Effort Estimation for Object-oriented System using Artificial Intelligence Techniques

Mukesh Kumar

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India
May 2013

Effort Estimation for Object-oriented System using Artificial Intellignce Techniques

Thesis submitted in partial fulfillment of the requirements for the degree of

Master of Technology

in

Computer Science and Engineering

(Specialization: Software Engineering)

by

Mukesh Kumar

(Roll- 211CS3295)

Under the supervision of

Prof. S. K. Rath

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela, Odisha, 769 008, India

May 2013

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India.

Certificate

This is to certify that the work in the thesis entitled ***Effort Estimation for Object - oriented System using Artificial Intelligence Techniques*** by ***Mukesh Kumar*** is a record of an original research work carried out by him 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 this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Place: NIT Rourkela
Date: June 3, 2013

(Prof. Santanu Ku. Rath)
Professor, CSE Department
NIT Rourkela, Odisha

Acknowledgment

I am grateful to numerous local and global peers who have contributed towards shaping this thesis. At the outset, I would like to express my sincere thanks to Prof. Santanu Ku. Rath for his advice during my thesis work. As my supervisor, he has constantly encouraged me to remain focused on achieving my goal. His observations and comments helped me to establish the overall direction to the research and to move forward with investigation in depth. He has helped me greatly and been a source of knowledge.

I am very much indebted to Prof. Ashok Kumar Turuk, Head-CSE, for his continuous encouragement and support. He is always ready to help with a smile. I am also thankful to all the professors at the department for their support.

I would like to thank Mr. Shashank Mouli Satapathy for his encouragement and support. His help can never be penned with words.

I would like to thank all my friends and lab-mates for their encouragement and understanding. Their help can never be penned with words.

I must acknowledge the academic resources that I have got from NIT Rourkela. I would like to thank administrative and technical staff members of the Department who have been kind enough to advise and help in their respective roles.

Last, but not the least, I would like to dedicate this thesis to my family, for their love, patience, and understanding.

Mukesh Kumar

Roll-211cs3295

Abstract

Software effort estimation is a vital task in software engineering. The importance of effort estimation becomes critical during early stage of the software life cycle when the details of the software have not been revealed yet. The effort involved in developing a software product plays an important role in determining the success or failure. With the proliferation of software projects and the heterogeneity in their genre, there is a need for efficient effort estimation techniques to enable the project managers to perform proper planning of the Software Life Cycle activates. In the context of developing software using object-oriented methodologies, traditional methods and metrics were extended to help managers in effort estimation activity.

There are basically some points approach, which are available for software effort estimation such as Function Point, Use Case Point, Class Point, Object Point, etc. In this thesis, the main goal is to estimate the effort of various software projects using Class Point Approach. The parameters are optimized using various artificial intelligence (AI) techniques such as Multi-Layer Perceptron (MLP), K-Nearest Neighbor Regression (KNN) and Radial Basis Function Network(RBFN), fuzzy logic with various clustering algorithms such as the Fuzzy C-means (FCM) algorithm, K-means clustering algorithm and Subtractive Clustering (SC) algorithm, such as to achieve better accuracy. Furthermore, a comparative analysis of software effort estimation using these various AI techniques has been provided. By estimating the software projects accurately, we can have software with acceptable quality within budget and on planned schedules.

Keywords: Software effort estimation, Class point approach, ANN, KNN, RBFN, Fuzzy Logic.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Project Management is the process of planning and controlling the development as a system within a specified time frame at a minimum cost with the right functionality. Much software fails due to faulty project management practices. Therefore, it is important to learn different aspects of software project management. Key features of Project Management-

- Project Scheduling
- Staffing
- Monitoring and control
- Project Estimation
- Risk Management
- Report Generation

Among all these Projects, Estimation is the most challenging task. Project estimation involves size estimation, effort estimation, cost estimation, estimation, time estimation, staffing estimation. First, we determine the size of the product. From size estimation, we determine the effort needed. From effort estimation, we can determine product duration and cost.

Software size estimation is important to determine the project effort. However, according to the last research reported by the Brazilian Ministry of Science and

Technology-MCT, in 2001, only 29% of the companies accomplished size estimation and 45.7% accomplished software effort estimate. So that effort estimation has motivated considerable research during recent years.
Effort Estimation: It is the process of predicting the effort required to develop or maintain software product in person months. Many ways are available for categorizing estimation approaches. Most efficient categories are as follows-

1. Expert estimation: The quantification step, on the basis of judgmental process estimation is done.

2. Formal estimation: the quantification step is based on mechanical processes, e.g., the use as a formula derived from historical data.

3. Combination-Based estimation: This estimation approach deals with a judgmental or mechanical combination of estimates from different sources.

Function Point Analysis (FPA), Use Case Point (UCP) Analysis and Class Point Analysis (CPA) comes under the Formal estimation model; that is based on size-based estimation approach. Here CPA has been used only because the class point has been inferred from one of the most important Unified Modeling Language (UML) diagrams, i.e. a class diagram. Hence one of the major advantages of using a class point approach (CPA) over FPA is that the number of function points is calculated at coding phase but class point is calculated from the design phase of the software-development life cycle (SDLC). Hence estimation can be done at an early stage of the SDLC.

1.1 Class Point Analysis

The class point approach was introduced by Gennaro Costagliola et al. in 1998 [1]. This was based on the function point analysis approach to represent the internal attributes of a software system in terms of counting. The idea using the *Class Point Approach* is the quantification of classes in a program similar to the FP measure, where the basic unit is function. It has been derived from the observations that in the procedural model, the basic programming units are functions or procedures;

whereas, in case of an object-oriented model, the logical building blocks are classes. The Class Point size estimation process is structured into three main phases, corresponding to similar phases in the function point approach, i.e.,

- Information processing size estimation:
  - Identification and classification of classes
  - Evaluation of complexity level of each class
  - Estimation of the Total Unadjusted Class Point
- Technical complexity factor estimation
- Final Class Point evaluation

During the first step, the design specifications are analyzed in order to identify and classify the classes into four types of system components, namely Problem Domain Type (PDT), Human Interaction Type (HIT), Data-Management Type (DMT), and Task Management Type (TMT).

During the second step, each identified class is assigned a complexity level, which is determined based on the local methods in the class and of the interaction of the class with the rest of the system. In case of CP1, the complexity level of each class is evaluated based on the Number of External Methods (NEM), and the Number of Services Requested (NSR). Similarly in case of CP2, apart from the above measures, the Number Of Attributes (NOA) measure is considered in order to evaluate the complexity level of each class. The block diagram shown in Figure-1.1 explains the steps to calculate the class point.

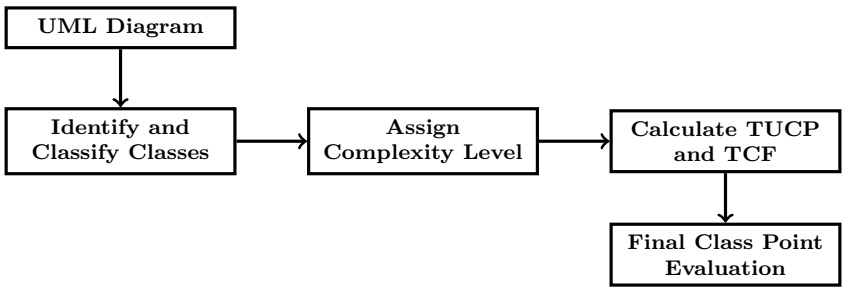

Figure 1.1: Steps to Calculate Class Point

For the calculation of CP1, the complexity level of the class is determined based on the value of NEM and NSR according to Table-1.1. For example, if a class is having NEM value 7 and NSR value 3, then the complexity level assigned to the class is Average.

Table 1.1: Complexity Level Evaluation for CP1

	0 - 4 NEM	**5 - 8 NEM**	**9 - 12 NEM**	≥ 13 **NEM**
0 - 1 NSR	Low	Low	Average	High
2 - 3 NSR	Low	Average	High	High
4 - 5 NSR	Average	High	High	Very High
> **5 NSR**	High	High	Very High	Very High

For the calculation of CP2, the complexity level of the class is determined based on the value of NEM, NOA and NSR according to Table- 1.2a, 1.2b and 1.2c. In all these tables, NEM and NOA range varies with respect to the fixed NSR range.

