Journal of Theoretical and Applied Information Technology
15th November 2018. Vol.96. No 21

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

HYBRID ARTIFICIAL NEURAL NETWORK AND FUZZY LOGIC FOR FUNCTION APPROXIMATION

ABDUL STTAR ISMAIL WDAA

Department of Mathematics, Faculty of Education for Pure Sciences, Anbar University, Anbar, Iraq

Email: sttarwdaa@yahoo.com

ABSTRACT

The problem of intelligent hybrid systems investigated in this study. Intelligent systems consist of fuzzy systems (FS) and neural networks (NN). This intelligent system has specific properties (modeling, ability of learning, obtaining empirical rules, solving optimizing tasks, classifying …) fitting certain type of applications. The combination of NN and FS systems makes fuzzy-NN system, neuron-fuzzy system. Such type of combination of systems is known as the hybrid intelligent systems (HIS). There are programs created in C++ and Matlab for these purposes, where many demo applications were made for different HIS in the area of system control and modeling. There are three programs have developed; Neural Network program (NNP), fuzzy program (FP) and Neural networks fuzzy program (NNFP), to investigate the effect of these approaches on ANN learning using several datasets. The results have explored that Neural networks fuzzy (NNF) give quite better results in terms of small errors and convergence rate. compared to NN and FUZZY. The aim of the paper is to prove that the process of hybridization between the algorithms gives better results than the use of separate algorithms. This is known as the soft computing. This is implementation on the approximation functions.

Keywords: *Function Approximation; Neural Network; Fuzzy logic*

1. INTRODUCTION

An intelligent system called hybrid system if it combines at least two intelligent systems. For instance, the combination of fuzzy system and neural network makes a hybrid system known as neuron-fuzzy system. The comparison of several intelligent technologies is presented in Table 1 [1]. The combination of fuzzy logic, probabilistic reasoning, evolutionary computation and neural networks makes the core of soft computing (SC). It is an Imprecise and uncertain environment. On the other hand, the hard computing or traditional Computing uses numbers and crisp values while the soft computing deals with fuzzy sets or soft values [2] . Evolving method to develop HIS, that have capability of learning and reasoning in an ‘’Soft computing is capable to handle incomplete information, uncertain and Imprecise information in such a way that Reproduces human thinking. Usually, soft data is used in the real life of humans and it Expressed by words instead of numbers. The sensory body part of humans handles the soft information such as brain create soft relations and inferences in imprecise and uncertain environments. We have an extraordinary Capability for reasoning and making decisions without the help of numbers. The soft computing tries to model our sense of words in decision making [3]. However, from last few years, the artificial intelligence area has extended speedily by including genetic algorithms, artificial, fuzzy set theory and even neural networks [4]. It connects the boundaries between soft computing and modern artificial intelligence elusive and vague. when one system becomes part of the other system no one can argue, but it provides an understanding about key principles to develop HIS. A HIS may be bad or good, but it depends on the parts that are used to make hybrid system. Therefore, the selection of right components to develop a good hybrid system is our aim [5-7].

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 1: Comparison of genetic algorithms (GA), neural networks (NN), Fuzzy systems (FS) and expert systems (ES),

	ES	FS	NN	GA
Knowledge representation	○	●	□	■
Uncertainty tolerance	○	●	●	●
Imprecision tolerance	□	●	●	●
Adaptability	□	■	●	●
Learning ability	□	□	●	●
Explanation ability	●	●	□	■
Knowledge discovery and data mining	□	■	●	○
Maintainability	□	○	●	○

The terms used for grading are: □ bad, ■ rather bad, ○ rather good and ● good

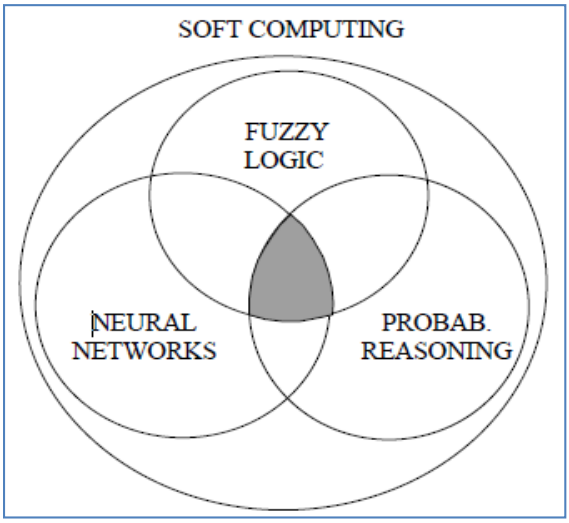
The component used for hybrid system has their own weaknesses and strengths. The mainly concerned is probabilistic reasoning with uncertainty, neural networks with learning, evolutionary computation with optimization and fuzzy logic with imprecision. A hybrid system is good if it carries the benefits of these technologies. Their combined effect allows a hybrid system to accommodate common sense, use human-like reasoning mechanisms, extract knowledge from raw data, deal with imprecision and uncertainty, learn to follow an unknown and speedily varying environment.

