Research Inventy: International Journal Of Engineering And Science
Vol.2, Issue 11 (April 2013), Pp 06-13
Issn(e): 2278-4721, Issn(p):2319-6483, Www.Researchinventy.Com

Hybrid Learning For Adaptive Neuro Fuzzy Inference System

[1] Dr.C.Loganathan and [2] K.V.Girija

[1] *Department of Mathematics, Maharaja Arts and Science College, Coimbatore,Tamilnadu, India – 641407,*
[2] *Department of Mathematics, Hindusthan College of Engineering & Technology,Coimbatore, Tamilnadu, India – 641432,*

Abstract: *Neuro fuzzy hybrid system is a learning mechanism that utilizes the training and learning neural networks to find parameters of a fuzzy system based on the symptoms created by the mathematical model. Adaptive learning is the important characteristics of neural networks. Adaptive Neuro Fuzzy Inference System (ANFIS) is used for system identification based on the available data. The main aim of this work is to determine appropriate neural network architecture for training the ANFIS structure inorder to adjust the parameters of learning method from a given set of input and output data. The training algorithms used in this work are Back Propagation, gradient descent learning algorithm and Runge-Kutta Learning Algorithm (RKLM). The experiments are carried out by combining the training algorithms with ANFIS and the training error results are measured for each combination. The results showed that ANFIS combined with RKLM method provides better training error results than other two methods.*

Key Words*: Neuro Fuzzy Hybrid System, Adaptive Neuro Fuzzy Inference System, Gradient Descent Learning Algorithm, Runge-Kutta Learning Algorithm.*

I. INTRODUCTION

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way of biological nervous systems, process information. The key element of this paradigm is the novel structure of the information processing system. Neural networks composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. It is, with their remarkable ability to derive meaning from complicated or imprecise data can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. Error measure is a mathematical expression that measures the discrepancy between the network's actual output and a desired output. Soft-computing is a practical alternative for solving computationally complex and mathematically intractable problems. The reason behind this is the fact that through the use of soft-computing methodologies, one can easily combine the natural system dynamics and an intelligent machine the most popular constituents of the soft-computing methodologies are the neural networks and fuzzy logic. Neural networks provide the mathematical power of the brain whereas the fuzzy logic based mechanisms employ the verbal power. The latter allows the linguistic manipulation of input state-Output data.

The next section dwells on back propagation algorithm. The training is performed by the error back propagation algorithm. Next Hybrid approach, The Runge-Kutta algorithm, is explained with their functional equivalence to Fuzzy Inference Systems (FIS). In the same reference, the details of Adaptive Neuro Fuzzy Inference Systems (ANFIS) structure can be found, proposed as a core neuro fuzzy model that can incorporate human expertise as well as adapt itself through repeated learning. Lastly the conclusions are presented.

II. ADAPTIVE NEURO FUZZY INFERENCE SYSTEMS (ANFIS)

Adaptive Neuro Fuzzy Inference System (ANFIS) was first proposed by Jang. ANFIS can be easily implemented for a given input/output task and hence it is attractive for many application purposes. It has been successfully applied in different areas. The first kind of NFS we apply for our data classification problem is the so called Adaptive Neuro-Fuzzy Inference System (ANFIS) model, which hybridizes an ANN and a FIS into a kind of NFS with a homogeneous structure. That is, the ANFIS model integrates the ANN and FIS tools into a 'compound', meaning that there are no boundaries to differentiate the respective features of ANN and FIS [9].

1.1. ANFIS ARCHITECTURE

In order to describe the architecture of an ANFIS, we briefly introduce the first order Sugeno-style FIS at first. A first-order Sugeno-style FIS model is a system that manages the process of mapping from a given crisp input to a crisp output, using fuzzy set theory.

$$If\ \ x_1\ is\ A_1, x_2\ is\ A_2, \ldots\ldots x_n\ is\ A_n$$
$$then\ y = k_0 + k_1x_1 + k_2x_2 + \ldots\ldots + k_nx_n$$

where $x_1, x_2, \ldots \ldots x_n$ are considered as input variables; $A_1, A_2, \ldots \ldots A_n$ are fuzzy sets; and *y* is the output variable, we can find that in such type of fuzzy rule. The output variable is a first-order polynomial on input variables.

