Effort Estimation of Agile and Web-based Software using Artificial Neural Networks

by

Aditi Panda

Department of Computer Science and Engineering
National Institute of Technology, Rourkela
Rourkela-769008, Odisha, India
May 2015

Effort Estimation of Agile and Web-based Software using Artificial Neural Networks

Dissertation submitted in partial fulfillment for the degree of

Masters of Technology

in

Computer Science and Engineering

(Specialization: Software Engineering)

by

Aditi Panda

(Roll No: 213CS3189)

under the supervision of

Prof. S. K. Rath

Department of Computer Science and Engineering

National Institute of Technology, Rourkela

Rourkela, Odisha, 769008, India

May 2015

Department of Computer Science and Engineering
National Institute of Technology, Rourkela
Rourkela-769008
Odisha, India

29th May 2015

Certificate

This is to certify that this thesis titled, '**Effort Estimation of Agile and Web-based Software using Artificial Neural Networks**' by **Aditi Panda** is a record of an original research work carried out by her under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of Master of Technology with the specialization of Software Engineering in the department of Computer Science and Engineering, National Institute of Technology, Rourkela. Neither the thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Santanu Kumar Rath
Professor
Department of CSE

Acknowledgements

I have many people to thank for helping me complete this research work. They have been of great help to me throughout this time. First of all, I would like to thank God for giving me the inspiration for taking up this work in the first place. I feel blessed for being able to take a good decision in choosing this topic for research. I attribute that to the Almighty.

Next I'd like to thank my parents for being there with me during all odds. Inspite of all the difficulties, I was able to maintain my cool and work only because of the soothing words and assurance given by my parents. Without them, I could never have been successful in doing my work. I feel so felicitous to see them proud.

I want to extend bighearted thanks to my guide here, Prof. S. K. Rath. He has trained me and helped me become a hard-working student. A fair teacher, who encourages you always is all you need to be on the right track. He is that teacher for me. I am so thankful to him for continuously supporting me.

I'd also like to thank my seniors who have helped me understand things easily. Their guidance was most required and I am indebted to them for their assistance.

Last but not the least, I'd like to thank my friends for being with me always, helping me move forward in life and make the most out of life. I thank NIT Rourkela for giving me a handful of friends for life.

Abstract

The agile methodology of software development is 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, software developers are required to estimate the effort necessary to develop projects by agile methodology in an efficient manner because the requirements keep on changing. Web has become a part and parcel of our lives. People depend on Internet for almost everything these days. Many business units depend on Internet for communication with clients and for outsourcing load to other branches. In such a scenario, there is a necessity of efficient development of web-based software. For improving the efficiency of software development, resource utilization must be optimum. For achieving this, we need to be able to ascertain effectively, what kind of people/materials are required in what quantity, for development. This research aims at developing efficient effort estimation models for agile and web-based software by using various neural networks such as Feed-Forward Neural Network (FFNN), Radial Basis Function Neural Network (RBFN), Functional Link Artificial Neural Network (FLANN) and Probabilistic Neural Network (PNN) and provide a comparative assessment of their performance. The approach used for agile software effort estimation is the Story Point Approach and that for web-based software effort estimation is the IFPUG Function Point Approach.

Keywords: Software Effort Estimation, Agile Software Development, Story Point Approach, Friction factors, Normalized work effort

Contents

List of Figures

List of Tables

Abbreviations

ANN	Artificial Neural Network
FFNN	Feed-Forward Neural Network
RBFN	Radial Basis Function Neural Network
FLANN	Functional Link Artificial Neural Network
PNN	Probabilistic Neural Network
LFLANN	Legendre-Polynomial Functional Link Artificial Neural Network
CFLANN	Chebyshev's-Polynomial Functional Link Artificial Neural Network
FSM	Functional Size Measurement
FPA	Function Point Analysis
FP	Function Point
AFP	Adjusted Function Point
LOC	Lines of Code
XP	Extreme Programming
CMMI	Capability Maturity Model Integration
CFP	COSMIC Function Points
SVR	Support Vector Regression
BN	Bayesian Networks
SWR	Stepwise Regression
CBR	Case-based Reasoning
GA	Genetic Algorithm
BPN	Back-Propagation Algorithm
CART	Classification and Regression Trees
OLSR	Ordinary Least-Squares Regression
CHAID	Chi-squared Automatic Interaction Detection
ISBSG	International Software Benchmarking Standards Group

Chapter 1

Introduction

Software development effort estimation is the methodology of anticipating the most practical measure of exertion (conveyed as individual hours or capital) needed to create or keep up development tasks in light of inadequate, questionable and uproarious data. The process of effort estimation needs to be optimized because proper estimates are necessary both on the developer side as well as client side. On the developer side, estimates help in planning the development and monitoring the progress. While on the client side, they are used for negotiating contracts, setting completion dates, prototype release dates etc. Unfortunately, the estimates done in the present time are not much accurate. According to the Molokken and Jorgensen report [1], about 30-40% extra effort in terms of man month is spent in software development on an average.

1.1 Agile Software Development

In the present day scenario, customer's requirements keep on changing, and hence, the conventional methodologies of software development are not suitable. As a result, agile software development has come to the forefront [2]. It has been proved to be way more flexible than the traditional methodologies of software development [3]. It involves a repetitive method of software development that aims to hand-over working software as soon as possible and evolve it for meeting the changing requirements. Another advantage with agile methodology is that it emphasizes on good communication between the customers and the development team. Further, the amount of documentation is limited in the agile approach, which helps to reduce overheads in the software development process and incorporate changes as and when demanded by the customer without excessive rework [4].

1.1.1 Effort Estimation in Agile Software Development

Effort estimation in agile methods is done in an ad-hoc manner. Abstract estimating methods like numeric sizing (1 through 10), t-shirt sizes, the Fibonacci sequence, and even dog breeds are used [5]. These abstract methods have no mathematical basis, due to which optimal results can't be guaranteed. A standard mathematical model which can be validated using project data is essential. Unfortunately, such models based on these abstract estimation methods don't exist. A regression model based on the Story Point estimation method exists in the literature [6], but it's performance is not satisfactory. Some other models exist such as the bayesian model proposed by Hearty et. al. in [7] and the Support Vectore Regression (SVR) model proposed by Satapathy et. al. in [8]). The model proposed in [7] performs well but it is a complex bayesian model. The model proposed in [8] also performs well but it is not based on ANNs. Thus, simple yet efficient models need to be developed for this purpose.

1.2 Web-based software Development

With the rising use of dependency on Web, there is a necessity of quick and efficient development of web-based software. For developing web-based software efficiently i.e. without any cost or resource (human or otherwise) overrun, the estimates that are done before the beginning of development need to be correct.

1.2.1 Effort Estimation in Web-based Software Development

As per Reifer [9], effort estimation models, utilized for a long time as a part of conventional software development, are not extremely precise for effort estimation of web-based software development. Traditional software size and effort estimation techniques are not adequate to capture specific features of the development that can influence the size and effort required in the development of web applications [10].

FSM is a concept on step by step instructions to evaluate the software size in terms of functional requirements requested by a user. The first method that was developed to support this concept was FPA developed by Allan Albrecht in 1979 [11]. Albrecht defined a FP as a unit of measure that represents the amount of business functionalities an information system provides to a client. FPA techniques can be used as software sizing methods in effort estimation. The advantage of these methods lies in the fact that they are independent of technology or programming language used and can be used through the entire development life cycle [12]. With FPA method, the size of a software application and, the development effort of the software application at the beginning of the development process can be estimated, which might not be the case of other methods. Researchers have made attempts to adopt FP to be used in web applications [10]. Different approaches are proposed by various authors in the literature on estimation of web-based applications. Broadly, there are generally two approaches for sizing web applications [13]: LOC and FPA. There are also some other custom solutions that have been illustrated in [14].

1.3 Literature Survey

Having stated the concepts of agile and web-based software development, and the importance of effort estimation in them, now a section is dedicated for looking into the existing literature. The survey has been done in three parts, for Agile Software Effort Estimation, for Web-based Software Effort Estimation and for use of ANNs for effort estimation.

1.3.1 A review of studies on Agile Software Effort Estimation

Keaveney et al. [15] investigated the applicability of conventional estimation techniques towards agile development approaches by underscoring on the case studies of agile methods utilized within diverse organizations. Andreas Schmietendorf et al. [4] have provided an investigation about estimation possibilities, especially for the extreme programming paradigm.

Coelho et al. [16] have 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 for further research. Ziauddin et al. [6] have developed an effort estimation model for agile software projects, where the model was fine-tuned with the help of the empirical data acquired from twenty one software projects. Hearty et al. [7] have proposed a Bayesian network model of an XP surrounding and indicated how it could gain from project data keeping in mind the end goal to predict the effort and risk appraisals without obliging any extra metrics. Satapathy et al. have used SVR-Kernel methods for optimizing the effort calculated using story point approach in [8]. Out of the four kernels used, the RBF-Kernel is shown to have comparably better accuracy.

Hussain et al. [17] have 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. [18] have introduced a process and methodology that guarantees relativity in software sizing while using agile story points. This proposed process and methodology was applied in a CMMI level three company on different projects. Ungan et al. [19] have compared SCRUM's native effort estimation method Story Points and poker planning, with effort estimation models based on CFP for a selection of projects by using regression models and ANN methodology and proved that COSMIC measurement is a better method for effort estimation than SCRUM's story points.

1.3.2 A review of studies on Web-based Software Effort Estimation

Filomena Ferrucci et. al.[20] have compared the prediction accuracy of effort estimation models built using single company data set and cross-company data set. They developed models using Manual Stepwise Regression, Linear Regression and Case-Based Reasoning on the Tukutuku data set. Ferrucci et. al. [21] also enquired the effectivity of Tabu Search in estimating effort for Web-based software development.