2. SOFT COMPUTING (SC).

"While traditional or 'hard' computing uses crisp values, or numbers, soft computing deals with soft values, or fuzzy sets. Soft computing is capable of operating with uncertain, imprecise and incomplete information in a manner that reflects human thinking. In real life, humans normally use soft data represented by words rather than numbers. Our sensory organs deal with soft information, our brain makes soft associations and inferences in uncertain and imprecise environments, and we have a remarkable ability to reason and make decisions without using numbers. Humans use words, and soft computing attempts to model our sense of words in decision making [12].

Soft computing exploits the tolerance for uncertainty and imprecision to achieve greater tractability and robustness, and lower the cost of solutions [13]. We also use words when the available data is not precise enough to use numbers. This is often the case with complex problems, and while 'hard' computing fails to produce any solution, soft computing is still capable of finding good solutions. However Figure (1) shows diagram of soft computing, neural networks and Fuzzy Logic Systems are often considered as a part of Soft Computing area".

Fig 1: Soft computing as a composition fuzzy logic, neural networks [9].



ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3. FUNCTION OF APPROXIMATION (FA).

The behavior of very complicated functions is described by function approximation. This approximation converts the complicated functions into simpler functions. Very important results have been set up in mathematics. In this study we will describe only a few that have a direct association with our aim. The Legendre and Gauss polynomials are used as function approximation for better understanding of NN. Chebychev established the idea of uniform approximation. Any continuous real can be approximated by using polynomial functions [6].

4. ADAPTIVE NEURON FUZZY INFERENCE SYSTEM (ANFIS)

The Sugeno fuzzy (SF) model was developed to generate fuzzy rules using input output data set in a systematic way. A typical SF model can be stated as:

IF R_1 is D_1
AND R2 is D2
. . . . .
AND R_m is D_m

THEN Z = f (R1; R2; . . . ; R_m)

Where R_1; R_2; . . . ; R_m are input variables; D1; D2; . . . ; Dm are fuzzy sets; and z is may be a constant or linear function of the input variables. In the case of constant z, the SF model becomes a zero-order in which the model results are specified by a singleton. If z is the polynomial of first-order as:

$Z = S_0 + S_1R_1 + S_2R_2 + \ldots + S_mR_m$

The SF model results as first-order model. Jang's ANFIS architecture is generally presented by a six layer feed forward neural network. This architecture corresponds to the SF first order model is presented in Figure 2 . It is assumed that the ANFIS has one output – z and two inputs – R_1 and R_2. Every input is denoted by two fuzzy [10]. First order polynomial output and sets. There are four rules implemented by ANFIS architecture:

Rule 1:
IF R_1 *is* D_1
AND R_2 *is* C_1
THEN $Z = f1 = S_{10} + S_{11}x_1 + S_{12}R_2$

Rule 2:
IF R_1 *is* D_1
AND R_2 *is* C_2
THEN $Z = f2 = S_{20} + S_{21}x_1 + S_{22}R_2$

Rule 3:
IF R_1 *is* D_1
AND R_2 *is* C_1
THEN $Z = f3 = S_{30} + S_{31}x_1 + S_{32}R_2$

Rule 4:
IF R_1 *is* D_1
AND R_2 *is* C_2
THEN $Z = f4 = S_{40} + S_{41}x_1 + S_{42}R_2$

Where the variables for inputs are R_1, R_2; D_1 and D_2 are fuzzy sets on the universe of discourse R_1; C_1 and C_2 are fuzzy sets on the inverse of discourse R_2; and S_{i0}, S_{i1} and S_{i2} is specified parameters set for *ith* rule. Moreover, in this study we explain the purpose of every layer of ANFIS architecture.

Layer 1: **Input Layer**

In layer 1 neurons simply pass external crisp signals to Layer 2 written as:

$$Z_i^{(1)} = R_i^{(1)} \quad (1)$$

Where the output and input for *ith* neuron in Layer 1 are denoted by $z_i^{(1)}$ and $R_i^{(1)}$ respectively.

