Software Engineering Project Effort Estimation using Fuzzy Neural Network

Submitted By

Sobia Khalid

Supervised By

Dr. Aasia Khanum

College of Electrical & Mechanical Engineering
National University of Sciences and Technology
2010

Software Engineering Project Effort Estimation using Fuzzy Neural Network

By

Sobia Khalid

2008-NUST-MS PhD-CSE (E)-22

Submitted to the Department of Computer Engineering
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Software Engineering

Advisor:
Dr. Aasia Khanum

College of Electrical & Mechanical Engineering
National University of Sciences and Technology
2010

DECLARATION

I hereby declare that I have developed this thesis entirely on the basis of personal efforts under the sincere guidance of my supervisor Dr Aasia Khanum. All the sources used in this thesis have been cited. No portion of the work presented in this thesis has been submitted in support of any application for any other degree of qualification to this or any other university or institute of learning.

Sobia Khalid

ACKNOWLEDGMENTS

All prayers and thanks to Allah Almighty, The most Merciful, and The most Beneficent. There is no success without the will of God. I am grateful to Almighty, for showing me the path to success in this work and ever before.

My special thanks are for my supervisor Dr. Aasia Khanum for the preparation of this thesis work. I would like to thank all the faculty members of the department of Computer Software Engineering, for their help and support all the time. I also thank to all the committee members for their acceptance to become the members of guidance and evaluation committee of this thesis and for sparing their precious time for reviewing the manuscript.

I am eternally thankful to my family, who have always stood by me and guided me through my career. I owe my thanks to my all friends and my classmates who gave me moral support.

In the end, I would like to thank Mr Munir Naveed, Mr Yasir Fayyaz, Mr Wasi Haider Butt, and Mr Imran Hassan for delivering their valuable knowledge and who have helped me when I got stuck.

It is impossible to remember all, and I apologize to those I've unintentionally left out.

I would like to dedicate this thesis work to

my teachers, family and friends.

ABSTRACT

"Estimation" is an important task in software engineering. Estimation has application in many areas e.g. cost estimation, effort estimation. The most crucial thing is the "effort estimation of the software project". Most of the software projects fail due to improper effort estimation. Effort estimation directly affects the budget of the software project. Software Project Mangers consider effort estimation of software engineering project a very difficult and challenging task because of its inherent imprecision. Therefore it is important and most crucial to have right effort estimate at the right time. Accurate effort estimation in software engineering projects is a challenging task, and it is one of the most crucial project management activities.

In this dissertation, we propose a Fuzzy Neural Network (FNN) model for effort estimation of software engineering project. This approach provides dual benefits of incorporating qualitative knowledge of experts and learning from historical data obtained from previous projects. The main focus of this research is to minimize the error by combining neural network and fuzzy logic and train the FNN with evolutionary algorithm. Three different datasets are used in this research work for training and testing purposes; each dataset is divided into three equal parts and each part is in turn used for testing purpose. For performance measurement, a total of 216 experiments were performed for the three datasets including the three combinations of each dataset, out of which 108 are with crossover operation and 108 are without crossover operations. The results show that the FNN is well trained by giving small values of Root Mean Square Error (RMSE) Moreover, the results show that greater the population size, lower the RMSE. The accuracy of the FNN is also based on the number of samples which are provided for training; the more the number of samples, the more the accuracy. Once trained, the FNN is supposed to predict the effort (in

person months) of a software engineering project. The results show that network is trained at an acceptable level of accuracy. Some of the examples are compared with the COCOMO model and the result shows that the proposed model behaves comparably well in these examples.

TABLE OF CONTENTS

Contents **Page No**

CHAPTER 1 INTRODUCTION

CHAPTER 2 LITERATURE SURVEY

CHAPTER 3 FRAMEWORK DESIGN

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In this chapter, we will discuss the main concepts which are used in this thesis. Artificial Intelligence in software engineering is discussed in section 1.1. Section 1.2 discusses the effort estimation. The detail of fuzzy neural network and evolutionary algorithm is discussed in section 1.3 and 1.4. Section 1.5 describes the problem statement and section 1.6 gives the summary of this chapter and the thesis organization is described in section 1.7.

1.1 Artificial Intelligence in Software Engineering

According to Lugar, Artificial Intelligence may be defined as the branch of computer science which deals with the mechanization of human/intelligent behavior [1]. There are several branches of artificial intelligence which include: natural language processing, computer vision, robotics, problem solving and planning, learning and expert system [2].

According to Somerville, software engineering is considered an engineering discipline which deals with all characteristics of software production. It also deals with the physical constraints of developing and delivering valuable software [3].

Software Engineering has been criticized from the beginning. The main criticism on software engineering is that it is not well developed science at all because software engineers do not know accurately that how long their projects will be applicable, project's cost, and either software work properly after release or not [4].

The models and methods developed by AI can be valuable for software engineering [5]. The models and methods developed by AI can be used to estimate the effort and cost of the software projects.

1.2 Effort Estimation

The estimate of the effort is considered as the most significant variables in the process of project management because it helps to plan the forthcoming activities [6]. Effort Estimation is defined as the prediction of number of hours and numbers of employees are needed to develop a project [6]. Software development effort estimation is based on incomplete, uncertain or noisy input and effort estimation is used as an input for investment, budget analysis and project plans [7]. Accurate estimation of effort/cost/time is a huge problem for software engineers [7]. In order to carry out the prediction with low error rate, a large volume of data is needed which is very difficult to obtain [6].

The categorization of estimation approaches are as follows:

- Expert Estimation
- Formal Estimation Model
- Combination Based Estimation

1.2.1 Expert Effort Estimation

In expert estimation, the quantization step is a judgment based, and the quantification step is defined as a step where the required effort for a software project is measured. The term "Expert" may be used for an individual or for a team. The quantification steps are based on the perception and sometimes may be based on the unambiguous, analytical argumentation. It is assumed that experts typically possessed more information and have the more flexibility that how information is processed [8].

1.2.2 Model based Effort Estimation

In model based estimation, the quantification step is mechanical. For example, use a formula derived from historical data like COCOMO. However, it may be complicated to build models for software development effort estimation because of lack of stable relationships and use of small data sets to build models [8].

1.2.3 Combination based Effort Estimation

In combination based effort estimation, the quantification step is based on a judgmental or mechanical combination [7].

1.3 Fuzzy Neural Network

1.3.1 Neural Network

The brain consists of a large number of neurons which are connected with each other by synapses. These networks of neurons are called neural networks, or natural neural networks. The artificial neural network (ANN) is a simplified mathematical model of a natural neural network. Basically it seems like a directed graph where a vertex corresponds to a neuron and an edge to a synapse [9].

1.3.1.1 A Neuron

The computational model of artificial neuron is motivated by the functionality of natural neuron. Natural neurons receives signals through synapses, and when the received signal is strong enough, then the neuron is activated and release the signal through axon, and this signal might sent to another neuron or synapse [10]. This all work is shown in Fig 1.1.

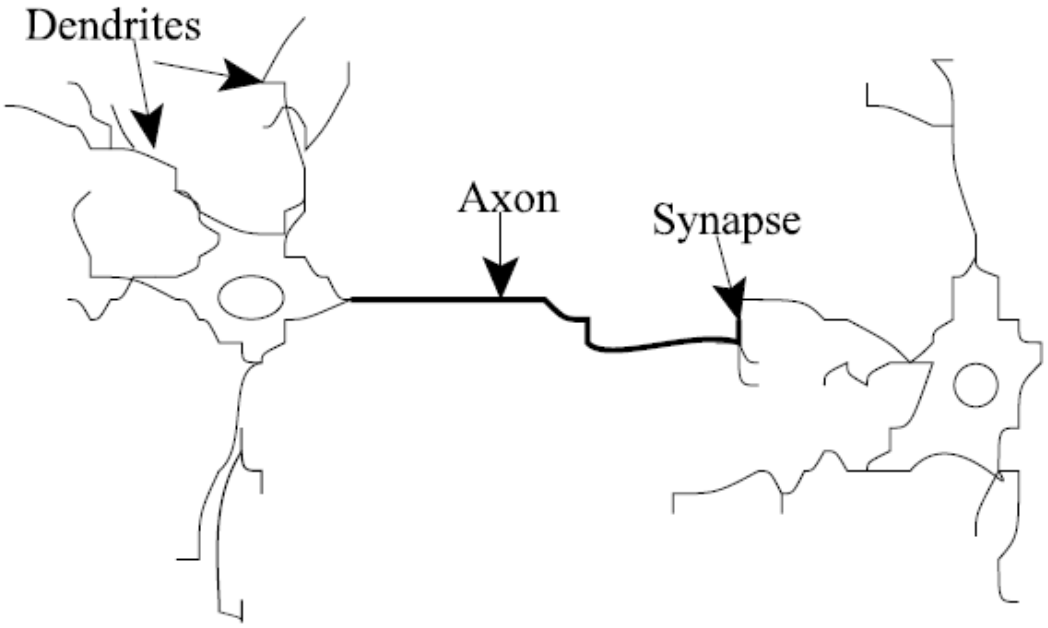

Fig 1.1 Natural Neuron [10]

In artificial neuron, the synapses are basically the inputs, which are multiplied by the connection weights which show the strength of the signal and then determine the activation of the neuron through some mathematical function. The artificial neural network is basically the combination of these artificial neurons [10]. The mathematical model of artificial neural network is shown in Fig 1.2.

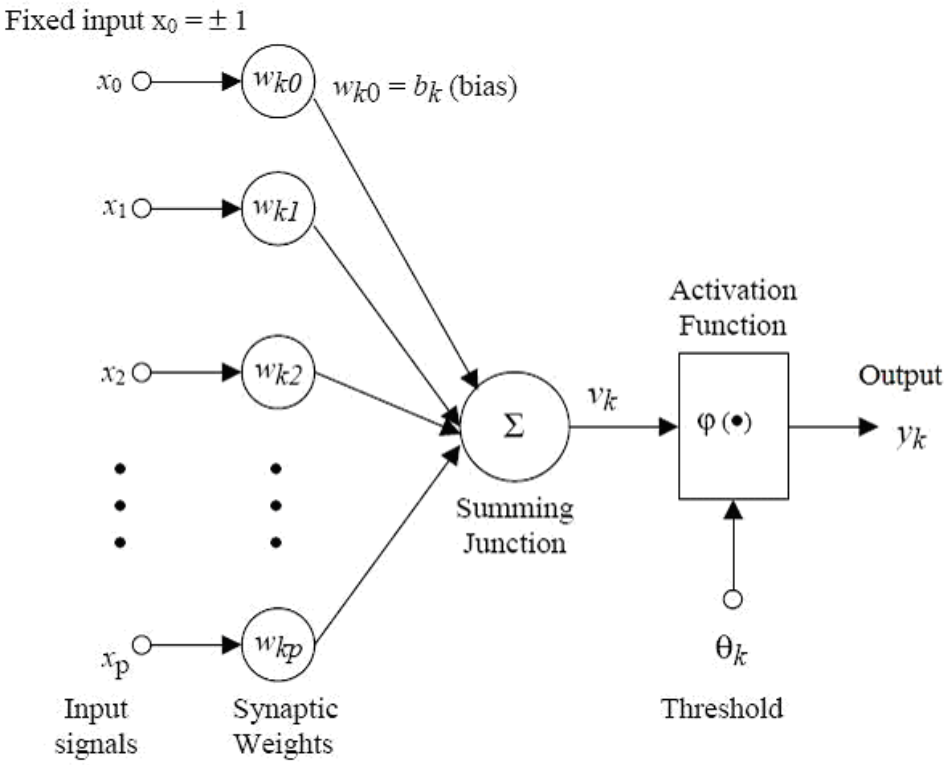

Fig 1.2 Model of a neuron [11]

From this model the interval activity of the neuron can be shown to be:

$$v_k = \sum_{j=1}^{p} w_{kj} . x_j \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{ 1.1 [11]}$$

The output of the neuron, yk, would therefore be the outcome of some activation function on the value of vk.

$$y_k = f(v_k) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{ 1.2}$$

Where f is any activation function.

1.3.1.2 Activation Function

There are three types of activation function namely:

i). Threshold Function

ii). Piecewise Linear Function

iii). Sigmoid Function

Threshold function has possible two values of 0 and 1. Threshold Function has a value of 0 if the summed input (v_k) is less than a certain threshold value (v), and the value 1 if the summed input (v_k) is greater than or equal to the threshold value [11].

In piecewise Linear function, the certain region or we can say that the certain values remain as it is while other changes to 0 and 1 [11]. For example, if we define the range of 0.3 and 0.6. Then if the value of the summed input comes between 0.3 and 0.6, then it remains as it is. However, if the value comes below 0.3 then the result is 0 and if it comes above 0.6, then the result is 1.

In sigmoid function, the output also comes between 0 and 1, and the output varies continuously depending on the summed input.

1.3.1.3 Learning Paradigms

We can categorize the learning situations in two distinct types. These are:

i). Supervised Learning

ii). Unsupervised Learning

1.3.1.3.1 Supervised Learning

Supervised learning is also called associative learning. In supervised learning, the network is trained by providing the input and desired output patterns. These patterns can be provided by a expert [11].

1.3.1.3.2 Unsupervised Learning

Unsupervised learning is also called self organization. In this type of learning, no input-output patterns are provided, only the input is provided. The system has supposed to discover the characteristics of the input [11].

1.3.1.4 Neural Network Topologies

The topologies of the neural network are based on the connection between the neurons and propagation of data. So we can distinct the topology in two different ways.

1.3.1.4.1 Feed Forward Neural Network

Feed-forward ANNs allow signals to travel one way only; from input to output. There is no feedback (loops) connections present i.e. the output of any layer does not affect that same layer or previous layers [11].

1.3.1.4.2 Recurrent Neural Network

In recurrent neural network, feedback connections are present [11]. The network feed the outputs from neurons to other adjacent neurons or to neurons in preceding layers.

1.3.2 Fuzzy Logic

The idea of fuzzy logic was first invented by Dr Lofti Zadeh. Fuzzy logic is considered as an approach to calculate based on the degree of fact rather than the Boolean values. Fuzzy logic includes Boolean values of 0 and 1 as extreme values of fact, but also includes the various cases of truth between these extreme values [12]. Fuzzy logic is one of the means that can model multi-input and multi-output system. It provides the ability for modeling states that are inaccurately defined. Fuzzy Logic techniques have been used in classification, clustering, feature extraction etc. It has the ability to mimic the human mind to utilize modes of reasoning that are approximate instead of exact [13]. The basic concepts which are used in fuzzy logic are discussed below.

1.3.2.1 Fuzzy Sets

Fuzzy set is considered as an extension of crisp set. However, crisp sets allow only full membership or membership not at all; it does not allow any partial membership state, where as, fuzzy set allows the partial membership state. Fuzzy set describe linguistic labels like low, medium, high. A given element can be the member of more than one fuzzy set at a time, and how much it belongs to one fuzzy set is indicated by their membership grade [13].

1.3.2.2 Membership Functions

Membership functions are used to convert the crisp value into fuzzy value. A membership function curve defines that how each point in the input space is converted into membership value. There are different types of membership function including triangular, trapezoidal, generalized bell shaped, guassian curve, polynomial curve, and sigmoid function [13].

1.3.2.3 Logical Operations

In fuzzy logic (FL), the truth of statement is the subject of degree. FL operators include min, max and complement operations. Most applications use min for fuzzy intersection, max for fuzzy union, and 1-μ (a) for complement operations [13].

1.3.2.4 IF-THEN Rules

If-then rules describe the relationship between fuzzy input and output sets. The “if” part of the rule is called antecedent or premise, and “Then” part of the rule is called consequent. If-then rule involve two discrete steps. The first step includes evaluation of the premise which involves fuzzification and applies any operator if required and second step is called implication in which the result of the antecedent is applied to the consequent [13].

1.3.2.5 Fuzzy Inference System (FIS)

FIS defines a nonlinear mapping of the input and output data with the help of fuzzy rules. The mapping method includes input/output membership function, FL operators, if-then rules, collection of output sets and defuzzification. FIS mainly has four components, fuzzifier, inference engine, rule base and defuzzifier. Fuzzifier converts the input into corresponding fuzzy membership value. The inference engine maps the fuzzy input sets to the fuzzy output sets. It calculates the degree to which premise is satisfied to the each rule. If more than one rule fires at the same time, then output of all the rules are aggregated. The defuzzifier maps the fuzzy output set into fuzzy number [13]. The block diagram of FIS is shown in Fig 1.3.

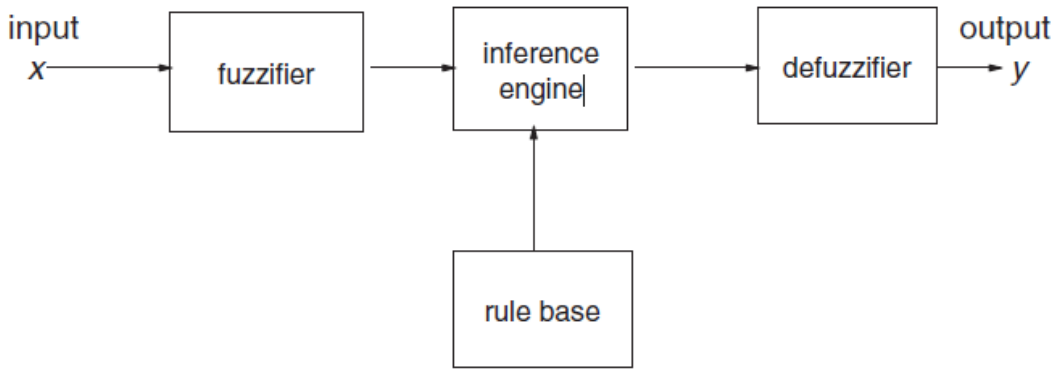

Fig 1.3 Block Diagram of FIS [13]

1.3.3 Fuzzy Neural Network

Neural Network and Fuzzy Logic, both are considered as the best possibility to deal with complex and poorly defined data [14]. With the integration of neural network and fuzzy logic, it is possible to deal with the expert knowledge and also capable of handle incomplete data. The strengths of fuzzy neural network includes fast and precise learning, good generalization capabilities, and ability to accommodate both data and existing expert knowledge [15].

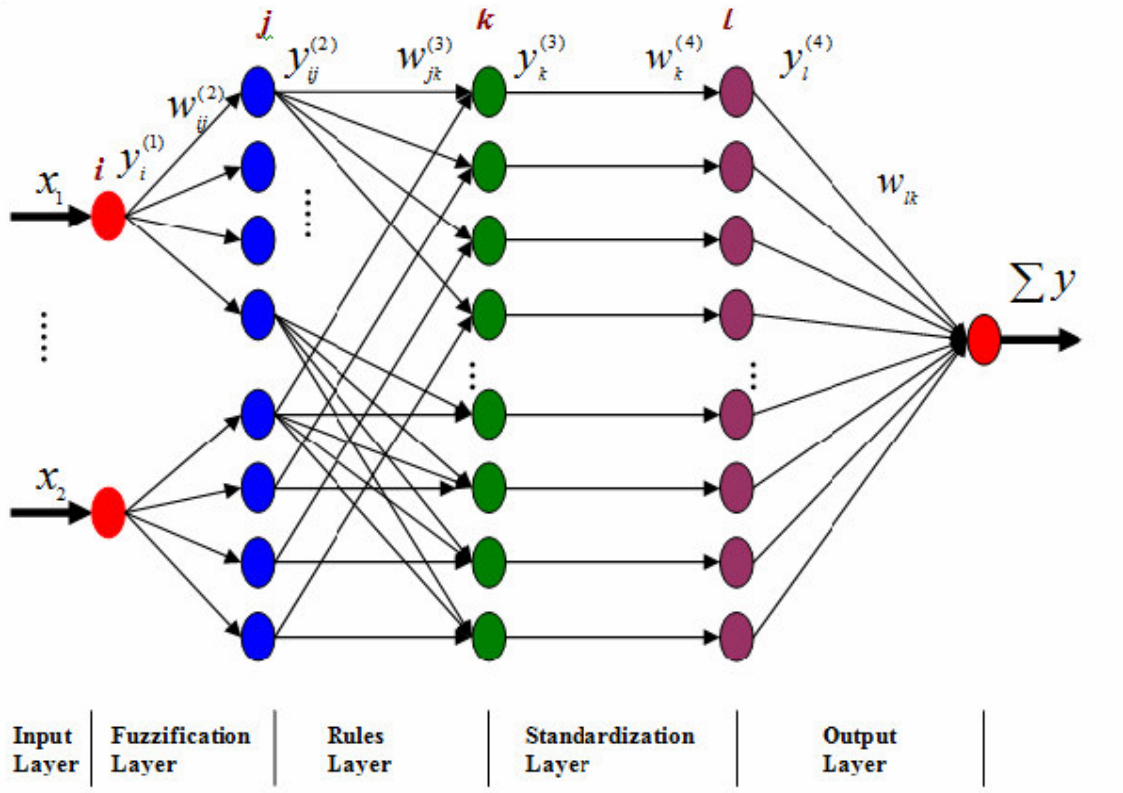

Fig 1.4 Structure of Fuzzy Neural Network [18]

1.3.3.1 Architecture of Fuzzy Neural Network

The fuzzy neural network may contain three or five layers.

1.3.3.1.1 Input Layer

The first layer is called input layer which simply passes the input to the next layer. The input nodes represent linguistic variables [14].

1.3.3.1.2 Conditional Element Layer

The second layer is called conditional element layer which performs fuzzification. The nodes at this layer act as the membership function [14] and output of this node is the degree that the input belongs to the given membership function [15].

1.3.3.1.3 Rule Layer

The nodes in the rule layer is called rule node and each rule node represents one fuzzy rule. The input to the rule node is the fuzzy input that comes after applying the fuzzy membership function at conditional element layer, and output of this node is the fuzzy output that passes to the action element layer. The association between layer 2 and 3 are called precondition link and association between layer 3 and 4 are called consequent link [14]. The links between conditional and rule layer perform precondition matching of fuzzy rules. So the connection weights are either set randomly or then trained through some algorithm [15].

1.3.3.1.4 Action Element Layer

In action element layer, the nodes represent the fuzzy labels of the output variable [15]. For example, if the output variable has three labels, then action element layer contain three nodes, each node represent one label. The activation of the node represents the degree to which this membership function is supported by the current data [15].

1.3.3.1.5 Output Layer

The output layer performs defuzzification to convert the fuzzy value into crisp output value. The Center of Gravity is the most commonly used defuzzification method that acquires the best result [15].

1.4 Evolutionary Algorithm

Evolutionary algorithm is a stochastic search method. It model natural processes such as selection, recombination, mutation and reinsertion. It works on population of individuals instead of a single solution. It provides a number of possible solutions to a given problem [16].

The pseudo-code of evolutionary algorithm [17] is given below.

Pseudo-Code

Initialize Population with random candidate solution.

Evaluate every candidate solution.

Repeat until Termination Condition is satisfied do

1. *Select Parents*
2. *Recombine pairs of best parents*
3. *Mutate the resulting offspring*
4. *Evaluate new candidates*
5. *Select Individuals for the next generation*

End

1.4.1 Initialize Population

The first step in the algorithm is to initialize the population. Population contains the number of individuals. Each individual contain one possible solution. The size of the population is defined by the user. The variety of population is the measure of number of different solutions [17].