1.2. DESCRIPTION OF THE METHOD

An ANFIS is, in essence, an ANN that is functionally equivalent to a first-order Sugeno-style FIS. Typically, there are six layers in an ANFIS model; one input layer, four hidden layers, and one output layer. Each layer performs a particular task to forward the signals. Such an ANFIS model is shown in Figure: 1.

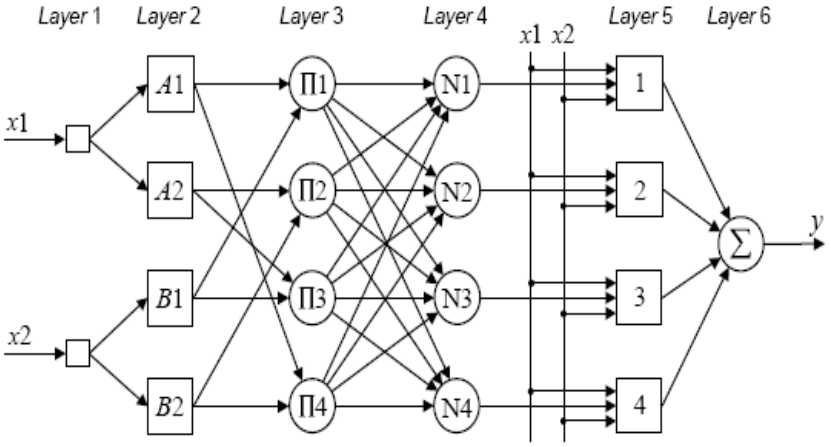

Figure:1 A typical ANFIS model with two inputs and one output.

The first layer, i.e. the input layer of the ANFIS model is the input layer. Neurons in this layer simply transmit the external input (crisp) signals to the next layer. Namely,

$$x_i^1 - y_i^1$$

where x_i^1 is the input signal and y_i^1 is the output signal of neuron i in the first layer.

The second layer, i.e. the first hidden layer of the ANFIS model is fuzzification layer. Neurons in this layer represent antecedent fuzzy sets of fuzzy rules. A fuzzification neuron here receives an input signal and determines the degree to which this signal belongs to the neuron's fuzzy set. If we let x_i^2 be the input and y_i^2 be the output signal of neuron i in the second layer, then we have:

$$y_i^2 = f(x_i^2)$$

where f represents the activation function of neuron i, and is set to a certain membership function.
The third layer, i.e. the second hidden layer is the fuzzy rule layer. Each neuron in this layer corresponds to a single first-order Sugeno fuzzy rule - a rule neuron receives signals only from the fuzzification neurons which are involved in the antecedents of the fuzzy rule it represents and computes the truth value of the rule. In an ANFIS, the 'product' operator is used to evaluate the conjunction of the antecedents. Therefore, we have:

$$y_i^3 = \prod_c^m x^3 c_i$$

where $\mathbf{x^3 c_i}$ is the signal from fuzzification neuron c in the second layer to neuron i in the third layer. $\mathbf{y_i^3}$ is the output signal of neuron i in this layer; and m is the number of antecedents of the fuzzy rule neuron i represents.

The fourth layer, i.e. the third hidden layer is the normalization layer. Each neuron in this layer receives signals from all rule neurons in the third layer and calculates the so-called normalized firing strength of a given rule. This strength value represents the contribution of a given rule to the final result and is obtained as:

$$y_i^4 = \frac{x_d^4}{\Sigma x_d^4}$$

where $\mathbf{x_d^4}$ the signal from rule neuron d in the third is layer to neuron i in the fourth layer; $\mathbf{y_i^4}$ is the output signal of neuron i in this layer; and n is the number of rule neurons in the third layer.

The fifth layer, i.e. the fourth hidden layer is the defuzzification layer. Each neuron in this layer is connected to the respective normalization neuron in the fourth layer and also receives initial input signals $x_1, x_2, \ldots \ldots x_n$. A defuzzification neuron computed the 'weighted consequent value' of a given rule as:

$$y_i^5 = x_i^5(k_{i0} + k_{i1}x_1 + k_{i2}x_2 + \ldots \ldots + k_{in}x_n)$$

where $\mathbf{x_i^5}$ is the input and $\mathbf{y_i^5}$ is the output signal of neuron i in the fifth layer; and $\mathbf{k_{i0}, k_{i1}, k_{i2}, \ldots . k_{in}}$ is a set of consequent parameters of rule i.