Idri et. al. [22] have designed Radial Basis Function Networks for software effort estimation using the COCOMO81 and Tukutuku datasets. Kitchenham et. al. [23] have used manual forward SWR to develop effort models and enquired about the applicability of cross-company and within-company cost estimation model for Web projects. Corazza et. al. [24] have developed SVR models for web development effort estimation using cross-company data set.

Mendes et.al. [25] have described a case study in which Bayesian networks were used to construct an expert-based Web effort model. Mendes et. al. [26] have developed three Case-based reasoning models for web effort estimation and found out the best one, then compared it with stepwise regression and regression trees. Mendes et. al. [27] further looked into the use of BN for effort estimation in web-based software development when a cross-company dataset is employed. Emilia Mendes [28] employed four techniques for web effort estimation BN, forward SWR, CBR and CART to obtain effort estimates and compared them. Mendes et. al. [29] investigated to what degree a cross-company cost model can be fruitfully utilized to estimate effort for projects that are owned by a single company, if none of these projects were used to construct the cross-company model. The models were developed using two techniques, forward SWR and CBR.

Di Martino et. al. [30] enquired the potency of the Web Objects measure as an indicator of Web-based software development effort. The effectiveness of the Web Objects measure as indicator of Web application development effort was confirmed, when assembled with OLSR and WebCOBRA, and this is true even when using CBR. It was observed that the Web Objects method yields better results than the FPA method when assembled with OLSR and Web-COBRA.

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

While going through the literature, it is observed that ANNs are frequently used for effort estimation purposes. Hence, a review on the use of ANNs for effort estimation has been prepared.

Wen et al. [31] reviewed various machine learning techniques-based software effort estimation models. It was observed from the analysis that machine learning-based effort estimation models performs better than non-machine learning models.

Idri et al. [32] have designed another RBFN network-based model for the purpose of software development effort estimation. Results showed that the estimates produced by RBFN network model are greatly improved on using an adequate formula for width. A FLANN [33] was proposed by Rao et. al. for effort estimation and to lessen the computational complexities so that the neural net becomes useful for on-line applications.

Parag C. Pendharkar [34] have proposed a PNN approach for predicting a software development parameter and a probability measure which denotes the chances of actual value of the parameter being less than its estimated value at the same time. This PNN approach was then compared with CHAID. Results indicate that PNN performs similar to the CHAID, but provides superior probability estimates.

Adriano L. Oliveira et al. [35] have proposed and investigated the application of the GA for selecting an optimized feature subset and improving SVR parameters all the while aiming at enhancing the exactness of the software effort estimates.

1.4 Why use ANNs for Software Effort Estimation?

The reason for choosing ANNs for this purpose can be attributed to the following facts:

- ANNs have been used in the past [36–40] for developing efficient effort estimation models due to their inherent learning ability and good interpretability.
- They are perceived for their capacity to give great results when managing issues where there are complex connections in the middle of inputs and yields, and where the information is bended by high commotion levels.

1.5 Motivation

Simple neural networks are very efficient (as observed from previous section and from literature survey). But they haven't been used for effort estimation in web-based projects. Also, no efficient neural network model (except [7]) has been designed for estimating the effort in agile software.

- For agile and web-based software, ANNs can be employed for obtaining simple yet efficacious effort estimation models with good prediction accuarcy.

1.6 Objectives of The Research

The main objective of the present work is to build efficient effort estimation models using various types of ANNs showing lower values of error and higher levels of prediciton accuracy values.

1.7 Organization of The Thesis

The rest of the thesis is organized as follows:

Chapter 2: Basic Concepts and Performance Metrics: This chapter describes the basic concepts required for estimating the development effort of agile and web-based software. The various performance metrics which are used for assessing the performance of the ANN models are also described.

Chapter 3: Proposed Work For Agile Software Effort Estimation: This chapter includes a detailed description of the approach proposed for effort estimation in agile software. The model design process is given, along with the graphical representation of the model's performance, for all the ANN models implemented. Finally, all the ANN models are compared by using the performance measures mentioned in Chapter 2.

Chapter 4: Proposed Work For Web-based Software Effort Estimation: This chapter includes a detailed description of the approach proposed for effort estimation in web-based software. The model design process is given, along with the graphical representation of the model's performance, for all the ANN models implemented. Finally, all the ANN models are compared by using the performance measures mentioned in Chapter 2.

Chapter 5: Conclusion and Future work: This chapter culminates the work done in this research, by suggesting possible future work.

Chapter 2

Basic Concepts and Performance Metrics

2.1 Neural Networks Used in this Study

Four different types of artificial neural networks and their variants are used here:

- Feed-Forward Neural Network
- Radial Basis Neural Network
- Functional Link Artificial Neural Network
- Probabilistic Neural Network

2.1.1 Feed-Forward Neural Network (FFNN)

It is composed of an input layer and an output layer with some hidden layers [43]. This neural network is trained with the help of BPN algorithm. The basic structure of a FFNN is shown in Figure 2.1. This FFNN has four nodes in both input and hidden layers and one node in the output node. In this paper, for input layer, linear activation function has been used; that is, the input of the input layer "I_i" becomes output of the input layer "O_i". For calculating outputs at hidden layers and output layers, the unipolar Sigmoid function is considered. The output of hidden layer O_h is represented as follows:

$$O_h = \frac{1}{1 + e^{-I_h}} \tag{2.1}$$

where I_h is the input to the hidden layer. Output of the output layer "O_o" is represented as follows:

$$O_o = \frac{1}{1 + e^{-O_i}} \tag{2.2}$$

where O_i is the input to the output layer. A neural network thus can be represented as follows:

$$EO' = f(W, EI) \tag{2.3}$$

where EI is the input vector, EO' is the output vector, and W is the weight vector. The weight vector W is updated in every iteration of training so as to reduce the value of mean square error (MSE).

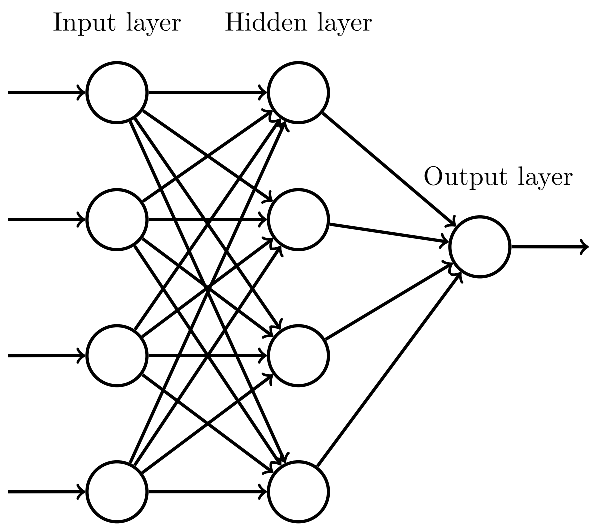

FIGURE 2.1: Basic Structure of FFNN

2.1.2 Radial Basis Function Networks (RBFN)

RBFN is a feed-forward type of ANN [44] which is trained with the help of supervised training algorithm. RBFN generally contains a single hidden layer, where the **basis functions** are used as activation functions. RBFN contains h number of hidden centers represented as $C_1, C_2, ..., C_h$. The basic structure of a RBFN is shown in Figure 2.2. The input in this diagram is p-dimensional and the hidden layer has h centers. The euclidean distance between inputs and centers is calculated and then the activation function is applied. The outputs of the activation function from all the hidden nodes are multiplied by weights and summed to get final output.

The target output is computed as follows:

$$y' = \sum_{i=1}^{n} \phi_i W_i \tag{2.4}$$

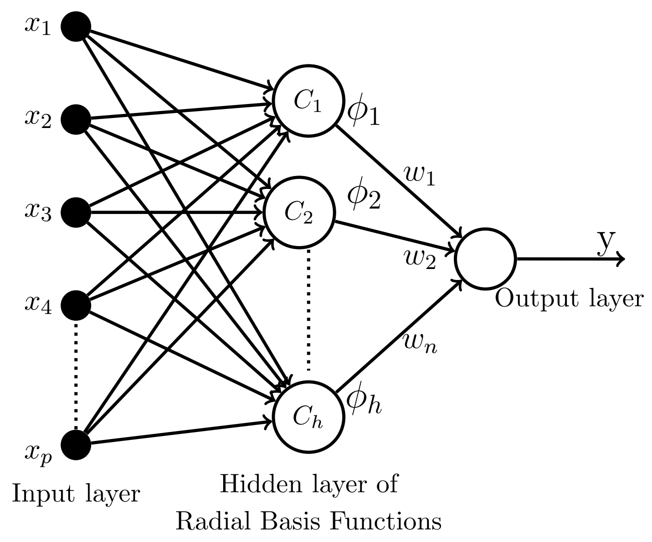

FIGURE 2.2: Basic Structure of RBFN

where W_i is the weight of the ith center, ϕ is the radial function, and y' is the target output. In this paper, the basis function used is the Gaussian function, and the distance vector is calculated as follows:

$$z = ||x_j - c_j|| \tag{2.5}$$

where x_j is input vector that lies in the receptive field for center c_j. The activation function is defined as:

$$\phi_i = \frac{e^{-z_i^2}}{2\sigma^2} \tag{2.6}$$

2.1.3 Function Link Artificial Neural Network (FLANN)

FLANN, initially proposed by Rao [33], is a flat network having a single layer; that is, the hidden layers are omitted. Input variables are converted into a collection of one-dimensional independent functions i.e., the inputs are functionally expanded using polynomials for a 1 to n expansion. As shown in Figure 2.3, the inputs X_1 and X_2 are first expanded using polynomials and then multiplied by weights and summed up. The adaptive algorithm used for learning in this network is BPN algorithm.