Table 1.2: Complexity Level Evaluation for CP2

0 - 2 NSR	**0 - 5 NOA**	**6 - 9 NOA**	**10 - 14 NOA**	≥ 15 **NOA**
0 - 4 NEM	Low	Low	Average	High
5 - 8 NEM	Low	Average	High	High
9 - 12 NEM	Average	High	High	Very High
≥ 13 **NEM**	High	High	Very High	Very High

(a)

3 - 4 NSR	**0 - 4 NOA**	**5 - 8 NOA**	**9 - 13 NOA**	≥ 14 **NOA**
0 - 3 NEM	Low	Low	Average	High
4 - 7 NEM	Low	Average	High	High
8 - 11 NEM	Average	High	High	Very High
≥ 12 **NEM**	High	High	Very High	Very High

(b)

≥ 5 ***NSR***	**0 - 3 NOA**	**4 - 7 NOA**	**8 - 12 NOA**	≥ 13 **NOA**
0 - 2 NEM	Low	Low	Average	High
3 - 6 NEM	Low	Average	High	High
7 - 10 NEM	Average	High	High	Very High
≥ 11 **NEM**	High	High	Very High	Very High

(c)

Once a complexity level of each class has been assigned, such information and its type are used to assign a weight to the class given in Table- 1.3. Then, the Total Unadjusted Class Point value (TUCP) is computed as a weighted sum of the number of classes of different component types.

$$TUCP = \sum_{i=1}^{4} \sum_{j=1}^{3} w_{ij} \times x_{ij} \tag{1.1}$$

where x_{ij} is the number of classes of component type i (problem domain, human interaction, etc.) with the complexity level j (low, average, or high), and w_{ij} is the weighting value of type i and complexity leveled j.

Table 1.3: Evaluation of TUCP for Each Class Type

System Component Type	Description	Complexity			
		Low	Average	High	Very High
PDT	Problem Domain Type	3	6	10	15
HIT	Human Interaction Type	4	7	12	19
DMT	Data Management Type	5	8	13	20
TMT	Task Management Type	4	6	9	13

The Technical Complexity Factor (TCF) is determined by adjusting the TUCP with a value obtained by 24 different target software system characteristics, each on a scale of 0 to 5. The sum of the influence degrees of all the general system characteristics forms the Total Degree of Influence (TDI) which is shown in Table-1.4. This is used to determine the TCF according to the following formula:

$$TCF = 0.55 + (0.01 * TDI) \tag{1.2}$$

Table 1.4: Degree of Influences of Twenty Four General System Characteristics

ID	System Characteristics	DI	ID	System Characteristics	DI
C1	Data Communication	....	C13	Multiple sites	....
C2	Distributed Functions	....	C14	Facilitation of change	....
C3	Performance	....	C15	User Adaptivity	....
C4	Heavily used configuration	....	C16	Rapid Prototyping	....
C5	Transaction rate	....	C17	Multiuser Interactivity	....
C6	Online data entry	....	C18	Multiple Interfaces	....
C7	End-user efficiency	....	C19	Management Efficiency	....
C8	Online update	....	C20	Developers' Professional Competence	....
C9	Complex processing	....	C21	Security	....
C10	Reusability	....	C22	Reliability	....
C11	Installation ease	....	C23	Maintainability	....
C12	Operational ease	....	C24	Portability	....
TDI				**Total Degree of Influence (TDI)**	....

Finally, the Class Point (CP) value is determined by multiplying the Total Unadjusted Class Point (TUCP) value by TCF.

$$CP = TUCP * TCF \tag{1.3}$$

The final class point of various projects is used to calculate the required effort to develop the project in a very scheduled time.

1.2 Various Performance Measures

The accuracy of the model can be evaluated by using the following criteria:

1.2.1 Mean Square Error (MSE)

It can be calculated as:

$$MSE = \frac{\sum_{i=1}^{N} (y_i - \bar{y})^2}{N} \tag{1.4}$$

1.2.2 Magnitude of Relative Error (MRE)

The **Magnitude of Relative Error (MRE)** is a very common criterion used to evaluate software cost estimation models. The MRE for each observation i can be obtained as:

$$MRE_i = \frac{|ActualEffort_i - PredictedEffort_i|}{ActualEffort_i} \tag{1.5}$$

1.2.3 Mean Magnitude of Relative Error (MMRE)

The **Mean Magnitude of Relative Error (MMRE)** can be achieved through the summation of MRE over N observations.

$$MMRE = \sum_{1}^{N} MRE_i \tag{1.6}$$

1.2.4 Root Mean Square Error (RMSE)

It is just the square root of the mean square error.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (y_i - \bar{y})^2}{N}} \tag{1.7}$$

1.2.5 Normalized Root Mean Square(NRMS)

The **Normalized Root Mean Square(NRMS)** can be calculated by dividing the RMSE value with standard deviation of the actual effort value for training data set.

$$NRMS = \frac{RMSE}{std(Y)} \tag{1.8}$$

where Y is the actual effort for training data set.

1.3 Dataset used for Effort Calculation

The dataset (Table-1.5) from forty Java systems is derived during two successive semesters of graduate courses on Software Engineering. The use of such data in the validation process has provided initial experimental evidence of the effectiveness of the *Class Point* approach [1]. It is clear that the use of student's projects may threaten the external validity of the experiment and, hence, for the assessment of the method; further analysis is needed by using data coming from the industrial world. Nevertheless, we have worked to make the validation process as accurate as possible.

1.4 Problem Definition

Traditional software estimation techniques like Constructive Cost Estimation Model (COCOMO) and Function Point Analysis (FPA) have proven unsatisfactory for measuring cost and effort of all types of software because the line of code(LOC) and function point(FP) were both used for procedural oriented [2, 3]. The procedural oriented design splits the data and procedure, whereas the object-oriented programming combines them.

COCOMO model is used for early rough, estimates of project cost, performance, and schedule; and gives the accuracy within 68%. Hence the objective is to increase the estimation accuracy of software products.

Function Point and COCOMO will be used in coding phase while CPA is used in design phase of SDLC. It provides more accurate results because classes are the major component of OO paradigm; and using a CPA, estimation of software projects can be done at the design phase.

1.5 Literature Review

Gennaro Costagliola, et al. have used two measures of size i.e. CP1 & CP2 and three metrics i.e. NEM, NSR & NOA to find the complexity of a class [1]. From the experiment over 40 project data set they have found that the aggregated MMRE of CP1 is 0.19 and CP2 is 0.18. Wei Zhou and Qiang Liu have extended this approach by adding another measure named as CP3 based on CPA and have taken 24 system characteristics instead of 18 considered by Gennaro Costagliola, et al [4]. By using this approach Wei Zhou and Qiang Liu found that the MMRE of CP1 and CP2 is 0.19 and 0.14 respectively. S. Kanmani, et al. have used the same CPA by using neural network in mapping the CP1 and CP2 into the effort and found that the aggregate MMRE is improved from 0.19 to 0.1849 for CP1 and from 0.18 to 0.1673 for CP2 [5]. SangEun Kim, et al. introduced some new definitions of class point to increase understanding of a system's architectural complexity [6]. They have taken the help of a number of extra parameters apart from NEM, NSR and NOA to calculate the total no of class points. Ziauddin et al. proposed an algorithm to implement the COCOMO model using fuzzy logic technique for software effort estimation [7]. S. Kanmani, et al. proposed another technique to use CPA with a fuzzy system using subtractive clustering technique for calculating the effort and have compared the result with that obtained using the concept of artificial neural network [8]. They found that fuzzy system using subtractive clustering technique yield better result than that of ANN. Veronica S. Moertini introduced five data clustering algorithm and show the implementation of two algorithms out of five using Matlab [9]. K. M. Bataineh, et al. compares the performance of Fuzzy c-means algorithm with subtractive clustering algorithm [10].

Alaa Sheta use Takagi-Sugeno (TS) technique to develop fuzzy models for two nonlinear processes [12]. They are the software effort estimation for a NASA software projects and the prediction of the next week S & P 500 i.e. Standard & Poor's 500 for stock market. Arshdeep Kaur et al. outlines the basic difference between the Mamdani-type Fuzzy Inference System and Sugeno-type Fuzzy Inference System for Air Conditioning System [13].

Adriano L.I. Oliveira [14] provides a comparative study on support vector regression (SVR), radial basis function neural networks (RBFNs) and linear regression for estimation of software project effort. The experiment is carried out using NASA project datasets and the result shows that SVR performs better than RBFN and linear regression. K. Vinay Kumar, et al. [15] have proposed the use of wavelet neural network (WNN) to forecast the software development effort and compares the result with other techniques such as multilayer perceptron (MLP), radial basis function network (RBFN), multiple linear regression (MLR), dynamic evolving neuro-fuzzy inference system (DENFIS) and support vector machine (SVM).

Iman Attarzadeh et al. [17] described an enhanced soft computing model for the estimation of software cost and time estimation and compare the result with algorithmic model. Vladimir Cherkassky et. al [18] use six representative methods implemented on artificial data sets to provide some insights on applicability of various methods. They conclude that no single method proved to be the best, since a method's performance depends significantly on the type of the target function (being estimated), and on the properties of training data (i.e., the number of samples, amount of noise, etc.).