Layer 2 : Fuzzification

Neurons performing fuzzification in layer 2. A bell activation function is associated with Fuzzification Neurons in the Jang's model. This function has a shape like regular bell and is defined as:

$$Z_i^{(2)} = \frac{1}{\left(\frac{R_i^{(2)} - d_i}{o_i}\right)^{2ci}} \quad (2)$$

Journal of Theoretical and Applied Information Technology
15th November 2018. Vol.96. No 21

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

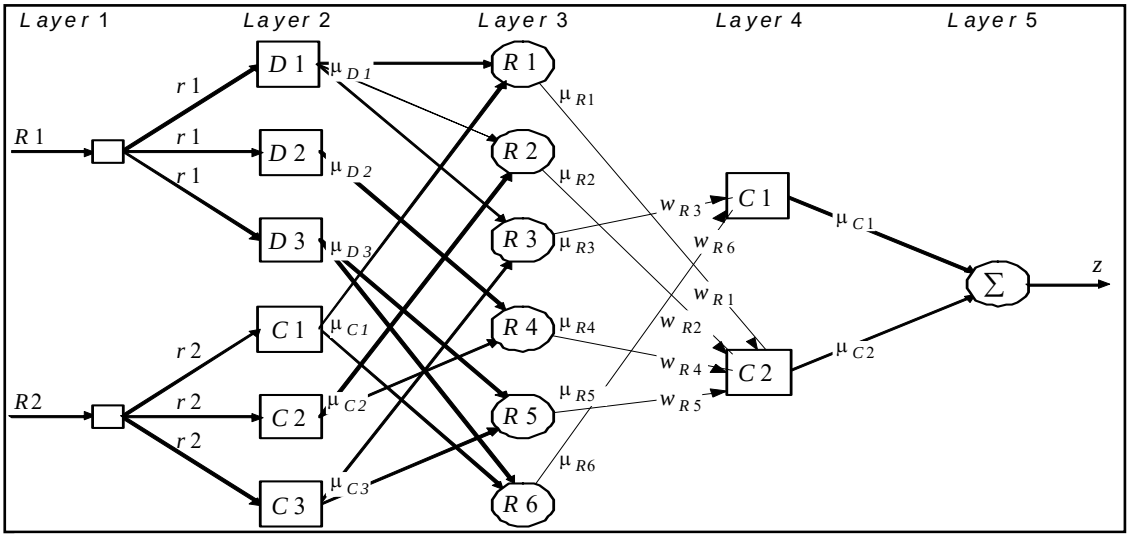


Fig 2: *Adaptive Neuron Fuzzy Inference System*

Where the output and input of *ith* neuron in Layer 2 are denoted by $Z_i^{(2)}$ and $R_i^{(2)}$ respectively ; *ci, di* and oi are ith neuron control parameters for the width, centre and slope of the bell-shaped activation function respectively.

Layer 3 : Rule Layer

Every neuron corresponds to a single (SF) rule in this layer. D rule neuron receives inputs from the respective neurons of fuzzification and calculates the firing strength of the rule. The product operator is used to calculate the conjunction of the rule antecedents in an ANFIS. Consequently, in Layer 3 the output of *ith* neuron is obtained as:

Where inputs and output are presented respectively as $R_{ji}^{(3)}$ and $z_i^{(3)}$ *i* for rule *ith* neuron in this Layer.

Layer 4: Normalization

Every neuron receives inputs from all neurons in the layer 3, and computes the normalized firing strength of a given rule in this layer. The normalized firing strength is the fraction of a given rule firing strength to the sum of all rules firing. It describes the role of a given rule to the final result. Therefore, the *ith* neuron output of Layer 4 is defined as:

$$z_i^{(4)} = \frac{R_{ji}^{(4)}}{\sum_{j=1}^{x} R_{ji}^{(4)}} = \frac{\mu_i}{\sum_{j=1}^{x} \mu_j} = \overline{\mu}_i \qquad (3)$$

Where $R_{ji}^{(4)}$ *ji* is the *jth* neuron input existed in Layer 3 to the *ith* neuron in Layer 4, n is the total number of rule neurons.

$$z_{x1}^{(4)} = \frac{\mu}{\mu_1 + \mu_2 + \mu_3 + \mu_4} = \overline{\mu} \qquad (4)$$

Layer 5: Defuzzification Layer.

Every neuron in this Layer is associated with respective normalization neuron. It also receives the initial inputs, *R1* and *R2*. A defuzzification neuron computes the value of weighted consequent for a given rule written as, s_{i0} , s_{i1} *and* $s_{i2} r_2$

Where the output and input of *ith* defuzzification neuron in Layer 5 are $z_i^{(3)}$, $R_{i1}^{(5)}$ respectively and , is a set of consequent parameters of *ith* rule. Layer 6 is denoted by a single summation neuron. This neuron calculates the sum of all outputs defuzzification neurons and gives the complete output z of ANFIS.

$$Z = \sum_{i=1}^{x} n_i^{(6)} = \sum_{i=1}^{x} \overline{\mu}_i [s_{i0} + s_{i1} r_1 + s_{i2} r_2] \qquad (5)$$

Therefore, the ANFIS is functionally equivalent to the SF first order as shown in Figure 8.10. Sometimes it is impossible or even hard to specify a rule results in the polynomial form. However, the information of previous rule consequent parameters

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

is not necessary for an ANFIS to handle a problem. The ANFIS architecture learns these parameters and tunes the related functions.

