Estimating Development Effort of Software Projects using ANFIS

E.Praynlin
Research Scholar
Department of CSE
Government college of Engineering
Tirunelveli

P.Latha
Associate Professor
Department of CSE
Government college of Engineering
Tirunelveli

ABSTRACT

Software Effort Prediction is the process of estimating the effort required to develop software. Effectively controlling the expensive investment of software development is achieved by accurately estimating the effort. Effort estimation at the early stage of software development is very difficult because of lot of uncertainty in input parameters which decides the software effort. Adaptive Neuro fuzzy Inference system (ANFIS) model deals effectively with uncertainty and provides reliable effort estimates In this paper ANFIS is proposed for software effort estimation is discussed. Dataset used for analysis purpose is of COCOMO II format which is the 93, 63 Historic dataset of NASA. COCOMO II consists of 17 Effort multipliers, 5 Scale factors, 1 LOC. Attributes like RUSE, PCON, and SITE play a least significant role in predicting the effort in COCOMO II Model these attributes are discarded in this approach. The ANFIS is modeled for several type of membership functions like Gaussian curve, Difference of sigmoidal membership, Gaussian combination membership, Generalized bell shaped membership, Product of sigmoidal membership, Trapezoidal membership, Triangular membership functions. From the experimental results, it was concluded that the proposed ANFIS model using Trapezoidal membership function has low MMRE (Mean Magnitude of Relative Error) than the above mentioned membership functions.

Keywords

Software Cost Estimation, Adaptive Neuro Fuzzy Inference System (ANFIS), Effort, Constructive Cost Model (COCOMO), Mean Magnitude of Relative Error (MMRE).

1. INTRODUCTION

Software development effort prediction is the process of estimating the cost required to develop software in its earlier stage of its development. Accurate effort prediction can help an organization to better analyze the feasibility of a project and to effectively manage the software development process, therefore greatly reducing the risk. Good software estimation is a difficult task[1, 2]. There are several methods available to predict the software development effort no approach is proven to be successful in effectively predicting software development effort. The several effort estimation methods are given below

A. Non-algorithmic Methods:

1) Parkinson: Using Parkinson's principle “work expands to fill the available volume” the cost is determined (not estimated) by the available resources rather than based on an objective assessment. If the software has to be delivered in 12 months and 5 people are available, the effort is estimated to be 60 person-months. Although it sometimes gives good estimation, this method is not recommended as it may provide very unrealistic estimates. Also, this method does not promote good software engineering practice.

*2) Price-to-win***:** The software cost is estimated to be the best price to win the project. The estimation is based on the customer's budget instead of the software functionality. For example, if a reasonable estimation for a project costs 100 person-months but the customer can
only afford 60 person-months, it is common that the estimator is asked to modify the estimation to fit 60 person months' effort in order to win the project. This is again not a good practice since it is very likely to cause a bad delay of delivery or force the development team to work overtime and dissatisfaction of developers.

Some of the popular methods use Neural Networks, Fuzzy logic, genetic algorithm, or combination of them. Here we will explain the use of Neuro fuzzy in detail.

B. Algorithmic methods: Most popular algorithmic method is **Constructive Cost Model** (**COCOMO**) software cost estimation model developed by Barry Boehm. The model uses a basic regression formula, with parameters that are derived from historical project data and current project characteristics. COCOMO was first published in 1981 Barry W. Boehm's Book Software engineering economics as a model for estimating effort, cost, and schedule for software projects. In 1997 COCOMO II was developed. COCOMO II is the successor of COCOMO 81 and is better suited for estimating modern software development projects [3]. COCOMO II consists of 17 cost drivers and five scale factor and one Lines of code (KLOC). These cost drivers are divided into four categories:

- Product
- platform
- Personnel
- Project.

The cost drivers are multiplicative factors that determine the effort required to complete the software project. COCOMO II can be used to determine the required development schedule by using the equation: SCED = Expected duration/nominal duration.