Adrian G. Bors [19] introduced a few RBF training algorithms and showed how RBF networks can be applied for real-life applications. Haralambos Sarimveis et al. [20] proposed a new algorithm for training RBF neural networks based on the subtractive clustering technique. Ali Idri et al. [21] provide a comparison between a RBF neural network using C-means and a RBF neural network using APC-III, in terms of estimate accuracy, for software effort estimation based on COCOMO'81 dataset. Chitra Panchapakesan et al. [22] have used another approach to test how much one can reduce the error by changing the centers in an RBF network.

1.6 Motivation

A survey says, almost one-third projects exceed their budget and is delivered late and two-thirds of all projects overrun their original estimates. It is impossible for a manager or system analyst to accurately predict the cost and effort required to

develop a software. Without accurate cost estimation capability, project managers can't determine how much time and manpower the project should take and that means the software portion of the project is out of control from its beginning.

To help the industry in developing quality products within the scheduled time, accurate software effort estimation is necessary.

1.7 Thesis Organization

The rest of the thesis is organized as follows.
Novel artificial intelligence (AI) techniquesis given in **Chapter-3**, in that Fuzzy logic system been discussed and implemented, then their results are compared to estimate the effort of software product development using OO paradigm.

In **Chapter-2** different types of adaptive regression techniques like MLP (2.1.1), KNN (2.1.2) and RBFN (2.1.3) has been proposed and implemented; and their accuracy is compared to estimate the effort of software product on the basis of Class Point count.

Table 1.5: Forty Project DataSet

Sr. No.	**EFH**	**CP1**	**CP2**	**NEM**	**NSR**	**NOA**
1	286	103.18	110.55	142	97	170
2	396	278.72	242.54	409	295	292
3	471	473.90	446.60	821	567	929
4	1016	851.44	760.96	975	723	755
5	1261	1263.12	1242.60	997	764	1145
6	261	196.68	180.84	225	181	400
7	993	178.80	645.60	589	944	402
8	552	213.30	208.56	262	167	260
9	998	1095.00	905.00	697	929	385
10	180	116.62	95.06	71	218	77
11	482	267.80	251.55	368	504	559
12	1083	687.57	766.29	789	362	682
13	205	59.64	64.61	79	41	98
14	851	697.48	620.10	542	392	508
15	840	864.27	743.49	701	635	770
16	1414	1386.32	1345.40	885	701	1087
17	279	132.54	74.26	97	387	65
18	621	550.55	481.66	382	654	293
19	601	539.35	474.95	387	845	484
20	680	489.06	438.90	347	870	304
21	366	287.97	262.74	343	264	299
22	947	663.60	627.60	944	421	637
23	485	397.10	358.60	409	269	451
24	812	678.28	590.42	531	401	520
25	685	386.31	428.18	387	297	812
26	638	268.45	280.84	373	278	788
27	1803	2090.70	1719.25	724	1167	1633
28	369	114.40	104.50	192	126	177
29	439	162.87	156.64	169	128	181
30	491	258.72	246.96	323	195	285
31	484	289.68	241.40	363	398	444
32	481	480.25	413.10	431	362	389
33	861	778.75	738.70	692	653	858
34	417	263.72	234.08	345	245	389
35	268	217.36	195.36	218	187	448
36	470	295.26	263.07	250	512	332
37	436	117.48	126.38	135	121	193
38	428	146.97	148.35	227	147	212
39	436	169.74	200.10	213	183	318
40	356	112.53	110.67	154	83	147

Chapter 2

Adaptive Regression Techniques

2.1 Introduction

The effort involved in developing a software product plays an important role in determining the success or failure. In the context of developing software using object-oriented methodologies, traditional methods and metrics were extended to help managers in effort estimation activity. Software project managers require a reliable approach for effort estimation. It is especially important during the early stage of the software-development life cycle. In this chapter, the main goal is to estimate the cost of various software projects using class point approach and optimize the parameters using various types of adaptive regression techniques such as Multi-Layer Perceptron (ANN), K Nearest Neighbor Regression (KNN) and Radial Basis Function Network(RBFN) to achieve better accuracy. Furthermore, a comparative analysis of software effort estimation using these various adaptive regression techniques has been provided. By estimating the software projects accurately, we can have software with acceptable quality within budget and on planned schedules.

2.1.1 Multi-Layer Perceptron (MLP)

MLP is a feed-forward neural network with one or more layers between input and output layer. Feed-forward means that data flows in one direction from input to output layer (forward). The back propagation learning (BPA) algorithm is basically used to train this type of model. MLPs are widely used for pattern

classification, recognition, prediction and approximation. Multi Layer Perceptron can solve problems, which are not linearly separable.

2.1.2 K Nearest Neighbor Regression (KNN)

K Nearest Neighbor Regression (KNN) is presented by LUC P. DEVROYE [35] in the year 1978. In pattern recognition, the KNN is a method for classifying objects based on closest training examples in the feature space. It is non-parametric and lazy algorithm. In this case, an object is classified by a majority vote of its neighbors, with the object being assigned for the class most common for its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned for the class of its nearest neighbor. Similarly, for regression, the same method can be used by simply assigning the property value for the object to be the average over the values of its k nearest neighbors.

2.1.3 Radial Basis Function Network (RBFN)

Radial basis function emerged as a variation of multi layer perceptron technique [19]. The theory of function approximation helps in deriving the idea of RBFN. The architecture of the RBFN is quite simple. An input layer which consists of a source's nodes; a hidden layer in which each neuron computes its output using a radial basis function, which is in general a Gaussian function, and an output layer which builds a linear weighted sum of hidden neuron outputs and supplies the response from the network (effort). An RBFN has only one output neuron.

$$F(x) = \sum_{j=1}^{L} w_j \phi_j(\|x - c_j\|) \tag{2.1}$$

where L is the number of hidden neurons, $x \epsilon R^p$ is the input, w_j are the output layer weights of the RBFN and $\phi(x)$ is Gaussian radial basis function given by:

$$\phi_j(\|x - c_j\|) = \exp(-\frac{\|x - c_j\|^2}{(\sigma_j)^2}) \tag{2.2}$$

where $\|.\|$ denotes the Euclidean distance, $c_j \epsilon R^p$ is the centre of the j^{th} hidden neuron and σ_j^2 is the width of the j^{th} hidden neuron.

2.2 Proposed Approach

The proposed work is based on data derived from forty student projects [1] developed using Java's language and intends to evaluate software-development effort. The use of such data on the validation process has provided initial experimental evidence on the effectiveness of the CPA. These data are used in the implementation of various adaptive methods for regression such as MLP, KNN and RBFN system model. The calculated result is then compared to measure the accuracy of the models. To calculate the effort of a given software project, basically the following steps have been used.

Steps in Effort Estimation

1. **Data Collection**: The data has been collected from previously developed projects.

2. **Calculate Class Point**: The class point will be calculated as per the steps described in the *Figure-1.1.*

3. **Select Data**: The generated CP2 value in *Step-2* has been used as input arguments.

4. **Normalize Dataset**: Input values were normalized over the range [0,1]. Let X be the dataset and x is an element of the dataset, then the normalization of the x can be calculated as :

$$Normalized(x) = \frac{x - \min(X)}{\max(X) - \min(X)} \tag{2.3}$$

   where
   $\min(X)$ = the minimum value of the dataset X.
   $\max(X)$ = the maximum value of the dataset X.
   if $\max(X)$ is equal to $\min(X)$, then Normalized(x) is set to 0.5.

5. **Division of data set**: Divide the number of data into three parts, i.e. learning set, validation set and test set.

6. **Perform Model Selection**: In this step, a 5-fold cross validation is implemented for model selection. The model which provides the least NRMSE value than the other generated models based on the minimum validation and prediction error criterion has been selected to perform other operations.

7. **Select Best Model**: By taking the average of all the 5-fold's corresponding validation and prediction error (NRMSE) value, the best model has been selected. Finally, the model has been plotted using training sample and testing sample.

Once the model is ready, the parameter of any new project can be given, and it will generate the estimated effort as output for that project.

2.3 Implementation

2.3.1 Model Design Using Multi-Layer Perceptron

This technique uses one parameter. This parameter sets the number of hidden neurons to be used in a three-layer neural network. The number of neurons used is directly proportional to the training time. The values are typically ranges between 2 to 40, but it can be increase up to 1000. While implementing the normalized data set using Multi-Layer Perceptron technique for a different number of hidden neurons, the following results have been obtained. The Table-2.3.1 provides minimum NRMSE value obtained from Training set and Test set using the Multi-layer Perceptron technique for each fold for a specific number of hidden neurons. Hence the average over the NRMSE values for training set and test set is treated as the final result. The proposed model generated using the Multi-Layer Perceptron technique is plotted based upon the training and testing sample as shown in Figure-2.1. From Figure-2.1, it has been observed that the predicted value is highly correlated with actual data.