The sixth layer, i.e. the output layer is the summation layer. There is only one neuron in the layer, which calculated the sum of outputs of all defuzzification neurons in the fifth layer and consequently produces the overall ANFIS output y as follows:

$$y = \sum_{i=1}^{n} x_i$$

where $\mathbf{x_i}$ is the signal from defuzzification neuron i in the fifth layer to this summation neuron; and n is the number of defuzzification neurons, namely the number of fuzzy rules in the ANFIS model .

1.3. TRAINING ANFIS MODEL

ANFIS can be trained to learn from given data. As we can observe from the ANFIS architecture, in order to configure an ANFIS model for a specific problem, we need to specify the fuzzy rules and the activation functions (i.e. membership functions) of fuzzification neurons. For the fuzzy rules, we can specify the antecedent fuzzy sets once we know the specific problem domain; while for the consequents of the fuzzy rules, the parameters (e.g.$\mathbf{k_{i0}}, \mathbf{k_{i1}}, \mathbf{k_{i2}}, \ldots . \mathbf{k_{in}}$) are formed and adjusted by certain learning algorithm in the training process. On the other hand, the shapes of activation functions can also be formed and adjusted in the training process [1].

For ANFIS models, the most commonly used activation function is the so-called bell-shaped function, described as:

$$y = \frac{1}{1+[\left(x-\frac{s}{r}\right)^2]^t}$$

where r, s and t are parameters that respectively control the slope, center and width of the bell-shaped function. And in the training process, these parameters can be specified and adjusted by the learning algorithm.

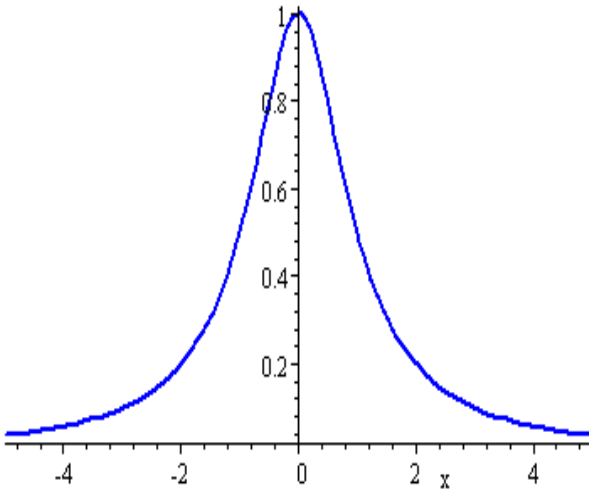

Figure 2 A Bell-Shaped Function with r = t = 1 and s = 0.

Specifically, an ANFIS uses a 'hybrid' learning (training) algorithm. This learning algorithm combines the so-called least-squares estimator and the gradient descent method finally with Runge Kutta Learning Method (RKLM). In the beginning, initial bell-shaped functions with certain parameters are assigned to each fuzzification neuron. The function centre of the neurons connected to input xi are set so that the domain of xi is divided uniformly, and the function widths and slopes are set to allow sufficient overlapping(s) of the respective functions[2].During the training process, the training dataset is presented to the ANFIS cyclically. Each cycle through all the training examples is called an epoch. In the ANFIS learning algorithm, each epoch comprises of a forward pass and a backward pass. The purpose of the forward pass is to form and adjust the consequent parameters, while that of the backward pass is to adjust the parameters of the activation functions.

III. BACK PROPAGATION LEARNING

Based on approach of error correction learning, Back propagation is a systematic method for training, provides a computationally efficient method for changing the synaptic weights in the neural network, with differentiable activation function units. The error back propagation algorithm uses method of supervised learning. We provide the algorithm with the recorded set of observations or training set. i.e. examples of the inputs and the desired outputs that we want the network to compute, and then the error (difference between actual and expected results) is computed.

