accuracy in the construction of efficient effort estimations models

4 Proposed Solution for Agile Software Effort Estimation

Data Set Description:

The data Set used here is the same as the data set used in (Ziauddin, tipu, & Zia , 2012) .It contains 21 record with 10 columns each.

Effort	Vi	D	V	Sprint Size	Work days	Team salary	Time	Cost

As discussed before agile teams must have specific characteristics that differ them from any other traditional software. Therefore the Estimation will not include personnel requirements, it will only include completion time for the agile project.

Math behind the collection of Agile Data set:

It is not easy to capture data from an agile project, so the Math behind gathering effort and velocity will be covered in the next section

Determining the Effort in the dataset

There is a huge amount of identified and vague factors that can affect effort. The challenge, however is what factor I should choose in effort calculation and what should I neglect. The perfect solution here is to neglect the factors that the company has no power to change and to concentrate on jobs and tasks where the company have influence. Minimizing the number of factors will keep the calculations simple as possible as well as it will help to maintain consistency over all areas.

Effort will be calculated by using 2 factors: Story size and complexity

Story Size

"Story size is an estimate of the relative scale of the work in terms of actual development effort" (Ziauddin, tipu, & Zia , 2012).Values are assigned to different type of types (5 is the largest while one is the lowest).small

Value	Description
1	• Very small story which require minor effort • Work accomplished in Few hours
2	• Small story • Work done in 1 to 2 days
3	• A relatively large story • Work done in 2 to 5 days
4	• Very large story • Requires a long period of time (more than week) • Require focused effort from the team for a long time • Should break the large story into a group of smaller ones.
5	• Extremely large story • Must be broken into smaller ones • Cannot be accurately estimated

Complexity:

It is defined as the technical complexity, Moreover More complexity means more uncertainty. The following figure shows user story complexity scale:

Value	Description
1	• Very straightforward • Very little uncertainty • No research required • Requires basic programming skills

2	• Easily understood but not straightforward • Little research needed • Little uncertainty • Requires basic or intermediate programming skills
3	• Relatively complex • Requires intermediate programming skills • Requires intermediate research • Moderate level of uncertainty
4	• Very complex • Requires relatively high amount of research • Multiple unknowns • High level of uncertainty
5	• Extremely complex • Requires extensive research • An extremely large amount of unknowns • Extremely high level of uncertainty

Using the previous two factors, Effort can be obtained by Multiplying complexity and size

Effort = complexity *size

In order to get the effort for the complete project, Efforts of different stories needed to be added together.

Equation

The unit of Effort is stated as Story Point (SP).So Story point is the quantity of effort done in a unit time.

Determining agile velocity:

- Velocity is defined as: Velocity = Distance /Time
- Team velocity is how much fast a team can finish a specific task Ex: Team 1 executed 3 tasks in an hour which means that the velocity is 3/1 = 3 tasks per hour.
- For Agile estimation purpose, Distance is equivalent to units of effort and Time is equivalent to the length of the sprint

Used Data:

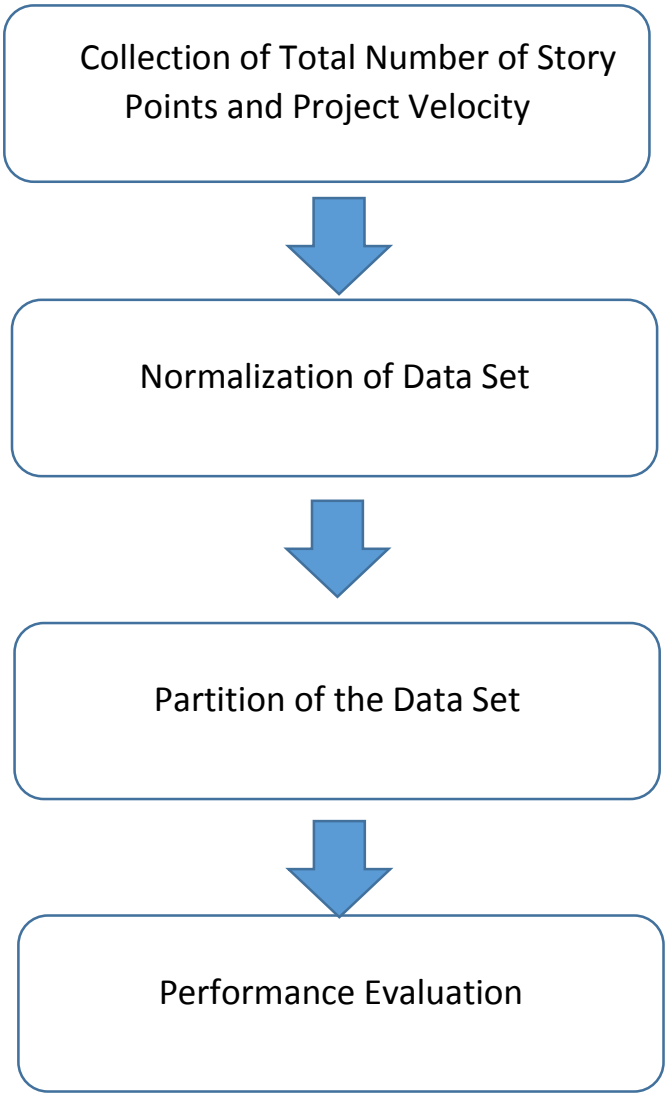
After gathering the idea of how information in (Ziauddin, tipu, & Zia , 2012) is calculated, we are now capable of using the data set. Effort and Velocity will be used as an input for various types of Neural Networks, while Time will be the estimate we need to figure. Predication Accuracy will be calculated for each ANN based on the time estimated.

4.1 Proposed Methodology:

The Proposed Approach will be fulfilled using the 21 record developed by six software houses (Ziauddin, tipu, & Zia , 2012).For Agile Effort estimation only Specific Fields will be used for greater efficiency and consistency .After Data set optimization, Three columns will be used only out of the original nine columns.