Fold	No. of Hidden Neurons	Training Set Validation Error (NRMSE)	Test Set Prediction Error (NRMSE)
1	15	0.356861543	0.2668688
2	40	0.389938789	0.2869995
3	25	0.283935873	0.5995649
4	5	0.335886118	0.2274501
5	20	0.292505942	0.3704862
Average		**0.3555415**	**0.39669798**

Table 2.1: RMSE Value Obtained using Multi-Layer Perceptron Technique for Different No. of Neurons

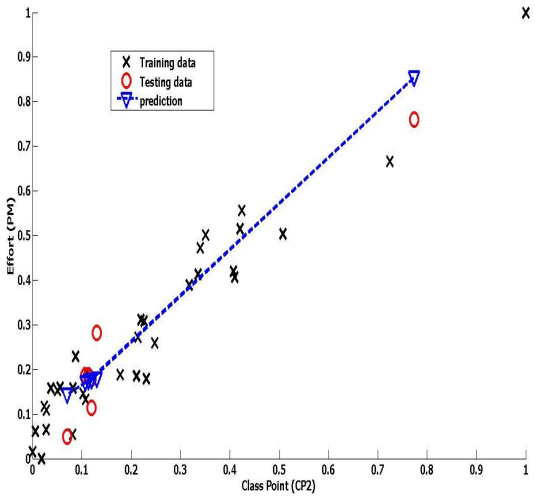

Figure 2.1: Multi-Layer Perceptron based Effort Estimation Model

2.3.2 Model Design Using K-Nearest Neighbor Regression

This technique uses one parameter called K which specifies the numbers of nearest neighbors that are averaged to form an estimate. The value for K must be greater than 0 but less than the number of samples in training file. The maximum value of K could be 100 . While implementing the normalized data set using K-Nearest Neighbor Regression technique for different numbers of nearest neighbors, the following results have been obtained. The Table-2.2 provides various estimation parameters value such as NRMS, RMSE and MMER obtained using the K-Nearest Neighbor Regression technique for different no. of nearest neighbors. From the above table, clearly the model with *two* no. Of nearest neighbors provides minimum value of NRMS and RMSE. The proposed model generated using the K-Nearest Neighbor Regression technique is plotted based upon the training and testing sample as shown in Figure-2.2. From Figure-2.2, it has been observed that the predicted value is less correlated with actual data.

Fold	No. of Nearest Neighbours	Training Set Validation Error (NRMSE)	Test Set Prediction Error (NRMSE)
1	3	0.4432182	0.2873718
2	2	0.4718944	0.4584063
3	4	0.3676056	0.6824517
4	4	0.4013564	0.2452088
5	2	0.4112930	0.4029506
Average		**0.41907352**	**0.41527784**

Table 2.2: NRMSE Value Obtained using K-Nearest Neighbor Regression for Different No. Of Nearest Neighbours

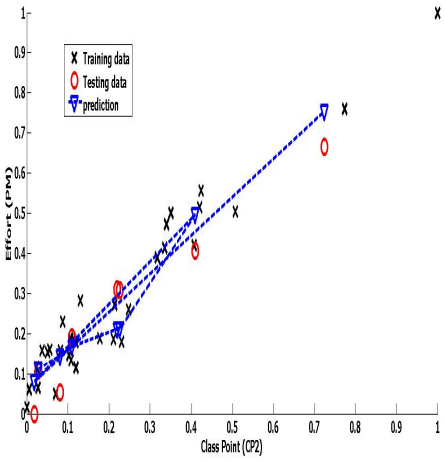

Figure 2.2: K-Nearest Neighbor Regression based Effort Estimation Model

2.3.3 Model Design Using Radial Basis Function Network

This technique uses one parameter, i.e., the number of basis functions. This parameter should be greater than 1. For multivariate input, this parameter should be a squared number (i.e., 4, 9, 25, 36, etc.). Moreover, this parameter should not be greater than the number of samples in the training data. While implementing the normalized data set using Radial Basis Function Network technique for a different number of basis functions, the following results have been obtained.

Fold	No. of Basis Functions	Training Set Validation Error (NRMSE)	Test Set Prediction Error (NRMSE)
1	2	0.4537595	0.2643657
2	2	0.3966360	0.3356930
3	2	0.3568095	0.5830884
4	2	0.3814956	0.2588177
5	2	0.3391349	0.3309974
Average		**0.3855671**	**0.35459244**

Table 2.3: NRMSE Value Obtained using Radial Basis Function Network Technique based on No. of Basis Functions

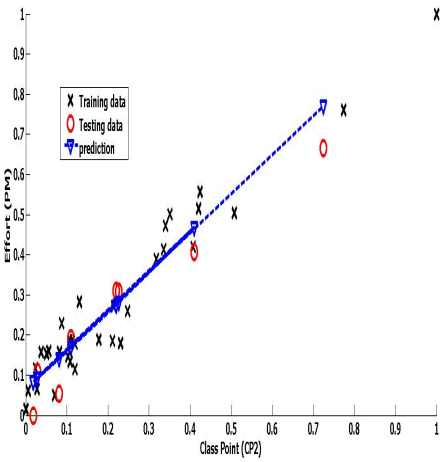

Figure 2.3: Radial Basis Function Network based Effort Estimation Model

The Table-2.3 provides minimum NRMSE value obtained from Training set, and Test set using the RBFN technique for each fold for a specific number of basis functions. Hence the result will be the average over the NRMSE values obtained from training set and test set. The proposed model generated using the RBFN technique is plotted based upon the training and testing sample as shown in Figure-2.3. From Figure-2.3, it has been observed that the predicted value is highly correlated with actual data, but fewer correlations than that of obtained using RBF technique.

2.3.4 Comparison

Based on results obtained, the estimated effort value using the adaptive methods for regression are compared. The results show that effort estimation using RBFN gives better values of NRMSE than those obtained using other methods.

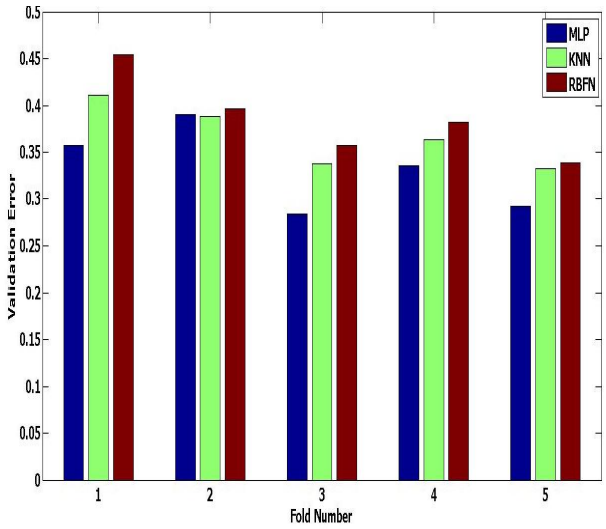

Figure 2.4: Comparison of Validation Error Obtained Using Six Adaptive Methods for Regression

The Figure-2.4 shows the comparison between validation error obtained using various adaptive regression methods.

The comparison between prediction error obtained using various adaptive regression methods is shown in the Figure-2.5.

The Figure-2.6 shows the comparison of average error values obtained from training set (validation error), and test set (prediction error) for various adaptive

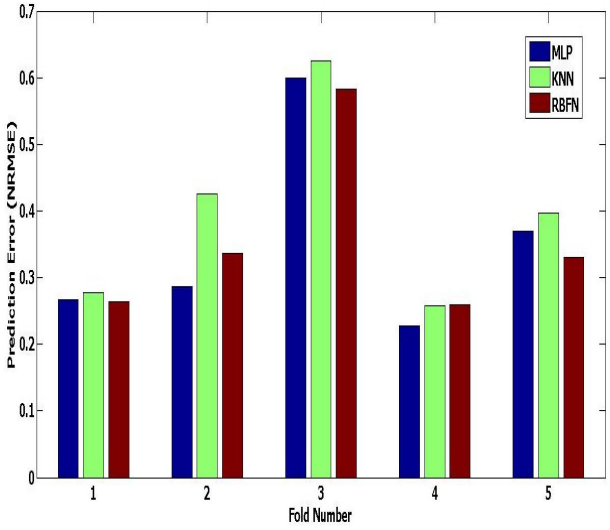

Figure 2.5: Comparison of Prediction Error Obtained Using various Adaptive Methods for Regression

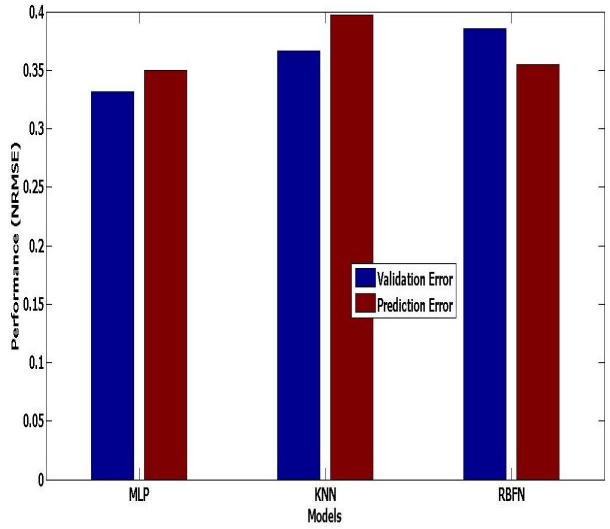

Figure 2.6: Comparison of Average Error values obtained from training, and test set using various Adaptive Methods for Regression

methods for regression techniques.

