

Solution to the Outlier Samples Problem in Function Approximation Based on an Adapted Neural Fuzzy Inference System

M. N Raheema¹, A. S Abdullah²

¹Assistant Professor, Ph.D. in Prosthetics & Orthotics Engineering Department, Engineering College, University of Kerbala, Kerbala, Iraq.

²Assistant Lecturer, M.Sc. in College Al Safwa University, Karbala, Iraq.

E-mail: mithaqnama76@yahoo.com

Abstract. This paper describes a novel application of the Adapted Neural Fuzzy Inference System (ANFIS) to function approximation. Several functions in one dimension are realised in this work, including a Gaussian function and a combination of sine waves with exponential functions, in order to confirm the efficiency of the ANFIS method; these results are then compared with those from a Radial Basis Function Neural Network (RBFNN) and Fuzzy Inference System (FIS), which have previously been successfully applied to function approximation problems. This paper introduces the ANFIS as a robust method that mitigates, or is insensitive to, outliers. The results show that the ANFIS method can solve the outlier samples problem, and that the performance of ANFIS proposed in this work is thus better than that of the NN and FIS methods; the function approximated outputs of the presented ANFIS are more faithful to the original test functions and RMSEs of the ANFIS are also lower, especially during the checking process.

Key Words: Function approximation, adapted neural fuzzy inference system.

1. Introduction

The mathematical formula needed required for modeling and control techniques may be difficult to obtain exactly; thus, a fuzzy approach can be used to model and control systems based solely on the behaviours of the unknown system [1]. A great deal of the literature has thus focused on applying fuzzy systems to function approximation, system identification, control, robot, expert systems, and industrial and power systems applications [2, 3]. Fuzzy systems are sets of if-then rules that map an input vector to an output vector. Fuzzy rules thus define fuzzy subsets of the input-output state space, and if these rules are change, the fuzzy system can adapt or learn [4]. Fuzzy systems have been proven to act as universal approximators in many different configurations. They can approximate many real functions to high degree of accuracy [5], with only very small error margins, by adding additional fuzzy rules and fuzzy sets [6].

However, good training of Neural Networks (NNs), which are systems organised similarly to human brains that can develop properties of generalisation and learning capability [7], are always considered first for modeling nonlinear functions, and NNs with sufficient numbers of layers and neurons may



approximate any real (nonlinear) continuous function [1]. An NN is used on the training set to find a mapping between the input vector and the output vector form, and this learning within the NN can be viewed as an approximation problem, if such mapping is seen as a function in an input space.

Much of the literature has been concerned with proving that the traditional Radial Basis Function Neural Network (RBFNN) is a good approximator to many continuous (real) functions [8] in that it can produce an interpolating surface that passes exactly through all the input-output training points. However, for good learning (or approximation) of an unknown nonlinear function, additional memory units may be required in an RBF network, along with an increase in the number of training data points [8]. However, an RBFNN structure with a fixed number of hidden neurons suffers from a lack of flexibility and cannot be adjusted effectively in line with dynamic behaviour of the model. Furthermore, outlier data of the complex function may affect the estimation method used to derive the weight values, thus reducing the approximation performance of RBFNN [9].

Previous literature has shown that both NN and Fuzzy methods can be applied to solve similar tasks such as system identification, and control and function approximation. It is also clear that Neural and Fuzzy networks are complementary on several levels: (1) Either can be used to enhance performance of the other; and (2) They deal with different types of knowledge representation, and provide different inferences for any given accuracy and fault tolerance. [10]. Thus, many works combine neural networks with Fuzzy set theory to mitigate any disadvantages such as slow convergence and large number of iterations of NN [1]. Fuzzy methods can be used to control the learning rate of NNs and to generate the initial weights of neurons; similarly, the efficiency of Fuzzy methods can be improved by using the learning ability of NN to tune both the weights of defuzzification and the parameters of membership functions [1]. NNs also can be used to change Fuzzy sets according to training set or errors.

In this paper, ANFIS is used as a hybrid neural-fuzzy system to handle function approximation problems, taking advantage of these complementary intelligent methods. While Fuzzy methods are used to perform inference, NN supports learning and classification. Section 2 thus provides the function approximation problem and fundamentals of the RBFNN, Fuzzy, and ANFIS methods. In Section 3, the proposed ANFIS configuration, training data, and learning processed are provided. In Section 4, function examples and simulation results are presented. The conclusions are given in Section 5, and, finally, suggestions for future work are given in Section 6.