- The first column will show how many story points required to complete the project
- The second column will represent the Team velocity or the velocity of the project
- The Third column will be the actual time to complete the project as well as our target (actual effort and it is measured in months).

The Following figure states the steps needed in order to compute predicted effort using various artificial neural network techniques



Steps in Effort Estimation

1. Gathering the number of Story points and team velocity: The number of Team Velocity and story points are collected.
2. Data set Normalization: The collected number of Story points and team velocity must be normalized in the range [0, 1].The normalization equation is described as :

$$X' = \frac{x - \min(x)}{\max(x) - \min(s)}$$

Where

X= is the original value

X'= the normalized value within the range of [0, 1]

Min(x) =minimum value of x

Max(x) =maximum value of x

3. Partition of Data set: The data set is partitioned into learning, Validation and test sets
    a. Training set : it is used to train the model by coupling the input with the expected output
    b. Validation set : It is used to select the best performing algorithm or approach
    c. Testing set : to estimate the accuracy of the Artificial neural network
    d. Both validation and testing sets are used to estimate how good the Artificial Neural Network is trained.
4. Performance Evaluation : In the last step the performance of the Neural network is evaluated using MMRE ,R^2 and Prediction values

4.2 Experiential Details

The input for all Neural Networks are Team velocity and Story points while the expected output should be the effort (completion time).Artificial Neural Networks are trained for a specific number of iterations, in every iteration the MSE is calculated. At the point the MSE value becomes constant, we should stop training as the best weights are obtained. After that the constant weights are used in the testing phase.

.

4.2.1 Model design in Feed forward artificial neural network:

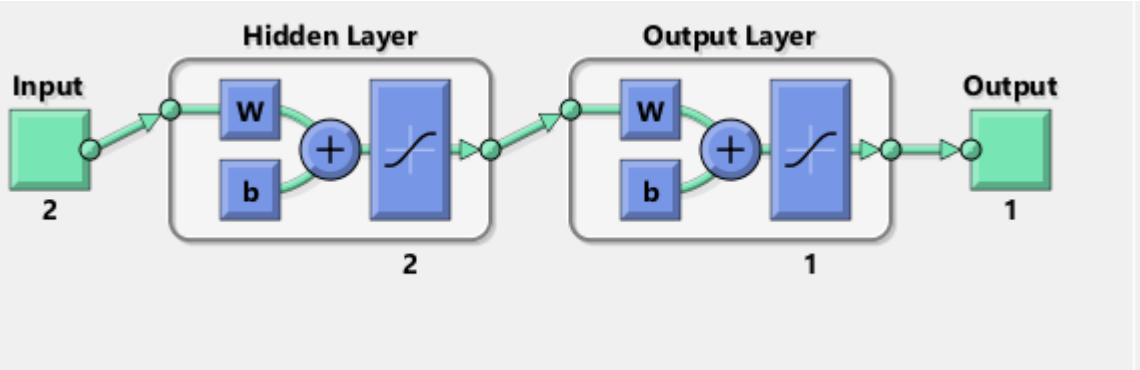
In the following Section, various versions of Feed forward artificial neural network will be tested and evaluated. Number of layers, Neurons, Training function, adaption learning function and transfer function will be changed until the best result is obtained.

Input layer of FFNN: team velocity and story points

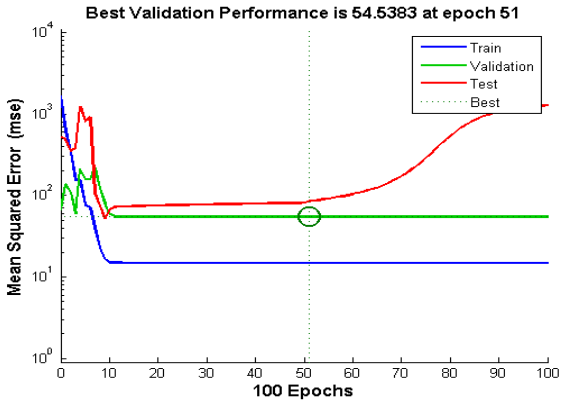
Output layer of FFNN: effort (time)

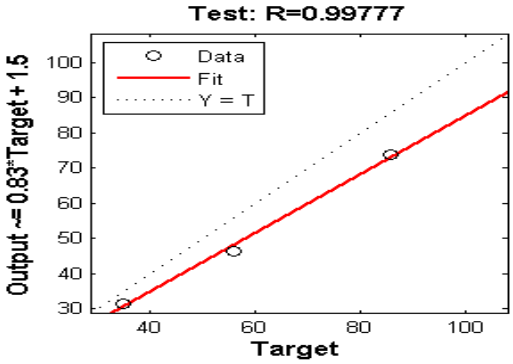
First test:

Number of Neurons	2
Number of layers	1
Training Function	Levenberg-Marquardt backpropagation
Transfer Function	Tan sigmoid



After training:





Results:

- R=0.997

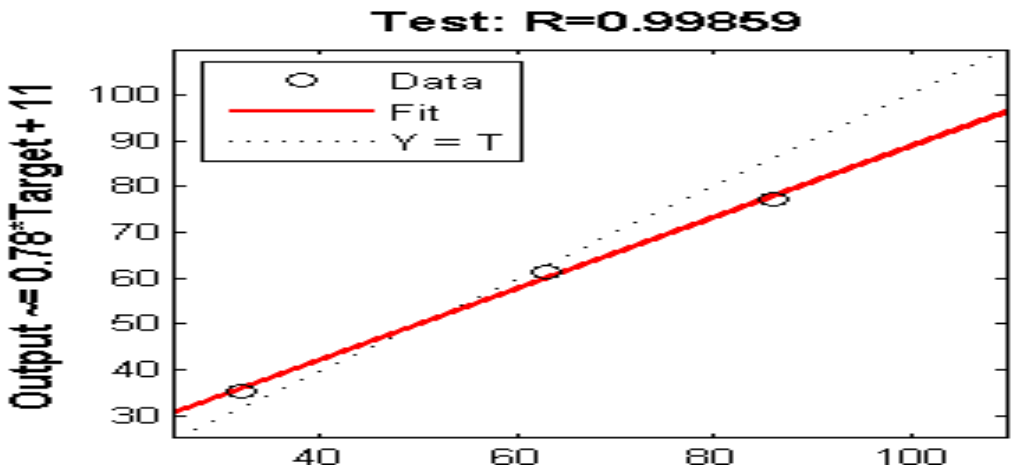
Expected Output (sample)	Actual Output	Error
35	31.496	3.504
93	95.7575	-2.7575
36	41.7868	-5.7868
62	68.4195	-6.419