The first 'n' terms of the polynomial series are found out using the input from data set. Hence, for one input parameter of the network, n different terms are obtained, which form the input layer nodes. The output in FLANN is calculated as follows:

$$\hat{y} = \sum_{i=1}^{n} W_i Z_i \tag{2.7}$$

where $\hat{y}$ is the predicted value, W is the weight vector, and Z is the functional block, and can be defined as follows (this is one of the many possible functional expansions):

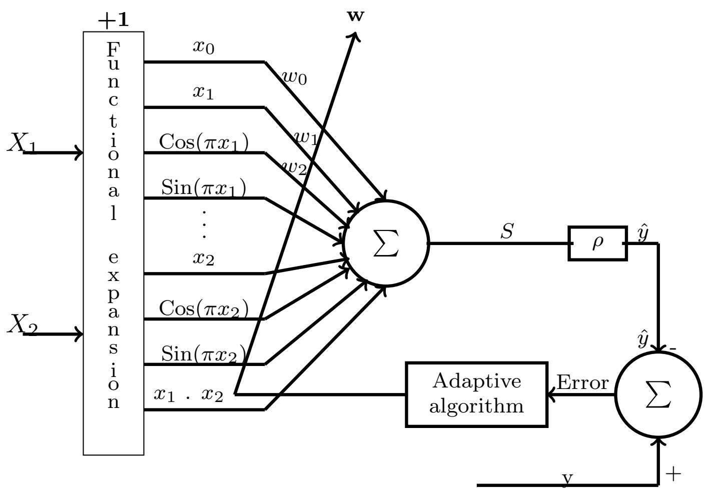

FIGURE 2.3: Basic structure of FLANN

$$\begin{aligned} Z = [&1, z_1, sin(\pi z_1), cos(\pi z_1), \\ &z_2, sin(\pi z_2), cos(\pi z_2), ...] \end{aligned} \tag{2.8}$$

This is an example of a polynomial used for functional expansion. Many such polynomials can be used like Legendre polynomial, Chebyshev's polynomial etc.

2.1.4 Probabilistic Neural Network (PNN)

It employs a supervised learning algorithm, which is a bit different from BPN algorithm [34]. It has a feed forward architecture. There are no weights in its hidden layer. An example vector is associated with each hidden node, which acts as the weights to that hidden node. A PNN is comprised of an input layer, which is basically the input vector. This input layer is entirely interlinked with the middle/hidden layer (which is associated with example vectors). Finally, the output layer constitutes each of the potential classes for which the inputs can be classified. The connections of a PNN can be well understood by the Figure 2.4. It shows example vectors for two classes and the respective layer organization.

The hidden layer is not entirely connected to the output layer. The hidden layer example vector nodes for a particular class are linked to only the output node of the respective class. For every class node, the sum of example vector activations is found out. Each hidden node's activation is the product of the example vector (E) and the input vector (F).

$$H_i = E_i F \tag{2.9}$$

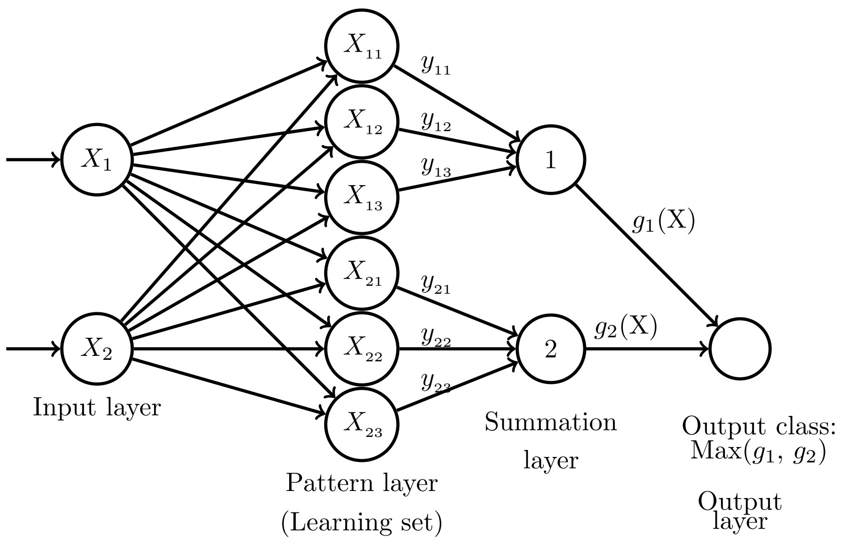

FIGURE 2.4: Basic structure of PNN

The class output activation is found out as:

$$C_j = \frac{\sum_{i=1}^{N} e^{\frac{H_i - 1}{\gamma^2}}}{N} \tag{2.10}$$

where N is the number of example vectors for this class, H_i is the activation of the hidden node and γ is the smoothing parameter. The class, for which the input layer conforms to, is determined through a winner-takes-all approach (i.e., the winning class is the output class node having the maximum class node activation).

2.2 Performance Measures

After the implementation of neural networks, their performance is assessed by using many performance metrics. These metrics are outlined below: [45]

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

$$MSE = \frac{\sum_{i=1}^{TP} (AE_i - PE_i)^2}{TP} \tag{2.11}$$

where
AE_i = Original effort value collected from the dataset for the i^{th} test data,
PE_i = Output (effort) obtained using the developed model for the i^{th} test data and TP = Total no. of projects in the test set.

- The **squared correlation coefficient ($\mathbf{R}^2$)**, otherwise called as the *coefficient of determination* is calculated as:

$$R^2 = 1 - \frac{\sum_{i=1}^{TD}(AE_i - PE_i)^2}{\sum_{i=1}^{TD}(AE_i - \bar{AE})^2} \tag{2.12}$$

where $\bar{AE}$ = Mean of Actual Effort Value.

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

$$PRED = (1 - (\frac{\sum_{i=1}^{TP} |AE_i - PE_i|}{TP})) * 100 \tag{2.13}$$

Chapter 3

Proposed Work For Agile Software Effort Estimation

3.1 Dataset Description

The dataset used in [6] is used here. It consists of the fields shown in the table below:

Effort	Vi	D	V	Sprint Size	Work Days	Team Salary	Time	Cost

Out of these fields, three are used in this study:

- Effort
- Velocity (V)
- Time

Effort and Velocity are taken as inputs to the neural networks and Time is considered as the output. The estimations done by ANNs are compared by Time to calculate prediction accuracy.

3.2 Proposed Methodology

The proposed approach is implemented using the twenty-one project data set developed by six software houses [6]. In the data set table, every row contains three columns. The first column indicates the number story points required to complete the project, the second column represents the velocity of the project, and the third column represents the actual effort (in terms of completion time) required to complete that project. This data set is used to determine agile software development effort and to assess the performance. The results obtained in the validation process prove the effectiveness of story point approach. The block diagram, demonstrated in figure 3.1, states the proposed steps used to compute the predicted effort using various neural networks.

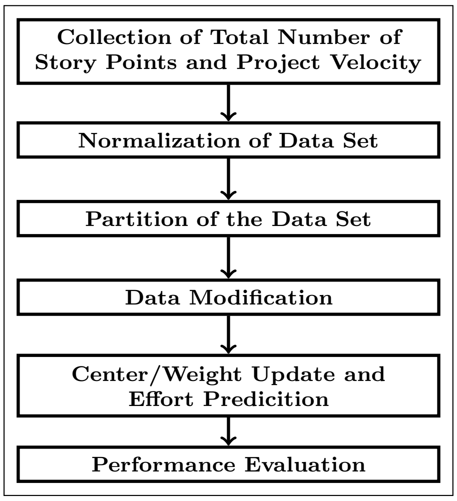
Collection of Total Number of Story Points and Project Velocity
Normalization of Data Set
Partition of the Data Set
Data Modification
Center/Weight Update and Effort Predicition
Performance Evaluation

FIGURE 3.1: Proposed Steps to Estimate Effort using Various Neural Networks

The steps taken to determine the effort of a software product to be developed by the agile methodolgy are outlined below.

Steps in Effort Estimation

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

2. **Normalization of Data Set**: The generated number of story points and project velocity values are employed as input arguments and are normalized in the range [0,1]. Let S be the data set and s is an element of the data set, then the normalized value of s i.e., s' is determined as :

$$s' = \frac{s - \min(S)}{\max(S) - \min(S)} \tag{3.1}$$

   where
   s' = Normalized value of S within the range [0,1].
   $\min(S)$ = min. value of S.
   $\max(S)$ = max. value of S.
   When $\max(S) = \min(S)$, $s' = 0.5$.

3. **Partition of the Data Set**: The data set is partitioned into learning set and validation set using 5-fold cross validation.

4. **Data Modification**: After partitioning the data set, in case of FFNN, RBFN and PNN, it was directly used for training the model. But in FLANN, the input data is functionally expanded using different types of polynomials like Chebyshev, Legendre, Power Series and Trigonometric.
5. **Center/Weight Update and Effort Predicition**: While training, only neural network weights are updated in case of FFNN and FLANN. But in case of RBFN Gradient-Descent learning, both centers as well as weights are updated. Validation data is applied to the model obtained at the end of training and the effort is predicted. This doesn't apply to off-line techniques (RBFN Pseudo-Inverse learning and PNN). Only their algorithms are run and the effort value is predicted by using learning and validation data simultaneously.
6. **Performance Evaluation**: The performance of various neural network models is assessed by evaluating MSE, R^2 and PRED values obtained from test samples.

The effort models (developed using various ANNs) are implemented using the above steps. Finally, a comparison of results obtained using various ANN-based effort estimation models is presented to critically examine the performance of individual techniques.