In the COCOMO II model, some of the most important factors contributing to a project's duration and cost are the Scale Factors. You set each Scale Factors to describe your project; these Scale Drivers determine the
exponent used in the Effort Equation.
The five Scale Factors are:

- Precedentedness
- Development Flexibility
- Risk resolution
- Team Cohesion
- Process Maturity

Calculation of Effort :

Effort in terms of person months can be given by the formula

$$PM = A * Size^{E} * \sum_{i=1}^{17} EM_i \quad (1)$$

Where $E = B + 0.01 * \sum_{j=1}^{5} SF_j$, A =2.94, B = 0.91
Where PM = Person Months.

The rest of this paper is organized as follows. In section II, we discuss about Software Effort estimation. In Section III, we discuss the ANFIS model, and in Section IV, we describe about the dataset used. In Section V, Experimentation part with results we describe how the input are given and how we validated using several membership function. Finally, we conclude in Section V. Future work is discussed in Section VI.

2. SOFTWARE EFFORT ESTIMATION

Software project managers usually estimate the software development effort. If the effort is estimated cost and duration can be easily calculated because cost is the product of working hours and wages, duration depends on how many hours a person works. Accurate Effort is necessary in the early stages of a software life cycle in order to appropriately plan, monitor and control the allocated resources for software project development activities. Hence, precision in estimating the required software development effort plays a critical factor on the success of a software development or maintenance project. Software effort prediction is needed for the following Major decision situations

- Making investment or other financial decisions involving a software development effort
- Setting project budgets and schedules as a basis for planning and control
- Deciding on or negotiating tradeoffs among software cost, schedule, functionality, performance or quality factors
- Making software cost and schedule risk management decisions
- Deciding which parts of a software system to develop, reuse, lease, or purchase
- Making legacy software inventory decisions: what parts to modify, phase out, outsource, etc

Hurdles in software cost estimation:

Software developers always have interest to know the effort estimation of software tasks. It could be done by comparing similar tasks that have already been developed. Although, estimating task has an uncertain nature, as it depends on several and usually not clear factors. Software schedule and cost estimation supports the planning and tracking of software projects. Effectively controlling the expensive investment of software development is of high importance.

Requirements are not clearly understood at the early stage of development. Some may be expert in one programming language another have expertise in some other programming language. For some programming language there will be a sophisticated development environment like GUI and the other may not have this options. So, there is huge uncertainty in Effort estimation and it is hard to be modelled mathematically. The reliable and accurate effort prediction in software engineering is an ongoing challenge due to it allows for considerable financial and strategic planning.

3.ANFIS MODEL

A. Architecture:

Neural network is good in learning and highly interpretable but fuzzy logic is good in handling imprecision[4]. The advantage of both the neural network and fuzzy logic can be combined by using the neuro fuzzy model. The neuro-fuzzy approach has added the advantage of reduced training time, not only due to its smaller dimensions but also because the network can be initialized with parameters relating to the problem domain.

A specific approach in neuro-fuzzy [5] development is the adaptive neuro-fuzzy inference system (ANFIS), which has shown significant results in modelling nonlinear functions (Jang et al., 1997). ANFIS uses a feed forward network to search for fuzzy decision rules that perform well on a given task. Using a given input-output data set, ANFIS creates a FIS whose membership function parameters are adjusted using a back propagation algorithm alone or a combination of a back propagation algorithm with a least squares method. This allows the fuzzy systems to learn from the data being modelled.

ANFIS (Adaptive Neuro-Fuzzy Inference system) used in this paper is of sugeno type Fuzzy inference system. It applies the combination of Least square method and the back propagation gradient descent method for training FIS membership function parameters to emulate the given training dataset[4

Consider a first order Takagi-Sugeno fuzzy model with a two input (x,y), one output system having two membership functions for each input. A first-order Sugeno fuzzy model has two rules:

• Rule1: If x is A1 and y is B1, then f1 = p1x + q1y + r1
• Rule2: If x is A2 and y is B2, then f2 = p2x + q2y + r2