- Mean magnitude relative error =9.844%
- Prediction =100-9.844=90.156%

Second test:

Number of Neurons	10
Number of layers	1
Training Function	Levenberg-Marquardt backpropagation
Transfer Function	Tan sigmoid

After training:



Results:

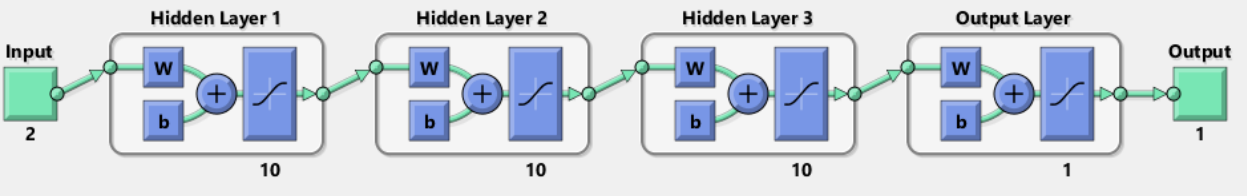
- R=0.997

Expected Output (sample)	Actual Output	Error
35	34.9985	0.001531
93	92.9581	0.041924
36	36.0216	-0.021569
62	62.3695	-0.3694

- Mean magnitude relative error =0.00148%
- Prediction =100-0.00148=99.98%
- With the increase of Neurons , Better results are obtained
- In the following test ,Multiple layers will be used

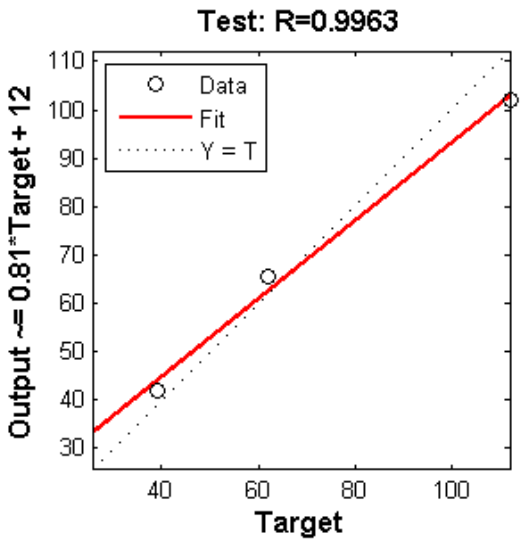
Third Test:

In the following Section, Number of layers will increase gradually till a good Prediction level is achieved. Each layer will include ten neurons due to its success in the previous phase



Number of Neurons	10
Number of layers	3
Training Function	Levenberg-Marquardt backpropagation
Transfer Function	Tan sigmoid

After training:



Results:

- R=0.9963

Expected Output (sample)	Actual Output	Error
35	34.2613	0.73872
93	101.7701	-8.7701
36	43.9387	-7.9387
62	65.459	-3.459

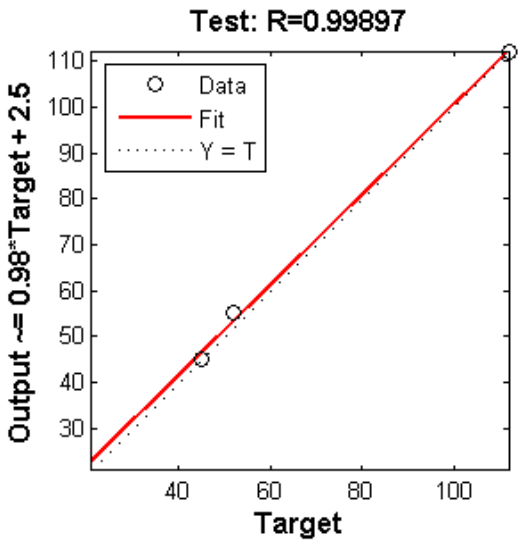
- Mean magnitude relative error =9.75%
- Prediction =90.25 %

Fourth Test:

In the following Section, the transfer Function will be changed .A log Sigmoid will be evaluated instead of the tan Sigmoid. Due to the success of a single layer with ten artificial neuron, a log sigmoid will be evaluated using the same features.

Number of Neurons	10
Number of layers	1
Training Function	Levenberg-Marquardt backpropagation
Transfer Function	Log sigmoid

After training:



Results:

- R=0.99897

Expected Output (sample)	Actual Output	Error
35	35.0015	-0.0014899
93	92.9998	0.00023077
36	36.0003	-0.00025132
62	62.0001	-8.0625e-05

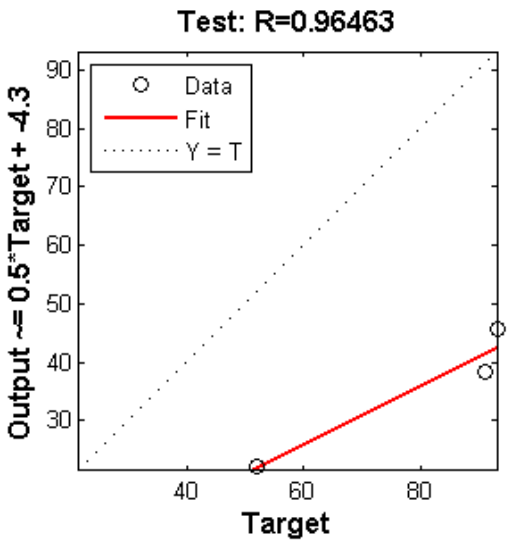
- Mean magnitude relative error =0.0000148%
- Prediction =99.99988 %

Fifth Test:

In the following Section, the Training function will be changed .A Gradient descent will be evaluated instead of the Levenberg-Marquardt back propagation.