3.3 Experimental Details

For implementation, the data set available in [6] is used. The inputs to different neural network models are total number of story points and project final velocity and the output is the effort (completion time). ANNs (online ones i.e. FFNN, RBFN Gradient Learning and FLANN) are trained for certain no. of epochs/iterations. The error parameter, MSE is calculated in every iteration. After some epochs, the MSE value becomes constant. At that point, MSE value saturates and the best weights are obtained. The weights corresponding to saturation point are taken and used for testing. The performance of the effort estimation models are shown in the graphs.

3.3.1 Model Design Using FFNN

The input layer of FFNN has two nodes, one for total no. of Story Points and the other for project velocity. And the output layer has one node, representing the effort (i.e. the completion time). The hidden layer has three nodes. The BPN algorithm is employed for optimizing the weights of the neural network in an iterative manner.

MSE value is calculated in every iteration. After certain number of epochs, the MSE value becomes constant. At this point, MSE value saturates and the best weights are obtained. These weights are used for testing. The proposed model shown in figure 3.2 is generated using the FFNN technique.

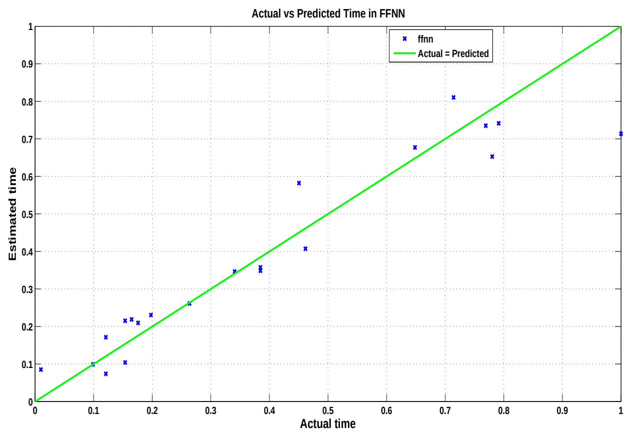

FIGURE 3.2: Effort Estimation Model using FFNN based on Story Points

Figure 3.2 shows the deviation of predicted effort value from actual effort value for FFNN. Nearer the points to the actual = predicted line in the figure, better is the prediction accuracy, because it shows that the degree to which the estimations are correlated to the actual values is high.

3.3.2 Model Design Using RBFN

RBFN model is implemented using the Gradient-Descent learning technique and the Pseudo-Inverse learning technique. Centers are initialized using three clustering techniques like K-means, Fuzzy C-means and Random. Gaussian function is used as the activation function.

3.3.2.1 Model Design Using RBFN Gradient Learning

Gradient Descent approach is one of the most popular approaches for updating centers and weights. It is a supervised training method which uses an error correcting term. The update rules are developed by differentiating the cost function (which is the error). Both weights and centers are updated while training.

MSE value is calculated in every iteration. After certain number of epochs, MSE value becomes constant. At this point, MSE value saturates and the best weights are obtained. These weights are used for testing.

The proposed model shown in figure 3.3 is generated using the RBFN technique implementing Gradient Descent learning algorithm. This figure shows the deviation of

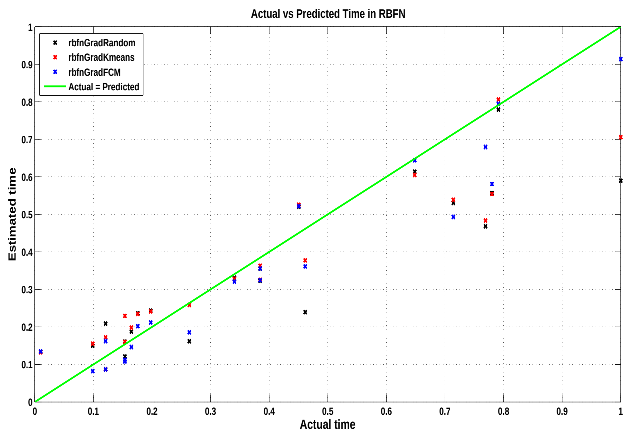

FIGURE 3.3: Effort Estimation Model using RBFN with Gradient Descent Learning based on Story Points

predicted effort value from actual effort for four types of clustering technique-based RBFN. The points nearer to the ideal prediction line shows more accurate prediction of effort.

3.3.2.2 Model Design Using RBFN Pseudo-Inverse Learning

Pseudo-Inverse learning is a least square problem. It employs gaussian functions for acting as radial centers. Centers are chosen only once. No further updates to the centers are made by this algorithm. It is an off-line technique in which no real training (network weights are not improved iteration by iteration; hence optimal weights are not obtained) takes place. This explains the reduced accuracy in prediction of this method. Since no real training takes place in Pseudo-Inverse learning, therefore the variation between MSE and epochs cannot be identified. The proposed model shown in Figure 3.4 is yielded using the RBFN technique implementing the Pseudo-Inverse learning algorithm.

Figure 3.4 shows the deviation of predicted effort value from actual effort for four types of clustering technique-based RBFN implementing the Pseudo-Inverse learning algorithm. The points nearer to the ideal prediction line show more accurate prediction of effort. Because of no real training, the best weights cannot be obtained. This justifies the sub-optimal performance and reduced prediction accuracy.

3.3.3 Model Design Using FLANN

In FLANN, the inputs need to be functionally expanded before they can be applied to the network. For functional expansion of inputs, four different polynomials were used such as Chebyshev, Legendre, Power Series and Trigonometric. The inputs are functionally

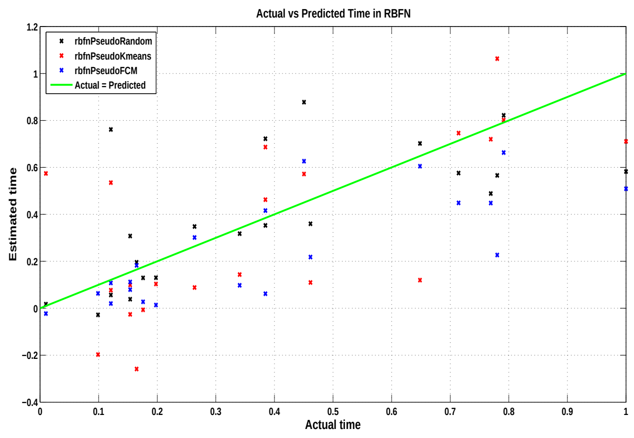

FIGURE 3.4: Effort Estimation Model using RBFN with Pseudo-Inverse Learning based on Story Points

expanded using polynomials for a 1 to n expansion. The first n terms of the polynomial series are found out using the input from data set. Hence, for one input parameter, n different terms are obtained, which form the input layer nodes. This process is called as *functional expansion.*

MSE value is calculated in every iteration. After certain number of epochs, MSE value becomes constant. At this point, MSE value saturates and the best weights are obtained. These weights are used for testing.

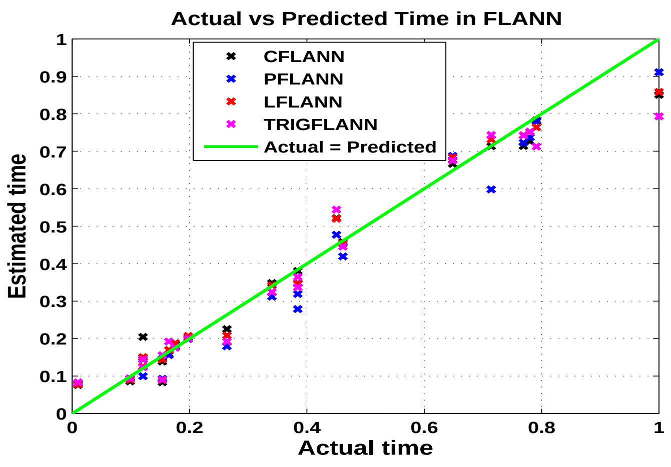

FIGURE 3.5: Effort Estimation Model using FLANN based on Story Points

The proposed model shown in figure 3.5 is generated using the FLANN technique. This figure shows the deviation of predicted effort value from actual effort for four type of polynomial technique-based FLANN. The points nearer to the ideal prediction line show more accurate prediction of effort.

3.3.4 Model Design Using PNN

Probablistic networks are mostly used for classification. For regression analysis i.e. employing PNN for prediction of continuous targets, first of all the number of classes in the dataset need to be found out. This can be done using any clustering mechanism. In this paper, K-means clustering technique is used. After finding out the number of classes and the inputs included under each class, some input vectors from each class are taken as **example vectors** and the dot product of example vectors and input vectors is found out. Then after applying the function mentioned above in section 2, the class node's activations are found out. The class node with highest activation is considered to be the predicted class for the current input and the numerical value (activation value) is used for performance evaluation.

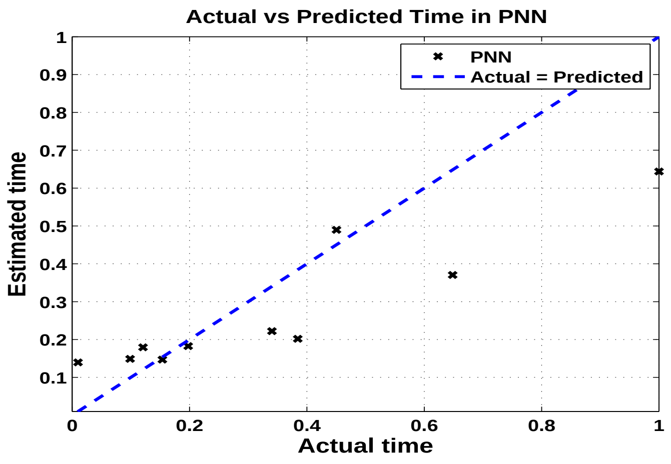

FIGURE 3.6: Effort Estimation Model using PNN based on Story Points

Figure 3.6 shows the deviation of predicted effort value from actual effort for PNN. The points nearer to the ideal prediction line show more accurate prediction of effort. The lower prediction accuracy is reflected by the points scattered away from actual = predicted line. Since the network isn't trained, optimal weights are not obtained. This leads to lower prediction accuracy.