1.4.2 Evaluation of Population

In evaluation of population, the solution of every individual is evaluated based on fitness function. There are different criteria of fitness function. For example, we can evaluate the solution of population based on their error; in this case error is the basis of fitness function.

1.4.3 Selection Process

In the selection process, the parents which are survived for the next generation are selected, and which are used to produce off springs. There are different methods of selection including random, absolute, roulette and rank selection [19].

In random selection process, choose the individuals at random. In absolute selection process, n-best fit individuals are selected which replaces the n-worst individuals. In roulette wheel selection, relative weights proportional to their fitness are assigned to individuals, and individuals are picked randomly. In rank selection process, rank is assigned to each individual based on their solution; the best fit individual has the rank number 1 while the least fit has the individual 10 [19].

1.4.4 Recombination

Recombination is also called cross over. In this operation, it merges the information from best parents into off springs. The principle behind recombination is that by crossovering the two individuals with different features can create the off spring by combining the features of both. Recombination operator can use two or more than two parents [17].

1.4.5 Mutation Process

A unary variation operator is called mutation. It is applied to one individual. It is a random process which a little modifies the genes of the individuals. Mutation process may include lengthening, shortening or modifying the genes of the individual [19].

1.4.6 Reinsertion

After producing the off springs, they must be inserted into the population. Reinsertion scheme determined that which individuals ought to be inserted in next generation and which individuals are to be replaced by the new offspring's [16].

1.5 Problem Statement

Accurate effort estimation in software engineering projects is a challenging task, and it is one of the most crucial project management activities. Project mangers consider it a very difficult and challenging task because of its inherent imprecision. It helps the project manager of a software project to plan the activities that are required for project completion and guides project manager to allocate their resources (e.g. hire the staff) as per requirement.

Most of the software projects fail due to improper effort estimation. Effort estimation directly affects the budget of the software project. Therefore it is important and most crucial to have right effort estimate at the right time.

A model is developed for software project effort estimation by combining artificial intelligence techniques of fuzzy logic and neural networks.

1.6 Summary

Accurate Effort Estimation is the one of the most challenging task. It helps project manager to plan upcoming activities. For effort estimation, fuzzy neural network is used. The approach provides dual benefits of incorporating qualitative knowledge of experts and learning from historical data obtained from previous projects. It is basically hybrid approach combining artificial intelligence techniques of fuzzy logic and neural networks. Fuzzy logic contain approximate reasoning whereas neural network is able to learn the ill defined data. For

training of fuzzy neural network, evolutionary algorithm is used which contain different solution among which best one is selected.

1.7 Dissertation Organization

The thesis is structured into chapters and organized as follows: Chapter 1 provides the brief introduction about the terms used in the thesis. Chapter 2 provides the literature review related to effort estimation. Chapter 3 discusses the framework design in detail. Chapter 4 provides the implementation details of the thesis. Chapter 5 provides the analysis of the results that comes from training and testing data. Chapter 6 concludes the thesis with future work.

CHAPTER 2

LITERATURE SURVEY

In this chapter, we will discuss the literature review regarding our work. This chapter initially describes the different approaches used for effort estimation. Section 2.1 name the estimation approaches, section 2.2 name the estimation tools and discusses few of them, whereas some models are discussed in section 2.3. The summary is given in section 2.4.

2.1 Estimation Approaches

There are many effort estimation models which have been established over the last decades. A number of estimation approaches including: analogy based estimation, WBS-based estimation, parametric models, size-based estimation models, group estimation, mechanical combination, judgmental combination [20].

2.2 Estimation Tools

There are a number of estimation tools available in market including: 20s estimation calculator, 20s reference estimator, ACEIT, COCOMO II, Construx Estimate, CostXpert, Coaster, Estimate Express, EstimatorPal, EZEstimate, FP Outline, Function Point WorkBench, knowledge Plan, Oracle Crystal Ball, QUEST, RASS Estimate, REVIC Software Estimation Model, SCOPE, SEER for software, SEER IT, SLIM Estimate, SystemStar, True Planning for IT [21].

20s estimator calculator estimate fixed price or time, and specify the restriction on time, cost or personnel, and it will work with excel version 97, 2000, 2002, whereas *20s reference*

calculator create precise estimate for future projects by indicating the historical projects. It accumulates the actual hours and original estimated hours that are required for delivery on project. *ACEIT* contains a number of applications that support project manager during all phases of project life cycle. It is a tool for analyzing, developing and sharing the cost estimate. *COCOMO II* is a model that estimates the cost, effort and schedule when planning a new software development activity. It contains the three sub models including application composition, early design and post architecture model. *Coaster* is a software development estimation tool based on the Constructive Cost Model described by Barry Boehm. Software Project Manager use Coaster to estimate project duration, staffing level, cost and effort. *Estimate Express* is a software project estimating tool. It estimates the cost, reliability, schedule and resources on big and small projects. *Function Point WORKBENCH* estimate project effort, time and cost, asses' software delivery productivity and quality. *SEER for Software* estimating software projects, it plan, analyze and manage complex software development projects. It evaluates software parameters not as isolated factor, but as inter dependent variables across project aim, constraints, work products and life cycle. *SLIM Estimate* helps to estimate the time, effort and cost required to satisfy a given set of software requirements and conclude the best strategy for designing and implementing the software project [21].

2.3 Literature Review

Some of the models are discussed below.

Sheppard et al used analogies to estimate software project effort. Estimation by analogy is the form of CBR (Case Based Reasoning). The key activities for estimation by analogy include identification of the problem, the retrieval of cases, and reuse of knowledge and suggestion of

solution for the new case. The approach used in this paper has been guided by the aims of feasibility and simplicity. The author takes a new project, one for which wish to predict effort, and attempt to find other similar projects. Since these projects were completed and their development effort will be known, and it can be used as a basis for estimating effort for the new project. The author analyzed the data in terms of features. Features may be categorical or continuous. The datasets used in the experimentation are quite diverse and drawn from different application domain ranging from telecommunication to commercial information system. A jack-knifing procedure was adopted for analogy based prediction. Each dataset is treated separately, since each one has different project features available, and therefore not able to merge data into a single all encompassing dataset. The result shows that for all the dataset, the MMRE performance of estimating by analogy is better than that of regression methods. This suggests that analogy is capable of yielding more accurate prediction for these dataset. Hence the author at the end, conclude that accurate estimation of software project effort at an early stage in the development process is a significant challenge for the software engineering community. The approach adopted in this paper allows user to assess the reasoning process behind a prediction by identifying the most analogous project, thereby increasing, or reducing their confidence in prediction [22].

In 2009, Reddy et al proposed a neural network model for effort estimation and compare the results with the COCOMO model. The architecture of neural network which is used is multi layer feed forward network trained with back propagation algorithm. According to Reddy et al, this new model improves the performance and accuracy in predicting the effort. The data used for this research is COCOMO data set. The data is divided into two sets: Training set and testing set. The proposed neural network is trained with the training sample and then its accuracy is calculated using tested sample. The proposed neural network model requires 22

input nodes in the input layer that corresponds to the effort multipliers, scale factors and bias values. The process of establishing the neural network involves initializing, training and then testing the network. First train the neural network with the expert judgment's input. Then the weights are updated with the back propagation algorithm. The back propagation procedure iteratively processing the set of training samples and compare the network results with the actual one. After each iteration, the weights are modified in order to minimize the error between network calculated output and actual output, and when it comes to certain stopping criteria, training stops. The stopping criteria can be the error is smaller than specific tolerance or the number of iteration which are exceed to a specific number. After training the network, the model is tested with testing samples and the results are compared with the COCOMO model effort estimate. The results are compared based on the criteria of MRE (Mean Relative Error). The results show that the proposed model produces more accurate and more significant results than the COCOMO model [23].

Braga et al introduces the effort estimation along with confidence interval based on machine learning. Machine learning uses the data from the past project to build the model and then use that model to predict the effort of the new projects. In this research work, the objective is to build the regression model using a training dataset that will be used to predict the effort of the development project in man-months. M5P algorithm is used to build the regression based tree. Regression tree is a special kind of decision tree. Robust confidence interval is computed from the prediction error that comes after the regression model is built. To evaluate the method, two data sets are used, the first one is Derharnais and second one was NASA dataset. The data set is divided into two sets called training and testing set. Training data set is used to build the model and testing is used to check the accuracy of the model. The

authors take two measurements: MMRE and PRED (25). The results show that the proposed method was able to build robust confidence interval [24].

Kodada et al experienced the case base reasoning to predict software project effort. assists in more effective use of CBR techniques for prediction system by providing experimental data. The analysis for this research work is based on the dataset collected from Canadian Software House. The pre-processing that is required for the dataset include the deletion of incomplete cases that are present. The dataset is divided into smaller dataset to explore the impact of size n. However, in order to generate the prediction, adopted a jack-knifing procedure. According to author, when more than one analogy is selected, ANGEL will calculate the mean of chosen analogies. Instinctively, the author expect that closest analogies to have more influence. In short, the analysis suggests the decision that how to configure CBR system [25].

Deng et al estimated software effort using harmonizing algorithms and domain knowledge in an integrated data mining approach. The authors present the integrated data mining framework. Integrated data mining framework incorporates the domain knowledge into series of data analysis and modeling processes. According to authors, integrating domain knowledge into data mining solution is challenging, because some domain knowledge is difficult to put in explicit form such as rules. The computational procedures like feature selection, visualization etc are not detached from each other, but they share active features between them. For experimentation, the author takes the dataset from PROMISE Software Engineering Repository. The feature selection index was employed to value the value of attributes in predicting effort. The normal-cut algorithm for spectral clustering was used to cluster the data into four clusters, and project the label into 2-D display. The authors used the nearest neighbor based algorithm for modeling of effort data. First work with the entire data

set for effort prediction, and from the results it was observed that when using log transformed effort data can significantly improve the indexes. Then working with the subset of data and the results obtained using homogenous subset are improved in two characteristics: prediction accuracy and prediction stability. However, from the experimentation, it was found that domain knowledge can be found to enhance, evaluate and confirm the computational outcome [26].

Song et al used grey relational analysis to predict software effort with small data sets. The authors in this paper particularly focus on feature subset selection and effort prediction at an early stage of project. For this purpose, the author proposes an approach using Grey Relational Analysis (GRA) of Grey System Theory (GST). The results are evaluated on the five publicly available data sets. The result shows that this approach is better than other machine learning techniques. Software development is a growing process because as it proceeds, collect and analyze deficient information, and try to optimize the development process. So according to author, effort prediction methods must be suitable for dealing with uncertainty and continuous effort estimating [27]. GST requires a limited amount of data to estimate the behavior of uncertainty system. In this paper, the authors focus on the feature subset selection and effort prediction with-between project data sets at an early stage of development process [27]. Grey Relational Analysis is used to measure all the influences of various factors and relationships among data series that is a collection of measurement. The authors view the feature data as series and software effort as output and applied Grey Relational Analysis to select the optimal feature subset for software effort prediction. The steps to select optimal feature subset using GRA includes: data series construction; normalization; grey relational grade calculation; feature subset selection. The GRA based software effort prediction method GRACE first construct data series from a project, and

measure the grey relational grade among the series, and then use most significant projects to predict the effort for new project [27].

Attarzadeh et al introduces fuzzy logic concept in fuzzy logic model which include three main stages, fuzzification, inference of rules and defuzzification. The evaluation criteria for the model are Magnitude of Relative Error. In order to develop a fuzzy system, the requirement is to develop a fuzzy system requires that the different categories of different input into fuzzy sets. The membership functions which are used in it are two sided guassian membership function, and some rules are also suggested for the model. The result shows that MMRE by applying fuzzy logic is smaller than other fuzzy logic models [28].

2.4 Summary

There are various approaches used for effort estimation, also there are a number of tools which are available for effort estimation. Some of the models which are developed for effort estimation are discussed. However, the main conclusion is that reliability of the effort estimation is still the big issue.

CHAPTER 3

FRAMEWORK DESIGN

In this chapter, the design of framework is discussed in detail, which will be adopted to build the model which will be used for effort estimation. For building the model, data is collected from different sources which are used to train and test the fuzzy neural network.

The design of the database is discussed in section 3.1. The method used to find the parameters of guassian membership is discussed in section 3.2. Section 3.3 describes the process that is adopted to extract rules from data. The topology and training algorithm of FNN is discussed in section 3.4. The summary of this chapter is given in section 3.5.

3.1 Database Design

The basic step for the implementation is to design the database, because in order to train the fuzzy neural network, we have to retrieve the data from the database, so its designing is important. The database engine which is used to store the data is MS Access which is easy to use by everyone. The structure and details of database are as follows:

3.1.1 Structure of Database

In the database, we have basic two tables, one is “Training Data”, which is used to train the fuzzy neural network, the other one is “Testing Data”, which is used to test or in other words to evaluate the performance the fuzzy neural network. The attributes of training and testing data are discussed below.

3.1.1.1 Training & Testing Data

The attributes for the training and testing data are the same which are given in table 3.1.

Training & Testing Data Attributes			
Field Name	**Data Type**	**Primary Key**	**Field Size**
ID	AutoNumber	Yes	Long Integer
PREC	Text		255
FLEX	Text		255
RESL	Text		255
TEAM	Text		255
PMAT	Text		255
RELY	Text		255
DATA	Text		255
CPLX	Text		255
RUSE	Text		255
DOCU	Text		255
TIME	Text		255
STOR	Text		255
PVOL	Text		255
ACAP	Text		255
PCAP	Text		255
PCON	Text		255

AEXP	Text		255
PEXP	Text		255
LTEX	Text		255
TOOL	Text		255
SITE	Text		255
SCED	Text		255
LOC	Number		Double
ACTUAL_EFFORT	Number		Double

Table 3.1 Attributes of Training & Testing Data

There are 25 fields in the table. All the input attributes are categorized into six different rankings ranges from very low, low, nominal, very high, extra high and high. The output/target field is categorized into low, medium, high. The description of the attributes is as follows.

3.1.1.1.1 ID

The field named ID is a primary key and it simply contains the row number.

3.1.1.1.2 PREC

PREC stands for “Precedentedness”. It reveals the experience the of the software developer to the present project context [29]. If the software developer had been working in present project context then PREC is high.

3.1.1.1.3 FLEX

FLEX represents “Development Flexibility”. It holds the number of constraints that the project has to meet. The more rigid the constraints lower the rating [30].

3.1.1.1.4 RESL

RESL indicates “Architecture/Risk Resolution”. It combines two factors “Design Thoroughness” and “Risk Elimination” [31]. The rating of this attribute is the weighted average of the several characteristics. For example, one of the characteristics is that risk management plan identifies all vital risks [31].

3.1.1.1.5 TEAM

TEAM stands for “Team Cohesion”, and it signifies the capability of a team to work as a team [29]. This factor consider for the causes of project instability due to complexity in coordinating the project stakeholder. This may be due to deficient of experience in stakeholders to work as a team [31].

3.1.1.1.6 PMAT

PMAT represents “Process Maturity” and it is organized around the Software Engineering Institute’s Capability Maturity Model and the time period for the rating is that when the project starts [31].

3.1.1.1.7 RELY

RELY stands for “Required Software Reliability”. Rely is defined as the measure of the amount of to which software must perform its projected role. If the software failure causes

slightly trouble, then Rely is low, and if it causes danger to human life, then Rely is very high [31].

3.1.1.1.8 DATA

DATA stands for “Database Size”. Data tries to measure the influence that large data requirement have on product development [31]

.

3.1.1.1.9 CPLX

CPLX indicates “Product Complexity”. According to [31], complexity is divided into five areas which are: control operations, computational operations, device-dependent operations, data management operations, and user interface management operations. The evaluation of the complexity depends on the individual weighted average of these areas [31].

3.1.1.1.10 RUSE

RUSE stands for “Required Reusability”. This cost driver is evaluated in terms of effort needed to build components which will be used in future. If the component is build to be used across multiple programs, then effort is high [31]

.

3.1.1.1.11 DOCU

DOCU indicates “Documentation match to Life-Cycle needs”. The evaluation of this driver is based on the appropriateness of the project’s documentation to its life cycle needs [31]. When the project documentation is appropriate according to the life cycle needs, then DOCU is high.

3.1.1.1.12 TIME

TIME represents “Execution Time Constraint”. Time is defined as the evaluation of the execution time constraint that is forced upon a software system and is expressed in terms of the percentage of the execution time used by the system or sub system out of the available execution time [31].

3.1.1.1.13 STOR

STOR represents “Main Storage Constraint”. Stor represents the degree of the storage constraint that will be consumed by the software [31].

3.1.1.1.14 PVOL

PVOL indicates “Platform Volatility”. The word platform is used to represent the software and hardware which are used by any program to perform its assignment. It can include any compilers and assemblers which support the development of the software system. If there is major change in platform after every twelve months, then PVOL is low [31].

3.1.1.1.15 ACAP

ACAP stands for “Analyst Capability”. Acap is a personnel factor and depends on the individual analysis and design capability, efficiency and carefulness, and ability to communicate and cooperate [31].

3.1.1.1.16 PCAP

PCAP shows “Programmer Capability”. The evaluation of the programmer capability is based on the aptitude of a programmer as a team, as well as ability, efficiency, carefulness and ability to communicate and cooperate [31].

3.1.1.1.17 PCON

PCON indicates “Personnel Continuity”. The evaluation of PCON is based on the project annual personnel turnover [30]. If the project annual personnel turnover rate is 48% per year, then PCON is very low [31].

3.1.1.1.18 AEXP

AEXP shows “Application experience”. Aexp shows the level of application experience the project team which are developing the software has with this type of application [31].

3.1.1.1.19 PEXP

PEXP stands for “Platform Experience” and accounts for the understanding of the platforms including more graphical user interface, database, networking and distributed middleware capabilities. If the developer has platform experience of 2 months, it is considered as very low [31].

3.1.1.1.20 LTEX

LTEX indicates “Language and Tool Experience” and it is a measure of level of experience of programming language and software tool. An experience of less than or equal to two months, it is counted as very low [31]

.

3.1.1.1.21 TOOL

TOOL represents the “Use of Software Tools”. The use of software tool ranges from simple edit and debug code to integrated lifecycle management tools [31].

3.1.1.1.22 SITE

SITE stands for “Multisite Development”. The evaluation of this attribute is based on measurement of site collocation and communication support. For example, if the site communication is through some phone, mail, it is considered as very low [31].

3.1.1.1.23 SCED

SCED show “Required Development Schedule”. Sced measure the schedule constraint imposed on the project team developing the software and is defined in terms of the percentage of the schedule stretch out [31].

3.1.1.1.24 LOC

LOC stands for “Lines of Code”. Loc is the measure of the size of project. The loc is based on the code which is delivered as a part of product, created by project staff [32].

3.1.1.1.25 ACTUAL_EFFORT

ACTUAL_EFFORT is the total effort required for project in terms of man month [33].

The next step which comes is to train the fuzzy neural network. For training of fuzzy neural network, we need to know rules and have to find the parameters of the guassian membership function for the continuous attribute named “LOC”.

3.2 Finding the Parameters of the Gaussian Function

We have to find the parameters of the guassian membership function for the attribute named "LOC". LOC contain continuous numeric data. The rating of "LOC" is categorized as

- Very Low
- Low
- Medium
- High
- Very High
- Extra High

The reason to use guassian membership function is that for effort estimation, it behaves well as compared to triangular membership function [34]. To find the parameters of the membership function of each category of the attribute, first we have to know the guassian membership function. The formula is

$$u(x) = \frac{e^{-(x-c)^2}}{2\sigma^2} \quad \text{..... 3.1 [34]}$$

From the formula, we know that we have to find the mean and variance for each category. To differentiate the ranking, the data is divided into six different clusters and each cluster represent one category, then we find the mean and variance of each cluster and the guassian membership function for each cluster is different. To make cluster of each category, we have used k-means clustering. First, we will discuss how to make clusters, and then from clusters how to find mean and variance.

The basic flow of the k-means clustering is shown in the figure 3.1.

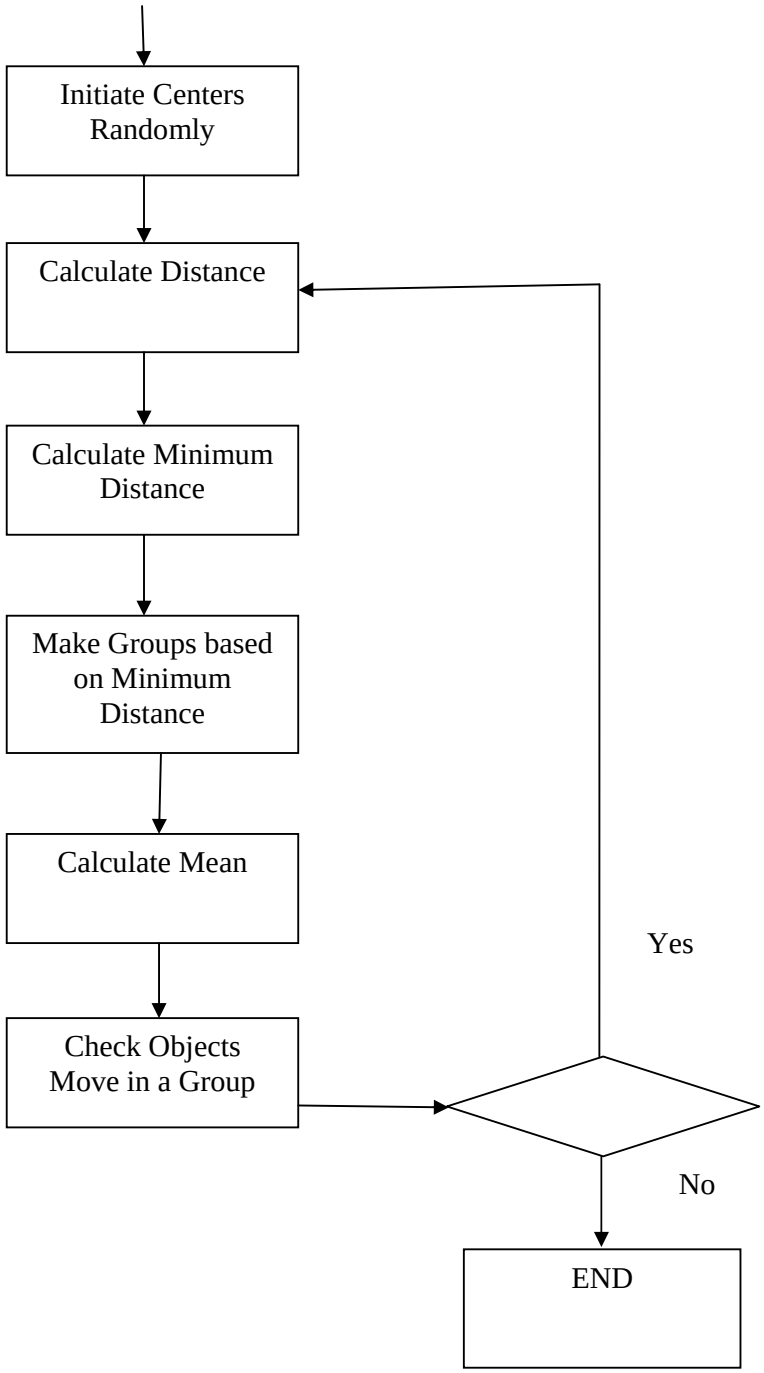

Fig 3.1 k-Means Clustering Flow

The steps which are adopted to make clusters are shown turn by turn:

3.2.1 Initiate Centers Randomly

The first step to make clusters is to initiate the centers randomly. The number of centers is equal to the number of clusters. So here, the number of centers which are initialized is six.