Number of Neurons	5
Number of layers	1
Training Function	Gradient descent
Transfer Function	Log sigmoid

After training:



Results:

- R=0.96463

Expected Output (sample)	Actual Output	Error
35	21.0002	13.9998
93	112	-19
36	21.1056	-14.8944
62	112	-50

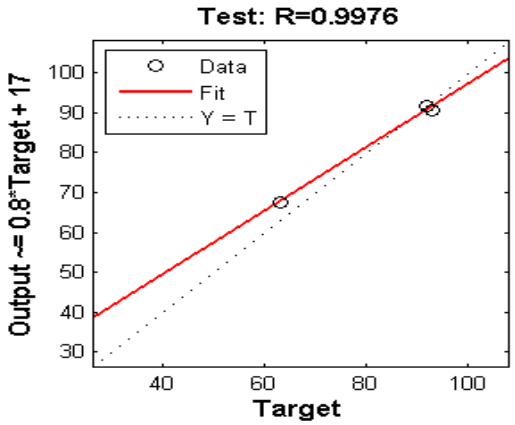
- Mean magnitude relative error =44.99%
- Prediction =55.01%

Sixth Test:

In the following Section, the Training function will be changed .A conjugate gradient back propagation will be evaluated instead of the gradient descent back propagation.

Number of Neurons	10
Number of layers	1
Training Function	Conjugate gradient back propagation
Transfer Function	Log sigmoid

After training:



Results:

- R=0.9976

Expected Output (sample)	Actual Output	Error
35	32.0942	2.9058
93	94.5663	--1.5663
36	35.9157	0.08433
62	60.8631	1.136

- Mean magnitude relative error =1.6%
- Prediction =98.4%

4.2.2 Model design in Cascade Forward back propagation Neural Network:

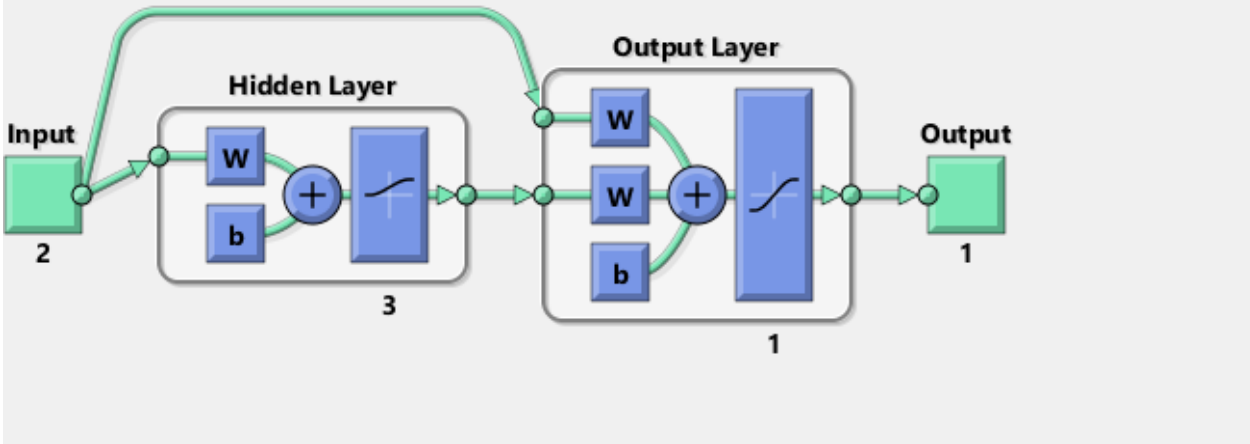
In the following Section, various versions of in Cascade Forward back propagation artificial neural network will be tested and evaluated. Number of layers, Neurons, Training function, adaption learning function and transfer function will be changed until the best result is obtained.

Input layer of CFBPNN: team velocity and story points

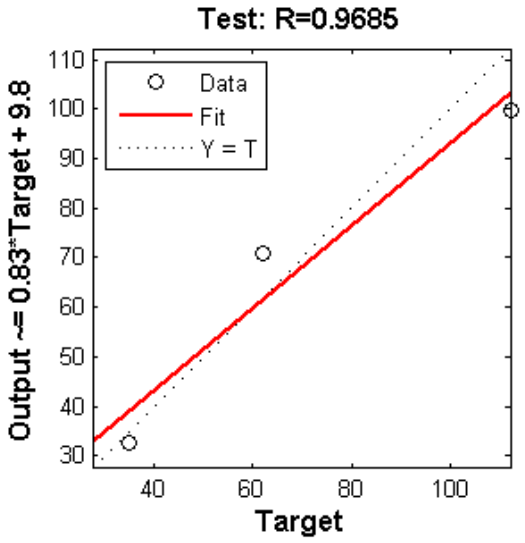
Output layer of CFBPNN: effort (time)

First test:

Number of Neurons	3
Number of layers	1
Training Function	Levenberg-Marquardt backpropagation
Transfer Function	Log sigmoid



After training:



Results:

- In R=0.9685

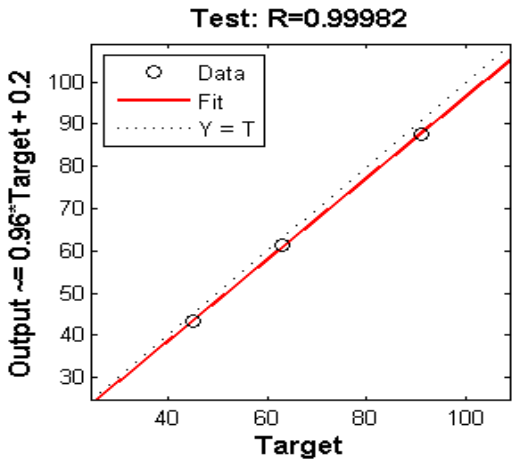
Expected Output (sample)	Actual Output	Error
35	32.7569	2.2431
93	89.4036	3.5964
36	38.6835	-2.6835
62	70.9462	-8.946