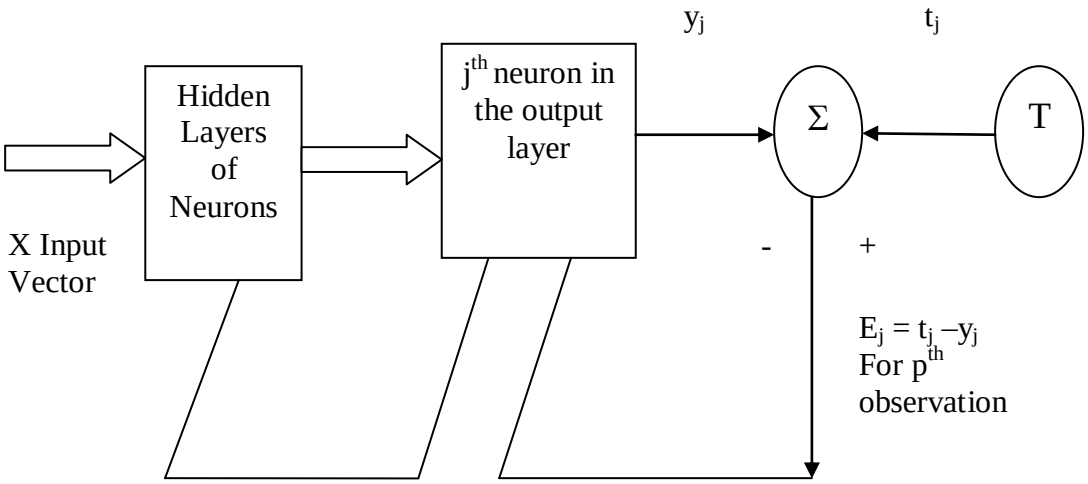

Figure 3. Error Correction Learning In Neural Network

These differences in output are back propagated in the layers of the neural networks and the algorithm adjusts the synaptic weights in between the neurons of successive layers such that overall error energy of the network, E is minimized. The idea of the back propagation algorithm is to reduce this error, until the ANN learns the training data. Training of network i.e. error correction is stopped when the value of the error energy function has become sufficiently small and as desired in the required limits[3].Total error for p^{th} observation of data set and j^{th} neuron in the output layer can be computed as:

$$E_i = t_i - y_i$$

where, t_i represents the desired target output and y_i represents the predicted from the system.

IV. GRADIENT DESCENT LEARNING

Training the neural network involves finding the minimum of a complicated nonlinear function (called' error function"). This function describes the error the neural network makes in approximating or classifying the training data, as a function of the weights of the network w.We want the error to become as small as possible and should thus try to move towards the point of minimum error. For that we are using Gradient descent method. Gradient descent simply means going downhill in small steps until you reach the bottom of error surface. This is the learning technique used in back propagation. The back propagation weight update is equal to the slope of the energy function that is further scaled by a learning rate η, (thus, the steeper the slope, the bigger the update but may cause a slow convergence)[4].

Let us train the model for p number of data observations. In back propagation, for the j^{th} neuron in the output layer, if output,
$t_j \neq y_j$ then for the p^{th} observation, the error signal for the neuron j in output layer can be given as:

$$E^P = \frac{1}{2}\sum_{j \in c} (t_j^p - y_j^p)^2 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

where set C includes all the neurons in the output layer of the network. For p^{th} observation, t_j represents the desired target output and y_i represents actual observed output from the system, for j^{th} neuron in the output layer. We have taken the square of error energy so that errors of opposite signs don't cancel out each other; also we are interested in overall magnitude of the error and not in the sign of error. The total error energy is obtained as $E^p = \sum_p E^p$, for the complete data set.Objective of the learning process is to adjust the ANN parameters (synaptic weights) to minimize the over all error energy. Weights are updated on pattern by pattern basis until complete set of training data is utilized (one epoch) for training of the network. We need to find out a new weight value so that E comes minimum, and also we know that changing the weight value either takes us away from minima or closer to it, so to tune for that weight value, we need to find the direction that will take us to local minima of the curve i.e. opposite to the direction of the gradient of error (GE). Gradient means change of error energy w. r. t. weight value.

$$GE = \partial E / \partial W_{ji}$$

$$GE = \partial E / \partial W_{ji} = \frac{\partial E}{\partial yj} \frac{\partial yj}{\partial w_{ji}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

where, W_{ji} represents the synaptic weight to j^{th} neuron in the output layer from the i^{th} neuron in the previous layer.