3.2.2 Calculate Distance

The next step is to calculate the distance of each object against every center. To find the distance, we used Euclidean distance.

$$d = \sqrt{(xi - yi)2}$$ ……..... 3.2 [35]

Where i represents the center of each cluster.

3.2.3 Find Minimum Distance

Then we find that to which center the distance is nearer, means which cluster distance is small.

3.2.4 Add Objects in a Cluster

Based on the “find minimum distance”, we added objects in the clusters. For example, if the distance of object to the center of cluster 3 is the smallest, then the object is added to that cluster.

3.2.5 Check Objects Move in a Different Cluster

After adding the objects in a cluster, check whether objects move in a cluster which is different from a previous cluster. If no object moves from one cluster to another cluster, then stop the algorithm, otherwise calculate mean and repeat the steps from 3.2.2 to 3.2.6.

3.2.6 Calculate Mean/Center

The mean of the clusters which are formed from the previous step, is calculated. The mean of each cluster is calculated as the sum of the values of the objects included in the cluster divided by the total number of objects in the cluster.

$$u_i = 1/n \sum_{i=0}^{n} x_i \ldots\ldots \quad \text{..... 3.3 [36]}$$

The parameters of the guassian membership function include the mean and variance, the mean has been calculated when making the clusters. Then after making the clusters, the variance of each cluster is calculated, the formula for calculating the variance is given below.

$$\sigma_i = 1/n \sum_{i-0}^{n} (x_i - u_i)^2 \ldots\ldots \quad \text{..... 3.4 [36]}$$

3.3 Adding & Extracting Rules

The next step after finding the parameters for the guassian membership function is to make the rules which will be passed to the fuzzy neural network. Some rules are hard coded, means that they are added by the developer based on the observation and their experience while other rules are extracted from the data. Some assumption that was made before adding and extracting the rules are given below.

RANK	NUMBER
Very Low	1
Low	2
Medium	3
High	4
Very High	5
Extra High	6

Table 3.2 Mapping of Rules from Text to Numeric Form

Some of the rules which were added by the developer based on their observation and experience, and also observed from [37]. Some of the hard-coded/observed rules are as follows:

- If ACAP is low, then effort is high
- If PCAP is high, then effort is low
- If AEXP is very low, then effort is high
- If TOOL is low, then effort is high
- If RELY is low, then effort is low
- If DATA is high, then effort is high
- If RELY is low, and DATA is low, then effort is low

While the steps which are adopted to extract the rules from the data are as follows:

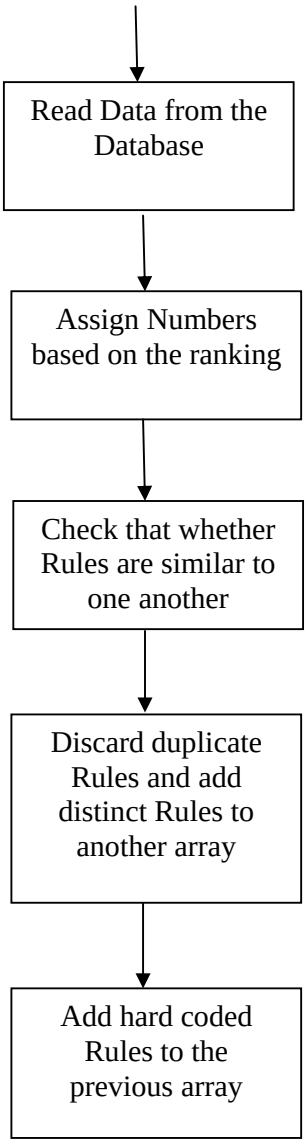

Fig 3.2 Basic Flow of Rule Extraction

3.3.1 Read Data from the Database

The first step in rule extraction is to read the data from the database which is in text form.

3.3.2 Assign numbers based on the Ranking

As the ranking are in text form, we convert it into numeric form for our easiness. It is converted into numeric form according to the assumption which is given in table 3.2. So we can say that the rules are in numeric form.

3.3.3 Check whether Rules are similar to one another

After assigning the numbers, the next step is to check that whether some rules are repeated or not. If there is a duplicate rule, then note down their index.

3.3.3 Discard Duplicate Rules

In order to discard the duplicate rules, copy the array of rules to another array, but not copy the duplicate rule.

3.3.4 Add hard coded Rules to the previous array

The last which we do is, add hard coded rules to the previous array.

3.4 Design of Fuzzy Neural Network (FNN)

3.4.1 Topology of Fuzzy Neural Network

The topology of FNN contains five layers. The first layer is called input layer, the second is conditional element layer which performs fuzzification, the third one is rule layer, the fourth is action element layer which performs defuzzification, and the last one is the output layer. The topology is shown in figure below.

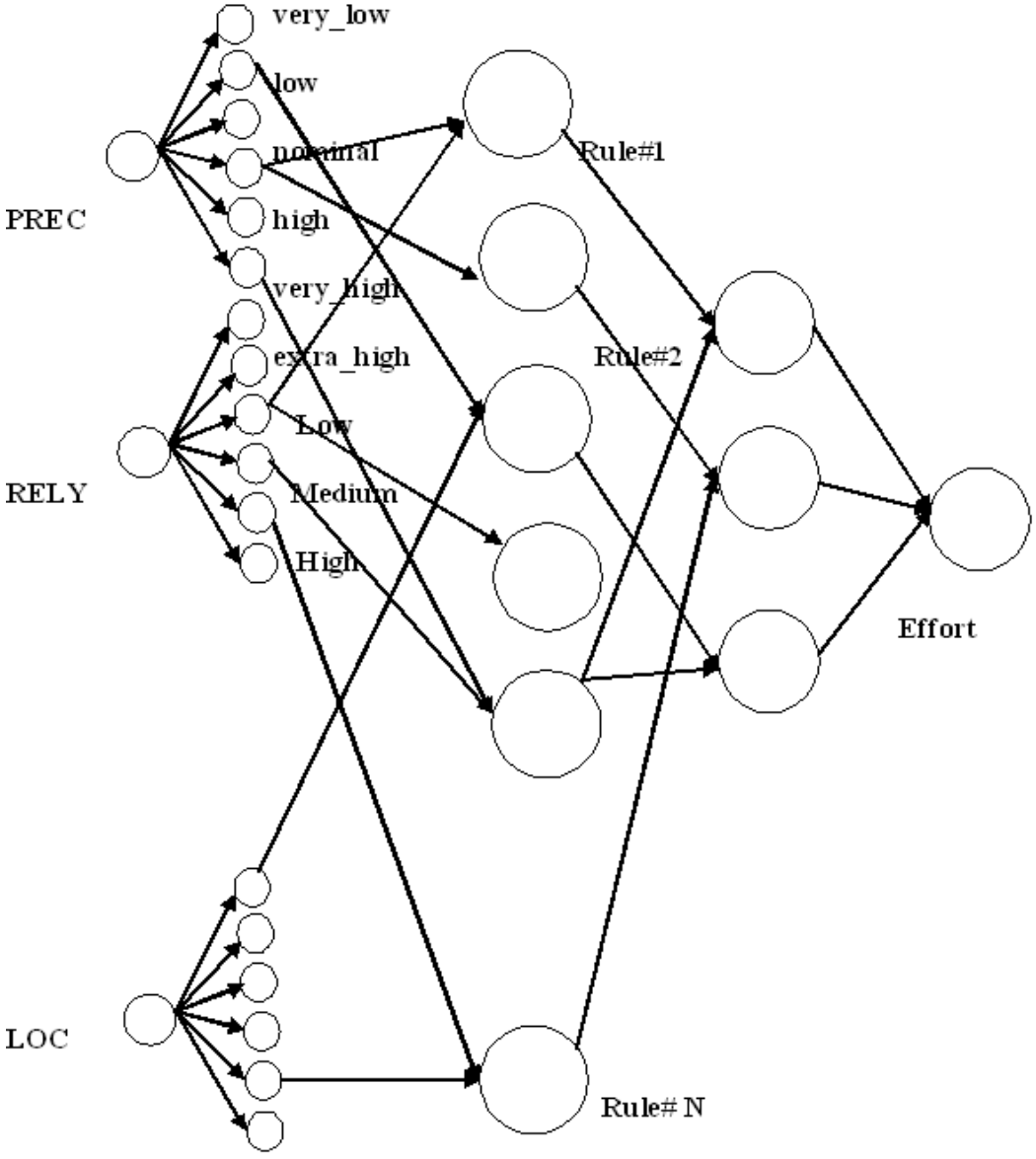

Fig 3.3 Topology of Fuzzy Neural Network

3.4.2 Functionality of Layers of Fuzzy Neural Network

As we know there are three or five layers in the fuzzy neural network. Each Layer depends on the other layer. We will discuss the functionality with respect to five layers turn by turn.

3.4.2.1 Input Layer

The input layer of fuzzy neural network simply passes the value to the conditional element layer.

3.4.2.2 Conditional Element Layer

The conditional element layer performs fuzzification. Fuzzification is defined as the conversion of fuzzy variable into membership value or grade for fuzzy sets [38]. The conditional element layer measures the membership grade of each ranking. In the data, we have categorical and continuous variables. For categorical attribute, we used singleton membership function and for continuous variable, we used guassian membership function whose parameters finding method is mentioned in section 3.2.

Singleton membership ship function is defined to be 1 on one rank while 0 on all other ranks. We can define fuzzy singleton with a membership function that is 1 one point while zero on everywhere else [39].

3.4.2.3 Rule Layer

The rule layer measures the strength of each rule to fire. The node of each rule layer multiplies the input which is coming to this node with their respective weight and adds all the inputs which are coming to this node, which is show in equation 3.5 and then applied the activation function. The activation function which is used over here is sigmoid activation function which is shown in equation 3.6.

$$v_k = \sum_{j=1}^{p} w_{kj}.x_j \text{ ……..... 3.5 [15]}$$

$$Output = 1/(1+e^{-x}) \text{ …… ..... 3.6 [15]}$$

3.4.2.4 Action Element Layer

Action element layer performs defuzzification. Defuzzification is defined as a process of transforming a fuzzy output into crisp input [39]. The nodes of action element layer depend on ranking which are used for the output. In this project, there are three nodes of action element layer because we have chosen three ranking for the output i.e. low, medium, high. The node of each action layer multiplies the input which is coming to this node with their respective weight and adds all the inputs which are coming to this node, which is show in equation 3.5 and then applied the activation function. The activation function which is used over here is sigmoid activation function which is shown in equation 3.6.

3.4.2.5 Output Layer

The node of output layer multiplies the input which is coming to this node with their respective weight and adds all the inputs which are coming to this node, which is show in equation 3.5 and then applied the activation function. The activation function which is used is shown in equation 3.7.

$$Output = v_k / \sum_{j=1}^{p} out(action) \text{ …….....} 3.7[15]$$

3.4.3 Training Algorithm for Fuzzy Neural Network

After deciding the topology for the respective data and understanding the functionality of layers, the next step is to train the FNN. For training FNN, variation of evolutionary algorithm is used to evolve the weights of FNN. An evolutionary algorithm is a stochastic search method and is very similar to biological evolution [40]. In FNN, there are five layers, and each neuron in a layer is connected to neurons of the conditional layer, and neurons in the conditional layer are connected to the neurons in the rule layer and neurons in the rule layer is connected to the neurons in the action layer and neurons in the action layer is connected output neuron in the output layer. The network is trained for several iterations, with each iteration; the error is computed as the difference between the network output and observable output. The combination of randomly assign weight, which gives the low error, replaces the weights with other combination of weights, which gives high error. This process is called training.

Training is the process to adjust the connection weights to enable the network to produce the expected output for all inputs.

The main steps which are used to train the FNN are as follows:

3.4.3.1 Initialize the Population

In algorithm, the first step is to create the population. So initially population of size 10 is created means 10 chromosomes are created. But we have experimented with different number of chromosomes. The size of chromosome is calculated by the following form.

Total_No_of_Weights = (size of input * 6) + no of rule layer nodes + no of action layer nodes……..... 3.8

Where total_No_of_Weights contains the total number of weights that are used to train the fuzzy neural network, and the weights are randomly assigned.

3.4.3.2 Feeding Input to the Network

After the initialization of population, the next step is to feed the inputs into the network. The Fuzzy network calculates the output by following the steps which is described in functionality of layers, and returns the results.

3.4.3.3 Calculation of Error

The steps which are mentioned above are repeated for each chromosome and error rate of each chromosome will be measured. Error is affirmed as the difference between the actual network output and the observable outcome shown in equation 3.9. Observable outcome is the effort per person months.

Error= actual network output – desired/observable output…….....3.9

After the calculation of error, the steps of algorithm are applied.

3.4.3.4 Selection Process

After calculation of error, selection step is applied. In this step the best individuals are selected. There are different methods of selecting it Its up to the developer or user to select any method for selection. Here we use Rank Selection. In this selection method, the individuals are ranked according to their fitness, and individuals having the high fitness value/lowest rank are selected and how many individuals are selected is defined by the user. We have experimented by selecting different number of individuals. For example if we want

to choose four best individuals, it means four individuals having the highest fitness value which are at the low rank are selected. We take the fitness criteria on the basis of error. The chromosomes whose error is low have high fitness function, so those chromosomes whose error is low are selected. It is summarized in equation 3.10.

Low Error = High Fitness
High Error = Low Fitness } ……..... 3.10

3.4.3.5 Crossover Operation

The crossover operation is defined as the combination of connection weights of two individuals which have been selected from the selection process. The cross over rate is defined by the user. The cross over rate may be 50% and 50%. It means that 50 percent of the connection weight of one individual is selected and 50 percent of the connection weight of second best individual is selected.

3.4.3.6 Mutation & Reinsertion Process

After crossover of chromosomes, the values of chromosomes are mutated. The number of chromosomes which are mutated depends on the process of reinsertion. In reinsertion process, the method which is used is Elitist Reinsertion. In this method, the number of offspring's created is less than the number of parents, and offspring replaces the worst parents. The worst parents are those whose fitness is low means whose error is high. In our case, the best chromosomes are kept the same which have high fitness value, and other chromosomes are also replaced by chromosomes which are cross over but with small variations with self adaptive parameter.

The variations by using self adaptive parameter are summarized in equation 3.11.

$$\sigma_j'[i] = \sigma_j'[i] * \exp(\tau N_j(0,1))$$

$$weight_i[j] = weight_i[j] \pm \sigma_j^{'} \cdot N_j(0,1) \quad \text{……..... 3.11 [41]}$$

$$\tau = 1/\sqrt{2 * \sqrt{size}}$$

where j=1, 2……..size means the total number of weights and bias in the neural network.

The above overall steps are repeated for each iteration in order to train FNN.

3.4.4 Testing for Fuzzy Neural Network

The fuzzy neural network has two phases: Training and Testing. In training phase, the network is trained by providing the complete information about the characteristics of the data and the observable outcomes to perform a particular task. FNN develops the model that learns the relationship between the input data and desired outcome in the training phase. Whereas in testing phase, the testing data is provided as input. The performance of this phase depends on the training phase, means it depends on the samples that are provided during training phase and also on the number of times, the network is trained and how much accurate the network is trained. It is impossible that the network output is 100% precise for any input.

For testing the FNN, cross validation method is used, and in cross validation, we use k-fold method, in which data is divided into different subsets, one subset is called testing data which is not used in training, the other subsets are called training data. We have done experimentation by taking each subset as a testing data. In testing phase, weights of the trained FNN and test data is provided to the network and output is calculated by performing the steps given in section 3.4.3.2 to 3.4.3.3.

3.5 Summary

We can teach a network to perform a specific task by using the following procedure. We present the network with the training examples which contain the sample of activities for the input units and also the desired output for the output unit, then present the network with different randomly assign weight i.e. with chromosomes. In others words, we can say that we initiate the population. Then determine how much difference exists between the network output and the desired output. Error is calculated by using RMSE. The next step is to determine the individuals /chromosomes with high fitness i.e. whose error is low. The best individuals whose error is low are kept as it is, and other chromosomes are also replaced by the weights of these chromosomes but with the little variations by using crossover operation and self adaptive parameter.

CHAPTER 4

IMPLEMENTATION DETAILS & USER INTERFACE

Our task is to develop a model for the effort estimation using fuzzy neural network. In this chapter, we discuss implementation details regarding the development of model, and also the user interface. Implementation details are discussed in section 4.1. Section 4.2 discusses the user interface and summary of this chapter is given in section 4.3.

4.1 Implementation Details

For the implementation, latest technologies are used; the language which is used for the implementation of this project is C# and the tool which is used for it is Microsoft Visual Studio.Net version 2005 and database engine is Microsoft Access.

Microsoft Visual Studio contains the set of development tools for building ASP.Net application, XML web services, desktop applications and mobile applications. It can be used to develop console and graphical user interface applications along with windows form applications, web applications etc [42]. It contains Visual Basic, Visual C#, and Visual J #. They all use the same integrated development environment [43].

The code which we implement should have:

- Proper User Interface in order to interact with the user for completion of task.
- Error Free
- Easy to Understand

4.2 User Interface

User interface of this application is divided into three parts.

i). Training

ii). Testing

iii). Effort Estimation

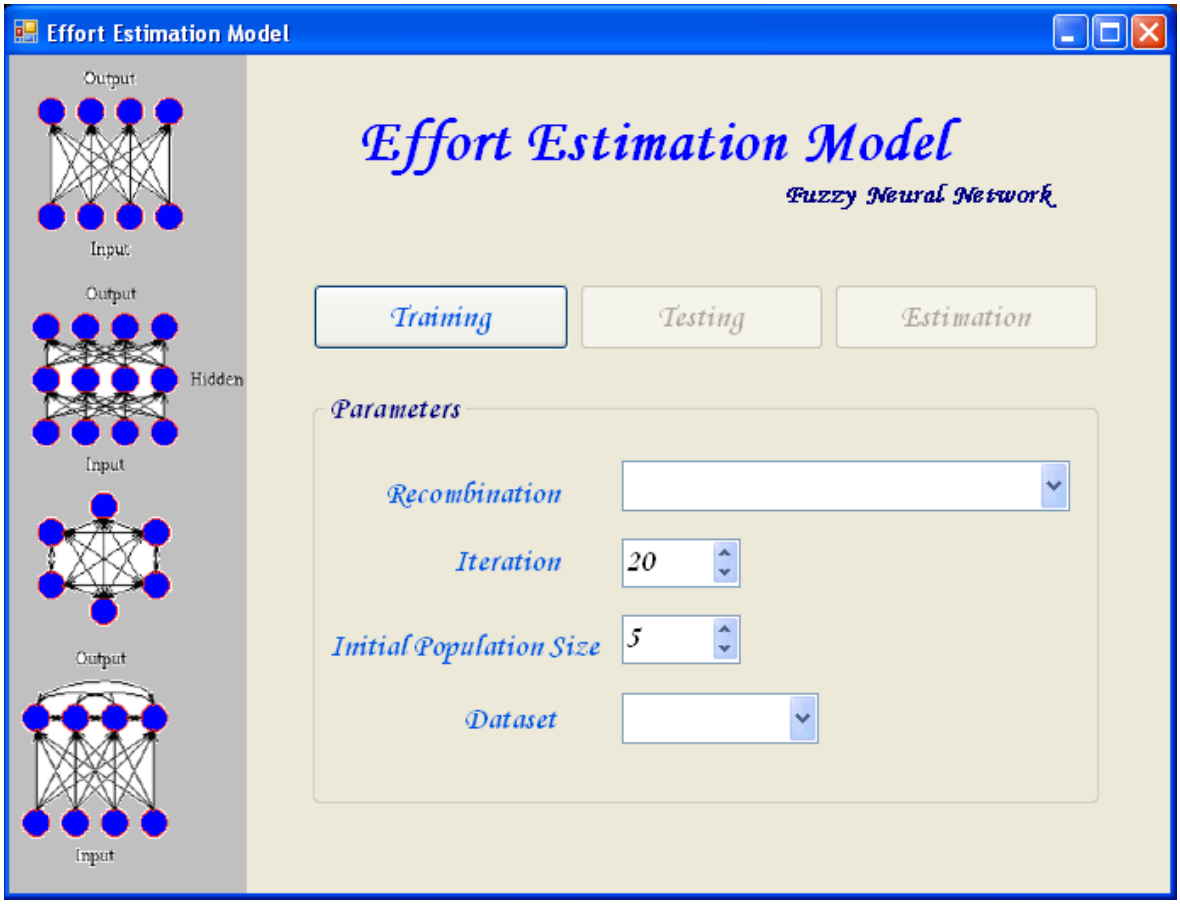

Fig 4.1 User Interface

4.2.1 Training

In order to train the fuzzy neural net, the parameters are selected. The first parameter is the recombination parameter in which the user decides that whether he/she wants to apply

mutation process on crossover offspring's or on without crossover chromosomes. Moreover, the number of iterations and the size of chromosomes which are taken initially are also selected by the user. As there are three datasets which can be used for training, the user also have to select the dataset on which training is to be done. Then click on the training button. After training, the message box is displayed which tells that Training is successfully completed.