5. HYBRID LEARNING ALGORITHM

Hybrid learning algorithm is used in an ANFIS architecture. This algorithm connects the gradient descent method and least-squares estimator. Firstly, every membership neuron is assigned by an initial activation functions. The neurons function centers are associated to input *Ri* in such a way that the domain of *Ri* is equally divided, the slopes and widths are adjusted to allow enough overlapping of the respective functions[11].

Each time is composed from a forward and backward pass in the training algorithm of an ANFIS. In the case of forward pass, a set of training input vector is presented to the ANFIS, the outputs of neuron are computed layer-by-layer basis and parameters of rule consequent are investigated by the least squares estimator. The output z in the SF model inference is a linear function. Therefore, for given membership parameters values and *t* input-output patterns of a training set, we can form *t* system of linear equations having consequent parameters as:

$$\begin{cases} z_i(1) = \overline{\mu_1}(1) f_1(1) + \overline{\mu_2}(1) f_2(1) + ...\overline{\mu_x}(1) f_x(1) \\ z_i(2) = \overline{\mu_1}(2) f_2(2) + \overline{\mu_2}(2) f_2(2) + ...\overline{\mu_x}(2) f_x(2) \\ \vdots \\ z_i(t) = \overline{\mu}_1(t) f_1(t) + \overline{\mu}_2(t) f_2(t) + ...\overline{\mu}_x(t) f_x(t) \\ \vdots \\ z_i(p) = \overline{\mu}_1(t) f_1(t) + \overline{\mu}_2(t) f_2(t) + ...\overline{\mu}_x(t) f_x(t) \end{cases} \quad (6)$$

OR

$$\begin{cases} z_i(1) = \overline{\mu_1}(1)[L_{10} + L_{11}R_1(1) + L_{12}R_2(1) + ...L_{1v}R_v(1)] \\ \quad + \overline{\mu_2}(1)[L_{20} + L_{21}R_1(1) + L_{22}R_2(1) + ...L_{2v}R_v(1)] + ... \\ \quad + \overline{\mu_x}(1)[L_{x0} + L_{x1}R_1(1) + L_{x2}R_2(1) + ...L_{xv}R_v(1)] \\ z_i(2) = \overline{\mu_1}(2)[L_{10} + L_{11}R_1(2) + L_{12}R_2(2) + ...L_{1v}R_v(1)] \\ \quad + \overline{\mu_2}(2)[L_{20} + L_{21}R_1(2) + L_{22}R_2(2) + ...L_{2v}R_v(2)] + ... \\ \quad + \overline{\mu_x}(2)[L_{x0} + L_{x1}R_1(2) + L_{x2}R_2(2) + ...L_{xv}R_v(2)] \\ z_i(t) = \overline{\mu_1}(t)[L_{10} + L_{11}R_1(t) + L_{12}R_2(t) + ...L_{1v}R_v(t)] \\ \quad + \overline{\mu_2}(t)[L_{20} + L_{21}R_1(t) + L_{22}R_2(t) + ...L_{2v}R_v(t)] + ... \\ \quad + \overline{\mu_x}(t)[L_{x0} + L_{x1}R_1(t) + L_{x2}R_2(t) + ...L_{xv}R_v(t)] \\ z_i(t) = \overline{\mu_1}(t)[L_{10} + L_{11}R_1(t) + L_{12}R_2(t) + ...L_{1v}R_v(t)] \\ \quad + \overline{\mu_2}(t)[L_{20} + L_{21}R_1(t) + L_{22}R_2(t) + ...L_{2v}R_v(t)] + ... \\ \quad + \overline{\mu_x}(t)[L_{x0} + L_{x1}R_1(t) + L_{x2}R_2(t) + ...L_{xv}R_v(t)] \end{cases} \quad (7)$$

Where *x* is the neurons number in the rule layer, *v* represents the number of input variables, and $z_i(t)$ is the required overall output of the ANFIS Corresponding to the inputs $R_1(t)$, $R_2(t)$,…,$R_v(t)$ are presented to it. In the matrix notation, we have :

$$z_i = R\,L \quad (8)$$

$$z_i = \begin{bmatrix} z_i(1) \\ z_i(2) \\ \vdots \\ z_i(t) \\ \vdots \\ z_i(t) \end{bmatrix}$$

Where z_i is a $t \times 1$ desired output vector?