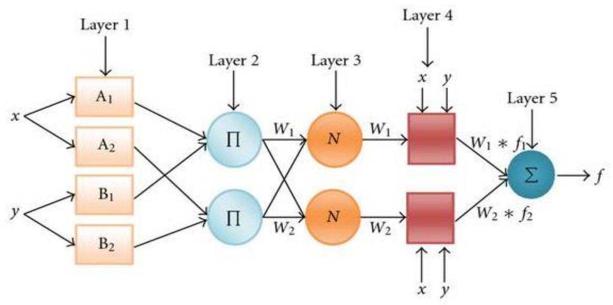

Fig.1 Architecture of ANFIS

Layer 1: Then, the functioning of ANFIS is a five-layered feed-forward neural structure, and the functionality of the nodes in these layers can be summarized as:

$$\boldsymbol{O_{1,i} = \mu_{A_i}(x)} \quad \textbf{for i=1,2} \quad \textbf{(2)}$$

$$\boldsymbol{O_{1,i} = \mu_{B_{i-2}}(y)} \quad \textbf{for i=3,4} \quad \textbf{(3)}$$

Layer 2: Where *x* or *y* is the input to the node, *Ai* or *Bi-2* is a fuzzy set associated with this node. At the first layer, for each input, the membership grades in the corresponding fuzzy sets are estimated. O1,i is the membership grade of a fuzzy set (A1,A2,B1,B2). At the second layer, all potential rules between the inputs are formulated by applying fuzzy intersection (AND). The product operation is used to estimate the firing strength of each rule.

$$\boldsymbol{O_{1,i} = w_1 = \mu_{A_i}(x) \times \mu_{B_{i-2}}(y),} \quad \textbf{i=1,2} \quad \textbf{(4)}$$

Layer 3: The third layer is used for estimation of the ratio of the *i*th rule's firing strength to the sum of all rule's firing strengths.

$$\boldsymbol{O_{3,i} = \overline{w_i} = \frac{w_i}{w_1 + w_2}} \quad \textbf{i=1,2} \quad \textbf{(5)}$$

Layer 4:

$$\boldsymbol{O_{4,i} = \overline{w_i} f_i = \overline{w_i}(p_i x + q_i y + r_i)} \qquad \textbf{(6)}$$

Where *wi* is the output of layer 3 and {p_i, q_i, r_i} is the parameter set. Parameters in this layer will be referred to as consequent parameters.

Layer 5: The final layer computes the overall output as the summation of all incoming signals from layer 4.

$$\text{Overall output} = \boldsymbol{O_{5,i} = \sum_i \overline{w}_i f_i = \frac{\sum_i \overline{w_i} f_i}{\sum_i w_i}} \qquad \textbf{(7)}$$

B. Membership function:

Membership function associates each point in the fuzzy set a real number in the interval [0,1] is called degree or grade of membership. [5]Membership function is generally represented in graphical form. The various type of membership function used are described below

1) Gaussian Membership function: Gaussian membership function is a simple and popular membership function. Gaussian function depend on two parameters σ and c as given as

$$\boldsymbol{f(x, \sigma, c) = e^{\frac{-(x-c)^2}{2\sigma^2}}} \qquad \textbf{(8)}$$

When σ = 2 and c = 5 the following plot is obtained

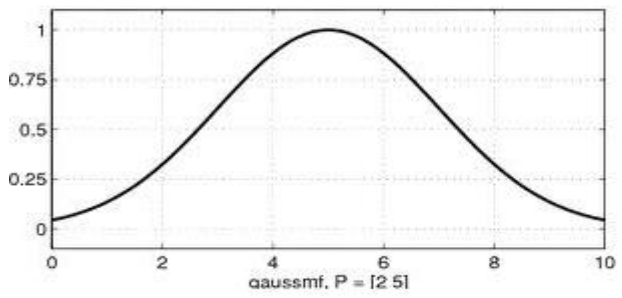

Fig.2 Gaussian membership function

2) Difference sigmoid membership function: Difference sigmoid membership function is just a difference between two sigmoid membership function

$$\boldsymbol{f(x; a, c) = \frac{1}{1+e^{-a(x-C)}}} \qquad \textbf{(9)}$$

Membership function depends on four parameters a_1, c_1, a_2, c_2 and is the difference between two of these sigmoidal functions $f_1(x, a_1, c_1) - f_2(x, a_2, c_2)$. If a_1= 5, c_1=2, a_2=-5, c_2=7 then the following plot is obtained