- Mean magnitude relative error =8.02%
- Prediction =100-8.02=91.97%

Second test:

Number of Neurons	10
Number of layers	1
Training Function	Levenberg-Marquardt backpropagation
Transfer Function	Log sigmoid

After training:



Results:

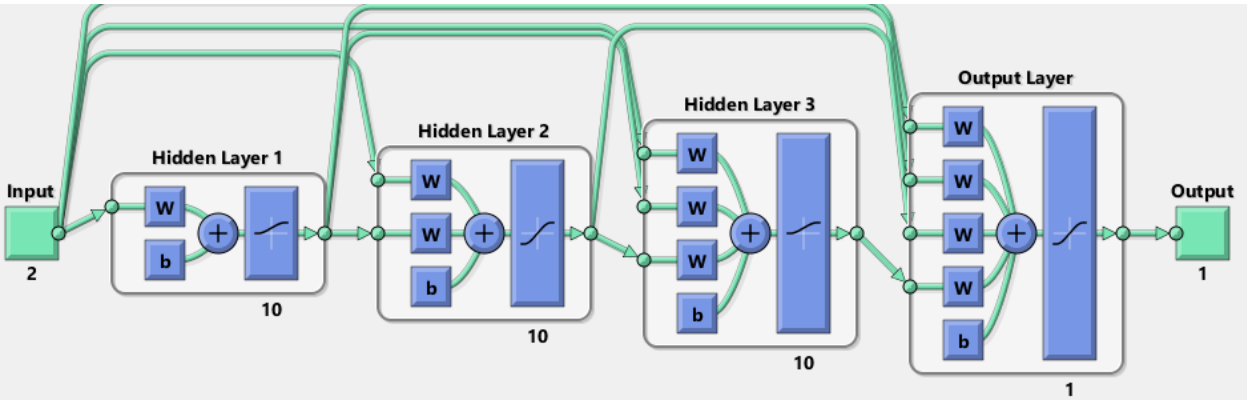
- In R=0.0.99982

Expected Output (sample)	Actual Output	Error
35	31.9152	3.0848
93	91.504	1.496
36	32.1355	3.8645
62	63.291	-1.291

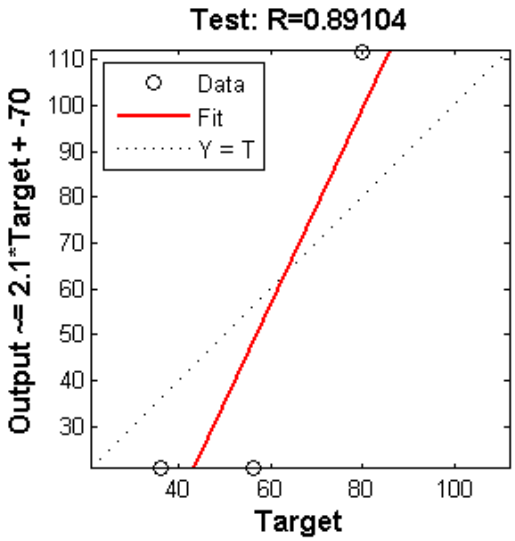
- Mean magnitude relative error =5.83%
- Prediction =100-5.83=94.17%

Third test:

Number of Neurons	10
Number of layers	3
Training Function	Levenberg-Marquardt backpropagation
Transfer Function	Log sigmoid



After training:



Results:

- In R=0.0.89104

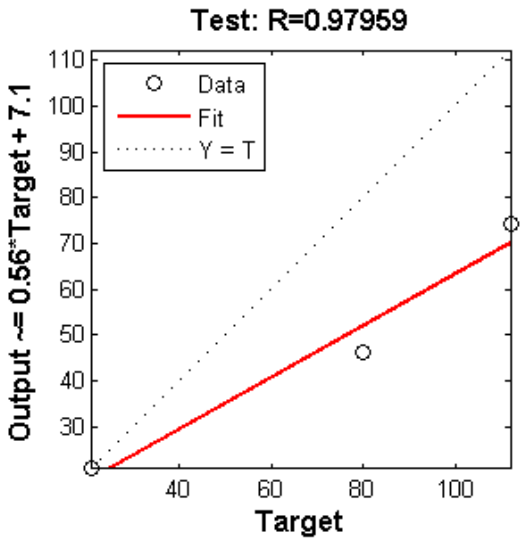
Expected Output (sample)	Actual Output	Error
35	25.7979	9.2021
93	94.5392	-1.5392
36	33.6908	2.3092
62	68.46	--6.46

- Mean magnitude relative error =11.15%
- Prediction =100-11.15=88.85%

Fourth test:

Number of Neurons	10
Number of layers	1
Training Function	Gradient descent
Transfer Function	Log sigmoid

After training:



Results:

- In R=0.0.97959

Expected Output (sample)	Actual Output	Error
35	35.0067	-0.0066916
93	93.0001	-0.00012153
36	23.4365	12.5635
62	49.882	12.117

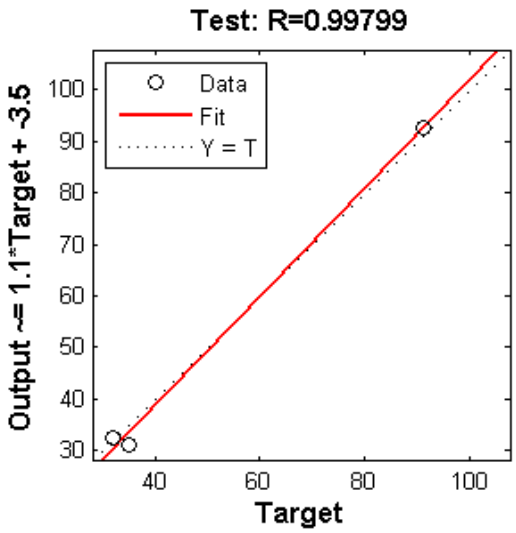
- Mean magnitude relative error =13.6%

- Prediction =100-13.6=86.4%

Fifth test:

Number of Neurons	10
Number of layers	1
Training Function	Conjugate gradient back propagation
Transfer Function	Log sigmoid

After training:



Results:

- In R=0.99799

Expected Output (sample)	Actual Output	Error
35	35.8599	-0.8599
93	92.938	0.061981
36	35.0439	0.95605
62	62.320	-0.3203

- Mean magnitude relative error =1.4%
- Prediction =100-1.4=98.5%

4.2.3 Model design in Layer Recurrent Neural Network:

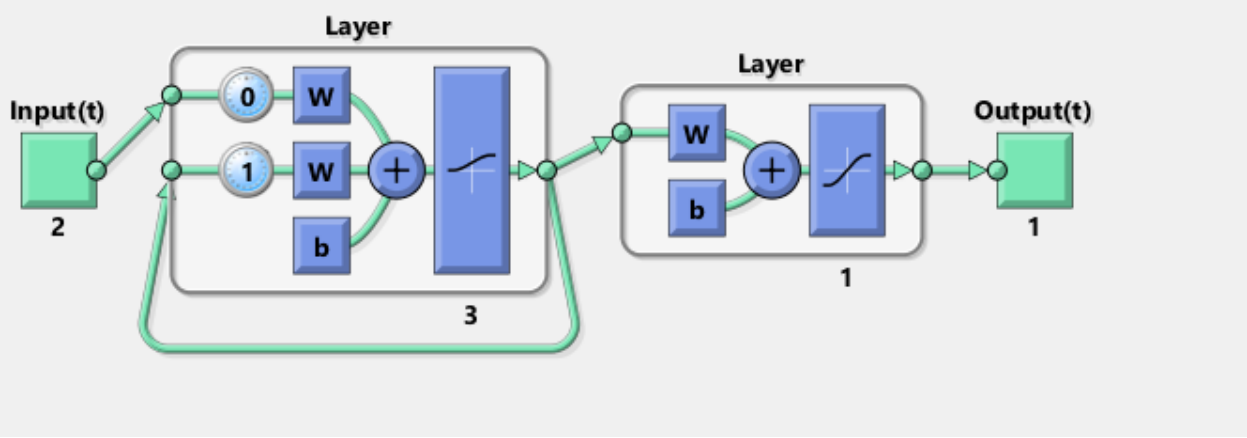
In the following Section, various versions of in in Layer Recurrent artificial neural network will be tested and evaluated. Number of layers, Neurons, Training function, adaption learning function and transfer function will be changed until the best result is obtained.

Input layer of LRNN: team velocity and story points

Output layer of LRNN: effort (time)

First test:

Number of Neurons	3
Number of layers	1
Training Function	Levenberg-Marquardt backpropagation
Transfer Function	Log sigmoid



Results:

Expected Output (sample)	Actual Output	Error
35	29.1299	5.8701
93	92.6771	0.32291
36	34.4154	1.5846
62	66.4504	-4.4504

- Mean magnitude relative error =7.07%
- Prediction =100-7.07=92.97%

Second test:

Number of Neurons	10
Number of layers	1
Training Function	Levenberg-Marquardt backpropagation
Transfer Function	Log sigmoid

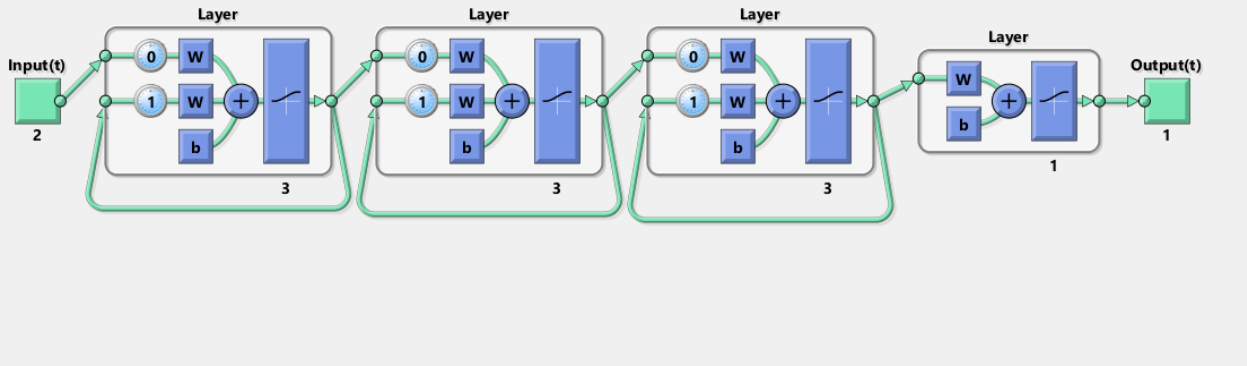
Results:

Expected Output (sample)	Actual Output	Error
35	66.5002	-31.5002
93	94.7349	-1.7349
36	66.5005	-30.5005
62	66.7011	-4.701

- Mean magnitude relative error =44.857%
- Prediction =100-44.857=55.14%

Third test:

Number of Neurons	3
Number of layers	3
Training Function	Levenberg-Marquardt backpropagation
Transfer Function	Log sigmoid



Results:

Expected Output (sample)	Actual Output	Error
35	66.5	--31.5
93	66.5	26.5
36	66.5	-30.5
62	66.5	--4.5

- Mean magnitude relative error =52.6%
- Prediction =100-52.6=47.33%

Fourth test:

Number of Neurons	3
Number of layers	1
Training Function	Gradient descent
Transfer Function	Log sigmoid

Results:

Expected Output (sample)	Actual Output	Error
35	66.5034	-31.5034
93	66.5035	26.4965
36	66.5032	-30.5032
62	66.503	-4.503

- Mean magnitude relative error =52.0%
- Prediction =100-52.0=48.00%

Fifth test:

Number of Neurons	3
Number of layers	1
Training Function	conjugate gradient back propagation
Transfer Function	Log sigmoid

Results:

Expected Output (sample)	Actual Output	Error
35	31.7229	3.2771
93	94.0347	-1.0347
36	32.5647	3.4353
62	63.100	-1.1005

- Mean magnitude relative error =5.38%
- Prediction =100-5.38=94.61%

5 Comparison and discussion:

5.1 Comparison

After obtaining the results of the mean median of relative error and predication for each Neural Network, the estimated effort value for feed forward, cascade forward and recurrent neural networks can now be compared. Comparison will not only exist between our models but will also exist with other researchers models (existing literature).