When using the NRMSE in evaluation, good results are implied by lower values of NRMSE. The Table-2.4 displays the final comparison of NRMSE value for various adaptive regression methods. From the table, it is clear that the effort estimation using rsdial basis function (RBFN) method gives least NRMSE value for validation error and prediction error than other methods.

Table 2.4: Comparison of NRMSE Values between MLP and RBFN

	AI Techniques	Average Validation Error (NRMSE)	Average Prediction Error (NRMSE)
1	Multi-Layer Perceptron	0.3555415	0.39669798
2	K Nearest Neighbor Regression	0.41907352	0.41527784
3	Radial Basis Function Network	0.3855671	0.35459244

2.4 Summary

In this chapter, various adaptive regression techniques have been proposed and implemented to estimate the effort of software product using optimized class point. Then the calculated class point values are being normalized and used to optimize the effort estimation result. The optimization is achieved by implementing different types of adaptive methods of regression techniques such as ANN, KNN and RBFN using normalized class point value. Finally, the generated minimum results of different have been compared for estimating the performance of different models. The result shows that RBFN based effort estimation model gives less value of NRMSE.

Chapter 3

TSK-Fuzzy Logic System

3.1 Introduction

The success of software development depends very much on proper estimation of effort required to develop the software. There are basically some points approach, which are available for software effort estimation such as Function Point, Use Case Point, Class Point, Object Point, etc. In this chapter, to estimate the effort of various software projects using Class Point Approach. The parameters are optimized using various AI techniques such as fuzzy logic to achieve better accuracy.

3.1.1 Fuzzy Logic System

Fuzzy sets were introduced by L. A. Zadeh (1965) [23]. This technique is used to represent and manipulate non-precise data, but rather fuzzy. This technique provides an inference morphology that enables approximate human reasoning capabilities to be applied to knowledge-based systems. Fuzzy system consists of three main components: *fuzzification process*, *inference from fuzzy rules* and *defuzzification process*. Among various fuzzy models, the model introduced by Takagi, Sugeno and Kang (TSK fuzzy system) [10, 24] is more suitable for sample-data based fuzzy modeling, because it needs fewer rules. Each rule's consequence with linear function can describe the input-output mapping in a large range, and the fuzzy implication used in the model is also simple. Since the TSK-Fuzzy system is used to predict the complexity problems [25], fuzzy system needs the antecedent and consequence to express the logical connection between the input and output

which is used as a basis to produce the desired output.The TSK fuzzy system has a high-output sensitivity to input data, because the fuzzy consequence is a variables-function system at the antecedent. TSK model is then structured as a set of IF-THEN rules of the following rules.

If x is A and y is B then z = f(x,y), where A and B are fuzzy sets in the antecedent and z=f(x,y) is a crisp function in the consequence. Usually f(x,y) is a polynomial in the input variables x and y, but it can be any function describe the output from the model within the fuzzy region specified by the antecedence of the rule. A generalized type-1 TSK model can be described by fuzzy IF-THEN rules, which represent input-output relations within a system. For a multi-input single-output first order type-1 TSK model; its kth rule can be expressed as :

IF x_1 is Q_{1k} and x_2 is Q_{2k} and .... and x_n is Q_{nk},
THEN

$$Z = P_0^k + P_1^k x_1 + P_2^k x_2 + ... + P_n^k x_n \tag{3.1}$$

The degree the input matches ith rule is typically computed using the minimum operator:

$$W_i = min(\mu_{A_j}(x), \mu_{B_k}(y)) \tag{3.2}$$

In this case, each rule is crisp output and the weighted average of crisp output is the overall output.

$$Z = \frac{\sum_i W_i Z_i}{\sum_i W_i} \tag{3.3}$$

The basic calculation procedure of TSK fuzzy model is shown below:

To interpret the rule of TSK fuzzy model choice of the center and standard deviation is required. Hence Fuzzy models can be built using different clustering algorithms as follows.

3.2 Methodology Used

To implement the Fuzzy system and to find the number of rules different types of the clustering algorithm are used that has been described in the following section.

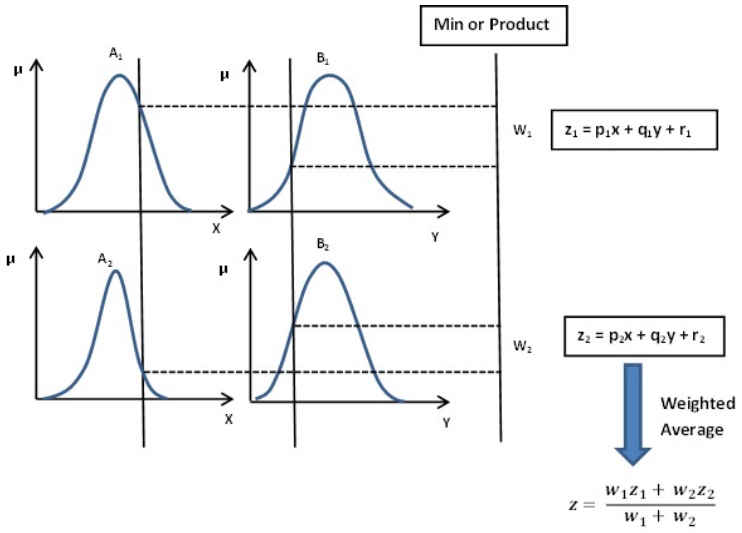

Figure 3.1: TSK Fuzzy Model

3.2.1 Subtractive Clustering (SC)

The subtractive clustering technique is proposed by Stephen L. Chiu [30] in 1994. Clustering has been often exercised as a preprocessing input phase used for the design of the RBF neural networks. To use subtractive clustering, four parameters should be pre-initialized [29] These parameters are *Hypersphere cluster radius in data space*, *Squash Factor* (η), *Reject Ratio* ($\bar{\epsilon}$), *Accept Ratio* (ϵ). *Hypersphere cluster radius in data space* defines neighborhood data points outside this radius has little influence upon the potential. *Squash Factor* (η) defines the neighborhood which will have the measurable reductions in potential, and it can be calculated as:

$$\eta = \frac{r_b}{r_a} \tag{3.4}$$

Reject Ratio specifies a threshold for the potential above which the data point is definitely accepted as a cluster center. *Accept Ratio* specifies a threshold below which the data point is definitely rejected. Consider a collection of q data points x_1, x_2,...., x_q where x_i is a vector to the feature space. Without the loss of generality, we assume that the feature space is normalized so that all data are bound by unit hypercube. The potential of each data point defines a measure of the data point to serve as a cluster center. The potential at each data point can

be calculated by using the following equation.

$$P_i = \sum_{q}^{j=1} \exp(-\frac{\|x_i - x_j\|^2}{(\frac{r_a}{2})^2}) \tag{3.5}$$

where $\|.\|$ denotes the Euclidean distance, and r_a is a positive constant called cluster radius. Then the highest potential data point is selected as the first cluster center. Let x_1^* be the center of the first cluster and p_1^* its potential value. The potential of each data points x_i* is revised as follows:

$$p_i = p_i - p_1^* \exp(-\frac{\|x_i - x_1^*\|^2}{(\frac{r_b}{2})^2}) \tag{3.6}$$

where η is a positive constant greater than 1 and is called the squash factor. When the revision of the potentials of all data points is done by using Equation-3.6, the data point with the highest remaining potential is selected as the second cluster center. In general, after the L_{th} cluster center has been obtained, the potential at each data point is revised as follows:

$$p_i = p_i - p_L^* \exp(-\frac{\|x_i - x_L^*\|^2}{(\frac{r_b}{2})^2}) \tag{3.7}$$

where x_L^* is the center of the L_{th} cluster and p_L^* is its potential value.