3.4 Comparison of Results Obtained

Using the results obtained, the estimated effort value using FFNN, RBFN, FLANN and PNN are compared. Table 3.1 compares the work existing in literature with the proposed best and worst model. It can be seen that the proposed LFLANN model outperforms all the existing models [6, 8] and the worst model performs better than that provided by [6].

Table 3.1: Comparison of Proposed Models with Existing Work

	PRED (in %)
Regression Analysis [6]	57.14
Support Vector Regression (SVR) Linear Kernel [8]	90.8112
Support Vector Regression (SVR) Polynomial Kernel [8]	68.7382
Support Vector Regression (SVR) RBF Kernel [8]	95.9052
Support Vector Regression (SVR) Sigmoid Kernel [8]	89.7646
Proposed Approach (Best)	96.42
Proposed Approach (Worst)	77.69

Table 3.2: Comparison of Proposed Models

	MSE	**R^2**	**PRED**
FFNN	0.0069	0.9295	94.0207
LFLANN	**0.0019**	0.9744	**96.4172**
PFLANN	0.0041	0.9535	94.3667
CFLANN	0.0033	0.9561	95.4782
TrigFLANN	0.006	0.9639	94.3076
RBF_Gradient_FCM	0.0074	0.8748	93.7716
RBF_Gradient_K-Means	0.0134	0.8445	91.8474
RBF_Gradient_Random	0.0192	0.8373	90.4400
RBF_Psuedo_FCM	0.0494	0.6863	83.6470
RBF_Psuedo_K-Means	0.0767	0.6028	77.6910
RBF_Psuedo_Random	0.0528	0.6878	84.0325
PNN	0.0276	0.6914	87.6561

Table 3.2 compares the performance of the ANN-based models implemented. FFNN sometimes get stuck at local minima thus yielding abnormally high accuracy values (when there are no improving neighbors, but in this case it clearly does not) while RBFN does not. Thus, FFNN trained with BPN learning shows more efficiency than RBFN. As it can be observed that, FFNN's performance is very good (prediction accuracy of 94.0207 %).

Radial basis function networks have the detriment of getting the input space covered by radial basis functions. RBF centres are determined by taking into account the input data distribution, with no reference to the prediction task. Due to this, resources may be thrown away on such areas of the input distribution that are insignificant to learning. This is the reason for RBFN's lower accuracy and higher error as compared to FFNNs (also proved by [46]). Each of the variants of RBFN has higher MSE value than that of FFNN. Among the variants of RBFN Gradient Descent learning, the one with initial FCM clustering performs best because FCM is the best clustering technique among the ones used here [47, 48]. The networks implemented using RBFN Pseudo-Inverse learning yield poor results, because pseudo-learning is an off-line technique. No real training occurs i.e., network parameters are not given optimal values. No network optimization criteria is employed for getting the best weights.

FLANN is known for it's lower computational complexity i.e. ability to handle non-linearity and low error rates [33]. Here also, it performs accordingly (lower error rates as compared to FFNN and RBFN). Each of the variants of FLANN has lower MSE value than that of FFNN and RBFN. As FLANN is capable of handling complexities of the input space and performs same as multilayer networks, it's prediction accuracy is the best among all. All the variants of FLANN perform almost the same. L-FLANN performs the best. PNN solves the optimization problem in an off-line manner, hence it has low accuracy of prediction. LFLANN gives best values for prediction accuracy (highest) and MSE (lowest) among all the techniques used in this work.

Chapter 4

Proposed Work For Web-based Software Effort Estimation

4.1 Dataset Description

The ISBSG dataset, Release 12 [41] is used in this study for developing effort estimation models for web-based software. It has 6006 rows of information pertaining to various project attributes like Project ID, Software Quality Rating, Software Age, Organisation Type, Development Type, Development Platform, Programming Language used, sizing attributes etc.

4.2 Proposed Methodology

The proposed approach is implemented using the ISBSG, Release 12 dataset. Figure 4.1, demonstrates the steps carried out in the proposed research work applied to compute the effort required to develop web-based applications using several neural networks.

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

1. **Collection of the web-based effort data**: The data used for developing effort estimation models is ISBSG Release 12 data. It is obtained from the ISBSG community.

2. **Filtering and Division of the Dataset:** The ISBSG data is filtered by using the following attributes:

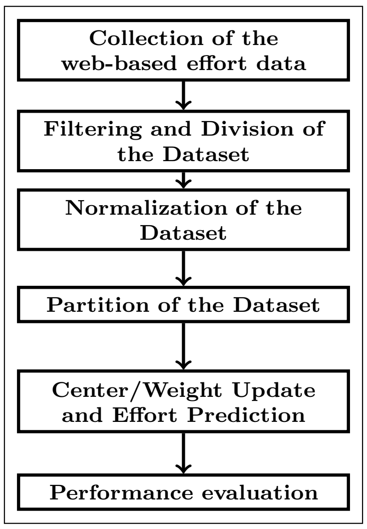
Collection of the web-based effort data
Filtering and Division of the Dataset
Normalization of the Dataset
Partition of the Dataset
Center/Weight Update and Effort Prediction
Performance evaluation

FIGURE 4.1: Proposed Steps Used for Web Effort Estimation using Neural Networks

- Web Architecture: Only web-based applications and web projects are taken into consideration.
- Data Quality Rating: Projects with a data quality rating of A and B are taken into consideration.
- Unadjusted Function Point Rating: Projects with an unadjusted function point rating of A and B are taken into consideration.

After the data is filtered, the development type attribute is used for separating the data into three major groups:

- Newly developed projects
- Enhanced projects
- Re-developed projects

After filtering and division, the Mean or Mode Single Imputation (MMSI) Method is used for filling the missing values. In this method, the missing values of an attribute are filled with the Mean value of that attribute.

3. **Normalization of the Dataset**: This step deals with generating the normalized values of the input vectors with in the range [0,1]. Let us consider Y as complete dataset and y as an element of the dataset, then normalized value of y is calculated as:
$$y' = \frac{y - \min(Y)}{\max(Y) - \min(Y)} \tag{4.1}$$

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

4. **Partition of the Dataset**: The entire dataset is partitioned into learning set and validation set using 5-fold cross validation.

5. **Center/Weight Update and Effort Prediction**: While training, only weights are updated in case of FFNN and FLANN. But in case of RBFN Gradient Descent Learning, both centers as well as weights are updated. In RBFN, centers are initialized both randomly and by using two other clustering techniques: Fuzzy C-means and K-means clustering. The network is trained only for on-line algorithms i.e., in those algorithms where the process is repeated for a certain number of iterations till certain criteria is fulfilled. In this study, the criteria is the saturation of MSE. Saturation means that MSE stays constant for some consecutive iterations, ten in this work. Since off-line algorithms (RBFN Pseudo-Inverse Learning and PNN) don't perform well, their results are not shown here.

6. **Performance Evaluation**: The performance of the models is accessed using MSE, R^2 and PRED values are obtained from test samples. The model giving lower values of MSE and higher values of PRED is considered as the best model.

The results obtained by using the above models are compared to assess their performance.

4.3 Experimental Details

After processing the input, the following number of data entires for each type are obtained:

- New development: 161 rows
- Enhancement Projects: 234 rows
- Re-development Projects: 12 rows

The re-development project data is too small, so it's not used. Only new and enhanced web project's data are used.

Adjusted Function Points (AFP) attribute is considered as input and Normalized Work Effort is considered as output of the effort estimation models. The reasons behind this are given below:

- The final stage of estimation in most of the Functional Size Measurement methods such as IFPUG, NESMA, and MARK-II calculates the effort from the AFP value. And this value is available in ISBSG Release 12 dataset. Whereas using the other attributes from initial phases of counting (like ILF, EIF, DET RET, EI, EO, VAF etc.) is not possible as these values are not individually provided in the dataset. Eventually, all calculations lead to AFP value, which is then used for calculating the final effort. So, AFP is taken as the input.
- Normalized Work Effort is the effort value of the full development cycle; whereas Summary Work Effort presents the total effort computed in terms of person-hours documented against the project. For some projects, the full development cycle time is not covered and Normalized Work Effort attribute is an approximation for full cycle effort. In other cases, where the full development cycle time is covered, the Normalized Work Effort and Summary Work Effort are same. Whether a project has completed the full developmental cycle time or not is not provided in the ISBSG dataset. So it is proposed to take the Normalized work effort as the output / Effort value instead of Summary Work Effort. It is more appropriate than Summary Work Effort.

After developing the models, a graph is drawn between actual effort and the estimated effort in all cases. This graph shows the performance of the model. Nearer the points to the 'Actual = Predicted' line, greater is the accuracy of the model (because the degree of correlation between actaul and predicted effort is higher).

4.3.1 Model design using FFNN

In this paper, a structure with three layers of FFNN is considered, in which one input node is used, four nodes are used for the hidden layer, and the output layer has one node. The BPN algorithm is employed for optimizing the weights of the neural network in an iterative manner. Training is done until the MSE value saturates i.e. stays constant for a certain number of iterations (10 here).