$$D = \begin{bmatrix} \overline{\mu}(1)\ \overline{\mu}(1)R_1(1)..\overline{\mu}(1)R_1(1)..\overline{\mu}_x(1)\ \overline{\mu}_x(1)R_1(1)..\overline{\mu}_x(1)R_v(1) \\ \overline{\mu}(2)\ \overline{\mu}(2)R_1(2)..\overline{\mu}(2)R_1(2)..\overline{\mu}_x(2)\ \overline{\mu}_x(2)R_1(2)..\overline{\mu}_x(2)R_v(2) \\ \vdots \\ \overline{\mu}(t)\ \overline{\mu}(t)R_1(t)..\overline{\mu}(t)R_1(t)..\overline{\mu}_x(t)\ \overline{\mu}_x(t)R_1(t)..\overline{\mu}_x(t)R_v(t) \\ \vdots \\ \overline{\mu}(t)\ \overline{\mu}(t)R_1(t)..\overline{\mu}(t)R_1(t)..\overline{\mu}_x(t)\ \overline{\mu}_x(t)R_1(t)..\overline{\mu}_x(t)R_v(t) \end{bmatrix}$$

D is a $t \times x(1+v)$ matrix,

And L is an $x(1+v)$ vector of unknown consequent parameters,

$$L = [\ L_{10}\ L_{11}\ L_{12}...L_{1v}\ L_{20}\ L_{21}\ L_{22}...L_{2v}...L_{x0}\ L_{x1}\ L_{x2}...L_{xv}\ ]^E$$

Normally, the number of consequent parameters $x(1+v)$ used in training is lesser than number of input-output patterns *t*. It shows that we are solving here an over determined problem, and it may be exact solution of Eq. (9) not exist. Therefore, we need to apply a least-square estimate on L , L^* to minimize the squared error. It is done by using the pseudo inverse method:

$$L^* = (D^E D)^{-1} D^E z_i \quad (9)$$

$\|DL - Z_i\|^2$

Where D is the transpose of D, and $(D^E D)^{-1} D^E$ is the pseudo inverse of A. The $(D^E D)$ matrix must be non-singular. Once, the parameters of rule consequent are developed, the actual output vector

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

of network y can be computed easily we can compute. The error vector also be computed using Equation 10:

$$er = z_i - z) \qquad (10)$$

The back-propagation algorithm is implemented in the case of backward pass. The antecedent parameters are updated according to the chain rule and the error signals are propagated back. For example, applying a correction to the bell activation function parameter 'a' and it may be expressed using the chain rule given as:

$$\Delta d = -\alpha \frac{\partial Er}{\partial d} = -\alpha \frac{\partial Er}{\partial d} \times \frac{\partial er}{\partial z} \times \frac{\partial z}{\partial (\overline{\mu_i} fi)} \times \frac{\partial (\overline{\mu_i} fi)}{\partial \mu_i} \times \frac{\partial \overline{\mu_i}}{\partial \mu_i} \times \frac{\partial \mu R1}{\partial d} \qquad (11)$$

Where α , and E are the learning rate and instantaneous value of the squared error for the ANFIS output neuron respectively[8].

$$Er = \frac{1}{2} er^2 = \frac{1}{2} (zi - z)^2 \qquad (12)$$

Thus, we have

$$\Delta d = -\alpha (zi - z)(-1) fi \times \frac{\overline{\mu_i}(1 - \overline{\mu_i})}{\mu_i} \times \frac{\mu_i}{\mu D1} \times \frac{\partial \mu D1}{\partial d} \qquad (13)$$

Or

$$\Delta d = \alpha (zi - z) fi \, \overline{\mu}_i (1 - \overline{\mu_i}) \times \frac{1}{\mu D1} \times \frac{\partial \mu D1}{\partial d} \qquad (14)$$

Where

$$\frac{\partial \mu D1}{\partial d} = -\frac{1}{\left[1 + \left(\frac{R1 - d}{o}\right)^{2c}\right]^2} \times \frac{1}{o^{2c}} \times 2c \times (R1 - d)^{2c-1} \times (-1)$$

$$= \mu^2 D1 \times \frac{2c}{o} \times \left(\frac{R1 - d}{o}\right)^{2c-1}$$

Following the similar way, we can get corrections for b and c parameters. The antecedent and consequent parameters are optimized in an ANFIS training algorithm suggested by Jang. The consequent parameters are tuned whereas the antecedent parameters remain constant in the forward pass. On the other hand, the antecedent parameters are tuned whereas the consequent parameters are remained fix in the case of backward pass. Moreover, in some cases related to relatively small data of input output, the human experts defined the membership functions. In such situations, the consequent parameters are tuned, and the membership functions are remained fixed during the training process.