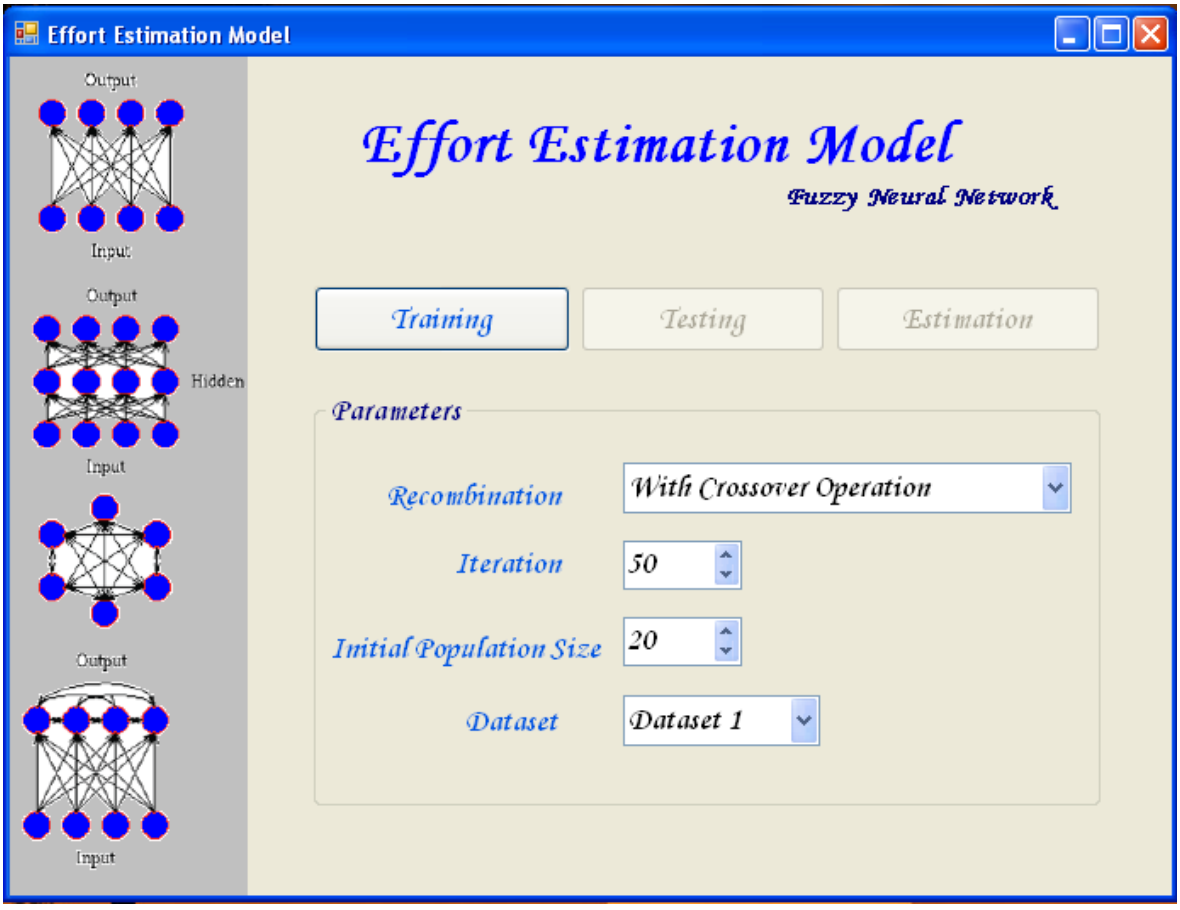

Fig 4.2 User Interface with Parameters Selected

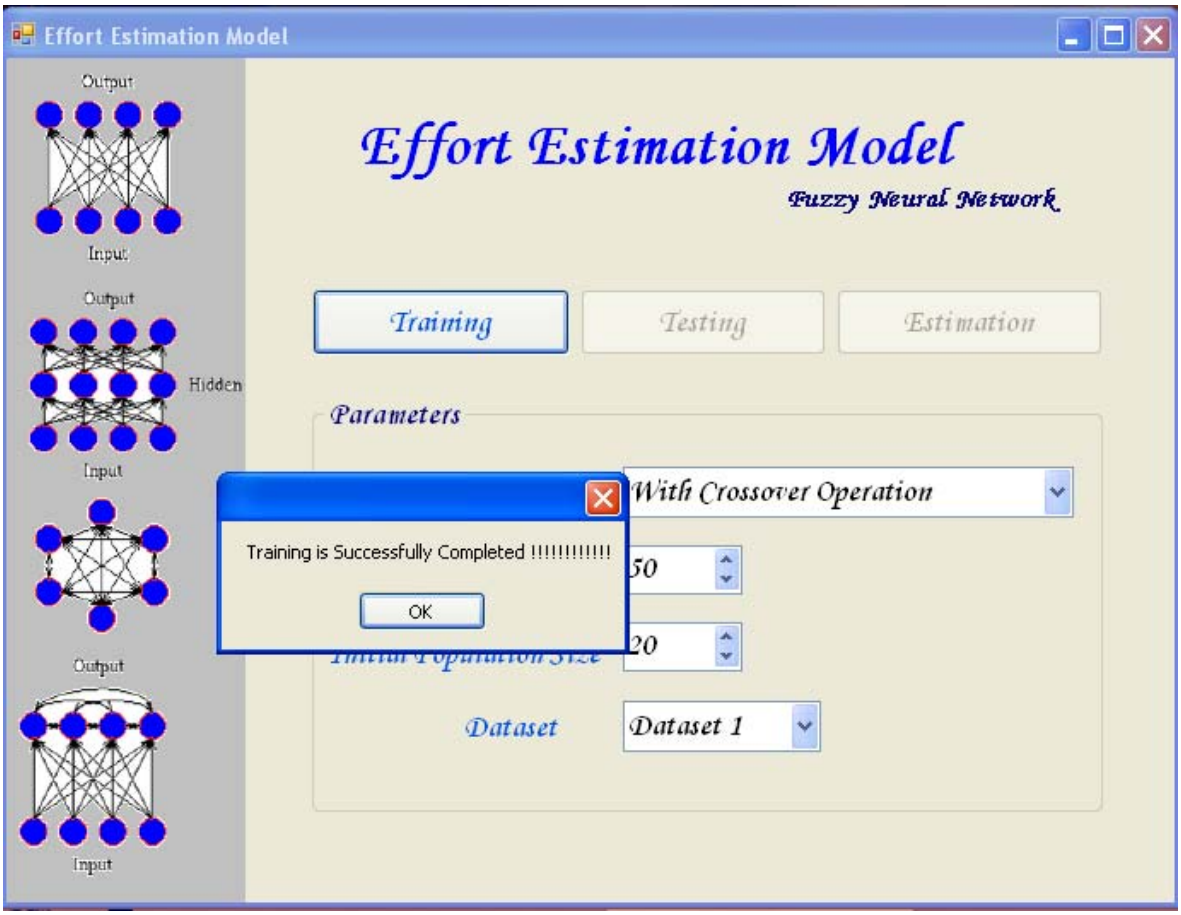

Fig 4.3 Training of FNN

4.2.2 Testing

In order to test the neural net, click on the testing button. . When testing is finished, the message box is displayed which tells that Testing is successfully completed.

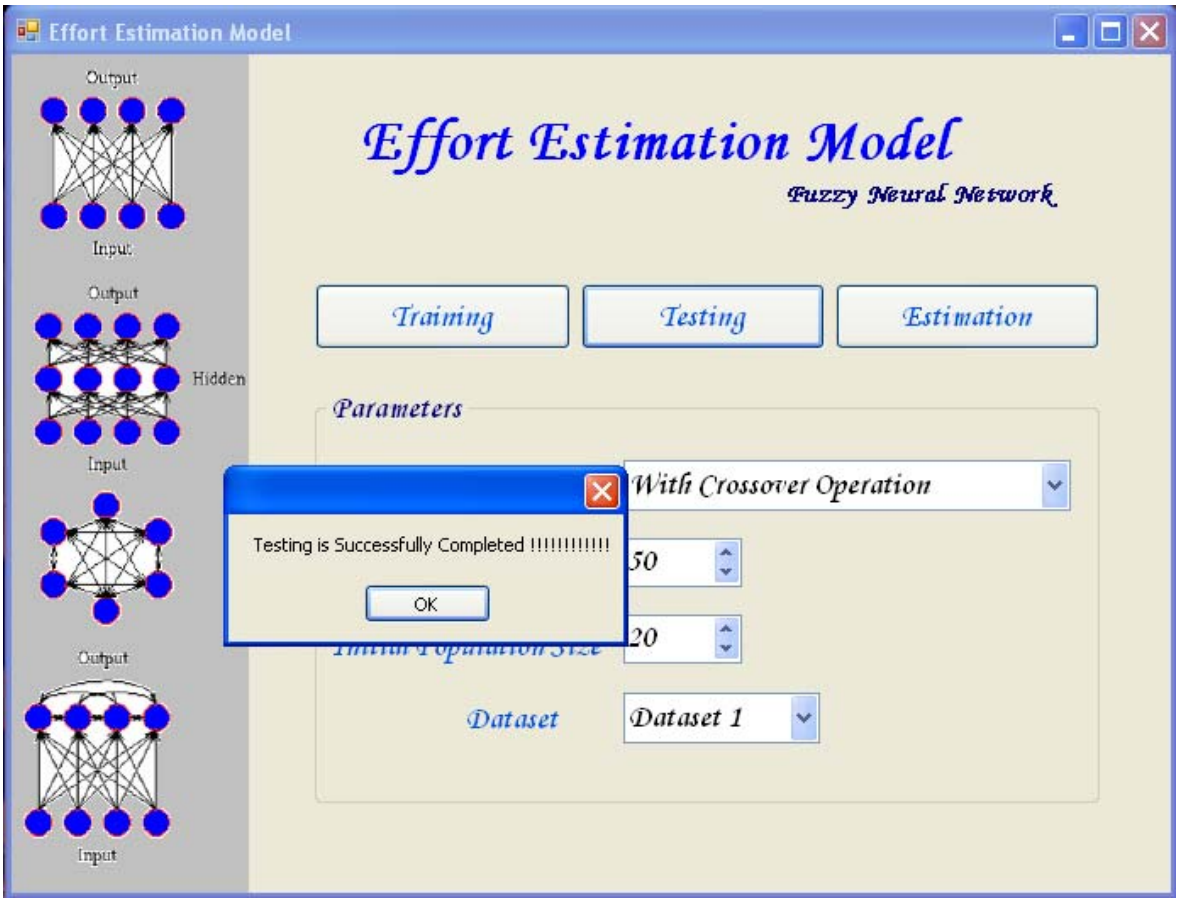

Fig 4.4 Testing of FNN

4.2.3 Effort Estimation

When the user clicks on the “Estimation” button, another form will open, in which user fills the specified fields and click on the “Estimate” button, the resultant value is displayed in the text box.

Fig 4.5 Effort Estimation

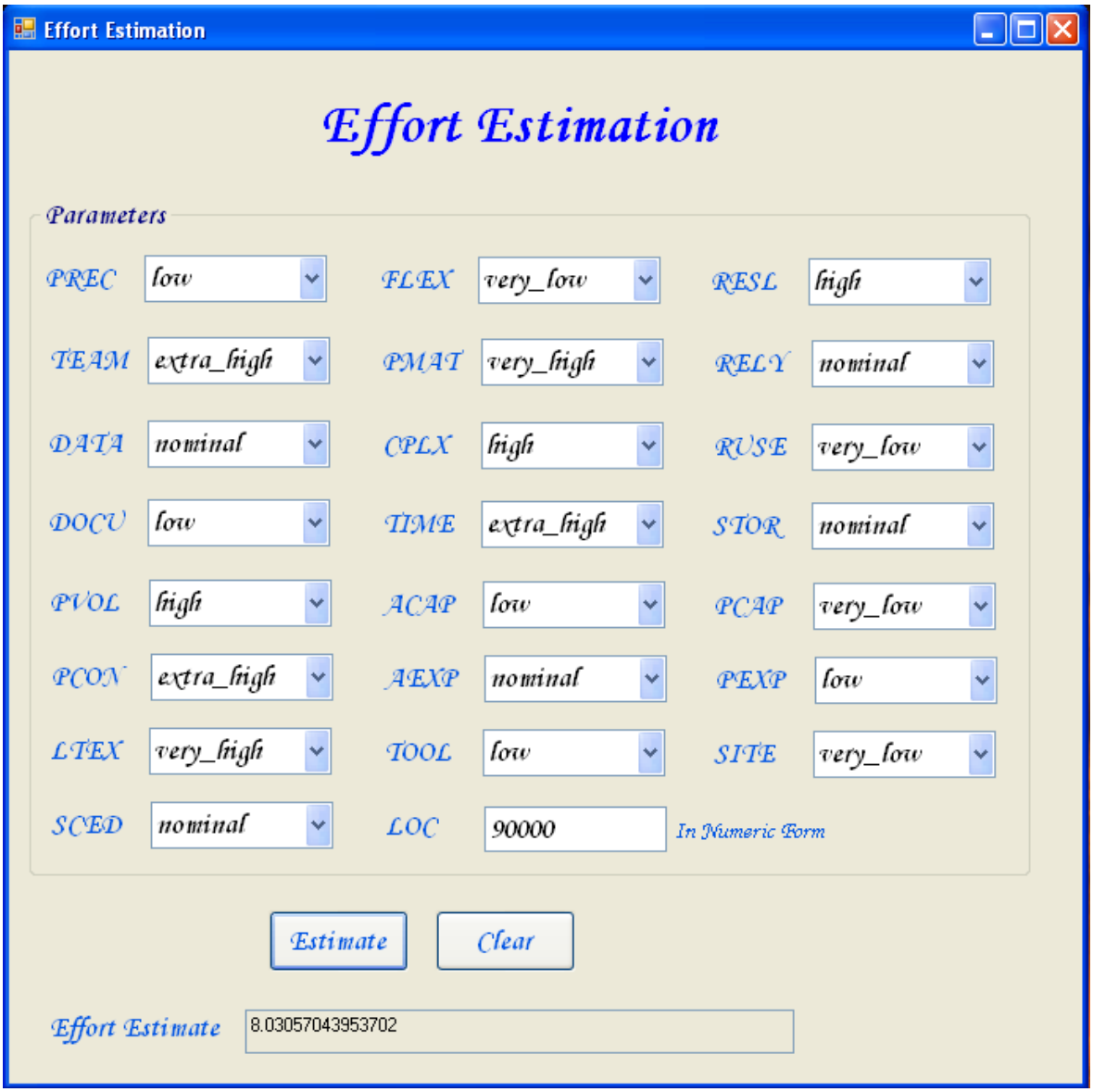

Fig 4.6 Example of Effort Estimation

4.3 Summary

In this chapter, we discussed the implementation details, and also the user interface, means how it looks and how it works.

CHAPTER 5

RESULTS & TESTING

In this chapter, we will detail the results of system testing on different datasets. Each dataset was divided into two parts: training dataset and testing dataset. For various experiments, the system was trained on the training dataset and then its results were analyzed on the test dataset.. The results are represented in the form of tables and graphs to increase readability for the user.

This chapter is organized as follows. Section 5.1 describes the datasets. The factors which are used for the analysis purpose are described in section 5.2. The training results of the datasets and their analysis are discussed in section 5.3. The testing results of the datasets and their analysis are discussed in section 5.4. The summary of this chapter is given in section 5.5.

5.1 Input Dataset Characteristics

Three different datasets was used to evaluate the performance of the Fuzzy Neural Network (FNN). The first dataset contain real data that is taken from [33] and from ERP team leader of College of E&ME. The second dataset was created by a professional developer based on his experience with various projects. The third dataset contain randomly generated data. For the analysis purpose, each dataset is divided into three equal parts, two parts are used for the training purpose and one is used for the testing purpose, and every part is used for the testing purpose in each experiment.

5.2 Factors of Comparisons

We evaluate the performance of FNN in terms of its Root Mean Square Error (RMSE). While taking the experiments, the factors which are used for the comparisons and analysis purpose are:

- RMSE effect with and without crossover operations.
- RMSE effect by changing the initial size of the population i.e generation of chromosomes at the start of algorithm.

5.3 Training Results & Analysis

Different experiments were conducted in order to observe how well the FNN is trained, or in other words, we can say measure the performance of the FNN. The experiments are carried out in various ways. For the purpose of performance measure, total of 216 experiments were performed for the three datasets including the three combinations of each dataset, out of which 108 are with crossover operation and 108 are without crossover operations. Crossover operation means that connection weights of two best chromosomes are combined. The population sizes which are used for the analysis purpose are 10, 20, 50 and 100. The results of the experiments for different datasets are discussed below.

5.3.1 Results

We will discuss the results of each dataset with the different combination of data turn by turn.

5.3.1.1 Dataset 1

We will discuss the results with different combinations of data one by one.

5.3.1.1.1 Data Combination 1 (DC11)

The results of the dataset with data part or data combination 1 (DC11) with and without crossover operation and by changing the population size, and by selecting the different number of best chromosomes are shown in Table 5.1 and 5.2.

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		1.63820667963474	1.63821115601164	1.63819126295196
2	20	**8**	**10**	**12**
		1.60307678886961	1.63819820670947	1.63844337162949
3	50	**20**	**25**	**30**
		1.61788060526262	1.63819177597436	1.63818952520204
4	100	**40**	**50**	**60**
		1.62247988886841	1.63819371843808	1.5314973753452

Table 5.1 RMSE of Training Results with Crossover (DC11)

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		1.63698678422563	1.63827307071342	1.63829625954059
2	20	**8**	**10**	**12**
		1.6381893467108	1.638189897977	1.63820892889931
3	50	**20**	**25**	**30**
		1.58064683566918	1.63818924680562	1.63818960239284
4	100	**40**	**50**	**60**
		1.58141026519184	1.62310124962382	1.63818932140575

Table 5.2 RMSE of Training Results without Crossover (DC11)

From the table 5.1, we observe that lowest RMSE is 1.5314973753452 with population size 100 and number of best chosen chromosome is 60, and their training pattern is shown in the Fig 5.1. And from table 5.2, the lowest RMSE is 1.58064683566918 with population size 50 and number of best chromosome is 20 and their training graph is shown in Fig 5.2. We observe from the graphs, as the number of iteration increases, the root mean square error decreases, and after some iteration, it becomes stable, means it does not learn more patterns after becoming stable.

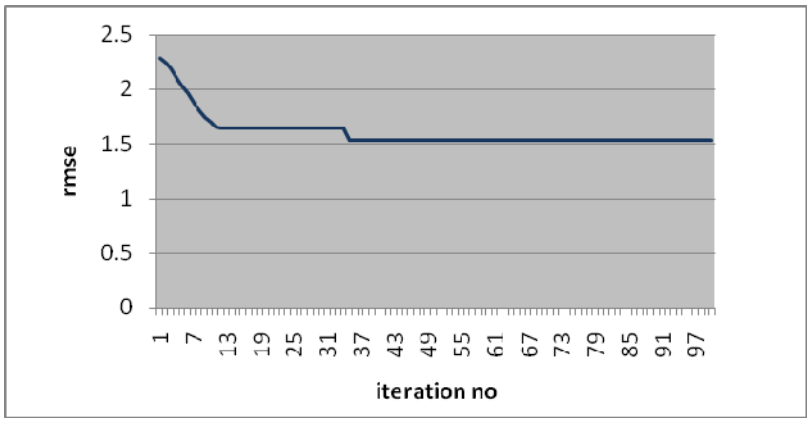

Fig 5.1 Graph with 100 Population Size and 60 Best Chosen Size of Chromosomes (DC11 with Crossover)

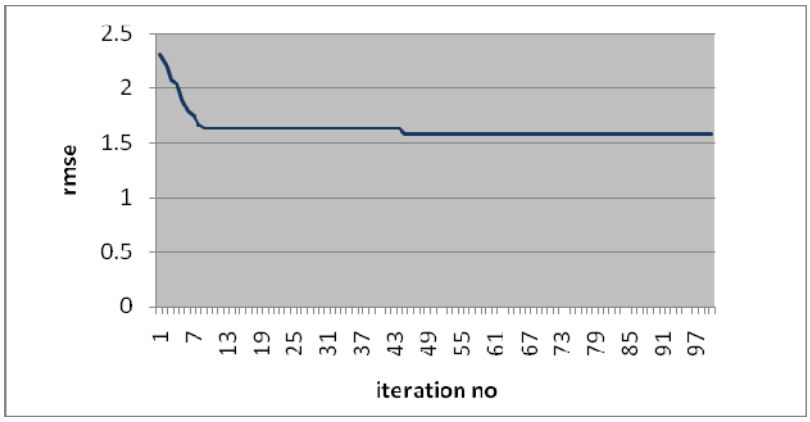

Fig 5.2 Graph with 50 Population Size and 20 Best Chosen Size of Chromosomes (DC11 without Crossover)

The summary of the RMSE's of the results given in table 5.1 and 5.2 with and without crossover are shown in Fig 5.3.

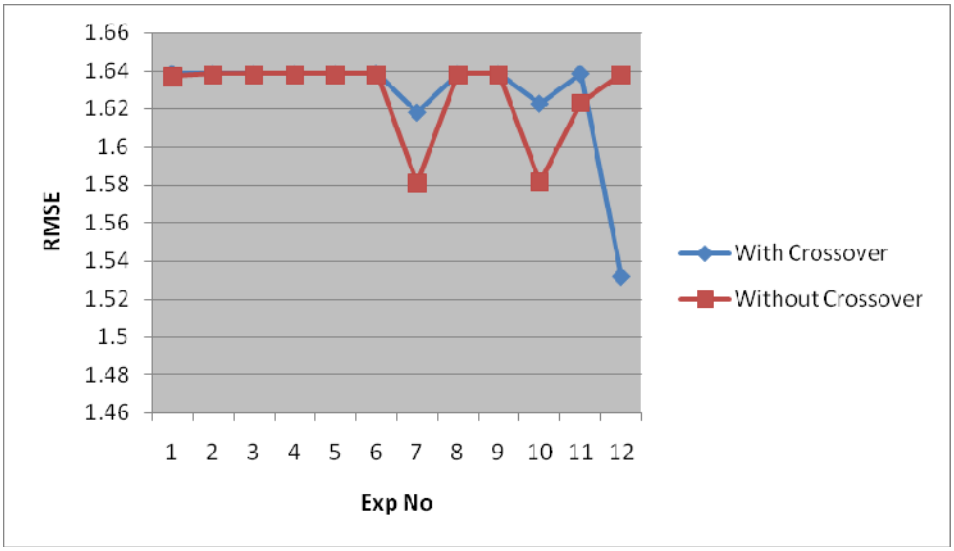

Fig 5.3 Summary of the Training Results of DC11 with & without Crossover Operation

The averages of the RMSE's of the results calculated from table 5.1 and 5.2 are given in Table 5.3.

Average RMSE of Training Results with Crossover	Average RMSE of Training Results without Crossover
1.626322902	1.627322567

Table 5.3 Average RMSE of Training Results with & without Crossover (DC11)

5.3.1.1.2 Data Combination 2 (DC12)

The results of the dataset with data part or data combination 2 (DC12) by changing the population size, and by selecting the different number of best chromosomes with and without crossover operation are shown in Table 5.4 and 5.5.

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		1.99399330192257	1.99416156726356	1.9941069267815
2	20	**8**	**10**	**12**
		1.99320061238867	1.99383184617617	1.99415418546137
3	50	**20**	**25**	**30**
		1.99433545139472	1.99394512166747	1.96405630896112
4	100	**40**	**50**	**60**
		1.96035569971199	1.9261357343588	1.84199952084285

Table 5.4 RMSE of Training Results with Crossover (DC12)

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		1.99404166149417	1.99421863259106	1.99402740910471
2	20	**8**	**10**	**12**
		1.99377517481042	1.99393735896395	1.99401180492594
3	50	**20**	**25**	**30**
		1.98003945625951	1.95941363637786	1.99401232615357
4	100	**40**	**50**	**60**
		1.92577461885759	1.99367552784335	1.97419501805537

Table 5.5 RMSE of Training Results without Crossover (DC12)

From the table 5.4, we observe that lowest RMSE is 1.84199952084285 with population size 100 and best chosen chromosome is 60, and their training pattern is shown in the Fig 5.4. And from table 5.5, the lowest RMSE is 1.92577461885759 with population size 100 and number of best chromosome is 40 and their training graph is shown in Fig 5.5.