Several Training have been done

5.1.1 Comparison of Proposed work:

N will indicate number of neurons

L will indicate number of layers

LM will indicate Levenberg-Marquardt

GD will indicate Gradient descent

CGB will indicate conjugate gradient back propagation

Model	Prediction (%)	Error (%)
Feed Forward N:2,L:1,Training function : LM, Transfer function: Tan sigmoid	90.156%	9.84%
Feed Forward N:10,L:1,Training function : LM, Transfer function: Tan sigmoid	99.98%	0.00148%

Feed Forward N:10,L:3,Training function : LM, Transfer function: Tan sigmoid	90.25%	9.75%
Feed Forward N:10,L:1,Training function : LM, Transfer function: log sigmoid	99.998%	0.000148%
Feed Forward N:5,L:1,Training function : GD, Transfer function: log sigmoid	55.01%	44.99%
Feed Forward N:10,L:1,Training function : CGB, Transfer function: log sigmoid	98.4%	1.6%
Cascade Forward N:3,L:1,Training function : LM, Transfer function: log sigmoid	91.97%	8.02%
Cascade Forward N:10,L:1,Training function : LM, Transfer function: log sigmoid	94.17%	5.83%
Cascade Forward N:10,L:3,Training function : LM, Transfer function: log sigmoid	88.85%	11.15%
Cascade Forward	86.4%	13.6%

N:10,L:1,Training function : GD, Transfer function: log sigmoid		
Cascade Forward N:10,L:1,Training function : GCD, Transfer function: log sigmoid	98.5%	1.4%
Recurrent neural network N:3,L:1,Training function : LM, Transfer function: log sigmoid	92.97%	7.07%
Recurrent neural network N:10,L:1,Training function : LM, Transfer function: log sigmoid	55.14%	44.857%
Recurrent neural network N:3,L:3,Training function : LM, Transfer function: log sigmoid	47.33%	52.6%
Recurrent neural network N:3,L:1,Training function : GD, Transfer function: log sigmoid	48.0%	52.0%
Recurrent neural network N:3,L:1,Training function : CGD, Transfer function: log sigmoid	94.6%	5.38%

5.1.2 Comparison of the proposed work with existing work:

Feed Forward	99.9988%

N:10,L:1,Training function : LM, Transfer function: log sigmoid(Best approach)	
Recurrent neural network N:3,L:3,Training function : LM, Transfer function: log sigmoid(worst approach)	47.33%
Regression Analysis (Ziauddin, tipu, & Zia , 2012)	57.14%
Support Vector Regression Linear Kernel (Satapathy, Panda, & Rath)	90.8112%
Support Vector Regression Polynomial Kernel (Satapathy, Panda, & Rath)	68.7382%
Support Vector Regression RBF Kernel (Satapathy, Panda, & Rath)	95.9052%
Support Vector Regression Sigmoid Kernel (Satapathy, Panda, & Rath)	89.7646%

5.2 Discussion:

In the following Section, Analysis upon the neural network results will be discussed and evaluated

5.2.1 Feed Forward Neural Network Discussion

- no effect has been observed When changing the transfer function from log sigmoid to tan sigmoid or vice versa , thus both log sigmoid and tan sigmoid are the same according to Feed forward neural network
- Increasing the number of neurons lead to a higher number of accuracy , thus Feed forward neural network efficiency increase as the number of neurons increase Ex:
    - 2 neuron :90.156
    - 10 neurons :99.98%
- LM back propagation has appeared to be the best training function when dealing with feed forward

- LM and Feed forward NN act well with each other Due to the simplicity of both and because Feed forward doesn't contain any loops.
- LM training function has outperformed other training function as gradient descent and conjugate gradient back propagation
- Increasing the number of hidden layer in FFNN has resulted in decreasing the accuracy of prediction ,while increasing the number of neurons has resulted in increasing the number of accuracy
- Gradient descent have failed to yield any good results due to the appearance of more than one local minimum, Gradient descent have failed to get the lowest cost function
- From the performance analysis , there seems to be a minimal amount of over fitting which means that the neural network has created an equation that will only work for our data set

5.2.2 Cascade forward Neural Network Discussion

- no effect has been observed When changing the transfer function from log sigmoid to tan sigmoid or vice versa , thus both log sigmoid and tan sigmoid are the same according to Cascade forward neural network
- Increasing the number of neurons lead to a higher number of accuracy , thus cascade forward neural network efficiency increase as the number of neurons increase Ex:
    - 3 neuron :91.97%
    - 10 neuron : 94.17%
- Conjugate gradient back propagation has appeared to be the best training function when dealing with cascade forward
- Conjugate gradient and Cascade forward NN act well with each other Due to the power of conjugate gradient when dealing with loops
- Conjugate gradient training function has outperformed other training function as gradient descent and LM back propagation
- Increasing the number of hidden layer in Cascade forward has resulted in decreasing the accuracy of prediction ,while increasing the number of neurons has resulted in increasing the number of accuracy
- From the performance analysis , there seems to be a minimal amount of over fitting which means that the neural network has created an equation that will only work for our data set