6. OPTIMIZATION ALGORITHMS

The ANFIS application for Function approximation is applied in this study to follow a non-linear function trajectory represented by the equation.

$$Z = \frac{\cos(2T1)}{e^{(T2)}} \qquad (6.1)$$

$$T2 = \sin(T1) \qquad (6.2)$$

An ANFIS necessarily have one output – *Z* and two inputs – *T1* and T2 –. The ANFIS training data $[T1 \;\; T2 \;\; Z]$,

Contains 151 training samples. These samples are represented by a 151 x 3 Array, where Z is an output vector and T1, T2 are input vectors. The input vector *T1*, starts from -10 with an increment of 0.10 and ends at 5. The input vector *T2*, calculate from equation number 2.

7. DISCUSSION AND RESULTS

This section highlights the results of this study which are obtained by experiments. Three different programs have been established: Neural network Forward (NN), Adaptive Neuron Fuzzy Inference System (ANFIS) and Fuzzy Logic (FL) using dataset: with Function Approximation. The results for each dataset are compared least error.

8. DISCUSSION

In this study, apply the concept of soft computers; the hybrid neural networks with the logic fuzzy resulting, (ANFIS). The rounding flag was used and then applied to three programs (NN) (FUZZY) (ANFIS), Table (2) indicates the application of the above function with neural networks where the result was compared between the actual output of the function with the neural network outputs. There is good convergence and the error rate is acceptable. Figure (3) indicates the convergence between the two outputs. When using the same logic for the same data, the result was better than the NN. As shown in Figure. (4), but when using the proposed system (ANFIS).

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

7.1 -Results on NN Dataset

Table 2: Result of NN

T1	T2	Target	Network	Error
-10.0000	0.5440	0.2369	0.6818	-0.4449
-9.9000	0.4575	0.3679	0.6992	-0.3313
-9.8000	0.3665	0.5070	0.7196	-0.2126
-9.7000	0.2718	0.6495	0.7432	-0.0937
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
5.0000	0.9589	-2.1890	-2.4634	0.2744

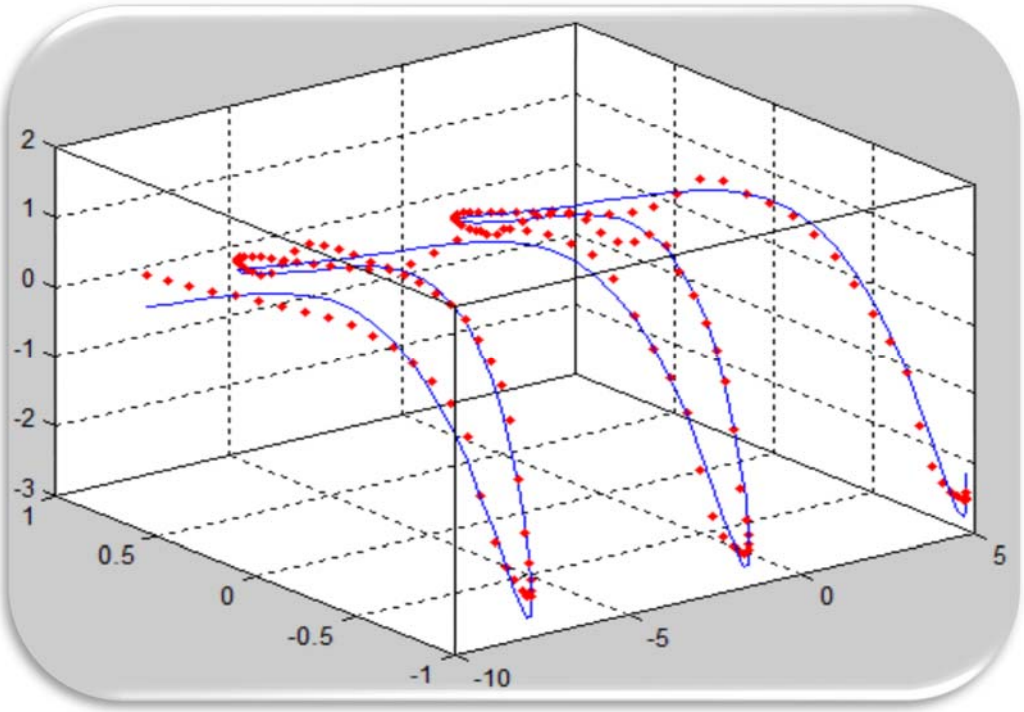


Figure 3 : NN data set

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

7.2 Results on Fuzzy Dataset

Table 3: Result of Fuzzy

T1	T2	Target	Fuzzy	Error
-10.0000	0.5440	0.2369	0.2069	0.0300
-9.9000	0.4575	0.3679	0.3429	0.0250
-9.8000	0.3665	0.5070	0.4951	0.0119
-9.7000	0.2718	0.6495	0.6547	-0.0052
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
5.0000	0.9589	-2.1890	-2.2205	0.0315