From equation. 1, we can write

$$\frac{\partial E}{\partial yj} = -(t_j - y_j) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ldots (3)$$

Also, we know in a neural network:

$$y_j = \sum_i w_{ji} x_i,$$

Therefore,

$$\frac{\partial yj}{\partial W_{ji}} = \frac{\partial}{\partial W_{ji}} \sum_i W_{ji} X_i = X_i \quad \ldots\ldots\ldots\ldots\ldots\ldots . \quad (4)$$

Now, from equations (2), (3) and (4), we can write

$$G = \partial E / \partial W_{ji} = -(t_j - y_j) x_i \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (5)$$

Correction in weight must be opposite to the gradient.

So we can write $\Delta w_{ji} = (t_j - y_j) x_i$.

And thus, updated new value of the synaptic weight can be written as:

$$w_{ji\,new} = w_{ji\,old} + \Delta w_{ji} \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (6)$$

By introducing a new parameter η, eqn. (6) can be rewritten as

$$w_{ji\,new} = w_{ji\,old} + \eta (t_j - y_j) x_i$$

The parameter, η controls the speed at which we do the error correction or decides for the rate at which the network learns, and therefore is known as learning rate parameter. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient at the current point [5].

V. RUNGE-KUTTA LEARNING ALGORITHM (RKLM)

Runge-Kutta Learning Method is the simplest approach to compute the values accurately. It is because, the values are computed based on the prior iteration values. Runge-Kutta method is a powerful way of solving the behavior of a dynamic system if the system is characterized by ordinary differential equations. In [6], the proposed method is applied to several problems and it is observed that the network shows a satisfactory performance in estimating the system states in the long run. It should be emphasized that the neural network architecture realizes the changing rates of the system states instead of the [x (k),τ(□ k)□]→□ [x(k+1)] mapping. Therefore, the RKLM approach alleviates the difficulties introduced by the discretization methods. This paper considers the approach with FNN with on-line tuning of the parameters. For fourth order Runge-Kutta approximation, the overall scheme is comprised of four times repeatedly connected neural network blocks and corresponding stage gains. The update mechanism is based on the error back propagation.

The derivation for FNN based identification scheme is given below.

$$\dot{\underline{x}} = \underline{f}(\underline{x}, \underline{\tau})$$

$$\underline{x}(i+1) = \underline{x}(i) + \frac{h}{6}(\underline{k}_0 + 2\underline{k}_1 + 2\underline{k}_2 + \underline{k}_3)$$

$$\underline{k}_0 = N(\underline{x};\ \varphi) = N(\underline{x}_0;\ \varphi)$$

$$\underline{k}_1 = N\left(\underline{x} + \frac{1}{2} h\underline{k}_0 ;\ \varphi\right) = N(\underline{x}_1;\varphi)$$

$$\underline{k}_2 = N\left(\underline{x} + \frac{1}{2} h\underline{k}_1 ;\ \varphi\right) = N(\underline{x}_2;\ \varphi)$$

$$\underline{k}_3 = N(\underline{x} + h\underline{k}_2;\ \varphi) = N(\underline{x}_3;\ \varphi)$$

where, φ is a generic parameter of neural network. There are two paths to be considered in this propagation. The first is the direct connection to the output summation; the other is through the FNN stages of the architecture. Therefore, each derivation, except the first one, will concern two terms. The rule is summarized below for the fourth order Runge-Kutta approximation.

$k_0 = N(x_0, \varphi)$

$$\frac{\partial k1}{\partial \Phi} = \frac{\partial k1}{\partial x1}\frac{\partial x1}{\partial k0}\frac{\partial k0}{\partial \Phi} + \frac{dk1}{d\varphi}$$

$$\frac{\partial k2}{\partial \Phi} = \frac{\partial k2}{\partial x2}\frac{\partial x2}{\partial k1}\frac{\partial k1}{\partial \Phi} + \frac{dk2}{d\Phi}$$

$$\frac{\partial k3}{\partial \Phi} = \frac{\partial k3}{\partial x3}\frac{\partial x3}{\partial k2}\frac{\partial k2}{\partial \Phi} + \frac{dk3}{d\Phi}$$

$$\Delta \Phi_{(i)} = \frac{\eta h}{6}(d^T(i) - x^T(i))(\partial k_0 + \frac{2\partial k1}{\partial \Phi} + \frac{2\partial k2}{\partial \Phi} + \frac{\partial k3}{\partial \Phi})$$

where η represents the learning rate and $d^T(i)$ represents measured state vector at time t. Thus the final values are calculated from the prior iteration values which help to produce the accurate results.