2. Function Approximation Problem and Methods

Assume several pairs of input points and output values of an unknown function; the problem is the need to ascertain the functional relational between these input and output values. If the mathematical description of this relationship is not required and this need can be satisfied with a software or hardware realisation of the function, then neural or Fuzzy networks can be used to approximate the unknown function [11, 12]. The purpose of function approximation is to identify a mapping from a given set of input points $\{x(n); 1 \leq n \leq N\}$ to the values of an unknown function evaluated on those points $\{y(n) = F(x(n)); 1 \leq n \leq N\}$, for the available set of training samples $\{x, y\}$, where N is the total number of available points of the tested function. An approximation of this function $\{\hat{F}(x)\}$ is required such that the sum of squares approximation error, E , for these sets of training samples is minimized. This error value, E , is given by equation (1) [13]:

$$E = \sum_{n=1}^N [y(n) - \hat{F}(x(n))]^2 \quad (1)$$

2.1. Radial Basis Function

The function approximated by RBFNN can be represented as a combination of many radial basis functions that are linearly weighted and have different scaling and translations, as seen in equation (2) [13]:

$$\hat{F}(x) = \sum_{j=1}^J \omega_j \varphi \left(\frac{\|x - m_j\|}{\sigma_j} \right) \quad (2)$$

where φ is the radial basis functions, ω_j is the output layer weights, x is the inputs to the network, m_j is the centres associated with the basis functions, J is the number of basis functions in the network, $\|\cdot\|$ denotes the Euclidean norm, and σ_j is the width vector of the j neuron.

During the supervised training of RBF, the centres, widths, and weights are adapted until the neuron combination approximates the given unknown function and the desired input-output mapping is achieved [14]. RBF has two layers: the first (hidden) layer has as many radial basis neurons as there are in the input vector. The N dimension input vector is thus passed to all K neurons in first layer. The output of each neuron in first layer is calculated as shown in equation (3):

$$h_k = \exp \left\{ - \sum_{n=1}^N [(x_n - c_{kn})^2 / \sigma_{kn}^2] \right\} \quad \text{for } k = 1, \dots, K \quad (3)$$

where x_n , c_{kn} , and σ_{kn} represent the n th element of the input vector, centre vector, and width vector of the k th neuron in first layer, respectively. There is a radial symmetry with respect to the centre seen in the bell-shaped graph that represents the function of each hidden neuron, which can be described by its width and its centre. It is from this symmetry that the name Radial Basis Function derives [14]. Figure 1 shows the general structure of RBFNN [15].

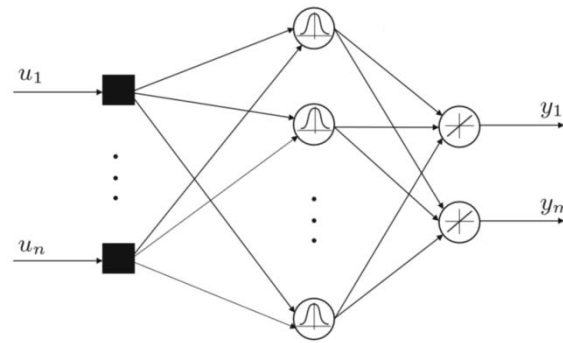


Figure 1. General structure of RBFNN [15].

In the first layer (hidden) the weighted input of each neuron is the distance between the input and its weight value. Each net input is calculated based on the product of its weighted input and its bias value. Then, the output is determined by applying the radial basis function of this net input [16]. The second layer has many pure linear neurons as target vectors, and its weights are set to the target vector, using the dot products to calculate weighted inputs and net summation to find net inputs. Both layers of RBF have biases [16]. The output for each of the M neurons in second layer is calculated using the weighted sum of the first layer outputs, as seen in equation (4):

$$y_m = \sum_{k=1}^K w_{mk} h_k \quad \text{for } m = 1, \dots, M \quad (4)$$

where w_{mk} represents the weight between the k th hidden node and the m th output node [14]. While other NNs have fixed numbers of nodes, RBFNN can add new neurons to the hidden layer of the network until the error goal is specified, based on the nature of the given problem and its complexity [16].