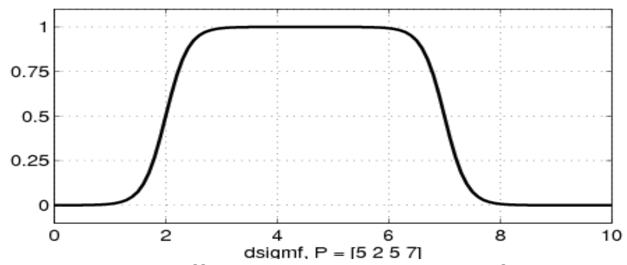

Fig.3 Difference sigmoid membership function

3) Gaussian combination membership function: The Double Gaussian membership function is a combination of two of these two parameters. The first function, specified by sig1 and c1, determines the shape of the left-most curve. The second function specified by sig2 and c2 determines the shape of the right-most curve. Whenever c1 < c2, the gauss2mf function reaches a maximum value of 1. Otherwise, the maximum value is less than one.

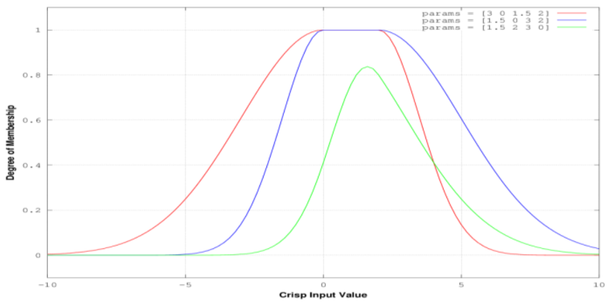

Fig.4 Gaussian combination membership function

4) Generalized Bell shaped membership function: Generalized Bell shaped membership function depends on three parameters a, b, and c as given by where the parameter b is usually positive. The parameter c locates the center of the curve

$$\boldsymbol{f(x; a, c) = \frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}}} \qquad \textbf{(10)}$$

When a=2,b=4,c=6. The plot is obtained as

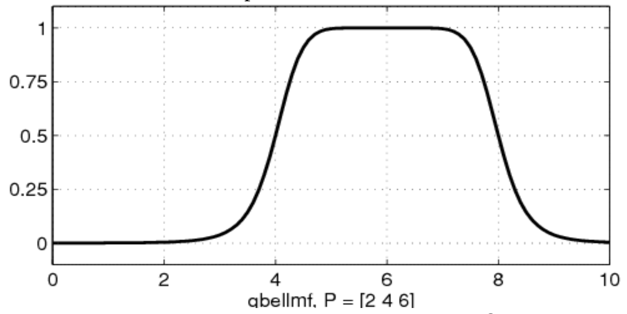

Fig.5 Generalized Bell shaped membership function

5) Product of two sigmoid membership function: Product of sigmoid membership function is just a product between two sigmoid membership function

$$\boldsymbol{f(x; a, c) = \frac{1}{1+e^{-a(x-C)}}} \qquad \textbf{(11)}$$

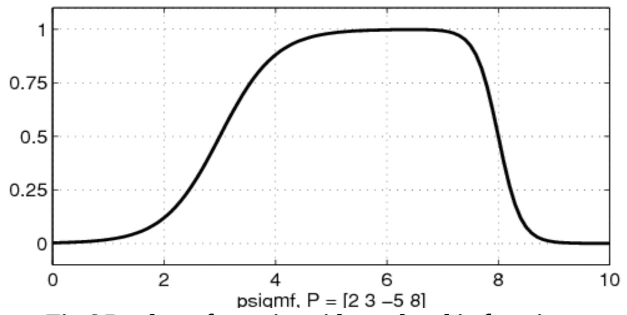

Fig:6 Product of two sigmoid membership function

Membership function depends on four parameters a_1, c_1, a_2, c_2 and is the difference between two of these sigmoidal functions $f_1(x, a_1, c_1) \times f_2(x, a_2, c_2)$.If a_1= 2, c_1=3, a_2=-5, c_2=8 then the following plot is obtained

6) Trapezoidal membership function:

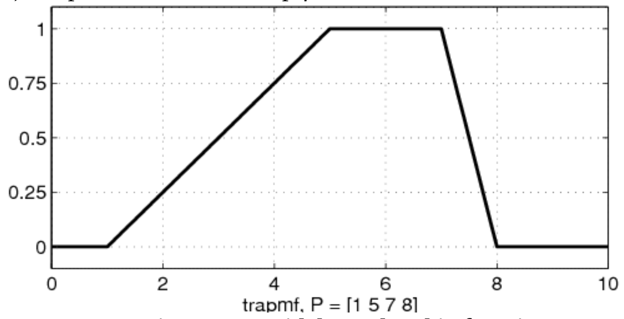

Fig.7 Trapezoidal membership function

The parameters a and d set the left and right "feet," or base points, of the trapezoid. The parameters b and c set the "shoulders," or top of the trapezoid.

$$f(x;a,c,d) = \max(\min\left(\frac{x-a}{b-a},1,\frac{d-x}{d-c}\right),o) \qquad (12)$$

If a=1, b=5, c=7, d=8 then the above plot is obtained

7) Triangular membership function: The parameters a and c locate the "feet" of the triangle and the parameter b locates the peak. The triangular curve is a function of a vector, x, and depends on three scalar parameters a, b, and c, as given by

$$f(x;a,c,d) = \max(\min\left(\frac{x-a}{b-a},\frac{c-x}{c-b}\right),o) \qquad (13)$$

If a=3, b=6, c=8 the following graph is obtained as triangular membership function.

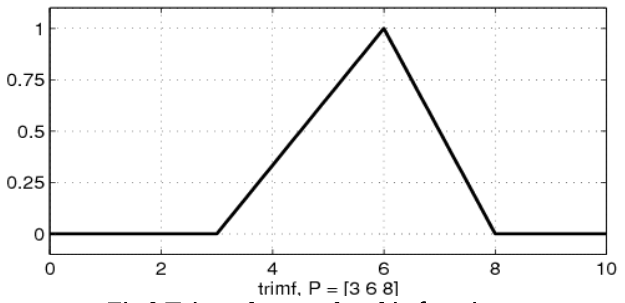

Fig.8 Triangular membership function

4. DATASET DESCRIPTION

Dataset used for analysis and validation of the model can be got from historic projects of NASA. One set of dataset consists of 63 projects and other consists of 93 datasets. The datasets is of COCOMO II format. In our experiment 93 datasets are used for training and 63 data is used for testing.

The Dataset need for training as well as testing is available in www.promisedata.org/?p=6 and in www.promisedata.org/?p=35 . The dataset available is of COCOMO 81 format which is to be converted to COCOMO II by following the COCOMO II Model definition manual [3] and Rosetta stone [7] COCOMO 81 is converted to COCOMO II. COCOMO 81 is the earlier version developed by Barry Boehm in 1981 and COCOMO II is the next model developed by Barry Boehm in year 2000. Some of the attributes like TURN are used only in COCOMO 81 and some new attributes like RUSE, DOCU, PCON, SITE are introduced in COCOMO II.

Table.1 Scale factors and their range

Scale Factor		Range
Precedentedness	PREC	0.00-6.20
Development Flexibility	FLEX	0.00-5.07
Architecture/Risk Resolution	RESL	0.00-7.07
Team Cohesion	TEAM	0.00-5.48
Process Maturity	PMAT	0.00-7.80

Table.2 Effort Multipliers and their Range

Effort Multipliers		Range
Required software Reliability	RELY	0.82-1.26
Data base size	DATA	0.90-1.28
Product Complexity	CPLX	0.73-1.74
Developed for Reusability	RUSE	0.95-1.24
Documentation Match to Life- Cycle needs	DOCU	0.81-1.23
Execution Time Constraints	TIME	1.00-1.63
Main storage Constraint	STOR	1.00-1.46
Platform Volatility	PVOL	0.87-1.30
Analyst capability	ACAP	1.42-0.71
Programmer capability	PCAP	1.34-0.76
Personal continuity	PCON	1.29-0.81
Applications Experience	APEX	1.22-0.81
Platform Experience	PLEX	1.19-0.85
Language and Tool Experience	LTEX	1.20-0.84
Use of software tool	TOOL	1.17-0.78
Multisite Development	SITE	1.22-0.80
Required Development Schedule	SCED	1.43-1.00