3.2.2 Fuzzy C-Means Clustering (FCM)

Fuzzy C-Means clustering (FCM), also known as ISODATA, is a data clustering algorithm in which each data point belongs to a cluster, to a degree, specified by a membership grade. It is first developed by Dunn [31] and improved by Bezdek [32]. FCM employs fuzzy partitioning such that a given data point can belong to several groups in the degree of belongings specified by membership grades between 0 and 1. However, FCM still uses a cost function which is to be minimized while trying to partition the data set. The membership matrix U elements value ranges between 0 and 1. However, the summation of degrees of belongings of a data point to all clusters is always equal to unity:

$$\sum_{i=1}^{c} u_{ij} = 1, \forall j = 1, ..., n \tag{3.8}$$

The cost function for FCM can be defined as:

$$J(U, c_1, ..., c_c) = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^m d_{ij}^2 \tag{3.9}$$

where u_{ij} is between 0 and 1; c_i is the cluster center of a fuzzy group i; $d_{ij} = \|c_i - x_j\|$ is the Euclidean distance between the ith cluster center and jth data point; $m\ \varepsilon\ [1, \infty)$ is a weighting exponent. The conditions for equation 3.9 to reach its minimum value are:

$$c_i = \frac{\sum_{j=1}^{n} u_{ij}^m x_j}{\sum_{j=1}^{n} u_{ij}^m} \tag{3.10}$$

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} (\frac{d_{ij}}{d_{kj}})^{\frac{2}{m-1}}} \tag{3.11}$$

The algorithm works iteratively through the preceding two conditions until there is no more improvement noticed. The performance of FCM depends on the initial membership matrix values; thereby it is advisable to run the algorithm for several times, each starting with different values of membership grades of data points.

3.2.3 K-Means Clustering

The K-means clustering (Hard C-means clustering), is a crisp clustering algorithm based on finding data clusters in a data set such that a cost function of dissimilarity measure is minimized. This algorithm partitions a collection of n vectors x_j , j = 1,...., n, is to be partitioned into c groups G_i, i = 1. c . Euclidean distance is chosen as a dissimilarity measure between a vector x_k in the group j and the corresponding cluster center c_i , can be defined by:

$$J = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} (\sum_{k, x_k \varepsilon G_i} \|x_k - c_i\|^2) \tag{3.12}$$

where $J_i = \sum_{k, x_k \varepsilon G_i} \|x_k - c_i\|^2$ is the cost function within the group i. Thus, the value of J_i depends on the geometrical properties of G_i and the location of c_i.

The partitioned groups are defined by a $c \times n$ binary membership matrix U, where the element u_{ij} is 1 if the jth data point x_j belongs to a group i, and 0

otherwise. Once the cluster centers $c_i]$ are fixed, the minimizing u_{ij} for Equation 3.12 can be derived as follows:

$$u_{ij} = \begin{cases} 1 \text{ if } x = \|x_j - c_i\|^2 \leq \|x_j - c_k\|^2, \\ \qquad\qquad for\ each\ k \neq i \\ 0\ Otherwise \end{cases} \tag{3.13}$$

where x_j belongs to a group i if c_i is the closest center among all centers.

Since a data point can only be in a group, the membership matrix, U have the properties as follows:

$$\sum_{i=1}^{c} u_{ij} = 1, \forall j = 1, ..., n \tag{3.14}$$

and

$$\sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij} = n \tag{3.15}$$

If u_{ij} is fixed, then the optimal center c_i that minimize equation 3.12 is the mean of all vectors in the group i :

$$c_i = \frac{1}{\mid G_i \mid} \sum_{k, x_k \varepsilon G_i} x_k \tag{3.16}$$

where $\mid G_i \mid$ is the size of G_i or $\mid G_i \mid = \sum_{j=1}^{n} u_{ij}$

The K-means algorithm is inherently iterative, and no guarantee can be made that it will converge to an optimum solution. The performance of the K-means algorithm depends on the initial positions of the cluster centers, thus it is advisable to run the algorithm several times, each with a different set of initial cluster centers.

3.3 Proposed Work

The proposed work is based on data derived from 40 student projects [1] and intend to evaluate a software-development efforts. These data are used in the implementation of a TSK based fuzzy system model using different clustering algorithm like subtractive clustering (SC), fuzzy C-means (FCM) clustering and k-means clustering algorithm. The calculated result generated using various methodologies are then compared. To calculate the effort of a given software project, basically

the following steps have been used.

Steps in Effort Estimation

1. **Calculate Class Point**: The class point (CP2) will be calculated as per the steps described in the *Figure-1.1.* Then the generated CP2 value has been used as input arguments.

2. **Normalize Dataset** : After calculating the final class point values, the data sets are then being normalized over the range [0,1]. Let X be the dataset and x is an element of the dataset, then the normalization of the x can be calculated as :

$$Normalized(x) = \frac{x - \min(X)}{\max(X) - \min(X)} \tag{3.17}$$

   where
   $\min(X)$ = the minimum value of the dataset X.
   $\max(X)$ = the maximum value of the dataset X.
   if $\max(X)$ is equal to $\min(X)$, then Normalized(x) is set to 0.5.

3. **Division of dataset**: The normalized data set is divided into different subsets using double sampling procedure. In the first step the normalized dataset is divided into *training set* and *test set*. The *training set* is used for learning (model estimation), whereas the *test set* is used only for estimating the prediction risk of the final model. The second step deals with selecting the model with optimal complexity. In this step, the *training set* is divided into *learning set* and *validation set*. The *learning set* is used for model parameters estimation, and the *validation set* is used for an optimal model complexity selection (usually via cross-validation).

4. **Perform Model Selection**: In this step, a 5-fold cross validation is used for model selection. The model which provides the least RMSE value than the other generated models based on the minimum validation and prediction error criteria has been selected to perform other operations.

5. **Select Best Model**: By taking the average of all the 5-fold's corresponding validation and prediction error (RMSE) value, the best model has been selected.

3.3.1 TSK Based Fuzzy Model Using Subtractive Clustering Algorithm

For the identification of each class, each cluster center found using the fuzzy model can be translated into a fuzzy rule. The process of selecting new cluster centers and revising potential is carried out iteratively until stopping criteria satisfied. The following criterion is used for the process of acquiring new cluster center and revising potential repeats:

Criterion: Cluster Center Finding.

Begin

if $p_L^* > \epsilon p_1^*$ **then**

Accept x_L^* as a cluster center and continue.

else if $p_L^* < \bar{\epsilon} p_1^*$ **then**

Reject x_L^* and end the clustering process.

else

d_{min} = shortest of the distance between x_L^* and all previously found cluster centers.

if $\frac{d_{min}}{r_a} + \frac{p_L^*}{p_1^*} \geq 1$ **then**

Accept x_L^* as a cluster center and continue.

else

Reject x_L^* and set the potential at x_L^* to 0. Select the data point with the next highest potential as the new x_L^* and reset.

end if

end if

Results and Discussion

The following Table-3.1 shows the values of the constants p_0 and p_1 for each generated fuzzy rule in TSK based fuzzy model using the subtractive clustering algorithm for CP2 in one fold.

Table 3.1: Type-1 TSK Fuzzy Model Developed Using Subtractive Clustering Algorithm for CP2

Fuzzy Rules	if x, then $z = p_1 \times x + p_0$
Rule - 1	if $x = exp(-\frac{1}{2}(\frac{x-0.1024}{0.3780})^2)$, then $z = 0.0075 \times x + 0.1460$
Rule - 2	if $x = exp(-\frac{1}{2}(\frac{x-0.3357}{0.3780})^2)$, then $z = 0.00765 \times x + 0.4109$
Rule - 3	if $x = exp(-\frac{1}{2}(\frac{x-0}{0.3780})^2)$, then $z = 0.00765 \times x + 0.0154$
Rule - 4	if $x = exp(-\frac{1}{2}(\frac{x-0.2140}{0.3780})^2)$, then $z = 0.1038 \times x + 0.2554$
Rule - 5	if $x = exp(-\frac{1}{2}(\frac{x-0.7243}{0.3780})^2)$, then $z = 0.3178 \times x + 0.5909$
Rule - 6	if $x = exp(-\frac{1}{2}(\frac{x-0.5079}{0.3780})^2)$, then $z = 0.4048 \times x + 0.3426$
Rule - 7	if $x = exp(-\frac{1}{2}(\frac{x-1.0000}{0.3780})^2)$, then $z = 0.7024 \times x + 0.5952$

By using Gaussian membership function, the type-1 TSK model developed using Subtractive Clustering Algorithm can be identified as TABLE 3.1. The

Table 3.2: RMSE Value using FIS (SC) for different Radius

Fold	Diff. radius	Training Set Validation Error (RMSE)	Test Set Prediction Error (RMSE)
1	0.4	0.0621	0.0762
2	0.3	0.0536	0.0959
3	0.5	0.0589	0.0909
4	0.4	0.0655	0.0589
5	0.4	0.0546	0.0884
Average		**0.0590**	**0.0823**

Table-3.2 shows the validation and prediction error in each fold and their average value has been given.

3.3.2 TSK Based Fuzzy Model Design Using Fuzzy C-Means Algorithm

In a batch mode operation, FCM determines the cluster centers c_i and the membership matrix U using the following steps:

1. The membership matrix U has been initialized with random values ranges between 0 and 1 such that the constraints in Equation 3.8 are satisfied.

2. Calculate c fuzzy cluster centers c_i, i = 1, ..., c using Equation-refeq:fcmmincond1.

3. Compute the cost function according to Equation 3.9. The computation process will be stopped if either cost function is below a certain tolerance value or its improvement over previous iteration is below a certain threshold.