2.2. Fuzzy

Fuzzy Logic (FL) was first presented by Lotfi Zadeh in 1965, and it is based on the IF-THEN rule that integrates an input-output mapping to a system based on the representation of linguistic variables such

as negative or positive, which are further represented by the Membership Function (MF). There are many types of MFs, including Triangular, Gaussian, and Trapezoidal. A combination of FL and Fuzzy Set (FS) theory produces the Fuzzy Logic Controller (FLC), and Fuzzy Inference (FI) is the process by which the input output mapping is formulated using FL. The most commonly used types of FI are Mamdani and Sugeno. In Mamdani, the Fuzzy set is consequent to the rule, while in Sugeno, the rule consequent is an input variables function. Any Fuzzy controller is constructed of four basic decision parameters: input, fuzzification, rule base, and defuzzification [17, 18].

2.3. ANFIS

The combination of the two intelligent systems, NN and FLS, forms the ANFIS. This combination of FLS and NN gives the advantages of both technologies, and this combination can be categorised into three models: concurrent, cooperative, and fully fused [17]. In a concurrent model, the NN continuously aids the FLS by providing all required parameters, particularly when it is not possible to measure the controller input variables directly. In the cooperative model, the NN determines the Membership Functions (MFs) from its training data, while the rules are formed using fuzzy clustering [17]. Finally, a fused Neuro-Fuzzy (NF) network shares structure information and knowledge representations, and the learning algorithm applied to the Fuzzy system is thus interpreted into the NN architecture. This model comes in many types, including ANFIS, Fuzzy Adaptive Learning Control Network (FALCON), and Neuro-Fuzzy Control (NEFCON) [17]. Typically, the ANFIS mechanism has two inputs and one output based on the common IF-THEN Sugeno rule [19]. Equation (5) is an example:

$$\text{IF } x_a \text{ is } A_i \text{ AND } x_b \text{ is } B_i \text{ THEN } R_j = k_{0j} + k_{1j}x_a + k_{2j}x_b \quad (5)$$

where: A_i and B_i are Fuzzy sets, x_a and x_b are input variables, R_j is the output, j is the rules number, and k is the consequent parameter. The ANFIS architecture has five basic function layers, as shown in Figure 2:

Layer 1 (Fuzzification layer): This layer generates the MFs; equation (6) represents the Gaussian MF (used as an example):

$$O_i^1 = \mu_{A_i}(x) = \exp\left(-\frac{\|x - d_i\|^2}{\sigma_i^2}\right) \quad (6)$$

where O_i is the layer1 output, x is the input to the i th node, μ is the membership function, A_i is a linguistic label associated with this node, and σ_i and d_i denote the width and the centre of the Gaussian function, respectively.

Layer 2 (Rules layer): The product of all inputs of layer 2, known as the firing strength, can be calculated using equation (7):

$$O_i^2 = w_i = \mu_{A_i}(x) \times \mu_{B_i}(y) \quad (7)$$

where w_i is the output rules of layer 2.

Layer 3 (Normalization layer): This layer normalizes the MFs, as shown in equation (8):

$$O_i^3 = \bar{w}_i = \frac{w_i}{w_1 + w_2} \quad (8)$$

Layer 4 (Defuzzification layer): This calculates the rules-weighted consequent parameters using equation (9):

$$O_i^4 = \bar{w}_i f_i = \bar{w}_i (a_i x + b_i y + c_i) \quad (9)$$

where $\bar{w}_i$ represents the output of layer 3 and $\{a_i, b_i, c_i\}$ are the parameters of the MFs.

Layer 5 (Summation layer): This calculates the overall output of the ANFIS, as shown in equation (10):

$$O_i^5 = f = \sum_i \bar{w}_i f_i = \frac{w_1 f_1 + w_2 f_2}{w_1 + w_2} \quad (10)$$

where f is the output of layer 5

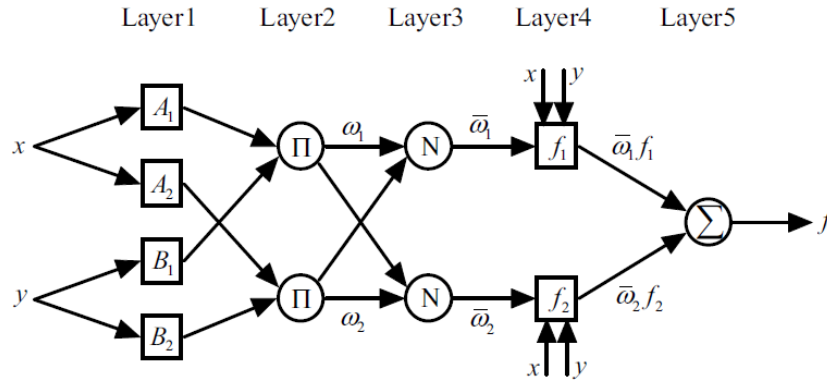


Figure 2. ANFIS Architecture [19]