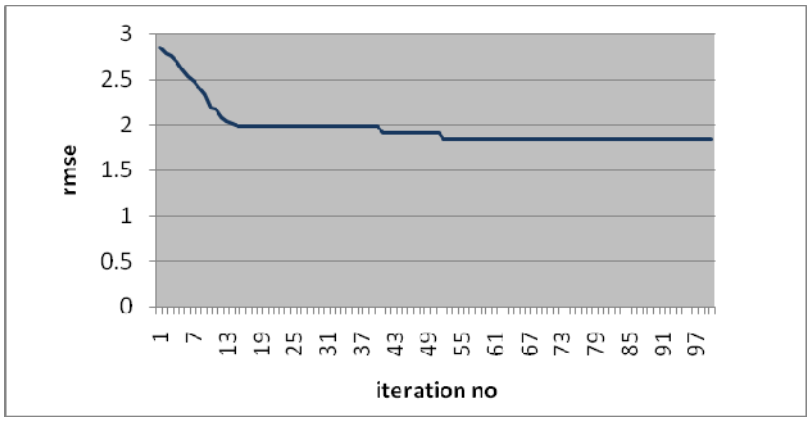

Fig 5.4 Graph with 100 Population Size and 60 Best Chosen Size of Chromosomes (DC12 with Crossover)

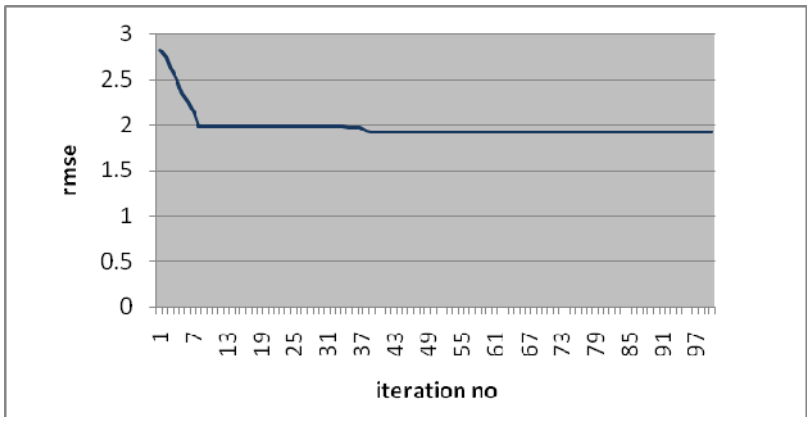

Fig 5.5 Graph with 100 Population Size and 40 Best Chosen Size of Chromosomes (DC12 without Crossover)

The summary of the RMSE's of the results given in table 5.4 and 5.5 with and without crossover are shown in Fig 5.6.

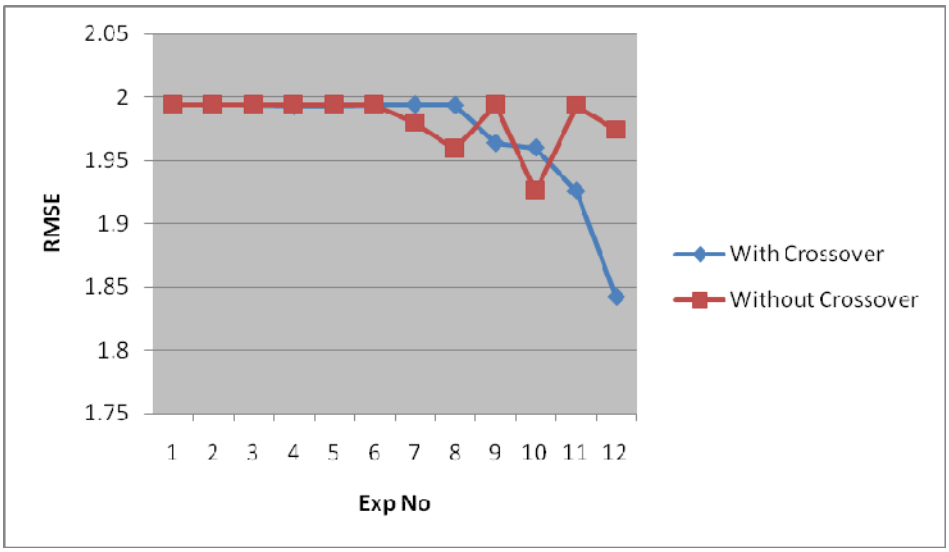

Fig 5.6 Summary of the Training Results of DC12 with & without Crossover Operation

The averages of the rmse's of the results calculated from table 5.4 and 5.5 are given in Table 5.6.

Average RMSE of Training Results with Crossover	Average RMSE of Training Results without Crossover
1.970356356	1.982593552

Table 5.6 Average RMSE of Training Results with & without Crossover (DC12)

5.3.1.1.3 Data Combination 3 (DC13)

The results of the dataset with data part or data combination 3 (DC13) by changing the population size, and by selecting the different number of best chromosomes with and without crossover operation are shown in Table 5.7 and 5.8.

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		1.66440318234985	1.66258289577062	1.66299836201039
2	20	**8**	**10**	**12**
		1.66237543532013	1.66229812330325	1.66267756484166
3	50	**20**	**25**	**30**
		1.66130138748718	1.65747375101583	1.64021174791185
4	100	**40**	**50**	**60**
		1.62974428634952	1.6619340641745	1.6329793984847

Table 5.7 RMSE of Training Results with Crossover (DC13)

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		1.66273619584422	1.66215288845445	1.65997856921291
2	20	**8**	**10**	**12**
		1.63749217758516	1.6622989208378	1.64448822959757
3	50	**20**	**25**	**30**
		1.63489799862911	1.64760910595616	1.65274769443576
4	100	**40**	**50**	**60**
		1.66258454395329	1.63931136926958	1.63209098819314

Table 5.8 RMSE of Training Results without Crossover (DC13)

From the table 5.7, we observe that lowest RMSE is 1.62974428634952 with population size 100 and best chosen chromosome is 40, and their training pattern is shown in the Fig 5.7.And from table 5.8, the lowest RMSE is 1.63209098819314 with population size 100 and number of best chromosome is 60 and their training graph is shown in Fig 5.8.

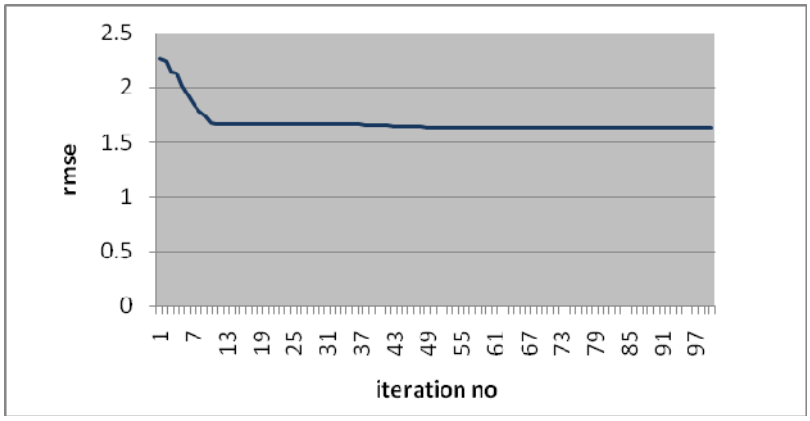

Fig 5.7 Graph with 100 Population Size and 40 Best Chosen Size of Chromosomes (DC13 with Crossover)

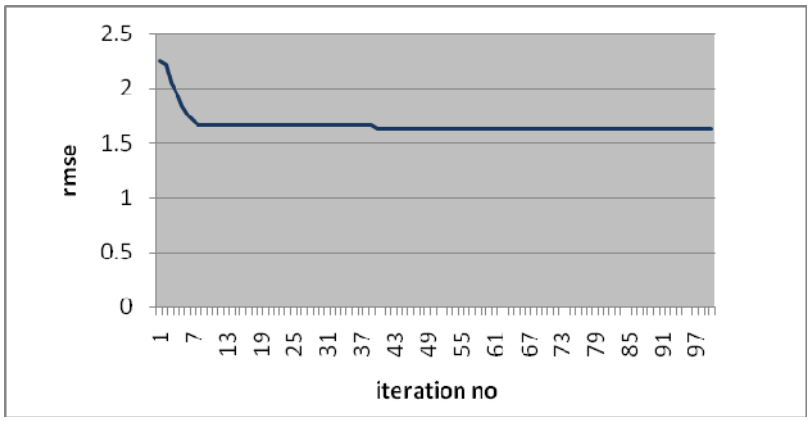

Fig 5.8 Graph with 100 Population Size and 60 Best Chosen Size of Chromosomes (DC13 without Crossover)

The summary of the RMSE's of the results given in table 5.7 and 5.8 with and without crossover are shown in Fig 5.9.

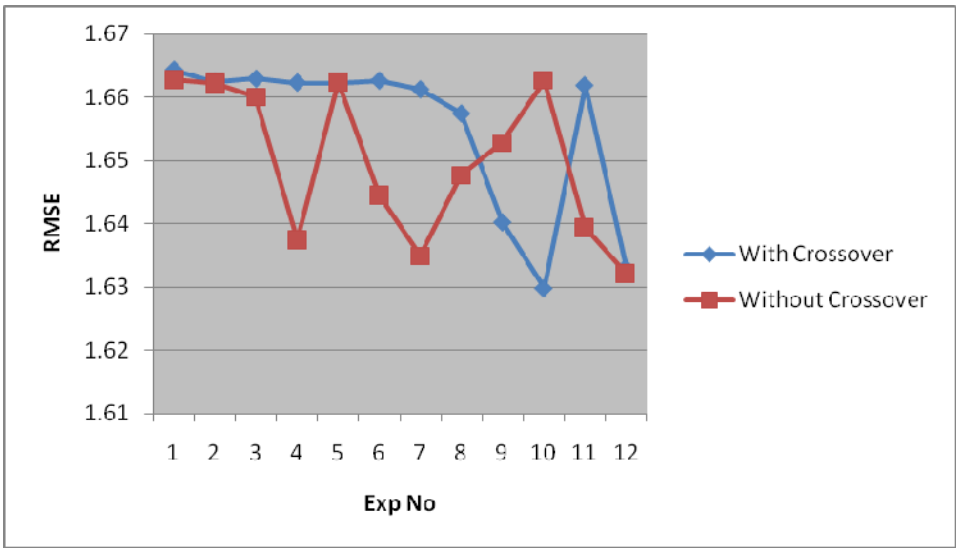

Fig 5.9 Summary of the Training Results of DC13 with & without Crossover Operation

The averages of the rmse's of the results calculated from table 5.7 and 5.8 are given in Table 5.9.

Average RMSE of Training Results with Crossover	Average RMSE of Training Results without Crossover
1.655081683	1.649865723

Table 5.9 Average RMSE of Training Results with & without Crossover (DC13)

As we observe from the above tables and graphs that FNN is not well trained for the data combination 2 as compared to the other combinations because it contains highest RMSE. The main drawback with this dataset is the small number of training examples, and the network

needs a large number of examples to better train a network, but on the other hand the main advantage of this dataset is that it contains real data, which is good for training.

5.3.1.2 Dataset 2

Now we will discuss the results of dataset 2 with different combinations of data turn by turn.

5.3.1.2.1 Data Combination 1 (DC21)

The results of the dataset with data part or data combination 1 (DC21) with and without crossover operation and by changing the population size, and by selecting the different number of best chromosomes are shown in Table 5.10 and 5.11.

S. No	**Population Size**	**Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)**		
1	10	**4**	**5**	**6**
		0.774147841943023	0.774087828804581	0.774321967242386
2	20	**8**	**10**	**12**
		0.77417087618872	0.774077129494083	0.77407686235788
3	50	**20**	**25**	**30**
		0.774076772862918	0.774088395035455	0.761375185886316
4	100	**40**	**50**	**60**
		0.739998872768895	0.7377853709561	0.751061950493052

Table 5.10 RMSE of Training Results with Crossover (DC21)

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		0.774085587508678	0.773391863030158	0.774078357483707
2	20	**8**	**10**	**12**
		0.765870447963984	0.774076607421646	0.774076644156088
3	50	**20**	**25**	**30**
		0.752880647900112	0.774077057228241	0.76895485819402
4	100	**40**	**50**	**60**
		0.72323205683826	0.754993139338827	0.765907444900027

Table 5.11 RMSE of Training Results without Crossover (DC21)

From the table 5.10, we observe that lowest RMSE is 0.7377853709561 with population size 100 and best chosen chromosome is 50, and their training pattern is shown in the Fig 5.10.And from table 5.11, the lowest RMSE is 0.72323205683826 with population size 100 and number of best chromosome is 40 and their training graph is shown in Fig 5.11.

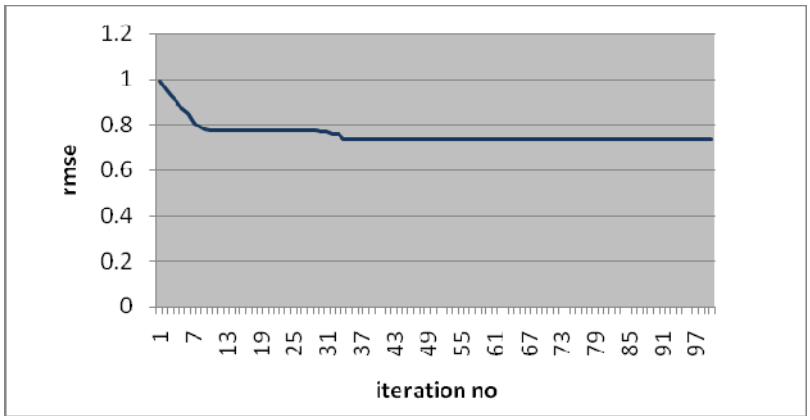

Fig 5.10 Graph with 100 Population Size and 50 Best Chosen Size of Chromosomes (DC21 with Crossover)

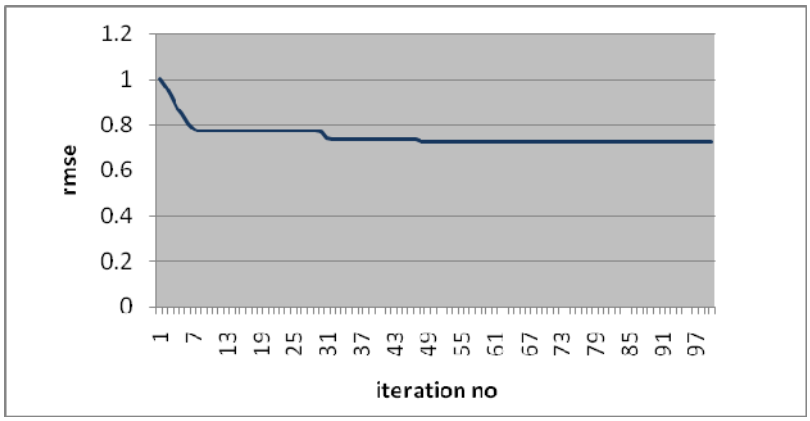

Fig 5.11 Graph with 100 Population Size and 40 Best Chosen Size of Chromosomes (DC21 without Crossover)

The summary of the RMSE's of the results given in table 5.10 and 5.11 with and without crossover are shown in Fig 5.12.

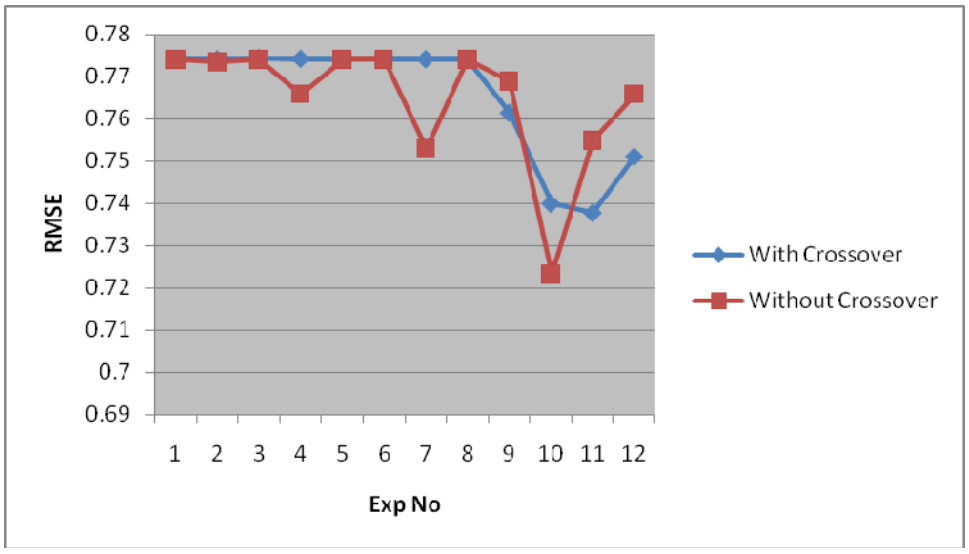

Fig 5.12 Summary of the Training Results of DC21 with & without Crossover Operation

The averages of the rmse's of the results calculated from table 5.10 and 5.11 are given in Table 5.12.

Average RMSE of Training Results with Crossover	Average RMSE of Training Results without Crossover
0.765272412	0.764635393

Table 5.12 Average RMSE of Training Results with & without Crossover (DC21)

5.3.1.2.2 Data Combination 2 (DC22)

Then results of the dataset with data part or data combination 2 (DC22) by changing the population size, and by selecting the different number of best chromosomes with and without crossover operation are shown in Table 5.13 and 5.14.

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		0.785220409133273	0.785183936061847	0.785183900985981
2	20	**8**	**10**	**12**
		0.785395703857917	0.785085388623097	0.78520099582178
3	50	**20**	**25**	**30**
		0.747352559679869	0.737673513249608	0.748861346785677
4	100	**40**	**50**	**60**
		0.785183368075852	0.634502449903263	0.757567461920363

Table 5.13 RMSE of Training Results with Crossover (DC22)

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		0.78518811646665	0.78522879206761	0.788783822690797
2	20	**8**	**10**	**12**
		0.785183363280739	0.785192005151932	0.785183572550179
3	50	**20**	**25**	**30**
		0.709138059073213	0.782552663752828	0.760856989596845
4	100	**40**	**50**	**60**
		0.772512979595968	0.756998074179578	0.773561205495457

Table 5.14 RMSE of Training Results without Crossover (DC22)

From the table 5.13, we observe that lowest RMSE is 0.634502449903263 with population size 100 and best chosen chromosome is 50, and their training pattern is shown in the Fig 5.13.And from table 5.14, the lowest RMSE is 0.709138059073213 with population size 50 and number of best chromosome is 20 and their training graph is shown in Fig 5.14.

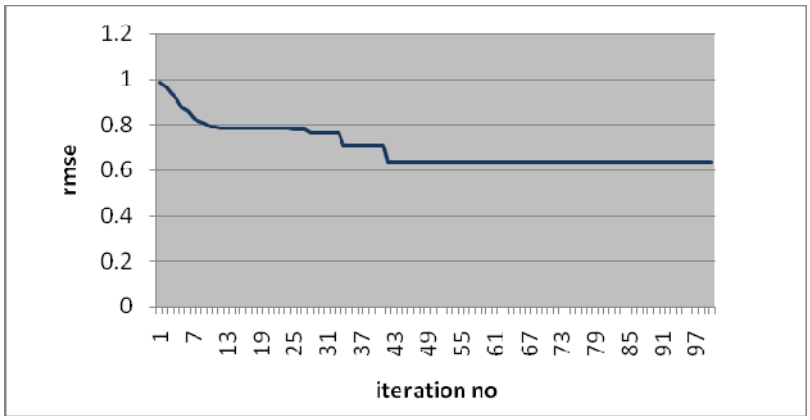

Fig 5.13 Graph with 100 Population Size and 50 Best Chosen Size of Chromosomes (DC22 with Crossover)

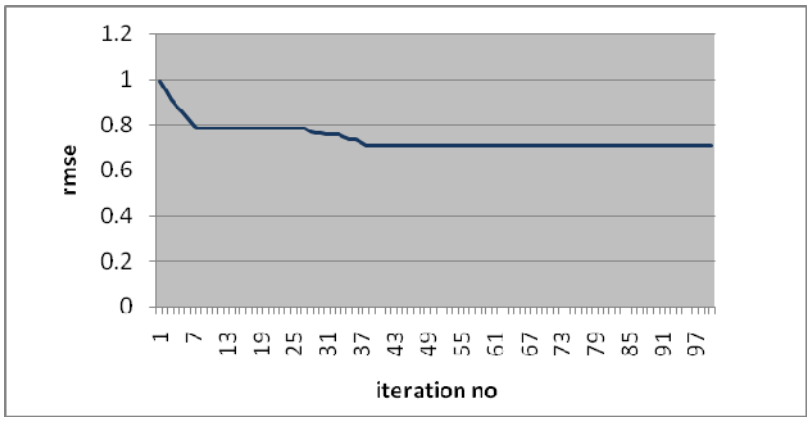

Fig 5.14 Graph with 50 Population Size and 20 Best Chosen Size of Chromosomes (DC22 without Crossover)

The summary of the RMSE's of the results given in table 5.13 and 5.14 with and without crossover are shown in Fig 5.15.

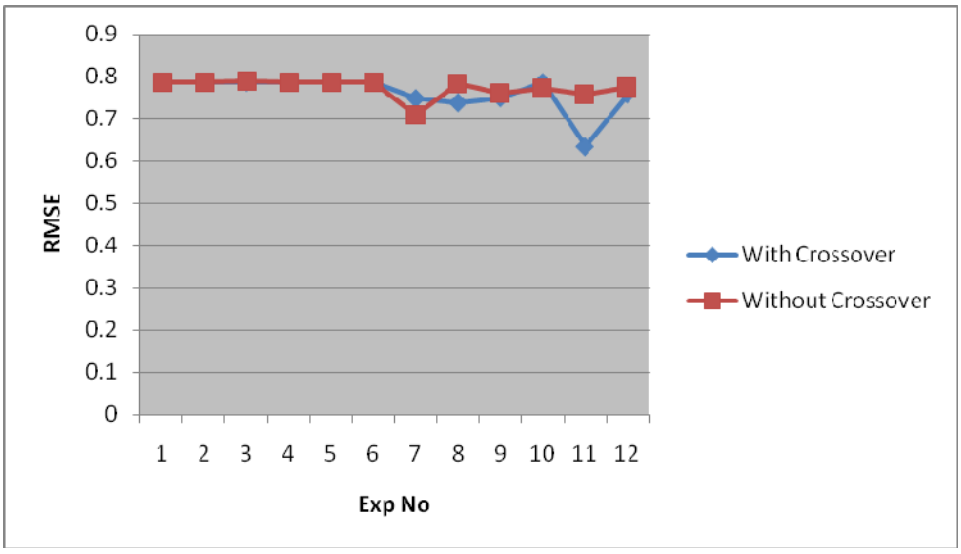

Fig 5.15 Summary of the Training Results of DC22 with & without Crossover Operation

The averages of the rmse's of the results calculated from table 5.13 and 5.14 are given in Table 5.15.

Average RMSE of Training Results with Crossover	Average RMSE of Training Results without Crossover
0.76020092	0.772531637

Table 5.15 Average RMSE of Training Results with & without Crossover (DC22)

5.3.1.2.3 Data Combination 3 (DC23)

Then results of the dataset with data part or data combination 3 (DC23) by changing the population size, and by selecting the different number of best chromosomes with and without crossover operation are shown in Table 5.16 and 5.17.

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		0.714630955109737	0.726538627533142	0.72695674529044
2	20	**8**	**10**	**12**
		0.726558485418481	0.726675478297488	0.726537368824476
3	50	**20**	**25**	**30**
		0.726537346599972	0.72653753627359	0.726557356446783
4	100	**40**	**50**	**60**
		0.723558351800119	0.726537474546244	0.726198786612324

Table 5.16 RMSE of Training Results with Crossover (DC23)

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		0.726551463478924	0.72653812290335	0.726538472674194
2	20	**8**	**10**	**12**
		0.726538185942841	0.726537382338909	0.701322929303188
3	50	**20**	**25**	**30**
		0.726544395107216	0.726539287464678	0.72531990226945
4	100	**40**	**50**	**60**
		0.723400376093425	0.720602851703414	0.726537329786993

Table 5.17 RMSE of Training Results without Crossover (DC23)

From the table 5.16, we observe that lowest RMSE is 0.714630955109737 with population size 10 and best chosen chromosome is 4, and their training pattern is shown in the Fig 5.16.And from table 5.17, the lowest RMSE is 0.701322929303188 with population size 20 and number of best chromosome is 12 and their training graph is shown in Fig 5.17.