The proposed model shown in fig. 4.2 and 4.3 are generated using the FFNN technique for new and enhanced web projects respectively. These figures show the deviation of predicted effort value from actual effort value for FFNN. The middle line in the graph is

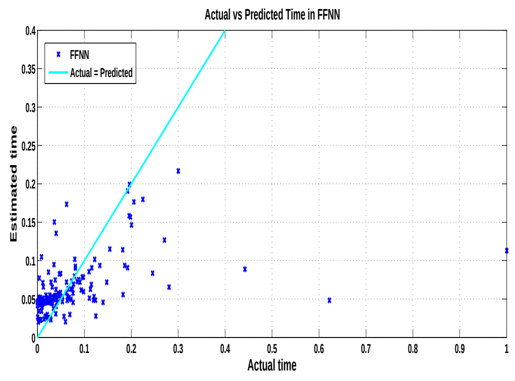

FIGURE 4.2: Effort Estimation Model using FFNN for new web projects

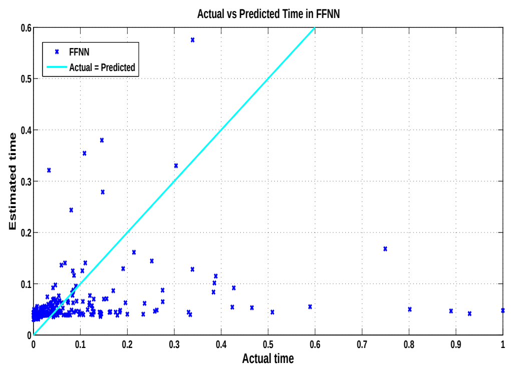

FIGURE 4.3: Effort Estimation Model using FFNN for enhanced web projects

a normal y = x line. Nearer the points to this line in the figure, better is the prediction accuracy.

4.3.2 Model design using RBFN

RBFN is implemented using the Gradient Descent learning and the Pseudo-Inverse learning algorithm. Initially, for both the learning methods, centers are initialized using various clustering techniques like Kmeans, Fuzzy C-means and Random.

4.3.2.1 Model Design Using RBFN Gradient Descent Learning

Gradient Descent approach is one of the most popular approaches for updating centers and weights. It is a supervised training method which uses an error correcting term. The update rules are developed by differentiating the cost function (which is the error). Both weights and centers are updated while training.

The proposed model shown in fig. 4.4, 4.5 and 4.6 are generated using the RBFN technique implementing gradient learning algorithm for new web projects respectively.

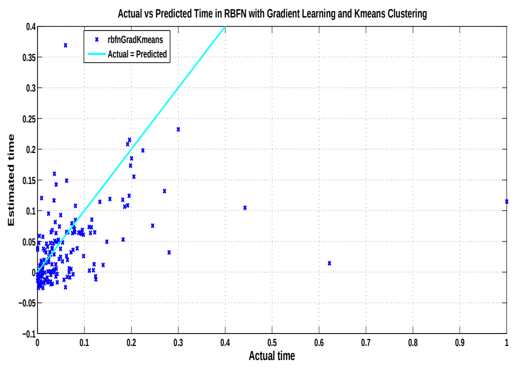

FIGURE 4.4: Effort Estimation Model using RBFN with Gradient Descent Learning and K-means Clustering for new web projects

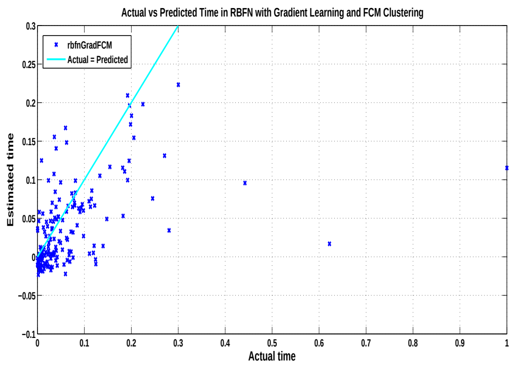

FIGURE 4.5: Effort Estimation Model using RBFN with Gradient Descent Learning and FCM Clustering for new web projects

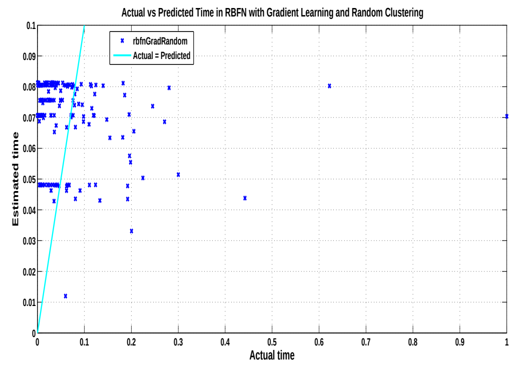

FIGURE 4.6: Effort Estimation Model using RBFN with Gradient Descent Learning and Random Clustering for new web projects

Likewise, fig. 4.7, 4.8 and 4.9 display the proposed model generated using the RBFN technique implementing Gradient Descent learning algorithm for enhancement type web projects.

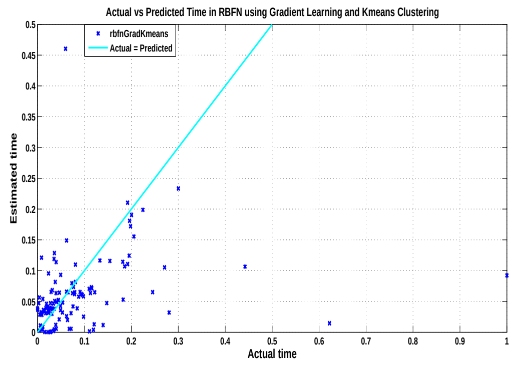

FIGURE 4.7: Effort Estimation Model using RBFN with Gradient Descent Learning and K-means Clustering for enhanced web projects

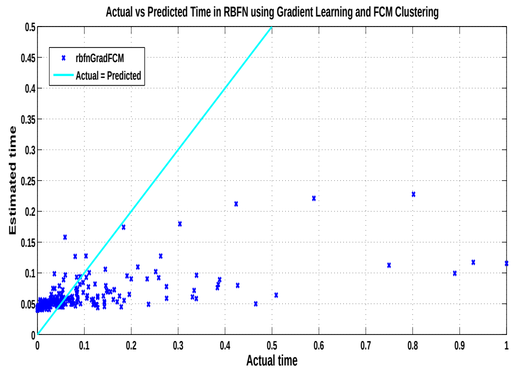

FIGURE 4.8: Effort Estimation Model using RBFN with Gradient Descent Learning and FCM Clustering for enhanced web projects

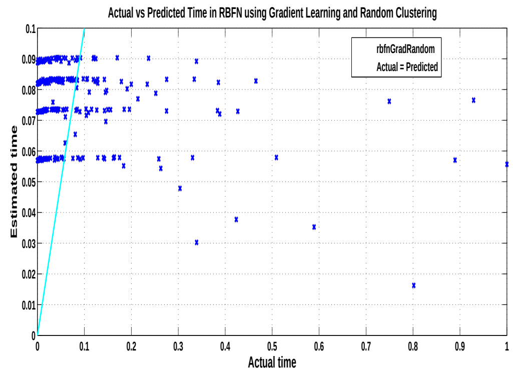

FIGURE 4.9: Effort Estimation Model using RBFN with Gradient Descent Learning and Random Clustering for enhanced web projects

4.3.3 Model design using FLANN

In case of FLANN, the inputs are functionally expanded using polynomials for a 1 to n expansion. The first n terms of the polynomial series are found out using the input from dataset. Hence, for one input parameter, n different terms are obtained, which forms the input layer nodes. This process is called as *functional expansion.* For functional expansion of inputs, four different polynomials are used. These are Chebyshev, Legendre, Power Series and Trigonometric polynomials. Five inputs are expanded to first five terms (n = 5) of the series thus giving twenty-five inputs. Thus there are twenty-five input nodes. Fifty nodes are considered in the hidden layer and one node as the output node.

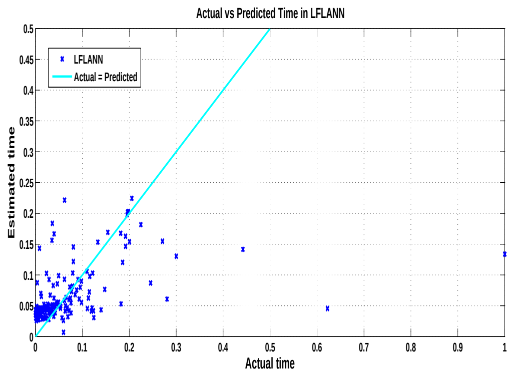

FIGURE 4.10: Effort Estimation Model using LFLANN for new web projects

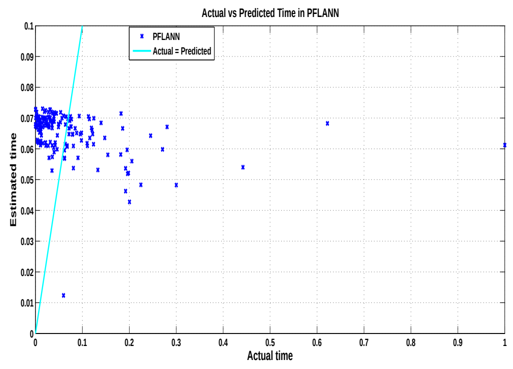

FIGURE 4.11: Effort Estimation Model using PFLANN for new web projects

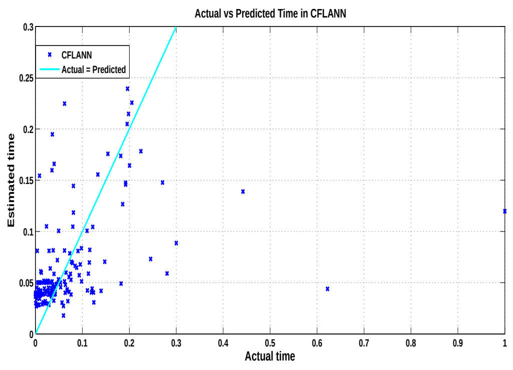

FIGURE 4.12: Effort Estimation Model using CFLANN for new web projects

The proposed model shown in fig. 4.10, 4.11, 4.12 and 4.13 are generated using the four types of FLANN techniques for new web projects. This figure shows the deviation of predicted effort value from actual effort for four type of polynomial technique-based FLANN.