VI. HYBRID LEARNING USING ANFIS WITH RKLM

The subsequent development of ANFIS approach, a number of methods have been proposed for learning rules and for obtaining an optimal set of rules. In this work, an application of RKLM, which is essential for classifying the behavior of datasets, is used for learning in ANFIS network for Training data.

The output of ANFIS is given as an input to the ANFIS RKLM. The inputs given to ANFIS RKLM are number of membership function, type of membership function for each input, type of membership function output training data & train target and epoch's[7].

VII. EXPERIMENTAL RESULTS

The experiment was carried out with 100 sample data for comparing the neural network learning Algorithms. MATLAB is used in our experiment to evaluate the results.ANFIS is relatively fast to convergence due to its hybrid learning strategy and its easy interpretation. It is a more transparent model and its behavior can be explained in human understandable terms, such as linguistic terms and linguistic rules. This provides for a better understanding of the data and gives the researchers a clear explanation for how the diagnostic results are arrived [8]. The back propagation stands for the position error bound for base position. The gradient method is combined with the least squares method to update the parameters of membership functions in an adaptive inference system. The amount of corrective adjustments applied to synaptic weight is called the learning rate η. Table 1 shows the comparison of experimental results of learning algorithms for training data in the ANFIS architecture. It is clear from the table that the error values obtained using RKLM method is very low (0.091077) when compared to Back Propagation and Gradient Descent.

Table 1: Comparison of Learning Algorithms

S.NO	LEARNING ALGORITHMS	TRAINING ERROR
1	Back Propagation	0.24432
2	Gradient Descent	0.21837
3	Runge-Kutta Learning Method	0.091077

The training error results obtained using ANFIS with Back Propagation, ANFIS with Gradient Descent and ANFIS with Runge-Kutta Learning Method is shown in the figure 4, 5 and 6 respectively.