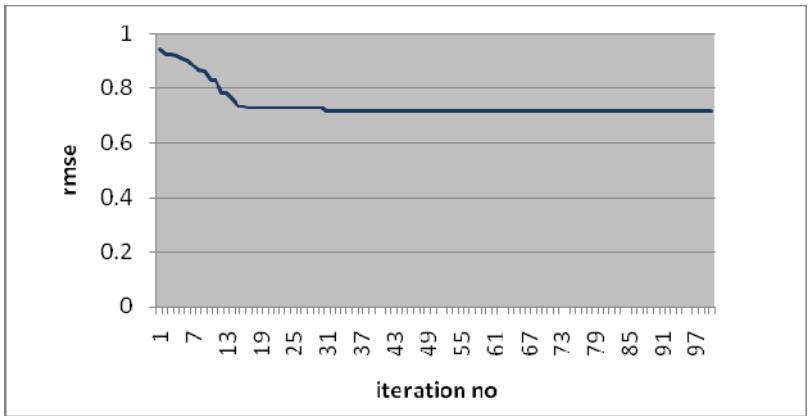

Fig 5.16 Graph with 10 Population Size and 4 Best Chosen Size of Chromosomes (DC23 with Crossover)

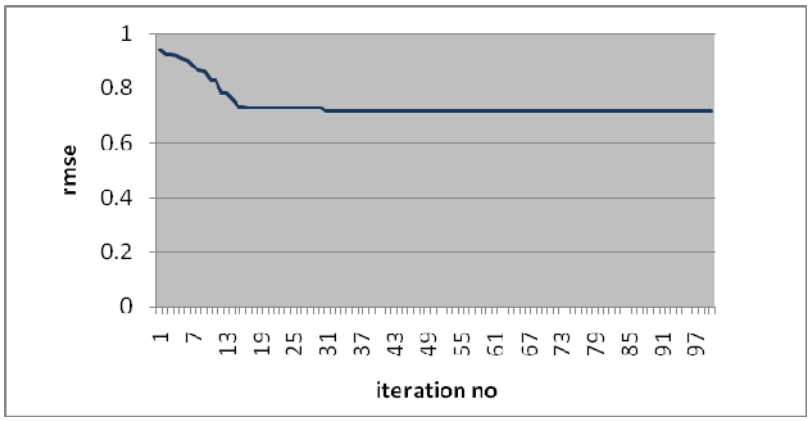

Fig 5.17 Graph with 20 Population Size and 12 Best Chosen Size of Chromosomes (DC23 without Crossover)

The summary of the RMSE's of the results given in table 5.16 and 5.17 with and without crossover are shown in Fig 5.18.

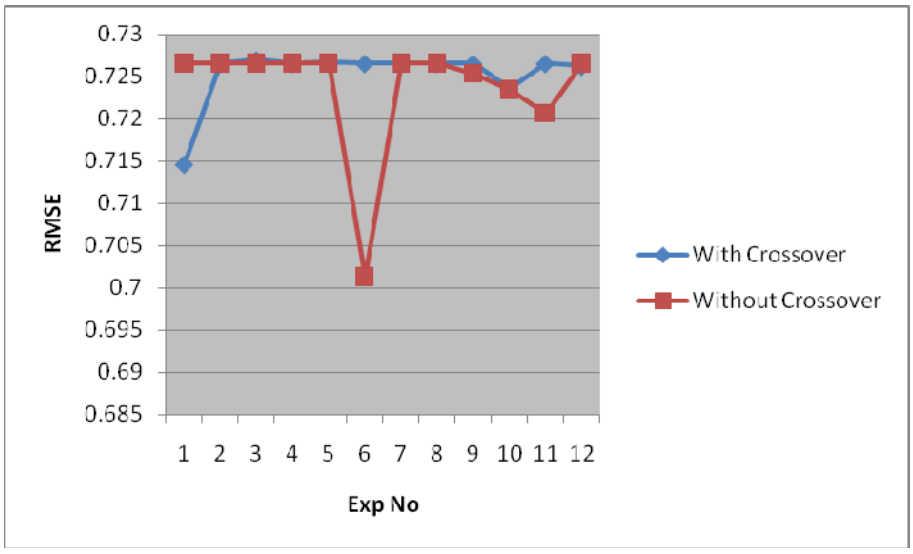

Fig 5.18 Summary of the Training Results of DC23 with & without Crossover Operation

The averages of the rmse's of the results calculated from table 5.16 and 5.17 are given in Table 5.18.

Average RMSE of Training Results with Crossover	Average RMSE of Training Results without Crossover
0.725318709	0.723580892

Table 5.18 Average RMSE of Training Results with & without Crossover (DC23)

5.3.1.3 Dataset 3

Now we will discuss the results of dataset 3 with different combinations of data turn by turn.

5.3.1.3.1 Data Combination 1 (DC31)

The results of the dataset with data part or data combination 1 (DC31) with and without crossover operation and by changing the population size, and by selecting the different number of best chromosomes are shown in Table 5.19 and 5.20.

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		0.853498565299813	0.853088253350499	0.854353166051225
2	20	**8**	**10**	**12**
		0.853094546950559	0.853090996977275	0.852154627366715
3	50	**20**	**25**	**30**
		0.853087763744071	0.853087753763717	0.853087641064629
4	100	**40**	**50**	**60**

		0.853087256802815	0.853088166640357	0.85308723411144

Table 5.19 RMSE of Training Results with Crossover (DC31)

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		0.853124744626693	0.85312580372016	0.853466542544103
2	20	**8**	**10**	**12**
		0.853104526049794	0.853265680607571	0.853219309384738
3	50	**20**	**25**	**30**
		0.853092096320901	0.853087411303574	0.853101455059848
4	100	**40**	**50**	**60**
		0.85308889437917	0.853087497982272	0.853087233009058

Table 5.20 RMSE of Training Results without Crossover (DC31)

From the table 5.19, we observe that lowest RMSE is 0.852154627366715 with population size 20 and number of best chosen chromosomes is 12, and their training pattern is shown in the Fig 5.19.And from table 5.20, the lowest RMSE is 0.853087233009058 with population size 100 and number of best chromosomes is 60 and their training graph is shown in Fig 5.20.

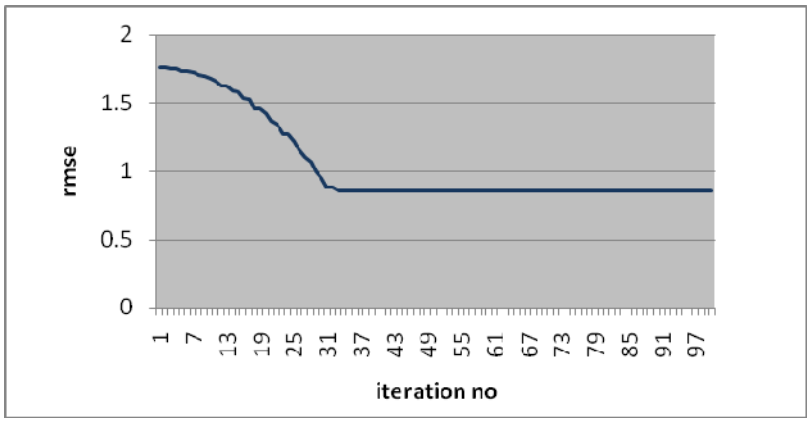

Fig 5.19 Graph with 20 Population Size and 12 Best Chosen Size of Chromosomes (DC31 with Crossover)

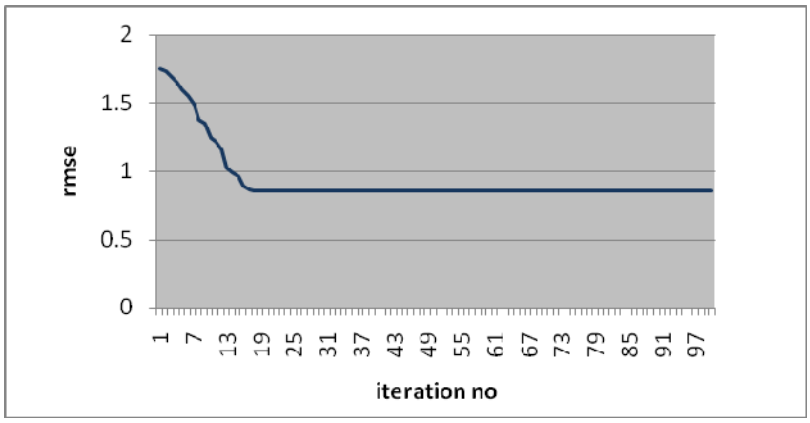

Fig 5.20 Graph with 100 Population Size and 60 Best Chosen Size of Chromosomes (DC31 without Crossover)

The summary of the RMSE's of the results given in table 5.19 and 5.20 with and without crossover are shown in Fig 5.21.

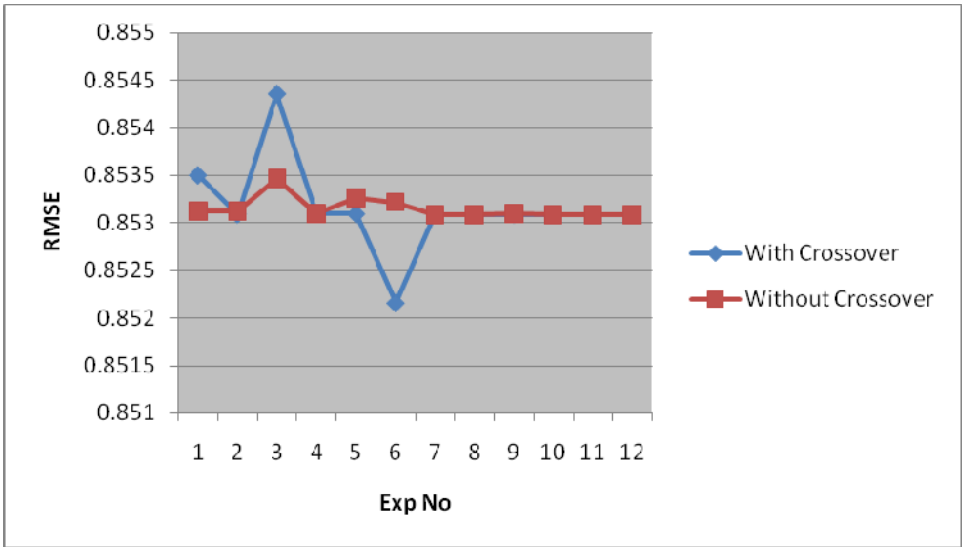

Fig 5.21 Summary of the Training Results of DC31 with & without Crossover Operation

The averages of the rmse's of the results calculated from table 5.19 and 5.20 are given in Table 5.21

Average RMSE of Training Results with Crossover	Average RMSE of Training Results without Crossover
0.853150498	0.853154266

Table 5.21 Average RMSE of Training Results with & without Crossover (DC31)

5.3.1.3.2 Data Combination 2 (DC32)

The results of the dataset with data part or data combination 2 (DC32) with and without crossover operation and by changing the population size, and by selecting the different number of best chromosomes are shown in Table 5.22 and 5.23.

S. No	Population	Selection of Best Chosen Chromosomes for Next Generation

	Size	(RMSE of Experiments)		
1	10	**4**	**5**	**6**
		0.84582991192323	0.845813122657213	0.847798286849673
2	20	**8**	**10**	**12**
		0.845812381376998	0.845816124754819	0.845915255931628
3	50	**20**	**25**	**30**
		0.845812693606922	0.845813358365221	0.845812360722539
4	100	**40**	**50**	**60**
		0.845111236090684	0.845574422632092	0.845812330815447

Table 5.22 RMSE of Training Results with Crossover (DC32)

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		0.845833345026194	0.845824221376483	0.845812320780172
2	20	**8**	**10**	**12**
		0.845815121172136	0.845813445229444	0.845814691805434
3	50	**20**	**25**	**30**
		0.845813154773314	0.845813113041145	0.845812275056077
4	100	**40**	**50**	**60**
		0.845812448250635	0.845812348582527	0.845813593878958

Table 5.23 RMSE of Training Results without Crossover (DC32)

From the table 5.22, we observe that lowest RMSE is 0.845111236090684 with population size 100 and number of best chosen chromosomes is 40, and their training pattern is shown in the Fig 5.22. And from table 5.23, the lowest RMSE is 0. 0.845812275056077 with population size 50 and number of best chromosomes is 30 and their training graph is shown in Fig 5.23.

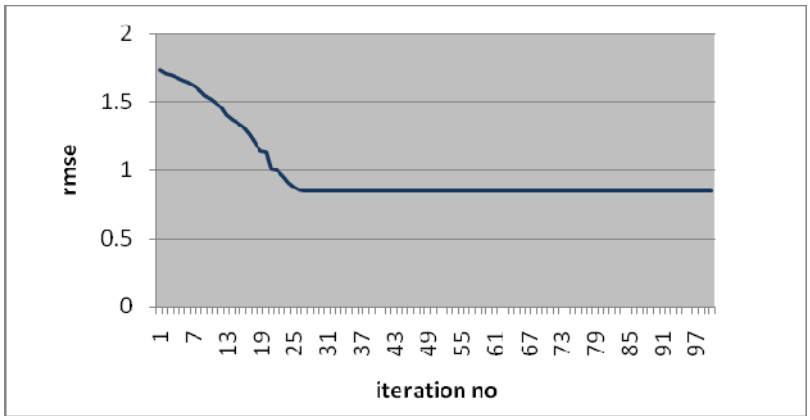

Fig 5.22 Graph with 100 Population Size and 40 Best Chosen Size of Chromosomes (DC32 with Crossover)

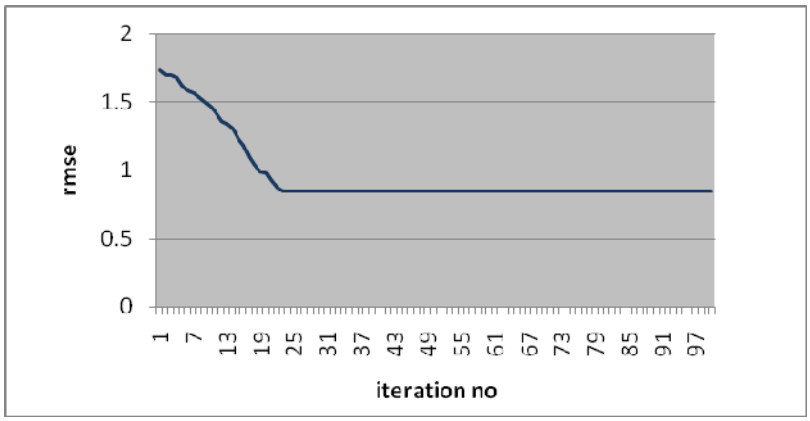

Fig 5.23 Graph with 50 Population Size and 30 Best Chosen Size of Chromosomes (DC32 without Crossover)

The summary of the RMSE's of the results given in table 5.22 and 5.23 with and without crossover are shown in Fig 5.24.

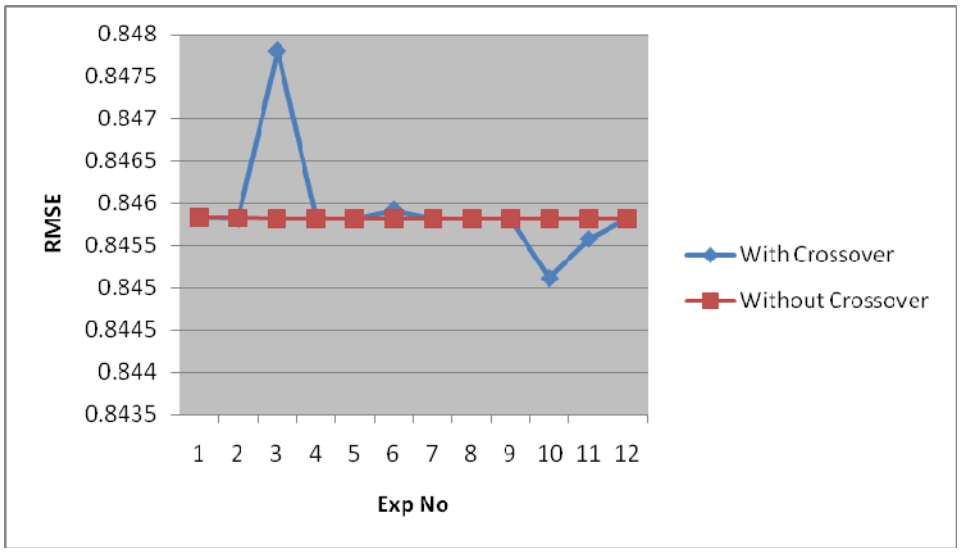

Fig 5.24 Summary of the Training Results of DC32 with & without Crossover Operation

The averages of the rmse's of the results calculated from table 5.22 and 5.23 are given in Table 5.24

Average RMSE of Training Results with Crossover	Average RMSE of Training Results without Crossover
0.845910124	0.84581584

Table 5.24 Average RMSE of Training Results with & without Crossover (DC32)

5.3.1.3.3 Data Combination 3 (DC33)

The results of the dataset with data part or data combination 3 (DC33) with and without crossover operation and by changing the population size, and by selecting the different number of best chromosomes are shown in Table 5.25 and 5.26.

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		0.889731133110635	0.889449872478944	0.889393387565423
2	20	**8**	**10**	**12**
		0.889343769475781	0.8894400201316	0.889402742047199
3	50	**20**	**25**	**30**
		0.889359318856807	0.886720720416569	0.887053659050378
4	100	**40**	**50**	**60**
		0.889342547916596	0.889342374392427	0.877870996195777

Table 5.25 RMSE of Training Results with Crossover (DC33)

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		0.889435152662864	0.889345442538824	0.889421363625855
2	20	**8**	**10**	**12**
		0.889366400431416	0.889342521962807	0.889342839631661
3	50	**20**	**25**	**30**
		0.889342903282971	0.879418433408268	0.889344019068735
4	100	**40**	**50**	**60**
		0.889342449852189	0.889342374392427	0.887606809612924

Table 5.26 RMSE of Training Results without Crossover (DC33)

From the table 5.25, we observe that lowest RMSE is 0.877870996195777 with population size 100 and number of best chosen chromosomes is 60, and their training pattern is shown in the Fig 5.25. And from table 5.26, the lowest RMSE is 0.879418433408268 with population size 50 and number of best chromosomes is 25 and their training graph is shown in Fig 5.26.

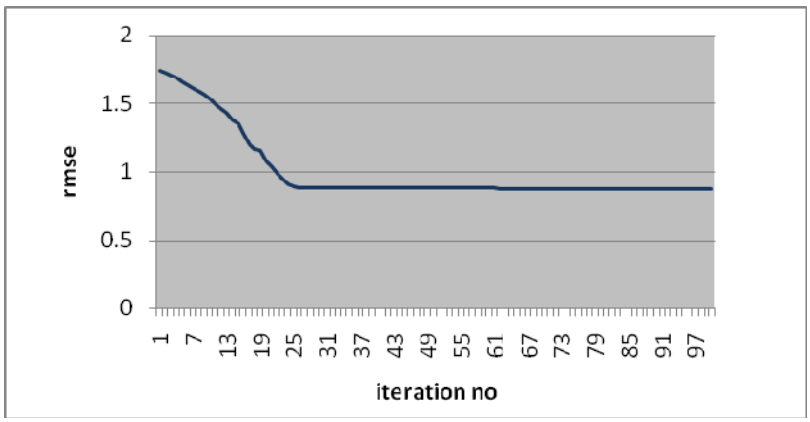

Fig 5.25 Graph with 100 Population Size and 60 Best Chosen Size of Chromosomes (DC33 with Crossover)

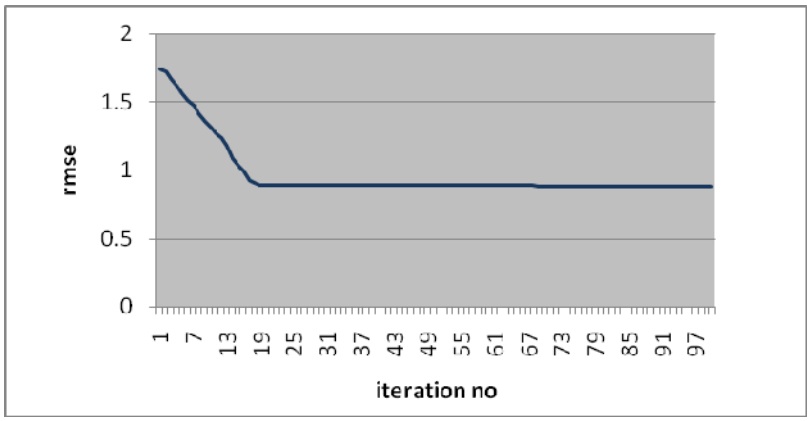

Fig 5.26 Graph with 50 Population Size and 25 Best Chosen Size of Chromosomes (DC33 without Crossover)

The summary of the RMSE's of the results given in table 5.25 and 5.26 with and without crossover are shown in Fig 5.27.

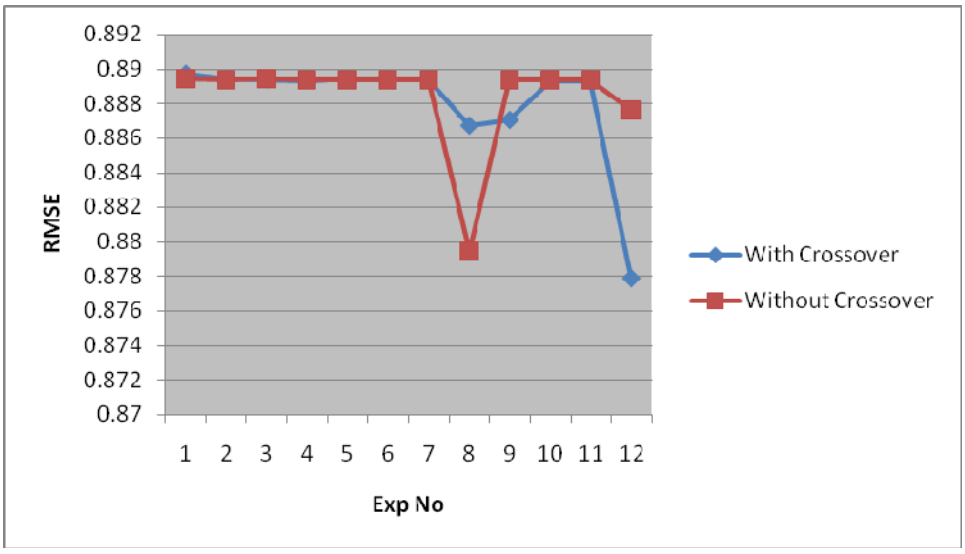

Fig 5.27 Summary of the Training Results of DC32 with & without Crossover Operation

The averages of the rmse's of the results calculated from table 5.25 and 5.26 are given in Table 5.27.