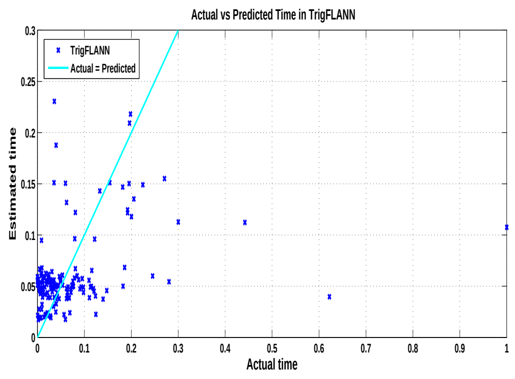

FIGURE 4.13: Effort Estimation Model using TrigFLANN for new web projects

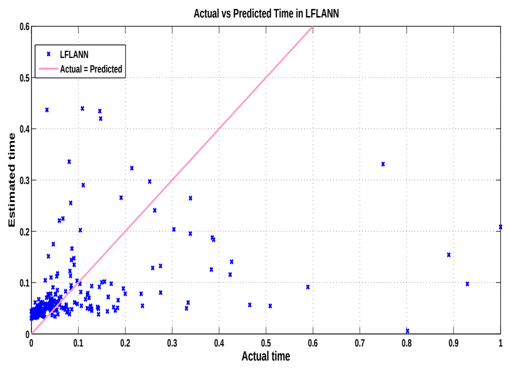

FIGURE 4.14: Effort Estimation Model using LFLANN for enhanced web projects

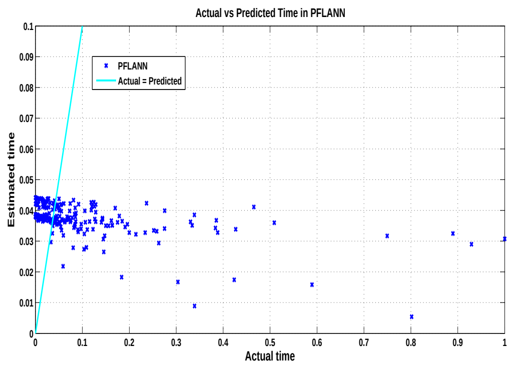

FIGURE 4.15: Effort Estimation Model using PFLANN for enhanced web projects

The proposed model shown in fig. 4.14, 4.15, 4.16 and 4.17 are generated using the four types of FLANN techniques for enhancement type of web projects respectively. These figures show the deviation of predicted effort value from actual effort for four type of polynomial technique-based FLANN models.

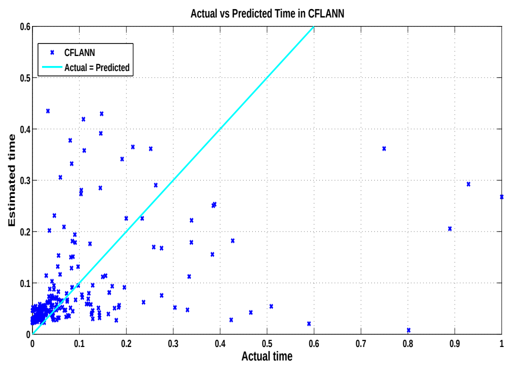

FIGURE 4.16: Effort Estimation Model using CFLANN for enhanced web projects

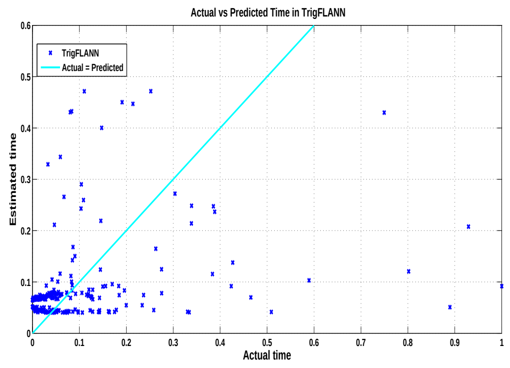

FIGURE 4.17: Effort Estimation Model using TrigFLANN for enhanced web projects

TABLE 4.1: Comparison of Proposed Models in New Web Projects

	MSE	**R^2**	**PRED**
FFNN	0.0106	0.9295	95.2735
LFLANN	**0.0097**	0.9744	**95.5087**
PFLANN	0.0127	0.9535	93.7334
CFLANN	0.0101	0.9561	95.3780
TrigFLANN	0.0105	0.9639	94.9453
RBF_Gradient_FCM	0.0107	0.8748	95.1619
RBF_Gradient_K-Means	0.0112	0.8445	94.9528
RBF_Gradient_Random	0.0127	0.8373	93.5576

TABLE 4.2: Comparison of Proposed Models in Enhanced Web Projects

	MSE	**R^2**	**PRED**
FFNN	0.0143	0.9295	94.0474
LFLANN	0.0124	0.9744	94.0953
PFLANN	0.0161	0.9535	93.7744
CFLANN	0.0118	0.9561	94.4934
TrigFLANN	0.0142	0.9039	92.6625
RBF_Gradient_FCM	**0.0103**	0.9748	**95.2544**
RBF_Gradient_K-Means	0.0115	0.9445	95.0056
RBF_Gradient_Random	0.0166	0.8873	91.7485

4.4 Comparison of Results Obtained

Tables 4.1 and 4.2 show the performance of various neural network models in effort estimation of web-based software, with both new and enhanced projects' effort models. It is observed that RBFN model with Gradient Descent learning and FCM clustering and LFLANN perform better than the rest. They are on-line techniques i.e. the network weights and centers are updated iteration by iteration until the MSE value saturates (stays constant for certain number of times, at least ten in this case). Due to this kind of rigorous training, the network weights and centers get optimal values, the error term (MSE) is low and performance is high.

Radial basis function networks get the input space covered by radial basis functions. RBF centers are determined by taking into account the input data distribution. Due to this kind of extensive coverage of the input space, the RBF network performs best in the case of enhanced projects. But at the same time, determination of RBF centers is done without any reference to the prediction task. Due to this, resources may be thrown away on such areas of the input distribution that are insignificant to learning. This reason does not affect the RBFN's accuracy in case of enhanced projects' effort models however, this is the reason for it's lower accuracy and higher error as compared to LFLANN in case of new project's effort models.

Furthermore, LFLANN (new web projects) and RBFN with gradient learning and FCM clustering (enhanced web projects) too follow an on-line means to optimize their network parameters, hence get high performance. LFLANN network performs good next to RBFN and CFLANN in case of enhanced web projects. It is again an on-line technique. FLANN is known for it's lower computational complexity i.e. ability to handle non-linearity and low error rates ([33]). It performs accordingly. Each of the variants of FLANN has lower MSE value than that of FFNN and RBFN in case of new web projects' effort models. All the variants of FLANN perform almost the same. L-FLANN performs the best by a slight margin in case of new web projects.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Effort estimation is a very crucial phase of software development. It needs to be done very efficiently in order to deliver the desired product on time, without any cost or resource overruns. Now a days, agile methods of software development and web-based software are given more priority due to their obvious benefits. So there is a requirement of efficient models for estimating the effort required to develop such software. A number of approaches have been proposed for effort estimaton in the past, and it was observed that a good number of researchers have considered ANN models. ANNs are an obvious choice for effort prediction because they have good interpretability and capacity to handle noisy input.

In this work, ANNs have been used for developing effort estimation models for agile and web-based software. Out of the existing models for agile software effort estimation [6] [7], the one proposed by Zia (Regression Model) [6] doesn't perform very well, but the one proposed by Hearty (Bayesian Network Model) [7] has very good performance. Other ANNS weren't used for this, so in this work, four different types of neural networks along with their variants are used to predict the effort. And they provide comparatively high quality results (high prediction accuarcy). All the computations carried out in this study were done using MATLAB software tool.

5.2 Future Work

The work presented here includes effort estimation based on various types of ANNs. This work can be extended to application of several Machine Learning techniques.

Bibliography

[1] Kjetil Molokken and Magen Jorgensen. A review of software surveys on software effort estimation. In *Empirical Software Engineering, 2003. ISESE 2003. Proceedings. 2003 International Symposium on*, pages 223–230. IEEE, 2003.

[2] A. B. M. Moniruzzaman and Syed Akhter Hossain. Comparative study on agile software development methodologies. *CoRR*, abs/1307.3356, 2013. URL `http://arxiv.org/abs/1307.3356`.

[3] Martin Fowler and Jim Highsmith. The agile manifesto. *Software Development*, 9 (8):28–35, 2001.

[4] Andreas Schmietendorf, Martin Kunz, and Reiner Dumke. Effort estimation for agile software development projects. In *5th Software Measurement European Forum*, pages 113–123, 2008.

[5] Trevor Hutt. Scrum Effort Estimation and Story Points. `http://bundlr.com/clips/513e60b02101880002001451`, 2013. [Online; accessed 12-March-2013].

[6] Ziauddin Khan Zia, Shahid Kamal Tipu, and Shahrukh Khan Zia. An effort estimation model for agile software development. *Advances in Computer Science and its Applications*, 2(1):314–324, 2012.

[7] Peter Hearty, Norman Fenton, David Marquez, and Martin Neil. Predicting project velocity in xp using a learning dynamic bayesian network model. *Software Engineering, IEEE Transactions on*, 35(1):124–137, 2009.