4. Compute a new U using Equation 3.11. Go to step 2.

Results and Discussion

The values of the constants p_0 and p_1 for each generated fuzzy rule in TSK based fuzzy model using fuzzy C-means clustering algorithm for CP2 is shown using the following table.

Table 3.3: Type-1 TSK Fuzzy Model Developed Using Fuzzy C-Means Clustering Algorithm for CP2

Fuzzy Rules	if x, then $z = p_1 \times x + p_0$
Rule - 1	if $x = exp(-\frac{1}{2}(\frac{x-0.0300}{0.3664})^2)$, then $z = 0.0010 \times x + 0.0659$
Rule - 2	if $x = exp(-\frac{1}{2}(\frac{x-0.2291}{0.3664})^2)$, then $z = 0.0346 \times x + 0.2932$
Rule - 3	if $x = exp(-\frac{1}{2}(\frac{x-0.7429}{0.3664})^2)$, then $z = 0.2878 \times x + 0.6817$
Rule - 4	if $x = exp(-\frac{1}{2}(\frac{x-0.0990}{0.3664})^2)$, then $z = 0.2949 \times x + 0.1421$
Rule - 5	if $x = exp(-\frac{1}{2}(\frac{x-0.9993}{0.3664})^2)$, then $z = 0.6470 \times x + 0.7047$
Rule - 6	if $x = exp(-\frac{1}{2}(\frac{x-0.3880}{0.3664})^2)$, then $z = 0.6890 \times x + 0.2167$
Rule - 7	if $x = exp(-\frac{1}{2}(\frac{x-0.2064}{0.3664})^2)$, then $z = 0.6940 \times x + 0.0482$

By using Gaussian membership function, the type-1 TSK model developed using Fuzzy C-Means (FCM) Clustering Algorithm can be identified as TABLE-3.3.

3.3.3 TSK Based Fuzzy Model Design Using K-Means Algorithm

An algorithm is presented with a data set x_i, i = 1, ..., n; it then determines the cluster centers c_i and the membership matrix U iteratively using the following steps:

1. The cluster center c_i, i = 1, ..., c is initialized by randomly selecting c points from among all of the data points.

2. The membership matrix U has been determined by Equation- 3.13.

3. Compute the cost function according to Equation- 3.12. The computation will be stopped if either cost function is below a certain tolerance value or its improvement over previous iteration is below a certain threshold.

4. Update the cluster centers according to Equation- 3.16. Go to step 2.

Results and Discussion

The values of the constants p_0 and p_1 for each generated fuzzy rule in TSK based fuzzy model using k-means clustering algorithm for CP2 is shown in the following table.

By using Gaussian membership function, the type-1 TSK model developed using K-Means Clustering Algorithm can be identified as TABLE-3.4.

3.3.4 Comparison

On the basis of results obtained, the effort obtained using Subtractive Clustering (SC), Fuzzy C-Means clustering(FCM) and K-Means clustering are compared. The results show that effort estimation using fuzzy system with Fuzzy C-Means clustering(FCM) gives better values of RMSE-1.7 than those fuzzy systems implemented using Subtractive Clustering and K-Means clustering.

Table 3.4: Type-1 TSK Fuzzy Model Developed Using K-Means Clustering Algorithm for CP2

Fuzzy Rules	if x, then $z = p_1 \times x + p_0$
Rule - 1	if $x = exp(-\frac{1}{2}(\frac{x-0.2182}{0.3020})^2)$, then $z = 0.0159 \times x + 0.1460$
Rule - 2	if $x = exp(-\frac{1}{2}(\frac{x-0.0462}{0.3020})^2)$, then $z = 0.0255 \times x + 0.4127$
Rule - 3	if $x = exp(-\frac{1}{2}(\frac{x-0.1099}{0.3020})^2)$, then $z = 0.0262 \times x + 0.0126$
Rule - 4	if $x = exp(-\frac{1}{2}(\frac{x-0.0338}{0.3020})^2)$, then $z = 0.0307 \times x + 0.2708$
Rule - 5	if $x = exp(-\frac{1}{2}(\frac{x-0.8328}{0.3020})^2)$, then $z = 0.2974 \times x + 0.6404$
Rule - 6	if $x = exp(-\frac{1}{2}(\frac{x-0.1113}{0.3020})^2)$, then $z = 0.3236 \times x + 0.4709$
Rule - 7	if $x = exp(-\frac{1}{2}(\frac{x-0.3906}{0.3020})^2)$, then $z = 0.4942 \times x + 0.8736$

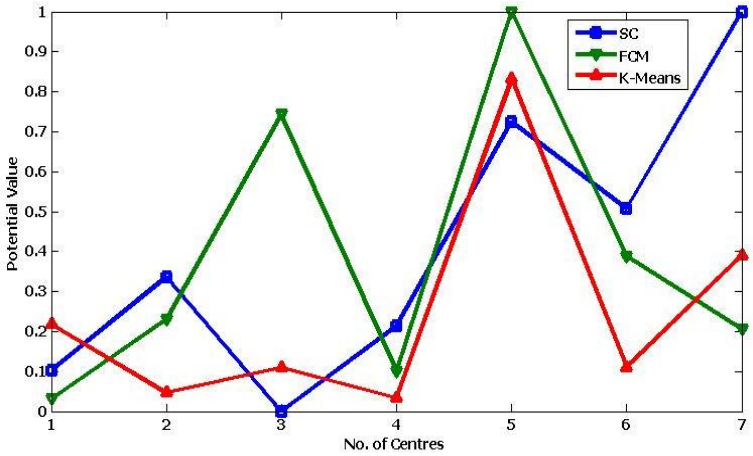

Figure 3.2: Center Points Generated Using SC, FCM and K-Means

The Figure-3.2 shows the center points generated for a TSK based fuzzy model using subtractive clustering algorithm fuzzy c-means clustering algorithm and k-means clustering algorithm.

Figure-3.3 shows the comparison of the average RMSE ($Equation - 1.7$) value of fuzzy systems using subtractive clustering, fuzzy c-means clustering and k-means clustering along with the Neuro-fuzzy inference system. When using the RMSE in evaluation, good results are implied by lower values of RMSE. The Table-3.5 displays the comparison of RMSE value for different fuzzy logic system.

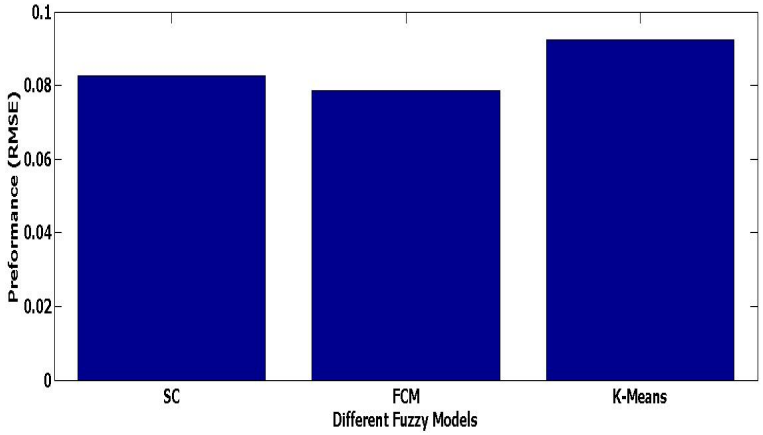

Figure 3.3: Comparison of RMSE values for SC, FCM and K-Means clustering

Table 3.5: Comparison of RMSE Value between SC, FCM and K-Means

	Subtractive Clustering	**Fuzzy C-Means**	**K-Means**
RMSE	**0.0823**	**0.0785**	**0.0925**

3.4 Summary

In this chapter, to estimate the effort of an object-oriented system, models using the fuzzy logic system has been proposed; and their results have been compared. The output shows that fuzzy system using FCM gives better results than Fuzzy logic based on various clustering algorithms such as subtractive clustering, and K-Means clustering.

Chapter 4

Conclusion and Future Work

Sr.No.	Different AI Techniques	Average Prediction Error (NRMSE)	Average Prediction Error (RMSE)
1	Multi-Layer Perceptron	0.3967	0.0856
2	K Nearest Neighbor Regression	0.4153	0.0896
3	Radial Basis Function Network	0.3546	0.0765
4	Fuzzy Logic (Subtractive Clustering)	0.3748	0.0823
5	Fuzzy Logic (Fuzzy C-Means)	0.3574	0.0785
6	Fuzzy Logic (K-Means)	0.4212	0.0925

Table 4.1: Comparison of NRMSE and RMSE.