Average RMSE of Training Results with Crossover	Average RMSE of Training Results without Crossover
0.888037545	0.888387559

Table 5.27 Average RMSE of Training Results with & without Crossover (DC33)

5.3.2 Cumulative Analysis on the Datasets

We have compared the results of the datasets in terms of root mean square error. By looking at the results, we have drawn some conclusions from it which we will discuss one by one. The analysis of the results with respect to different parameters is as follows.

5.3.2.1 Crossover Operation

By looking at the table which shows the summaries of the training results of the different combination of the datasets, we have observed that in more than half of the results, average of RMSE of the experiments is low with crossover operation. Moreover, by looking at the graphs which show the summaries of the experiments with and without crossover operation, we have observed that lowest root mean square error was associated with the crossover operation. So from this observation, we can say that crossover operation improves the performance of the network, or in other words the network is better trained with crossover operation, because crossover operation have combined the connection weights of best chromosomes, so there is more diversity in the connection weights of chromosomes and more chance that RMSE is more reduced.

5.3.2.2 Population Size of Chromosomes

The other thing which we observe from the results is that most of the time the lowest RMSE comes with population size of 50 and 100, so the conclusion which we derived from it is that "greater the population size, lower rmse", the reason for this is that larger the population size means more variety in the connection weights of the chromosomes which are used to train the network.

5.4 Testing Results & Analysis

Testing is a building block in the development of any application. Testing is defined as a process which are used to identify the correctness and performance of the computer software or application, means how well it performs. It only discovers the correctness or defects, not correcting it. [44] In testing, we are generalizing the error, means how well it performs on the untrained data.. There are further categorization of cross validation methods which are holdout cross validation, k-fold cross validation, leave one out cross validation [45]. In this thesis work, we use the k-fold cross validation method. In k-fold cross validation method, the whole data is divided into three subsets and every time one subset is used for testing purpose called testing set while other two subset are used for training purpose called training set.. Training dataset is used to train the fuzzy neural network and testing data is used to estimate that how well the fuzzy neural network is trained. The performance of training and testing data is measured in terms of root mean square error.

5.4.1 Results

As we discussed above, that each dataset is divided into three parts and each part is used in testing. So we will discuss the results of testing of dataset with respect to their combination.

One important thing is that the part of dataset which is used for testing are not included in training.

5.4.1.1 Dataset 1

We will discuss the results with different combinations of data one by one.

5.4.1.1.1 Data Combination 1 (DC11)

The testing results of the dataset with data part or data combination 1 (DC11) with and without crossover operation and by changing the population size, and by selecting the different number of best chromosomes are shown in Table 5.28 and 5.29.

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		1.45955365436334	1.45881584577382	1.45931353345552
2	20	**8**	**10**	**12**
		1.67221054070569	1.45944892615229	1.46071225461028
3	50	**20**	**25**	**30**
		1.45565267726206	1.45905814550451	1.45924126844571
4	100	**40**	**50**	**60**
		1.46871459041243	1.45901631002939	1.6888694613283

Table 5.28 RMSE of Testing Results with Crossover (DC11)

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		1.45880531400417	1.45848865944384	1.46013526416725
2	20	**8**	**10**	**12**
		1.45915590193692	1.45927174001992	1.45883454739994
3	50	**20**	**25**	**30**
		1.43894845855592	1.45916768663753	1.45924645568662
4	100	**40**	**50**	**60**
		1.45078652756554	1.5116934911351	1.4592239133083

Table 5.29 RMSE of Testing Results without Crossover (DC11)

From the table 5.28, we observe that lowest RMSE is 1.45565267726206 with population size 50 and number of best chosen chromosome is 20. And from table 5.29, the lowest RMSE is 1.43894845855592 with population size 50 and number of best chromosome is 20.

The summary of the RMSE's of the results given in table 5.28 and 5.29 with and without crossover operation are shown in Fig 5.28

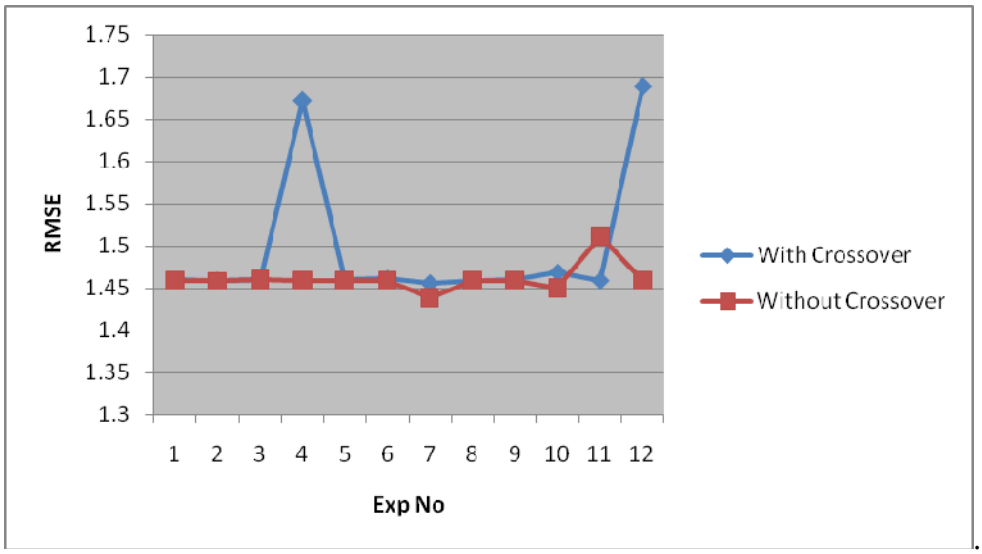

Fig 5.28 Summary of the Testing Results of DC11 with & without Crossover Operation

The averages of the rmse's of the results calculated from table 5.28 and 5.29 are given in Table 5.30.

Average RMSE of Testing Results with Crossover	Average RMSE of Testing Results without Crossover
1.49677267	1.461146497

Table 5.30 Averages RMSE of Testing Results with & without Crossover Operation (DC11)

5.4.1.1.2 Data Combination 1 (DC12)

The testing results of the dataset with data part or data combination 2 (DC12) with and without crossover operation and by changing the population size, and by selecting the different number of best chromosomes are shown in Table 5.31 and 5.32.

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		1.47513596603901	1.47376113598488	1.47290812012457
2	20	**8**	**10**	**12**
		1.47384067984055	1.4750779409451	1.47311013970736
3	50	**20**	**25**	**30**
		1.47513652636656	1.47700049059649	1.45148418343979
4	100	**40**	**50**	**60**
		1.40763694062448	1.42044066494135	1.400836484229

Table 5.31 RMSE of Testing Results with Crossover (DC12)

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		1.47464973133103	1.47493553816494	1.47714669594494
2	20	**8**	**10**	**12**
		1.4751508626539	1.47725922065732	1.47419741553959
3	50	**20**	**25**	**30**
		1.45278359746026	1.49852533949348	1.47522702234024
4	100	**40**	**50**	**60**
		1.3989814530978	1.47434108641329	1.43583337259369

Table 5.32 RMSE of Testing Results without Crossover (DC12)

From the table 5.31, we observe that lowest RMSE is 1.400836484229 with population size 100 and number of best chosen chromosome is 60. And from table 5.32, the lowest RMSE is 1.3989814530978 with population size 100 and number of best chromosome is 40.

The summary of the RMSE's of the results given in table 5.31 and 5.32 with and without crossover operation are shown in Fig 5.29

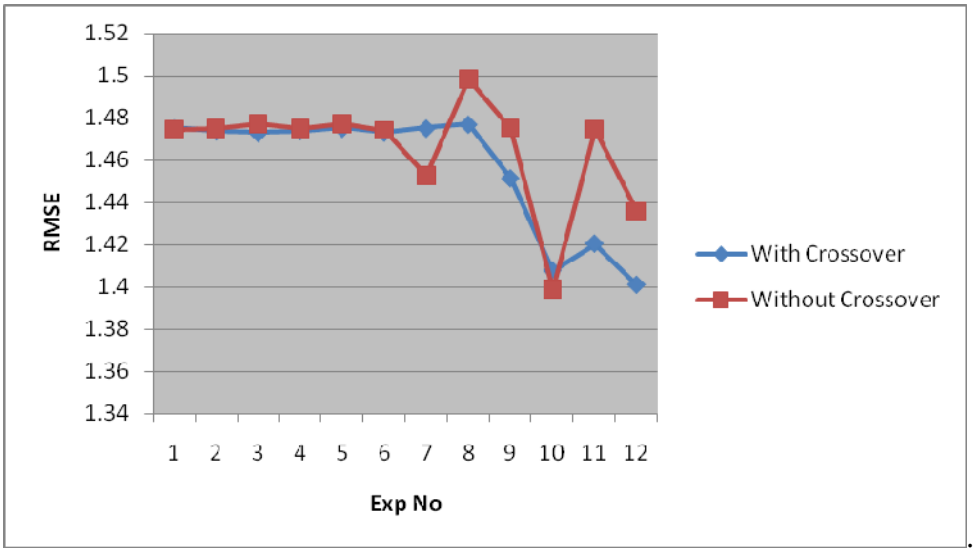

Fig 5.29 Summary of the Testing Results of DC12 with & without Crossover Operation

The averages of the rmse's of the results calculated from table 5.31 and 5.32 are given in Table 5.33.

Average RMSE of Testing Results with Crossover	Average RMSE of Testing Results without Crossover
1.456364106	1.465752611

Table 5.33 Averages RMSE of Testing Results with & without Crossover Operation (DC12)

5.4.1.1.3 Data Combination 3 (DC13)

The testing results of the dataset with data part or data combination 3 (DC13) with and without crossover operation and by changing the population size, and by selecting the different number of best chromosomes are shown in Table 5.34 and 5.35.

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		1.46095700758555	1.46470528137562	1.46407424858062
2	20	**8**	**10**	**12**
		1.4655400421302	1.4635136727361	1.46590707359042
3	50	**20**	**25**	**30**
		1.46132408883306	1.46462699359236	1.49989087970591
4	100	**40**	**50**	**60**
		1.43813540919528	1.46463913909041	1.4735391488274

Table 5.34 RMSE of Testing Results with Crossover (DC13)

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		1.46449313727159	1.46622866650681	1.46235976642011
2	20	**8**	**10**	**12**
		1.53792666932487	1.46593584384851	1.50729283946837
3	50	**20**	**25**	**30**
		1.46347287625809	1.46130603533105	1.50993813968947
4	100	**40**	**50**	**60**
		1.46495282592517	1.47031788961447	1.47128255848725

Table 5.35 RMSE of Testing Results without Crossover (DC13)

From the table 5.34, we observe that lowest RMSE is 1.43813540919528 with population size 100 and number of best chosen chromosome is 50. And from table 5.35, the lowest RMSE is 1.46130603533105 with population size 50 and number of best chromosome is 25.

The summary of the RMSE's of the results given in table 5.34 and 5.35 with and without crossover operation are shown in Fig 5.30.

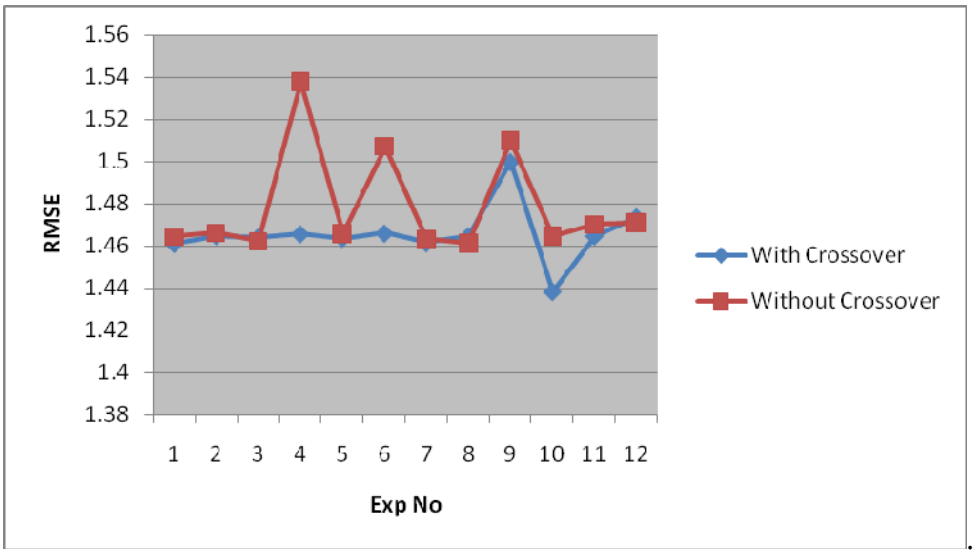

Fig 5.30 Summary of the Testing Results of DC13 with & without Crossover Operation

The averages of the rmse's of the results calculated from table 5.34 and 5.35 are given in Table 5.36.

Average RMSE of Testing Results with Crossover	Average RMSE of Testing Results without Crossover
1.465571082	1.478792271

Table 5.36 Averages RMSE of Testing Results with & without Crossover Operation (DC13)

5.4.1.2 Dataset 2

We will discuss the results with different combinations of data one by one.

5.4.1.2.1 Data Combination 1 (DC21)

The testing results of the dataset with data part or data combination 1 (DC21) with and without crossover operation and by changing the population size, and by selecting the different number of best chromosomes are shown in Table 5.37 and 5.38.

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		0.623797558621138	0.623538533570821	0.624238670691255
2	20	**8**	**10**	**12**
		0.623868818729786	0.623422041782224	0.623413127481597
3	50	**20**	**25**	**30**
		0.623409218163806	0.623259508502428	0.611247527459716
4	100	**40**	**50**	**60**
		0.59637985004315	0.592343638935951	0.602246525092826

Table 5.37 RMSE of Testing Results with Crossover (DC21)

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		0.623275141377187	0.621852255854453	0.623337868980252
2	20	**8**	**10**	**12**
		0.617233072521659	0.623385036235649	0.623401289872932
3	50	**20**	**25**	**30**
		0.611177421073346	0.623419895082388	0.617176669421871
4	100	**40**	**50**	**60**
		0.57455614266472	0.609017226055767	0.610719908521369

Table 5.38 RMSE of Testing Results without Crossover (DC21)

From the table 5.37, we observe that lowest RMSE is 0.592343638935951 with population size 100 and number of best chosen chromosome is 50. And from table 5.38, the lowest RMSE is 0.57455614266472 with population size 100 and number of best chromosome is 40.

The summary of the RMSE's of the results given in table 5.37 and 5.38 with and without crossover operation are shown in Fig 5.31.

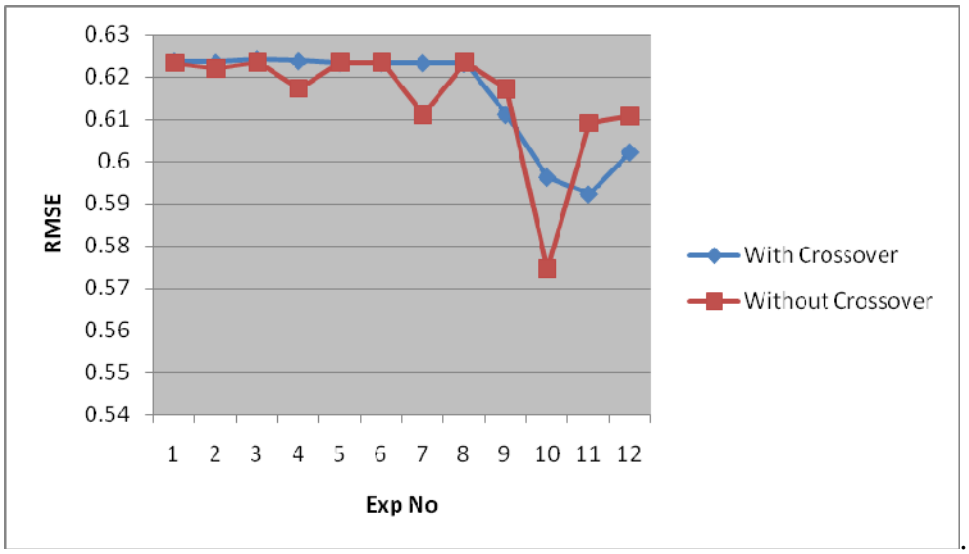

Fig 5.31 Summary of the Testing Results of DC21 with & without Crossover Operation

The averages of the rmse's of the results calculated from table 5.37 and 5.38 are given in Table 5.39.

Average RMSE of Testing Results with Crossover	Average RMSE of Testing Results without Crossover
0.615930418	0.614879327

Table 5.39 Averages RMSE of Testing Results with & without Crossover Operation (DC21)

5.4.1.2.2 Data Combination 2 (DC22)

The testing results of the dataset with data part or data combination 2 (DC22) with and without crossover operation and by changing the population size, and by selecting the different number of best chromosomes are shown in Table 5.40 and 5.41.

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		0.622975665776128	0.622891823880451	0.62289197696855
2	20	**8**	**10**	**12**
		0.623188885109456	0.622846086957695	0.62294468668883
3	50	**20**	**25**	**30**
		0.58982919780186	0.588744508131284	0.596092113020063
4	100	**40**	**50**	**60**
		0.622897891600409	0.514331789486978	0.605450018145771

Table 5.40 RMSE of Testing Results with Crossover (DC22)

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		0.622884282861822	0.622987788249734	0.625457732457799
2	20	**8**	**10**	**12**
		0.622896867585704	0.622927345739741	0.622900985553704
3	50	**20**	**25**	**30**
		0.561867556016215	0.62091265587529	0.603772968141184
4	100	**40**	**50**	**60**
		0.614880272582332	0.601210704057985	0.616513793827449

Table 5.41 RMSE of Testing Results without Crossover (DC22)

From the table 5.40, we observe that lowest RMSE is 0.514331789486978 with population size 100 and number of best chosen chromosome is 50. And from table 5.41, the lowest RMSE is 0.561867556016215 with population size 50 and number of best chromosome is 20.

The summary of the RMSE's of the results given in table 5.40 and 5.41 with and without crossover operation are shown in Fig 5.32.

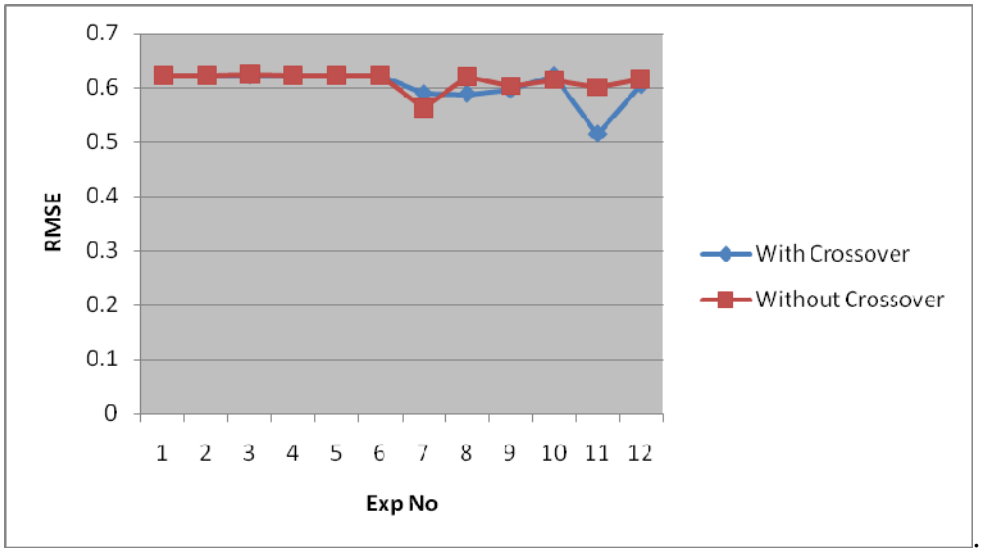

Fig 5.32 Summary of the Testing Results of DC22 with & without Crossover Operation

The averages of the rmse's of the results calculated from table 5.40 and 5.41 are given in Table 5.42.

Average RMSE of Testing Results with Crossover	Average RMSE of Testing Results without Crossover
0.604590387	0.613267746

Table 5.42 Averages RMSE of Testing Results with & without Crossover Operation (DC22)

5.4.1.2.3 Data Combination 3 (DC23)

The testing results of the dataset with data part or data combination 3 (DC23) with and without crossover operation and by changing the population size, and by selecting the different number of best chromosomes are shown in Table 5.43 and 5.44.

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		0.618832131210131	0.623656720784124	0.622956818490383
2	20	**8**	**10**	**12**
		0.623835195377885	0.624264280728401	0.623587144740045
3	50	**20**	**25**	**30**
		0.623589076970229	0.623624771451243	0.623404320134514
4	100	**40**	**50**	**60**
		0.618175155988828	0.623580337226775	0.624399407611922

Table 5.43 RMSE of Testing Results with Crossover (DC23)

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		0.623790045410455	0.6235581339618	0.623550150533192
2	20	**8**	**10**	**12**
		0.623646464644337	0.623586094742587	0.603144551188115
3	50	**20**	**25**	**30**
		0.623732553292529	0.623535387841109	0.622505051745046
4	100	**40**	**50**	**60**
		0.619572643653266	0.617596428802631	0.623590786314214

Table 5.44 RMSE of Testing Results without Crossover (DC23)

From the table 5.43, we observe that lowest RMSE is 0.618175155988828 with population size 100 and number of best chosen chromosome is 40. And from table 5.44, the lowest RMSE is 0.603144551188115 with population size 20 and number of best chromosome is 12.

The summary of the RMSE's of the results given in table 5.43 and 5.44 with and without crossover operation are shown in Fig 5.33.

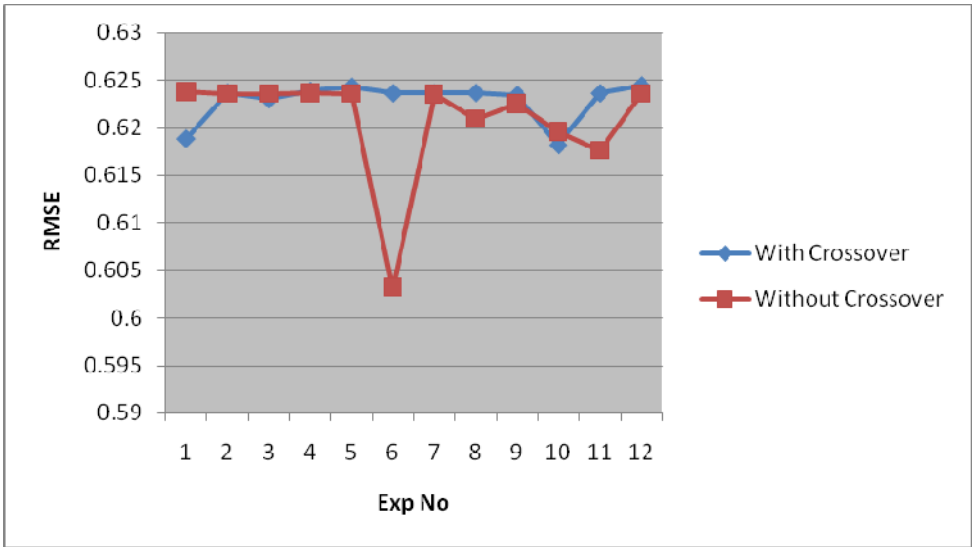

Fig 5.33 Summary of the Testing Results of DC23 with & without Crossover Operation

The averages of the rmse's of the results calculated from table 5.43 and 5.44 are given in Table 5.45.