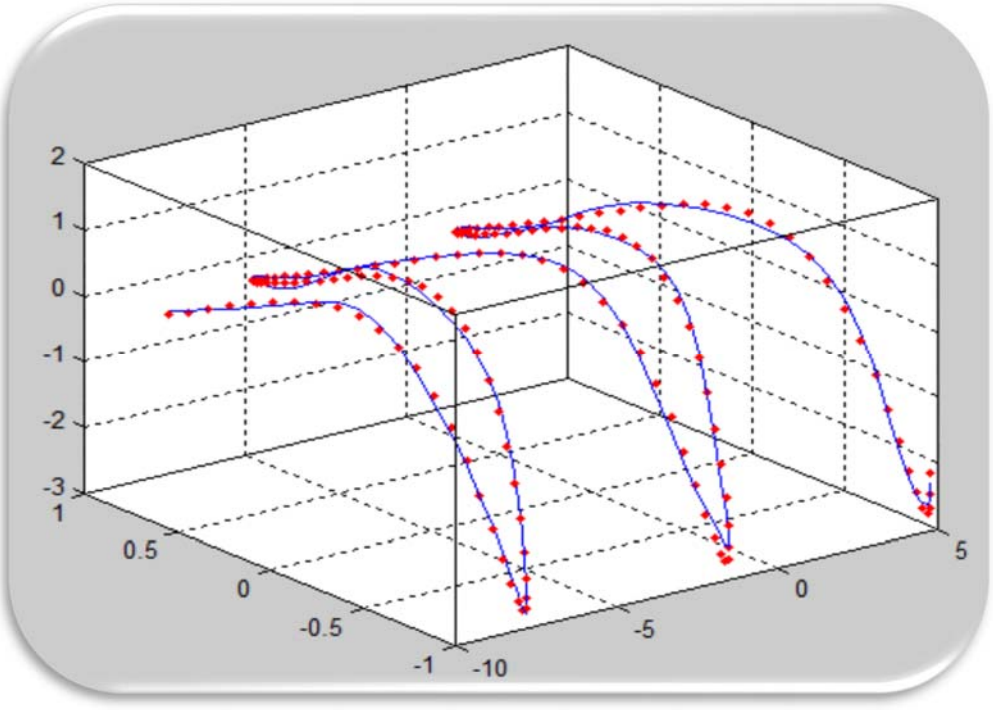


Figure 4: Fuzzy data set

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

7.3: Result of ANFIS

Table 4: Result of ANFIS

T1	T2	Target	ANFIS	Error
-10.0000	0.5440	0.2369	0.2319	0.005
-9.9000	0.4575	0.3679	0.3599	0.008
-9.8000	0.3665	0.5070	0.4969	0.0101
-9.7000	0.2718	0.6495	0.6376	0.0119
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
5.0000	0.9589	-2.1890	-2.1882	-0.0008