Modeling of systems may be difficult where the available number of points in the training data set is small. The training data may also contain samples with low noise or with high distance from the range of other points, which are called outlier samples. This work focuses on the outliers' effects on the ANFIS.

3. Computer Simulations

This work uses MATLAB 20014a to simulate an approximation of three different functions. The performance of the presented ANFIS for this approximation is then compared with NN and FIS results. In each function test, the training data set consists of 100 input points over an input range [0,10], with corresponding values existing as target values evaluated from the underlying true function. The outliers set of target values was obtained by adding random values from a uniform distribution range in [-1.5, 1.5]. The simulation work also contains test examples where the outlier noise is increased. Root mean square error (RMSE) is used to evaluate the results, as seen in equation (11):

$$RMSE = \left[\sum_{n=1}^N (y(n) - (f(n))^2 / N \right]^{1/2} \quad (11)$$

Three different data sets were used to implement the approximation of the three functions: Points from a simple function were used as a data set for the first approximation problem, as in equation (12):

$$f(x) = \sin(ax) * e^{bx} \quad (12)$$

over an input range [0,10]. Where a and b are constants, the outliers set of the target values is obtained using equation (13):

$$f_o(x) = f(x) + 3 * (\text{rand} - 0.5) \quad (13)$$

where rand is a function that returns a random real value between 0 and 1.

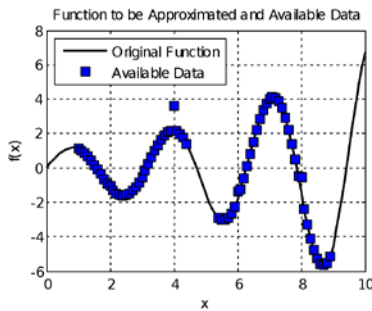
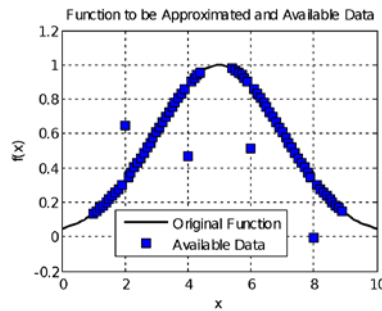
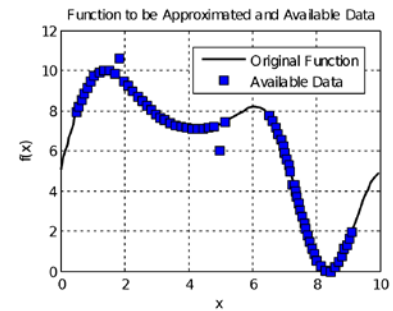
The standard Gaussian function in equation (14) is used in the second approximation problem:

$$f(x) = \text{gaussmf}(x, [a, b]) \quad (14)$$

over an input range [0,10]. The outliers set of the target values is obtained using equation (15):

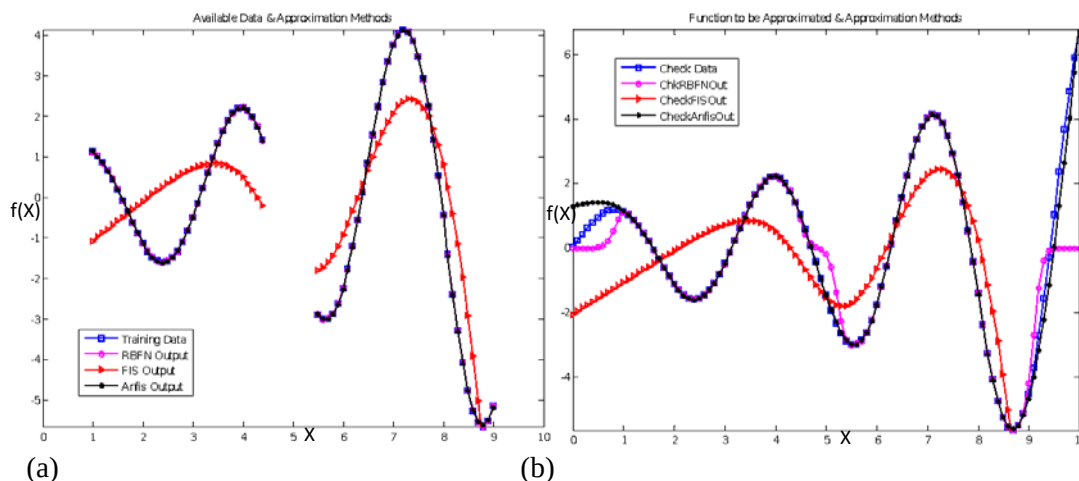
$$f_o(x) = f(x) + (\text{rand} - 0.5) \quad (15)$$