Average RMSE of Testing Results with Crossover	Average RMSE of Testing Results without Crossover
0.622825447	0.620749033

Table 5.45 Averages RMSE of Testing Results with & without Crossover Operation (DC23)

5.4.1.3 Dataset 3

We will discuss the results with different combinations of data one by one.

5.4.1.3.1 Data Combination 1 (DC31)

The testing results of the dataset with data part or data combination 1 (DC31) with and without crossover operation and by changing the population size, and by selecting the different number of best chromosomes are shown in Table 5.46 and 5.47.

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		0.705564096058224	0.704807347256134	0.705136928139166
2	20	**8**	**10**	**12**
		0.704778543144133	0.704790966247574	0.708374718288802
3	50	**20**	**25**	**30**
		0.704841774484552	0.70481270779954	0.704814270731231
4	100	**40**	**50**	**60**
		0.704829969631705	0.704846848749679	0.704827832546508

Table 5.46 RMSE of Testing Results with Crossover (DC31)

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		0.704979520400426	0.704733628296321	0.705522169914631
2	20	**8**	**10**	**12**
		0.704757509274665	0.704703510113762	0.704703417487986
3	50	**20**	**25**	**30**
		0.704786549297968	0.704818402416826	0.70491366185805
4	100	**40**	**50**	**60**
		0.704853886066065	0.704837294546055	0.70482589252514

Table 5.47 RMSE of Testing Results without Crossover (DC31)

From the table 5.46, we observe that lowest RMSE is 0.704778543144133 with population size 20 and number of best chosen chromosome is 8. And from table 5.47, the lowest RMSE is 0.704703417487986 with population size 20 and number of best chromosome is 12.

The summary of the RMSE's of the results given in table 5.46 and 5.47 with and without crossover operation are shown in Fig 5.34.

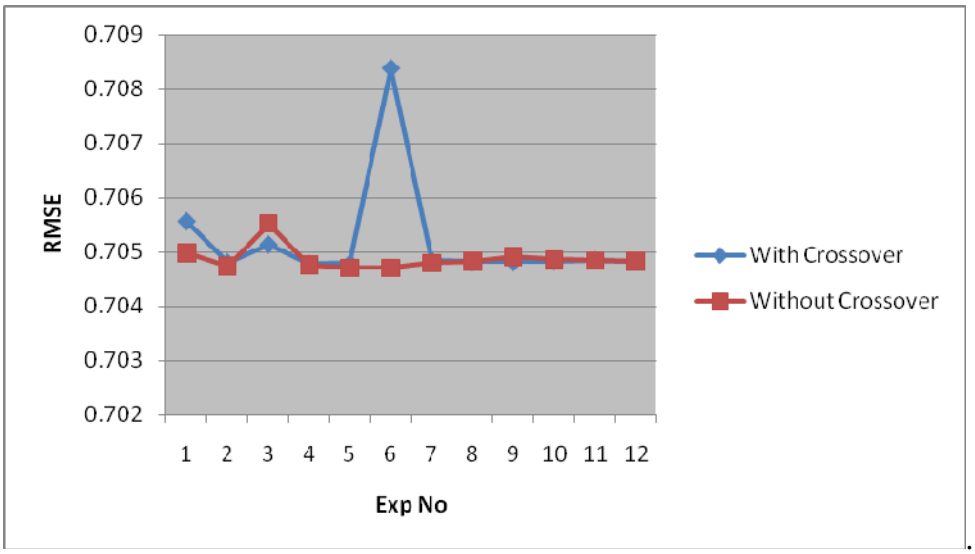

Fig 5.34 Summary of the Testing Results of DC31 with & without Crossover Operation

The averages of the rmse's of the results calculated from table 5.46 and 5.47 are given in Table 5.48.

Average RMSE of Testing Results with Crossover	Average RMSE of Testing Results without Crossover
0.705202167	0.70486962

Table 5.48 Averages RMSE of Testing Results with & without Crossover Operation (DC31)

5.4.1.3.2 Data Combination 2 (DC32)

The testing results of the dataset with data part or data combination 2 (DC22) with and without crossover operation and by changing the population size, and by selecting the different number of best chromosomes are shown in Table 5.49 and 5.50.

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		0.704736920588566	0.704711241673261	0.706462866881421
2	20	**8**	**10**	**12**
		0.704709156375465	0.704702870822574	0.704751618732011
3	50	**20**	**25**	**30**
		0.704709513297396	0.70471188055663	0.704708999928931
4	100	**40**	**50**	**60**
		0.703941739666852	0.705900664970475	0.704706258641629

Table 5.49 RMSE of Testing Results with Crossover (DC32)

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		0.704706833309395	0.704729559394198	0.704707693262924
2	20	**8**	**10**	**12**
		0.704715703837385	0.704712106972528	0.704703255482934
3	50	**20**	**25**	**30**
		0.7047113225316	0.704704370506739	0.704707299006516
4	100	**40**	**50**	**60**
		0.704705689995826	0.704708184841005	0.704706114937425

Table 5.50 RMSE of Testing Results without Crossover (DC32)

From the table 5.49, we observe that lowest RMSE is 0.703941739666852 with population size 100 and number of best chosen chromosome is 40. And from table 5.50, the lowest RMSE is 0.704703255482934 with population size 20 and number of best chromosome is 12.

The summary of the RMSE's of the results given in table 5.49 and 5.50 with and without crossover operation are shown in Fig 5.35.

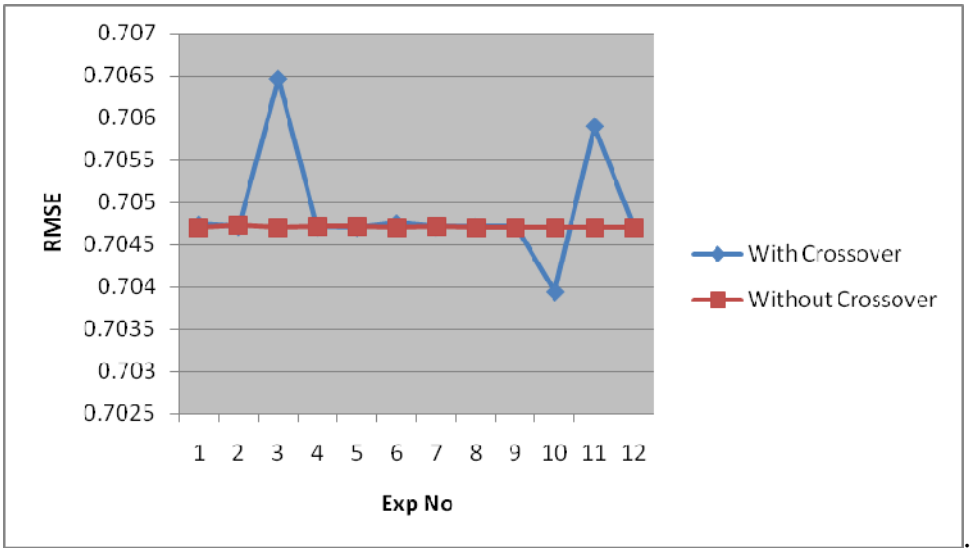

Fig 5.35 Summary of the Testing Results of DC32 with & without Crossover Operation

The averages of the rmse's of the results calculated from table 5.49 and 5.50 are given in Table 5.52.

Average RMSE of Testing Results with Crossover	Average RMSE of Testing Results without Crossover
0.704896144	0.704709845

Table 5.51 Averages RMSE of Testing Results with & without Crossover Operation (DC32)

5.4.1.3.3 Data Combination 3 (DC33)

The testing results of the dataset with data part or data combination 3 (DC33) with and without crossover operation and by changing the population size, and by selecting the different number of best chromosomes are shown in Table 5.53 and 5.54.

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		0.552674956660219	0.552597956401514	0.552756725104372
2	20	**8**	**10**	**12**
		0.552658834043939	0.55281721097396	0.55276972317618
3	50	**20**	**25**	**30**
		0.552609195876144	0.551646078048397	0.55602380449261
4	100	**40**	**50**	**60**
		0.552650313255273	0.552647962712029	0.547890916774405

Table 5.52 RMSE of Testing Results with Crossover (DC33)

S. No	Population Size	Selection of Best Chosen Chromosomes for Next Generation (RMSE of Experiments)		
1	10	**4**	**5**	**6**
		0.552811302967002	0.552666101921924	0.552596081562961
2	20	**8**	**10**	**12**
		0.552605033040785	0.552650018427661	0.552653018739183
3	50	**20**	**25**	**30**
		0.552635719171274	0.546228460989004	0.552660084155147
4	100	**40**	**50**	**60**
		0.552649115243442	0.552647962712029	0.555740380961404

Table 5.53 RMSE of Testing Results without Crossover (DC33)

From the table 5.53, we observe that lowest RMSE is 0.547890916774405 with population size 100 and number of best chosen chromosome is 60. And from table 5.54, the lowest RMSE is 0.546228460989004 with population size 50 and number of best chromosome is 25.

The summary of the RMSE's of the results given in table 5.53 and 5.54 with and without crossover operation are shown in Fig 5.36.

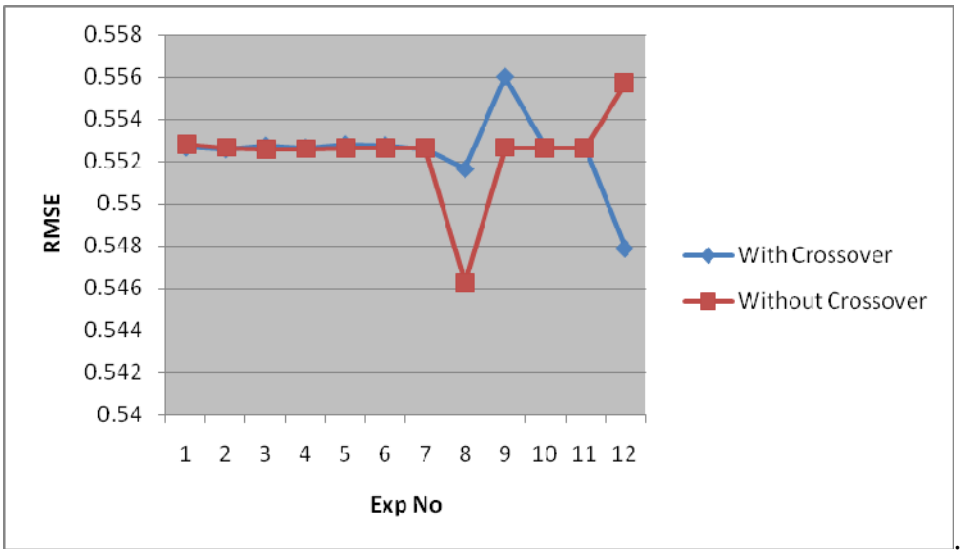

Fig 5.36 Summary of the Testing Results of DC33 with & without Crossover Operation

The averages of the rmse's of the results calculated from table 5.53 and 5.54 are given in Table 5.55.

Average RMSE of Testing Results with Crossover	Average RMSE of Testing Results without Crossover
0.55247864	0.552378607

Table 5.54 Averages RMSE of Testing Results with & without Crossover Operation (DC33)

5.4.2 Cumulative Analysis on the Datasets

We have compared the results of the datasets in terms of root mean square error. By looking at the results, we have observed that testing RMSE is lower than training RMSE which indicates that network is also well trained for unknown data. The parameters which we have discussed in training results will also be discussed in testing results.

5.4.2.1 Crossover Operation

By looking at the table which shows the summaries of the testing results of the different combination of the datasets, we have observed that in more than half of the results, average of RMSE of the experiments is low without crossover operation. Moreover, by looking at the graphs which show the summaries of the experiments with and without crossover operation, we have observed that most of the time lowest root mean square error was associated without the crossover operation. So from this observation, we can say that without crossover operation shows improved performance for unseen data, or in other words the network shows better behavior without crossover operation for unknown data.

5.4.2.2 Population Size of Chromosomes

The other thing which we observe from the testing results is same as we have observed in training results, that most of the time the lowest RMSE comes with population size of 50 and 100, so the conclusion which we derived from it is that “greater the population size, lower rmse”.

5.5 Summary

In this chapter, we have discussed the training and testing results of the datasets with their different combination. The result of the analysis have shown that crossover operation behaves well for the known data, but on the other hand the experiments which we have done without crossover operation shows better behavior for the unknown data. Moreover, the greater the population size, the lower the rmse. Furthermore, the experiments which we have done to analyze the selection of best chromosomes for the next generation does not draw any useful conclusion.

CHAPTER 6

CONCLUSIONS & FUTURE WORK

This chapter concludes the thesis, summarizing the work presented so far and provides directions for the future work. Conclusion is discussed in section 6.1 and section 6.2 discusses the future work. The summary of this chapter is discussed in section 6.3.

6.1 Conclusion

Accurate effort estimation in software engineering projects is a challenging task, and it is one of the most crucial project management activities. It helps the project manager of a software project to plan the activities that are required for project completion. In the thesis work, we have done effort estimation with a hybrid approach combining artificial intelligence techniques of fuzzy logic and neural networks. The approach provides dual benefits of incorporating qualitative knowledge of experts and learning from historical data obtained from previous projects. The fuzzy neural network (FNN) is trained with the help of evolutionary algorithm. The evolutionary approach offers the possible pathways of a problem, the solutions are compared, and indicate which one is better and identifies and collects the useful solution. This process can optimize the solution for the given problem. It demonstrates that evolving weights can be of major advantage. The evolutionary algorithm evolve the weights with two variations, in one variation we have used crossover operation and self mutation process, while in other variation we only use self mutation process. Experiments are also done by changing the population size of chromosomes, in other words we have changed the number of possible solutions.

In this research, I have achieved the following goals:

i). Training fuzzy neural network with different variations of evolutionary algorithm by using training data.

ii). Analysis of training of FNN, with and without using crossover operation, changing the population size and number of best chromosomes that does not change in next iteration.

iii). Testing of FNN is done by using different parts of each dataset called testing data, which are not used in the training phase.

iv). Use the training weights for effort estimation.

The conclusion which are derived from the experimental results are

i). Training of FNN including crossover operation shows better results.

ii). Testing of FNN without crossover operation show better results.

iii). Overall without crossover operation, training is better, because our aim to achieve accurate results for unknown data.

iv). Greater the Population size, lower the RMSE.

6.2 Future Work

There is a lot of research work available, related to neural network and data mining, which helps the researcher to continue the work based on the previous literature, and contribute faster to the pool of information.

This research may further be extended by training FNN with other algorithms like gradient descent, different variations of backpropogation algorithm e.g. Momentum strategy. Different architecture FNN can also be used, for example in this research work, we first extract the rules and then train the network, but we can also design a architecture that extract the rules

from the data during training. As training of FNN takes more time to train large set of data, we can research on some technique that takes less time for training.

6.3 Summary

In this chapter, we conclude the thesis with the observation, that FNN is well trained by using crossover operation, but for unknown data, including self mutation process is better. The other conclusion is that greater the number of possible solutions, lower the RMSE. The research work may be extended by training the FNN with other algorithms, then compared that which algorithm is better. Moreover, also research some method that takes less time to train the FNN when the size of dataset is large.

REFERENCES

[1] George Lugar, (2004), Artificial Intelligence, structure and strategies for Complex Problem Solving, fifth Edition, Published by Addison Wesley, ISBN 978-81-317-2327-2

[2] Badiru Adediji B, Cheung Jihn Y, (2002), Fuzzy Engineering Expert System with Neural Network Application, Published by John Wiley & Sons, New York

[3] Somerville, Software Engineering, (2006), Eight Edition, Published by Addison Wesley, SBN 978-81-317-2461-3

[4] On Effort Estimation in Software Projects, https://oa.doria.fi/handle/10024/31235, (20th March, 2010)

[5] L. Ford, Artificial Intelligence and Software Engineering, Artificial Intelligence Review (1987) 1, 255-273

[6] Effort Estimation, 11th Aug, 2004, http://tracer.lcc.uma.es/problems/estimation/estimation.html, (25th July, 2010)

[7] Software Development Effort Estimation, 22nd April, 2010, http://en.wikipedia.org/wiki/Sofftware_effort_estimation, (30th April, 2010)

[8] Jorgenson Magne, Forecasting of Software Development Work Effort: Evidence on Expert Judgment and Formal Models, May 2007, 1-34

[9] Sunyoung Lee, Sungzoon Cho, Patrick M. Wong, 1998, Rainfall Prediction Using Artificial Neural Networks, Journal of Geographic Information and Decision Analysis, 2(2); 233 – 242

[10] Carlos Gershenson, Artificial Neural Network for Beginners, http://arxiv.org/ftp/cs/papers/ 0308/0308031.pdf, (5th Feb, 2010)

[11] Artificial Neural Network, http://www.learnartificialneuralnetworks.com/ (16th Mar, 2010)

[12] Fuzzy Logic, http://whatis.techtarget.com/definition/0,,sid9_gci212172,00.html, (23th April, 2010)

[13] Chapter 3, Fuzzy Logic Fundamentals, March 26, 2001, http://ptgmedia.pearsoncmg.com /images /0135705991/samplechapter/0135705991.pdf, (6th July, 2010)

[14] Lan et al, An Approach to Early prediction of Software Quality, Journal of Electronic Science and Technology of China, Vol 5, Mar 2007; 1-6

[15] Kasabov N.K, Kim J.S, Gray A.R, Watts M.J, "A Fuzzy Neural Network Architecture for Adaptive learning and knowledge Acquistation", Pages 155-175, Volume 101, Issues 3-4, Journal of Information Sciences, Special Issues on Advanced Neuro-Fuzzy Techniques and Their Applications, October 1997

[16] Evolutionary Algorithm, Overview, http://www.geatbx.com/docu/algindex-01.html, (16th May, 2010)

[17] Eiben Smith, Introduction to Evolutionary Algorithm, Chapter 2, http://www.cs.vu.nl/~gusz /ecbook /Eiben-Smith-Intro2EC-Ch2.pdf

[18] Bayo Adeloye, Modeling Activated Sludge Process Using Adaptive Network- Based Fuzzy Inference System (ANFIS), 22 Aug 2002, http://www.sbe.hw.ac.uk/Researchand Business/SWMRG/Project%20Pages/Modelling%20activated%20sludges%20process%20using%20ANFIS.htm, (6th April, 2010)

[19] Blokhead, A Beginning Guide to Evolutionary Algorithm, Oct 13, 2003, http://www.perlmonks.org/?node_id=298877, (7th June, 2010)

[20] Software Development Effort Estimation, 26th June 2010, http://en.wikipedia.org/wiki/Software_development_effort_estimation, 27th June 2010

[21] Effort Estimation Tools, http://www.esendem.eu/en/project-management/55-effort-estimation-tools, (14th April, 2010)

[22] Shepperd. M, Schofield. C, “Estimating Software Project Effort Using Analogies”, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 23, NO. 12, pp. 736-743, NOVEMBER 1997

[23] Reddy CH, Raju K, A Concise Neural Network Model for Estimating Software Effort, International Journal of Recent Trends in Engineering, Issue. 1, Vol. 1, May 2009

[24] Petronio L. Braga, Adriano L. I. Oliveira, Silvio R. L. Meira, “Software Effort Estimation using Machine Learning Techniques with Robust Confidence Intervals,” pp.352-357, 7th International Conference on Hybrid Intelligent Systems (HIS 2007), 2007

[25] Kadoda G, Cartwright M, Chen L, and Shepperd M. Experiences Using Case-Based Reasoning to Predict Software Project Effort, 1-23, March 15,2000,

[26] Deng Jeremiah. D, Purvis Martin. K, “Software Effort Estimation: Harmonizing Algorithms and Domain Knowledge in an Integrated Data Mining Approach “, The Information Science Discussion paper Series, Jun 2009, ISSN 1177-455X, pp 1-13

[27] Qinbao Song, Martin Sheppard, Carolyn Mair, "Using Grey Relational Analysis to Predict Software Effort with Small Data Sets," metrics, pp.35, 11th IEEE International Software Metrics Symposium (METRICS'05), 2005

[28] Attarzadeh. I, Ow S.H, Software Development Effort Estimation based on New Fuzzy Logic Model, International Journal of Computer Theory and Engineering, Vol 1, No 4, October 2009, pp 473-476

[29] Application of COCOMO II regarding Off Shore Software Projects, Betz. S, Makio. J, http://www.outshore.org/LinkClick.aspx?fileticket=UivJBCgTPCg%3D&tabid=58&mid=387, 1-11

[30] COCOMO II and COQUALMO Data Collection Questionnaire, sunset.usc.edu/research/COCOMOII/Docs/cform22.pdf

[31] Cocomo II Cost Driver & Scale Driver Help, http://sunset.usc.edu/research/COCOMO II/expert_cocomo/drivers.html, (5th March, 2010)

[32] Overview of Cocomo, April 18, 2005, http://www.softstarsystems.com/overview.htm (10th Dec, 2009)

[33] Ayse Bener, Ekrem, Kocaqueneli, PROMISE Repository of empirical software engineering data http://promisedata.org/ repository, Software Research Laboratory (Softlab), Bogazici University, Istanbul, Turkey, May 19, 2009

[34] Reddy. Satyananda Ch, Raju KVSVN, An Improved Fuzzy Approach for COCOMO's Effort Estimation using Guassian Membership Function, Journal of Software, Vol 4, No 5, July 2009, 452-459

[35] Distance, http://mathworld.wolfram.com/Distance.html, (8th April, 2010)

[36] Mean and Standard Deviation, http://www.fmi.uni-sofia.bg/vesta/virtuallabs/freq/freq 2.html, (8th April, 2010)

[37] Tim Menzies, PROMISE Repository of empirical software engineering data**,** http://promisedata.org/repository/data/coc81/coc81_1_1.arff**,** December 2, updated (5/28/2008)

[38**]** Christian Schmid, Fuzzification**,** http://www.esr.ruhr-uni-bochum.de/rt1/syscontrol/nod e122.html**,** 2005-05-09

[39] Documentation – Fuzzy Logic Toolbox, http://www.mathworks.com/access/helpdesk /help/ toolbox/fuzzy/fp998738.html, (9th April, 2010)

[40] Alba Enrique, Cotta Carlos, Evolutionary Algorithm, 1-25, Feb 19, 2004

[41] Chellapilla Kumar, Fogel David .B, Evolving Neural Network to Play Checkers without Relying on Expert Knowledge, IEEE Transaction on Neural Networks, Vol 10, Issue 6, pp 1382-1391, Nov 1999

[42] Microsoft Visual Studio, 30th Aug, 2010, http://en.wikipedia.org/wiki/Microsoft_ Visual_Studio, (30th Aug, 2010)

[43] Visual Studio 2005, http://msdn.microsoft.com/en-us/library/ms950416.aspx, (8th Aug, 2010)

[44] Software testing, http://www.wordiq.com/definition/Software_testing, (5th July, 2010)

[45] Jeff Schneider, Cross validation, http://www.cs.cmu.edu/~schneide/tut5/node42.html, Feb 7(30th June, 2010)