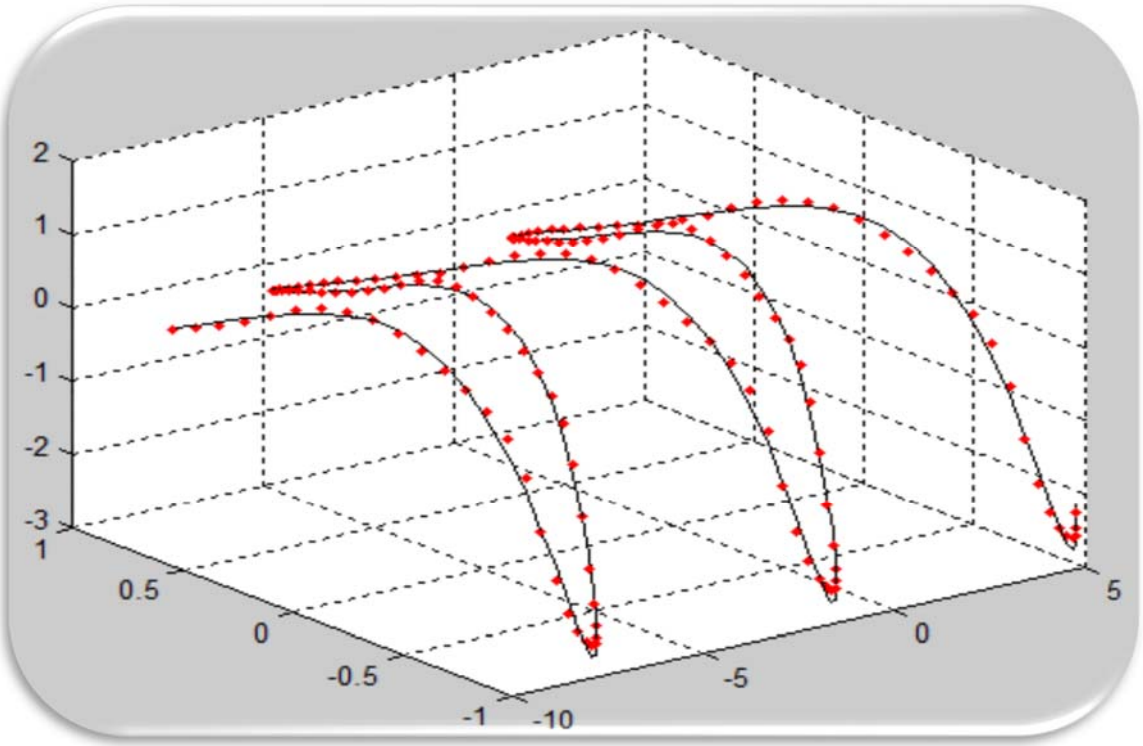


Figure 5: ANFIS data set

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The results were excellent. But when using the proposed system (ANFIS), the results were excellent.

The convergence between the output of the function and the hybrid system was very promising.

As Table (4) and Figure (5) .

Indicate where the error rate was as low as Possible. The proposed hybrid system has important and useful results.

8. CONCLUSION

A neuron-fuzzy system corresponding to the Mamdani fuzzy inference model can be expressed by a feed forward neural network containing of five layers: fuzzification, input, output, defuzzification and fuzzy rule. A neuron-fuzzy system may be applied on standard learning algorithms which established for NN with the algorithm of back-propagation. Skillful knowledge of fuzzy rules and linguistic variables may be included in the model of neuron-fuzzy system.

A Neuron-fuzzy system cans automatically transform. It into a set of fuzzy IF-THEN Rules when a representative set of examples are available. An adaptive neuron-fuzzy inference system, ANFIS, corresponds to the first. In the end, ANFIS has an extraordinary potential to generalize and rapidly converge. Particularly, it is important in an on-line learning. Consequently, the model along with its variables have several applications, especially in the area of adaptive control. And finally the use of hybridization or what is known as soft computers. Gives excellent results compared to discrete algorithmsas demonstrated in this paper.

REFERENCES

[1] T. Ba¨ck, D.B. Fogel, and Z. Michalewicz, "Handbook of Evolutionary Computation", *Institute of Physics Publishing, Bristol, Philadelphia and Oxford University Press*, (New York), 1997.

[2] L.M. Fu, "Knowledge-based connectionism for revising domain theories", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 23, No. 1, 1993, pp. 173–182.

[3] S.I. Gallant, "Connectionist expert systems", *Communications of the ACM*, Vol. 31, No. 2, 1988, pp. 152–169.

[4] S.I. Gallant, "Neural Network Learning and Expert Systems", *MIT Press*, (Cambridge, MA), 1993.

[5] Y. Ichikawa, and T. Sawa, "Neural network application for direct feedback controllers", *IEEE Transactions on Neural Networks*, Vol. 3, No. 2, 1992, pp. 224–231.

[6] H. Ishibuchi, , K. Nozaki, and H. Tanaka, " Distributed representation of fuzzy rules and its application to pattern classification", *IEEE Transactions on Fuzzy Systems*, Vol. 3, No. 3, 1992, pp. 260–270.

[7] X. Yao, "Evolving artificial neural networks", *Proceedings of the IEEE*. Vol. 87, No. 9, 1999, pp. 1423 -1447.

[8] A. Hussein Abbass, Ruhul Sarker, and Charles Newton. "A pareto differential evolution approach to vector optimization problems", *In Proceedings of the IEEE Congress on Evolutionary Computation, CEC2001*, (Seoul, Korea, IEEE Press), 2001.

[9] J.-S.R. Jang, "ANFIS: Adaptive Network-based Fuzzy Inference Systems", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 23, No. 3, 1993, 665–685.

[10] G.F. Miller, P.M. Todd, and S.U. Hedge, "Designing neural networks using genetic algorithms", *Proceedings of the Third International Conference on Genetic Algorithms, J.D. Schaffer, ed., Morgan Kaufmann*, (San Mateo, CA), 1989, pp. 379–384.

[11] S. Abe and M.-S. Lan, "A classifier using fuzzy rules extracted directly from numerical data", in: Proceedings of IEEE Internet Conf. *on Fuzzy Systems*, (San Francisco), 1993, pp.1911-1198.

[12] N. Kasabov, "Foundations of Neural Networks, Fuzzy Logic, and Knowledge Engineering", *MIT Press*, (Cambridge, MA), 1996.

[13] C. Nikolopoulos, "Expert Systems: Introduction to First and Second Generation and Hybrid Knowledge Based Systems", *Marcel Dekker, Inc.,* (New York), 1997.