The third data set is generated from a `simplefit_dataset` performed in MATLAB over an input range [0,10] for a real operation problem. The outlier values were obtained using equation (13). To approximate these three functions, 100 training data points were used, including some outliers, as shown in Figures 3 to 5, where the solid line represents the original function and the square notation represents training data.

**Figure 3.** First Test Function.**Figure 4.** Second Test Function.**Figure 5.** Third Test Function.

4. Results

For the training process, 70% of the data set was used, while all 100 points in the testing process were used to reconstruct the function. After sufficient iterations of the learning epochs, the value of RMSE, representing objective function error, was below 10⁻² in most test cases. The RMSE value determined for the approximation has an effect on the performance of the methods used. Figures 6 to 8 show the output signals in the training and checking process for the training data without outlier samples, as obtained by RBFNN, FIS, and the proposed ANFIS methods; and Table 1 presents their RMSEs. As Figures 6 to 8 show, the performance of the ANFIS proposed in this work is better than that of the NN and FIS methods, and the function approximated outputs of the presented ANFIS are more exact matches with the original test functions, especially in terms of the checking process. Table 1 confirms these results.



(a)

(b)

Figure 6. Output signal for the first test function in (a) training process and (b) checking process.

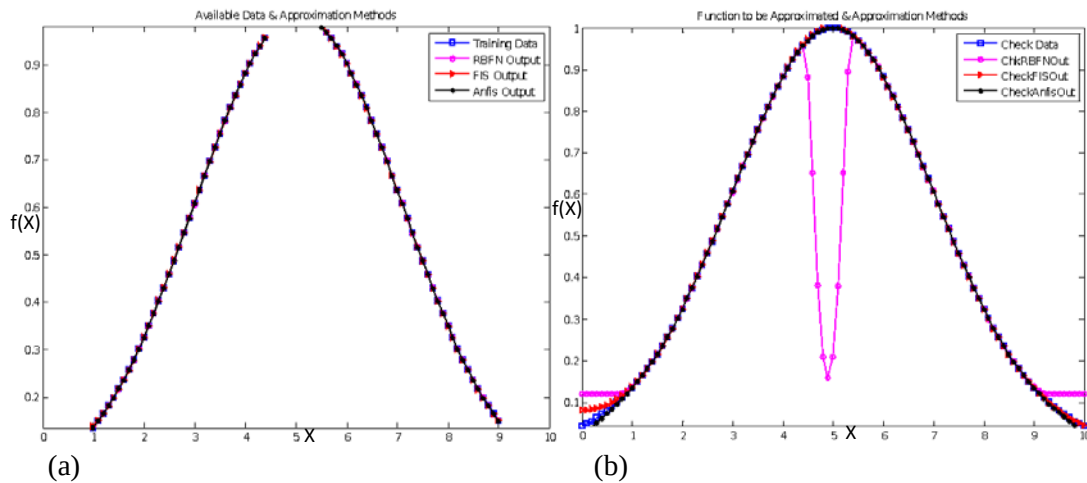


Figure 7. Output signal for the second test function in (a) training process and (b) checking process.