Every input as Effort multiplier has been tuned by following the COCOMO II model definition manual [3]. The scale factor and Effort multiplier and their range is given in Table 1 and 2. One of such inputs RELY can be discussed below in Table 2:

The Rating levels are fixed by the developer. If the failure of the software causes slight inconvenience and the corresponding rating level is very low, then the effort multiplier is fixed to be 0.82. In case of some software failure can easily be recoverable then the corresponding rating level is Nominal and rating level is fixed to be 1. If the failure of the software causes risk to human life the rating level given by the developer is very high then the effort multiplier is fixed to be 1.26

Table.3 Fixing Input attributes

RELY Descriptors	Slight inconvenience	Low, easily recoverable	Moderate, easily recoverable	High financial loss	risk to human life
Rating Levels	Very Low	Low	Nominal	High	Very High
Effort Multipliers	0.82	0.92	1	1.1	1.26

5. EXPERIMENTATION & RESULTS

ANFIS Uses two set of datasets One set of dataset consists of 63 projects and other consists of 93 datasets both are from the historic projects of NASA. Here we use 93 projects for training the ANFIS and 63 projects for testing. The Number of Epoch is given as 20 and the number of membership function used is 2. The simulation is done in MATLAB 10b environment.

ANFIS generates FIS by two main methods: (i). Grid partitioning and (ii). Clustering; Hence we have splited the inputs we are using Grid partition which generates the rule base. We had used 'genfis1' command in MATLAB to generate a new fuzzy inference system with grid partitioning.

Input to Adaptive fuzzy Inference model is given separately, not more than three input then the input is linearized using logarithmic function. In the similar way the target is also normalized. The calibration method is also a modification done to improve the predictive accuracy [6]. First the network is trained with the set of inputs and another set of input is used for testing. The Actual effort is compared with the predicted effort which gives the MMRE.

Another major modification done in this paper is RUSE, PCON, SITE are the set of attributes which shows the no change in the entire dataset hence these attributes are removed for calculating the MMRE and these attribute are discarded as inputs.

For evaluating the different software effort estimation models, the most widely accepted evaluation criteria are the mean magnitude of relative error (MMRE) The magnitude of relative error (MRE) is defined as follows

$$MRE_i = \frac{|actual\ effort_i - predicted\ effort_i|}{actual\ effort_i} \qquad (14)$$

The MRE value is calculated for each observation I whose effort is predicted. The aggregation of MRE over multiple observations (N) can be achieved through the mean MMRE as follows

$$MMRE = \frac{1}{N}\sum_i^N MRE_i \qquad (15)$$

The below table represents the MMRE values for the set of inputs Trapmf has the very low MMRE compared to all other membership functions.

Table 4 MMRE for various MFs Vs Effort Multipliers

	Effort Multipliers (Units in MMRE)				
Fuzzy Membership Function	RELY, DATA, CPLX.	TIME, STORE, PVOL	ACAP, PCAP, PEXP	AEXP, ITEX, TOOL	SCED, DOCU, KLOC
gaussmf	0.4992	0.5569	0.6771	0.5945	0.5845
Dsigmf	0.4484	0.5372	0.3872	0.6028	0.8796
Gauss2mf	0.3517	0.4687	0.3935	0.7739	1.0672
Gbellmf	0.4697	0.5466	0.3901	0.6134	0.3794
Psigmf	0.4484	0.5372	0.3872	0.6028	0.8796
trapmf	0.3504	0.3378	0.3451	0.921	0.2571
Trimf	0.5304	0.6779	3.1254	0.5372	0.2362

Table 5 MMRE for various MFs Vs Scale factors

Fuzzy Membership Function	Scale factors (MMRE)
Gaussmf	0.5114
Dsigmf	0.7794
Gauss2mf	0.8546
Gbellmf	0.5087
Psigmf	0.8517
Trapmf	0.4542
Trimf	0.6375