5.2.3 Recurrent neural network Discussion

- no effect has been observed When changing the transfer function from log sigmoid to tan sigmoid or vice versa , thus both log sigmoid and tan sigmoid are the same according to Cascade forward neural network
- Increasing the number of neurons lead to a lower number of accuracy , thus recurrent neural network efficiency decreases as the number of neurons increase Ex:
    - 3 neuron :92.97%
    - 10 neuron : 55.14%
- Conjugate gradient back propagation has appeared to be the best training function when dealing with recurrent neural network
- Conjugate gradient and Recurrent NN act well with each other Due to the power of conjugate gradient when dealing with loops
- Conjugate gradient training function has outperformed other training function as gradient descent and LM back propagation
- Increasing the number of hidden layer in Recurrent NN has resulted in decreasing the accuracy of prediction ,Moreover increasing the number of neurons has resulted in decreasing the number of accuracy
- Gradient descent have failed to yield any good results due to the appearance of more than one local minimum, Gradient descent have failed to get the lowest cost function
- conjugate gradient back propagation training function has outperformed other training function as gradient descent and LM
- From the performance analysis , there seems to be a minimal amount of over fitting which means that the neural network has created an equation that will only work for our data set

5.3 Over all analysis

- All neural networks suffer from a minimal over fitting due to the small number of records in the database
- Models efficiency will increase with the increase of the dataset number of records
- Neural Network training is done until the number of MSE tends to be constant.
- Many samples were used and the sample with the highest prediction is used to calculate error and prediction accuracy.
- Validation and testing MSE(Mean square error)exceed number of training which means that over fitting exist
- Increasing number of neurons increases prediction accuracy except for recurrent neural network due to the overlapping of loops
- Due to the simplicity of agile effort estimation ,Increasing number of layers in all neural networks have resulted in decreasing accuracy prediction
- Conjugate gradient has proven to be the suitable learning function for all types of neural networks
- Tan Sigmoid and log sigmoid have the same effect on all neural networks
- Gradient descent have proved to be the worst learning function in all types of neural networks

6 Conclusion and future work

6.1 Conclusion

Now a days, agile methods are given more priority than any other traditional method because of their inherited benefits and advantages. In a parallel world, Effort estimation need to be done efficiently to deliver a good quality product as well as preventing cost or resource overrun. Various researches have considered ANNS in effort prediction as they have good capability and capacity to handle both linear and nonlinear equations. In this paper Feed forward, cascade forward and recurrent neural networks have been used to generate effort estimation models for agile software. Out of the existing models, no one has ever tried using the three previously mentioned neural network. So the paper has covered the three neural networks models used for effort estimation prediction, moreover the aim of the work was to examine the impact of changing the number of neurons, hidden layers ,activation function and learning function on the accuracy of feed forward , cascade forward and recurrent Neural network . All the computations carried out in this study were done using MATLAB software tool.

6.2 Future Work

This work can be extended to include other machine learning techniques.

References

Bataineh, K. M., Najia, M., & Saqera, M. (2011). A Comparison Study between Various Fuzzy Clustering Algorithms. *Jordan Journal of Mechanical and Industrial Engineering*.

Britto, R., Mendes, E., & B¨orstler, J. (2015). *An Empirical Investigation on Effort Estimation in Agile Global Software Development.* diva-portal.org.

Coelho, E., & Basu, A. (2012). Effort Estimation in Agile Software Development using Story Points. *International Journal of Applied Information Systems*.

Georgsson, A. (2011). Introducing Story Points and User Stories to Performe Estimations in a Software Development Organisation.

Hearty, P., Fenton, n., Marquez, D., & Neil, M. (2009). Predicting project velocity in xp using a learning dynamic bayesian network model.

Hussain, I., Kosseim, L., & Ormandjieva, O. (2012). Approximation of COSMIC functional size to support early effort. *data and knowledge engineering*.

Idri, A., Zakrani, A., & Zahi, A. (2010). Design of Radial Basis Function Neural Networks for Software.

Javdani, T., Zulzalil1 , H., & Ghani, A. (n.d.). On the Current Measurement Practices in Agile Software Development.

K.R, S., Sree P , R., & S.N.S.V.S.C, R. (2010). Software Effort Estimation using Radial Basis and Generalized Regression Neural.

Keaveney, S., & Conboy, K. (n.d.). *COST ESTIMATION IN AGILE DEVELOPMENT PROJECTS.*

Kitchenham, B. A., & Mendes, E. (n.d.). *A Comparison of Cross-company and Within-company Effort Estimation Models for Web Applications.* Australia.

Krenker, A., Bešter, J., & Kos, A. (2011). *Introduction to the Artificial Neural Networks.* Intech.

Litoriya, R., & Kothari, A. (2013). An Efficient Approach for Agile Web Based Project Estimation: AgileMOW. *Journal of Software Engineering and Applications,*.

Litoriya, R., & Kothari, A. (2013). An Efficient Approach for Agile Web Based Project Estimation: AgileMOW.

MENDES, E., Watson, I., Triggs, C., Mosley, N., & Counsol, S. (2003). *A Comparative Study of Cost Estimation Models for web Hypermedia Applications.* Netherlands.: Kluwer Academic Publishers.

Moniruzzaman, A. B. (n.d.). *Comparative Study on Agile software development methodologies.*

Oliveira, A. L., L. Braga, P., Lima, R., & L., M. (2010). GA-based method for feature selection and parameters optimizationfor machine learning regression applied to software effort estimation.

Panda, A. (2015). *Effort Estimation of Agile and Web-based Software using Artificial Neural Networks.* India: National Institute of Technology, Rourkela.

Panda, A., Satapathy, S., & Rath, S. (2015). *Empirical Validation of Neural Network Models for Agile Software.* india: ScienceDirect.

Rao, B. T. (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*.

Satapathy, S. M., Panda, A., & Rath, S. (n.d.). Story Point Approach based Agile Software Effort Estimation using Various SVR Kernel Methods. *National Institute of Technology*.

Schmietendorf, A., Kunz, M., & Dumke, R. (2008). *Effort estimation for Agile Software Development Projects.*

Taghi Javdani, H. Z. (n.d.). *On the Current Measurement Practices in Agile Software Development.*

Wena, J., Li a, S., Lin b, Z., & Huc, Y. (2011). Systematic literature review of machine learning based software development.

Zhang. (n.d.). Neural Networks for RF and Microwave Design. ieee.

Ziauddin, tipu, S., & Zia , S. (2012). *An effort estimation model for agile software development.* United states: ACSA.