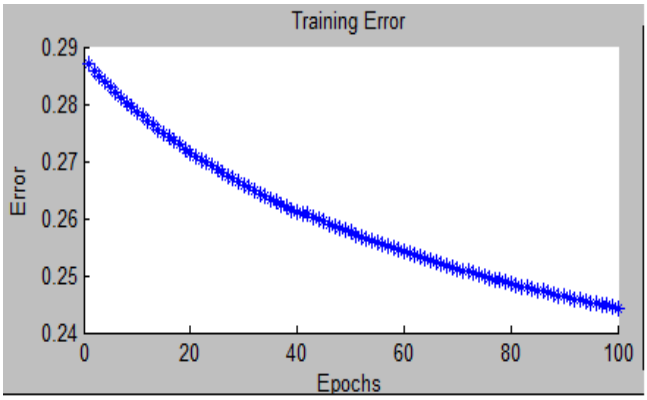

Figure 4. Training Error results obtained using Back Propagation with ANFIS

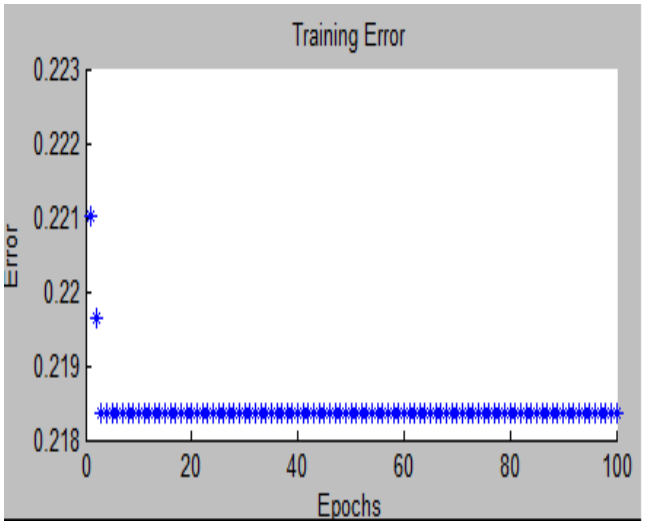

Figure 5. Training Error results obtained using Gradient Descent with ANFIS

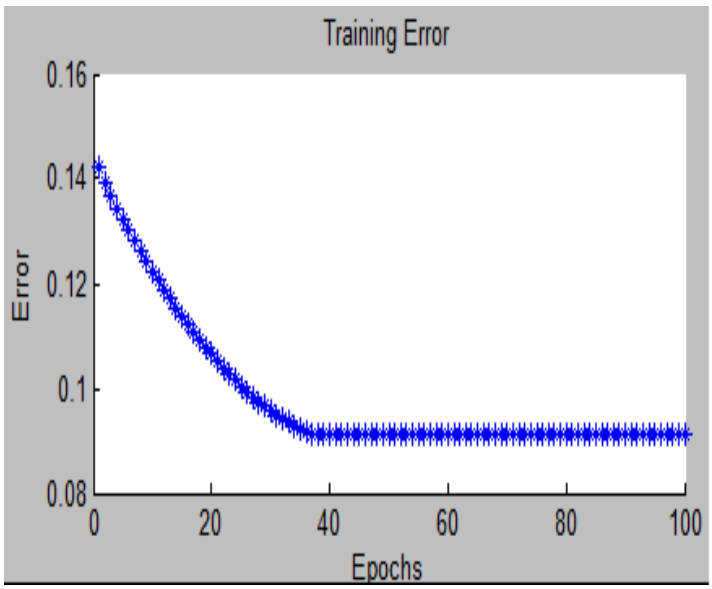

Figure 6. Training Error results obtained using Runge-Kutta Learning Method with ANFIS

Thus the training error results show the use of RKLM with ANFIS architectures result in a good estimation performance.

VIII. CONCLUSION

In this work, the performance of learning algorithms such as back propagation, gradient descent learning algorithm and Runge-Kutta learning algorithm for system identification are analyzed. In this assessment, the performance of training algorithms is considered as the major comparison measures. RKLM is the simplest approach among all the three proposed approaches in the sense of computational complexity. An additional measure of RKLM is the dependency on a priori knowledge of the parameters. Back propagation and gradient descent method requires a pre-training phase whereas RKLM does not require pre-training. The work, reported in this paper, indicates that ANFIS structure is a good candidate for identification purposes, additionally the elegant performance of the RKLM approach with on-line operation and with ordinary feed forward neural network. The estimated results proved that the ANFIS with RKLM provides better results than other two approaches. The work is in progress in the direction of improving the estimation performance through the use of ANFIS with RKLM approach.

REFERENCES

[1] J.S. R. Jang, C.T. Sun and E. Mizutani, "Neuro-Fuzzy and Soft Computing", PTR Prentice Hall, (1997).

[2] Jamal M. Nazzal, Ibrahim M. Elemary and Salam A. Najim, *"*Multilayer Perceptron Neural Network (MLPs) For Analyzing the Properties of Jordan Oil Shale*",* World Applied Sciences Journal 5, Issue 5, (2008), pp. 546 - 552.

[3] Er. Parveen Sehgal, Sangeeta Gupta and Dharminder Kumar, *"*A Study of Prediction Modeling Using Multilayer Perceptron (MLP) With Error Back Propagation*",* Proceedings of AICTE sponsored National Conference on Knowledge Discovery & Network Security: (February 2011), pp. 17-20.

[4] Er. Parveen Sehgal, Sangeeta Gupta and Dharminder Kumar, "Minimization of Error in Training a Neural Network Using Gradient Descent Method", International Journal of Technical Research, Vol. 1,Issue 1, (2012), pp.10-12.

[5] Walter H. Delashmit and Michael T. Manry, *"*Recent Developments in Multilayer Perceptron Neural Networks*",* Proceedings of the 7th Annual Memphis Area Engineering and Science Conference, (2005), pp. 1-3.

[6] Y.J. Wang and C.T. Lin, "Runge-Kutta Neural Network for Identification of Dynamical Systems in High Accuracy", IEEE Transactions on Neural Networks, Vol. 9., No. 2, (1998), pp. 294-307.

[7] Mehmet Onder Efe and Okyay Kaynak, "A Comparative Study of Neural Network Structures in Identification of Non-linear Systems", Mechatronics Research and Application Center, Bogaziçi University, Turkey, (1999)**.**

[8] J.S.R Jang and C.T Sun, "Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence", Prentice-Hall, USA, (1997).

[9] J.S.R. Jang, Adaptive Network Based Fuzzy Inference System", IEEE Trans. On Systems Man. and Cybemetics, Vol.23, No.3, (1993), Pp 665-685.