Table 5 shows the MMRE value for various membership function where Scale factor is given as the input. MMRE 0.4542 for Trapezoidal membership function is good compared to 0.5087 of Generalised Bell membership function prediction is based on available data. If certain error may exist in the actual data, that error data will enter into the model and as a result the prediction will not be accurate. The fuzzification plays a major role in handling the data in fuzzy environment. The fuzzification is based on certain membership function. The selection of a particular membership function depends on the nature of data value to be used. If the selected membership function is not proper, the input fuzzy data is wrong and as a result the output fuzzy data will also go wrong and the defuzzified output data will be improper and error will be very high. Hence the selection of membership function plays an important role in Prediction model.

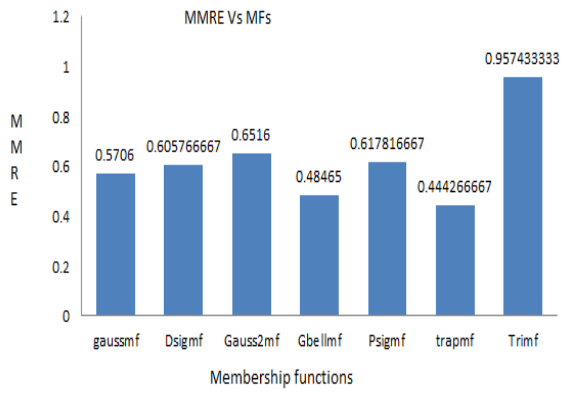

Fig.9 Plot of Average MMRE Vs Membership functions

6. CONCLUSION

Software effort prediction involves of dealing with uncertainty as inputs. ANFIS can handle this kind of uncertainty; however, selecting the particular membership function plays a crucial role in effort prediction. Comparison of membership functions like Gaussian curve, Difference of sigmoidal membership, Gaussian combination membership, Generalized bell shaped membership, Product of sigmoidal membership, Trapezoidal membership and Triangular membership was done. The Results shows the trapezoidal membership function has produced the least MMRE compared to other kind of membership function. The objective of this work is to provide which membership function is to be used in ANFIS which gives accurate predictive effort.

7. FUTURE WORKS

Some of the future enhancements made to this research work are given below

1. Analysis can be made for other type of datasets like ISBSG, IBM e.tc
2. Calculation of Pred(25), Spearman's rank can lead to better validation of prediction models.
3. Analysis can also be done using artificially generated dataset.
4. Analyzing the performance of the model by varying the number of epoch, number of membership functions.

8. REFERENCES

[1] M. Sheppard, G. Kadoda, Comparing software prediction techniques using simulation, IEEE Trans. Software Eng. 27 (11) 1014–1022, November 1999.

[2] S. Chulani, Bayesian Analysis of Software Cost and Quality Models, Ph.D. Dissertation, University of Southern California, Los Angeles, 1999.

[3] Barry Boehm, COCOMO II: Model Definition Manuel. Version 2.1, Center for Software Engineering, USC, 2000.

[4] Xishi Huang, Danny Ho, Jing Ren, Luiz F. Capretz, "Improving the COCMO model using the Neuro fuzzy

approach", Journal of applied soft computing, pp 29- 40, 2007.

[4] J.S.R.Jang, Chuen-Tsai Sun, Eiji Mizutani, "Neuro – Fuzzy and Soft Computing", Prentice-Hall, 2006

[5] Anish Mittal, Kamal Prakash, Harish Mittal, " Software Cost estimation using fuzzy logic", ACM SIGSOFT software Engineering Notes, Vol.35, Nov 2010.

[6] Taeho Lee, Donoh Choi, Jongmoon Baik, "Empirical Study on Enhancing the Accuracy of Software Cost Estimation Model for Defense Software Development Project Applications", ISBN 978-89-5519-146-2 " ICACT, pp.1117- 1122, Feb.2010.

[7]Donald J. Reifer, Barry W. Boehm and Sunitha chulani, "The Rosetta stone: Making COCOMO 81 Estimates work with COCOMO II", CROSSTALK The Journal of Defense Software Engineering, pp 11 – 15, Feb.1999.