[8] Shashank Mouli Satapathy, Aditi Panda, and Santanu Kumar Rath. Story point approach based agile software effort estimation using various svr kernel methods. In *The Twenty Sixth International Conference on Software Engineering & Knowledge Engineering (SEKE), July 1-3, Hyatt Regency, Vancouver, Canada*, pages 304–307, 2014.

[9] Donald J Reifer. Web development: estimating quick-to-market software. *IEEE software*, 17(6):57–64, 2000.

[10] Gennaro Costagliola, Sergio Di Martino, Filomena Ferrucci, Carmine Gravino, Genoveffa Tortora, and Giuliana Vitiello. Effort estimation modeling techniques: a case study for web applications. In *Proceedings of the 6th international conference on Web engineering*, pages 9–16. ACM, 2006.

[11] Allan J Albrecht. Measuring application development productivity. In *Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium*, volume 10, pages 83–92, 1979.

[12] Adam Trendowicz, Jürgen Münch, and Ross Jeffery. State of the practice in software effort estimation: a survey and literature review. In *Software Engineering Techniques*, pages 232–245. Springer, 2011.

[13] Edilson JD Candido and Rosely Sanches. Estimating the size of web applications by using a simplified function point method. In *WebMedia and LA-Web, 2004. Proceedings*, pages 98–105. IEEE, 2004.

[14] Damir Azhar, Emilia Mendes, and Patricia Riddle. A systematic review of web resource estimation. In *Proceedings of the 8th International Conference on Predictive Models in Software Engineering*, pages 49–58. ACM, 2012.

[15] Siobhan Keaveney and Kieran Conboy. Cost estimation in agile development projects. In *ECIS*, pages 183–197, 2006.

[16] Evita Coelho and Anirban Basu. Effort estimation in agile software development using story points. *development*, 3(7), 2012.

[17] Ishrar Hussain, Leila Kosseim, and Olga Ormandjieva. Approximation of cosmic functional size to support early effort estimation in agile. *Data & Knowledge Engineering*, 85:2–14, 2013.

[18] Alaa El Deen Hamouda. Using agile story points as an estimation technique in cmmi organizations. In *Agile Conference (AGILE), 2014*, pages 16–23. IEEE, 2014.

[19] Erdir Ungan, Numan Cizmeli, and Onur Demirors. Comparison of functional size based estimation and story points, based on effort estimation effectiveness in scrum projects. In *Software Engineering and Advanced Applications (SEAA), 2014 40th EUROMICRO Conference on*, pages 77–80. IEEE, 2014.

[20] Emilia Mendes, Sergio Di Martino, Filomena Ferrucci, and Carmine Gravino. Effort estimation: how valuable is it for a web company to use a cross-company data set, compared to using its own single-company data set? In *Proceedings of the 16th international conference on World Wide Web*, pages 963–972. ACM, 2007.

[21] Filomena Ferrucci, Carmine Gravino, Rocco Oliveto, Federica Sarro, and Emilia Mendes. Investigating tabu search for web effort estimation. In *Software Engineering and Advanced Applications (SEAA), 2010 36th EUROMICRO Conference on*, pages 350–357. IEEE, 2010.

[22] Ali Idri, Abdelali Zakrani, and Azeddine Zahi. Design of radial basis function neural networks for software effort estimation. *IJCSI International Journal of Computer Science Issues*, 7(4), 2010.

[23] Barbara A Kitchenham and Emilia Mendes. A comparison of cross-company and within-company effort estimation models for web applications. In *Proceedings of the 8th International Conference on Empirical Assessment in Software Engineering, Edinburgh, Scotland, UK*, pages 47–55, 2004.

[24] Anna Corazza, Sergio Di Martino, Filomena Ferrucci, Carmine Gravino, and Emilia Mendes. Using support vector regression for web development effort estimation. In *Software Process and Product Measurement*, pages 255–271. Springer, 2009.

[25] Emilia Mendes, Carmel Pollino, and Nile Mosley. Building an expert-based web effort estimation model using bayesian networks. In *13th International Conference on Evaluation & Assessment in Software Engineering*, 2009.

[26] Emilia Mendes, Ian Watson, Chris Triggs, Nile Mosley, and Steve Counsell. A comparative study of cost estimation models for web hypermedia applications. *Empirical Software Engineering*, 8(2):163–196, 2003.

[27] Emilia Mendes. The use of bayesian networks for web effort estimation: further investigation. In *Web Engineering, 2008. ICWE'08. Eighth International Conference on*, pages 203–216. IEEE, 2008.

[28] Emilia Mendes. A comparison of techniques for web effort estimation. In *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*, pages 334–343. IEEE, 2007.

[29] Emilia Mendes and Barbara Kitchenham. Further comparison of cross-company and within-company effort estimation models for web applications. In *Software Metrics, 2004. Proceedings. 10th International Symposium on*, pages 348–357. IEEE, 2004.

[30] Sergio Di Martino, Filomena Ferrucci, Carmine Gravino, and Federica Sarro. Using web objects for development effort estimation of web applications: a replicated study. In *Product-Focused Software Process Improvement*, pages 186–201. Springer, 2011.

[31] Jianfeng Wen, Shixian Li, Zhiyong Lin, Yong Hu, and Changqin Huang. Systematic literature review of machine learning based software development effort estimation models. *Information and Software Technology*, 54(1):41–59, 2012.

[32] Ali Idri, Abdelali Zakrani, and Azeddine Zahi. Design of radial basis function neural networks for software effort estimation. *IJCSI International Journal of Computer Science Issues*, 7(4), 2010.

[33] B Tirimula Rao, B Sameet, G Kiran Swathi, K Vikram Gupta, Ch RaviTeja, and S Sumana. A novel neural network approach for software cost estimation using functional link artificial neural network (flann). *International Journal of Computer Science and Network Security*, 9(6):126–131, 2009.

[34] Parag C Pendharkar. Probabilistic estimation of software size and effort. *Expert Systems with Applications*, 37(6):4435–4440, 2010.

[35] Adriano LI Oliveira, Petronio L Braga, Ricardo MF Lima, and Márcio L Cornélio. Ga-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation. *information and Software Technology*, 52(11):1155–1166, 2010.

[36] Vachik S Dave and Kamlesh Dutta. Neural network based models for software effort estimation: a review. *Artificial Intelligence Review*, 42(2):295–307, 2014.

[37] Farhad Soleimanian Gharehchopogh, Isa Maleki, and Sahar Sadouni. Artificial neural networks based analysis of software cost estimation models. *algorithms*, 20: 15, 2014.

[38] F.S. Gharehchopogh. Neural networks application in software cost estimation: A case study. In *Innovations in Intelligent Systems and Applications (INISTA), 2011 International Symposium on*, pages 69–73, June 2011. doi: 10.1109/INISTA.2011.5946160.

[39] Abbas Heiat. Comparison of artificial neural network and regression models for estimating software development effort. *Information and Software Technology*, 44 (15):911– 922, 2002.

[40] Heejun Park and Seung Baek. An empirical validation of a neural network model for software effort estimation. *Expert Systems with Applications*, 35(3):929 – 937, 2008. doi: http://dx.doi.org/10.1016/j.eswa.2007.08.001.

[41] International Software Benchmarking Standard Group. ISBSG, ISBSG dataset release 12. `http://www.isbsg.org/`, 2013.

[42] IFPUG fp description. `http://it-gost.ru/`. Accessed: 2015-03-25.

[43] S. N. Sivanandam, S. Sumathi, and S. N. Deepa. *Introduction to Fuzzy Logic using MATLAB*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 3540357807.

[44] Sundaramoorthy Rajasekaran and GA Vijayalakshmi Pai. *NEURAL NETWORKS, FUZZY LOGIC AND GENETIC ALGORITHM: SYNTHESIS AND APPLICATIONS (WITH CD)*. PHI Learning Pvt. Ltd., 2003.

[45] Tim Menzies, Zhihao Chen, Jairus Hihn, and Karen Lum. Selecting best practices for effort estimation. *Software Engineering, IEEE Transactions on*, 32(11):883–895, 2006.

[46] Rejane B Santos, Markus Ruppb, Santiago J Bonzi, and Ana Maria F Filetia. Comparison between multilayer feedforward neural networks and a radial basis function network to detect and locate leaks in pipelines transporting gas. *Chem. Eng. Trans*, 32:1375–1380, 2013.

[47] Soumi Ghosh and Sanjay Kumar Dubey. Comparative analysis of k-means and fuzzy c-means algorithms. *IJACSA*, 4:35–38, 2013.

[48] KM Bataineh, M Naji, and M Saqer. A comparison study between various fuzzy clustering algorithms. *Jordan journal of mechanical and industrial engineering*, 5 (4):335–343, 2011.

Published

1. S. M. Satapathy, A Panda and S. K. Rath, "Story Point Approach based Agile Software Effort Estimation using Various SVR Kernel Methods," *The 26th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pp. 304-308, Vancouver, Canada, 1-3 July 2014.

2. A Panda, S. M. Satapathy and S. K. Rath, " Neural Network Models for Agile Software Effort Estimation based on Story Points," *The Third International Conference On Advances in Computing, Control and Networking (ACCN)*, Bangkok, Thailand, 21-22 February 2015.

3. A Panda, S. M. Satapathy and S. K. Rath, "Empirical Validation of Neural Network Models for Agile Software Effort Estimation based on Story Points," *The Third International Conference on Recent Trends in Computing (ICRTC)* , Delhi, India, 12-13 March 2015.

Accepted

1. A Panda, S. M. Satapathy and S. K. Rath, "Story Point Approach based Agile Software Effort Estimation using Machine Learning Techniques" *The Twenty-Third International Conference on Software Engineering and Data Engineering (SEDE)*, Loiusiana, USA, 13-15 October 2014.