Several approaches have already been defined in literature for software effort estimation. However, the CPA is one of the different cost estimation models that has been widely used because it is simple, fast, accurate to a certain degree. Fuzzy-logic technique is further used to find out the complexity level of the class and to calculate optimized class point. Then the calculated class point values are being normalized and used to optimize the effort estimation result. The optimization is achieved by implementing different artificial (AI) techniques such as ANN, KNN, RBFN, and fuzzy logic system with different clustering algorithm using normalized class point value. Finally, the generated minimum results of different have been compared for estimating the performance of different models. The result shows that RBFN based effort estimation model gives less value of NRMSE. Hence it can be concluded that the effort estimation using the RBFN model will provide more accurate results than other AI techniques. The results are summarized in

Table-4.1. The computations for above procedure have been implemented using MATLAB. This approach can also be extended by using other AI techniques such as genetic algorithm (GA), particle swarm optimization (PSO), Random Forest and Gradient Boosted Trees.

Bibliography

[1] G. Costagliola, F. Ferrucci, G. Tortora, and G. Vitiello, "Class point: an approach for the size estimation of object-oriented systems," *Software Engineering, IEEE Transactions on*, vol. 31, no. 1, pp. 52–74, 2005.

[2] J. Matson, B. Barrett, and J. Mellichamp, "Software development cost estimation using function points," *Software Engineering, IEEE Transactions on*, vol. 20, no. 4, pp. 275–287, 1994.

[3] F. Heemstra and R. Kusters, "Function point analysis: Evaluation of a software cost estimation model," *European Journal of Information Systems*, vol. 1, no. 4, pp. 229–237, 1991.

[4] W. Zhou and Q. Liu, "Extended class point approach of size estimation for oo product," in *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*, vol. 4, pp. 117–122, IEEE, 2010.

[5] S. Kanmani, J. Kathiravan, S. S. Kumar, and M. Shanmugam, "Neural network based effort estimation using class points for oo systems," in *Proceedings of the International Conference on Computing: Theory and Applications*, ICCTA '07, (Washington, DC, USA), pp. 261–266, IEEE Computer Society, 2007.

[6] S. Kim, W. Lively, and D. Simmons, "An effort estimation by uml points in early stage of software development," *Proceedings of the International Conference on Software Engineering Research and Practice*, pp. 415–421, 2006.

[7] Z. Zia, S. Tipu, K. Khan, and S. Zia, "Software cost estimation using soft computing techniques," *Advances in Information Technology and Management*, vol. 2, no. 1, pp. 233–238, 2012.

[8] S. Kanmani, J. Kathiravan, S. S. Kumar, and M. Shanmugam, "Class point based effort estimation of oo systems using fuzzy subtractive clustering and artificial neural networks," in *Proceedings of the 1st India software engineering conference*, ISEC '08, (New York, NY, USA), pp. 141–142, ACM, 2008.

[9] V. S Moertini, "Introduction to five data clustering algorithms," *INTEGRAL Majalah Ilmiah Matematika dan Ilmu Pengetahuan Alam*, vol. 7, no. 2, 2008.

[10] K. Bataineh, M. Naji, and M. Saqer, "A comparison study between various fuzzy clustering algorithms," *EDITORIAL BOARD*, vol. 5, no. 4, p. 335, 2011.

[11] K. Hammouda and F. Karray, "A comparative study of data clustering techniques," *Tools of intelligent systems design. In Course Project, SYDE*, vol. 625, 2000.

[12] A. Sheta, "Software effort estimation and stock market prediction using takagi-sugeno fuzzy models," in *Fuzzy Systems, 2006 IEEE International Conference on*, pp. 171–178, IEEE, 2006.

[13] A. Kaur and A. Kaur, "Comparison of mamdani-type and sugeno-type fuzzy inference systems for air conditioning system," *International Journal of Soft Computing and Engineering (IJSCE), ISSN*, pp. 2231–2307, 2012.

[14] A. L. Oliveira, "Estimation of software project effort with support vector regression," *Neurocomputing*, vol. 69, no. 13, pp. 1749–1753, 2006.

[15] K. Vinay Kumar, V. Ravi, M. Carr, and N. Raj Kiran, "Software development cost estimation using wavelet neural networks," *Journal of Systems and Software*, vol. 81, no. 11, pp. 1853–1867, 2008.

[16] P. L. Braga, A. L. Oliveira, and S. R. Meira, "A ga-based feature selection and parameters optimization for support vector regression applied to software effort estimation," in *Proceedings of the 2008 ACM symposium on Applied computing*, pp. 1788–1792, ACM, 2008.

[17] I. Attarzadeh and S. Ow, "Soft computing approach for software cost estimation," *Int. J. of Software Engineering, IJSE*, vol. 3, no. 1, pp. 1–10, 2010.

[18] V. Cherkassky, D. Gehring, and F. Mulier, "Comparison of adaptive methods for function estimation from samples," *Neural Networks, IEEE Transactions on*, vol. 7, no. 4, pp. 969–984, 1996.

[19] A. Bors, "Introduction of the radial basis function (rbf) networks," in *Online symposium for electronics engineers*, vol. 1, pp. 1–7, 2001.

[20] H. Sarimveis, A. Alexandridis, and G. Bafas, "A fast training algorithm for rbf networks based on subtractive clustering," *Neurocomputing*, vol. 51, pp. 501–505, 2003.

[21] A. Idri, A. Abran, and S. Mbarki, "An experiment on the design of radial basis function neural networks for software cost estimation," in *Information and Communication Technologies, 2006. ICTTA'06. 2nd*, vol. 1, pp. 1612–1617, IEEE, 2006.

[22] C. Panchapakesan, M. Palaniswami, D. Ralph, and C. Manzie, "Effects of moving the center's in an rbf network," *Neural Networks, IEEE Transactions on*, vol. 13, no. 6, pp. 1299–1307, 2002.

[23] M. Sugeno and T. Yasukawa, "A fuzzy-logic-based approach to qualitative modeling," *IEEE Transactions on fuzzy systems*, vol. 1, no. 1, pp. 7–31, 1993.

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

[25] R. John, "Type 2 fuzzy sets: an appraisal of theory and applications," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 06, pp. 563–576, 1998.

[26] P. Sandhu, P. Bassi, and A. Brar, "Software effort estimation using soft computing techniques," *World Academy of Science, Engineering and Technology*, vol. 46, p. 2008, 2008.

[27] H. Leung and Z. Fan, "Software cost estimation," *Handbook of Software Engineering, Hong Kong Polytechnic University*, 2002.

[28] R. Fuller, "Neural fuzzy systems," in *IN ADVANCES IN SOFT COMPUTING SERIES. BERLIN/HEILDELBERG: SPRINGER-VERLAG, 2000, ISBN*, pp. 3–7908, Springer, 1995.

[29] Q. Ren, L. Baron, and M. Balazinski, "Type-2 takagi-sugeno-kang fuzzy logic modeling using subtractive clustering," in *Fuzzy Information Processing Society, 2006. NAFIPS 2006. Annual meeting of the North American*, pp. 120–125, IEEE, 2006.

[30] S. Chiu, "Fuzzy model identification based on cluster estimation," *Journal of intelligent and Fuzzy systems*, vol. 2, no. 3, pp. 267–278, 1994.

[31] J. C. Dunn, "A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters," 1973.

[32] J. C. Bezdek, *Pattern recognition with fuzzy objective function algorithms*. Kluwer Academic Publishers, 1981.

[33] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," *Advances in neural information processing systems*, pp. 155–161, 1997.

[34] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[35] L. Devroye, "The uniform convergence of nearest neighbor regression function estimators and their application in optimization," *Information Theory, IEEE Transactions on*, vol. 24, no. 2, pp. 142–151, 1978.

Dissemination of Work

Accepted

1. Mukesh Kumar, Shashank Mouli Satapathy, Santanu Kumar Rath. Class Point Approach for Software Effort Estimation using Various Fuzzy Clustering Algorithms: A Comparative Study. *International conference on Fuzzy Systems,* Hyderabad, India, 2013.

2. Sugandha Saha, Mukesh Kumar, Santanu Kumar Rath. Comparison of different Neural Network Models for Stock Market Prediction. *25th International Conference on Software Engineering and Knowledge Engineering,* Boston USA, 2013.

Communicated

1. Shashank Mouli Satapathy, Mukesh Kumar, Santanu Kumar Rath. Optimized Class Point Approach for Software Effort Estimation Using Adaptive Neuro-Fuzzy Inference System Model. *International Journal of Computer Applications in Technology (IJCAT), Special Issue on: "Current Trends and Improvements in Software Engineering Practices".* Inderscience Publishers, 2013.

2. Shashank Mouli Satapathy, Mukesh Kumar, Santanu Kumar Rath. Fuzzy-Class Point Approach for Software Effort Estimation Using Various Adaptive Regression Methods. *CSI Transactions on ICT,* Springer, 2013.

3. Mukesh Kumar, Shashank Mouli Satapathy, Santanu Kumar Rath. Class Point Approach for Software Effort Estimation using Soft Computing Tech-

niques. *International Conference on Advances in Computing, Communications and Informatics* Mysore, India, 2013.