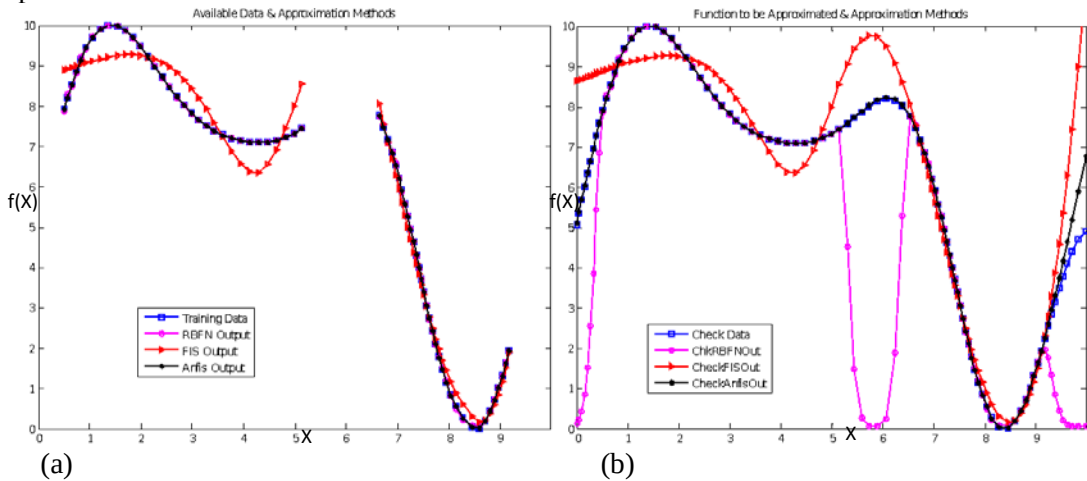


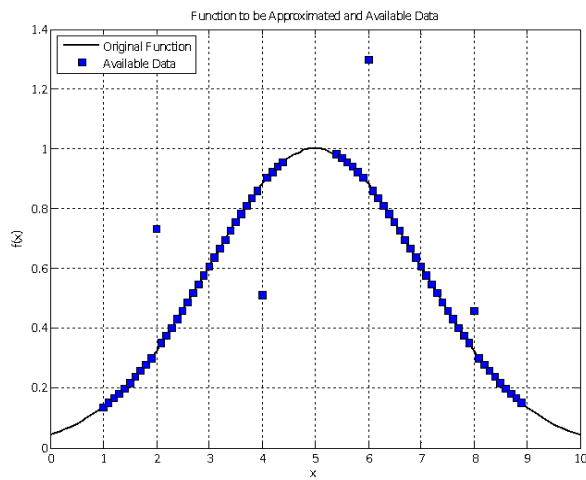
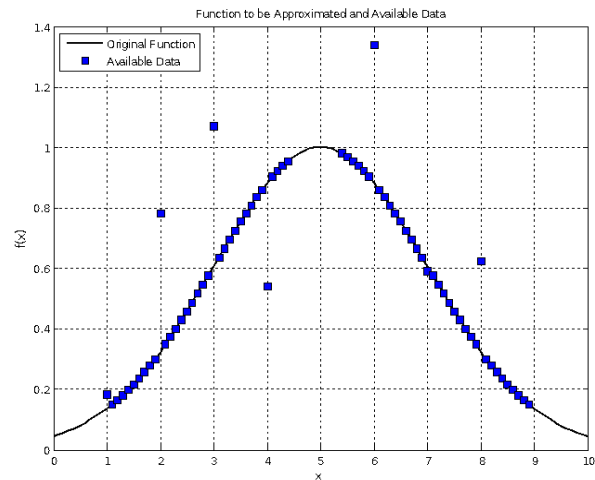
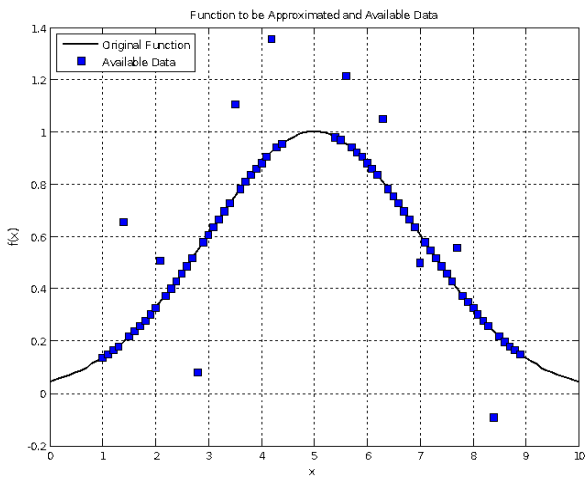
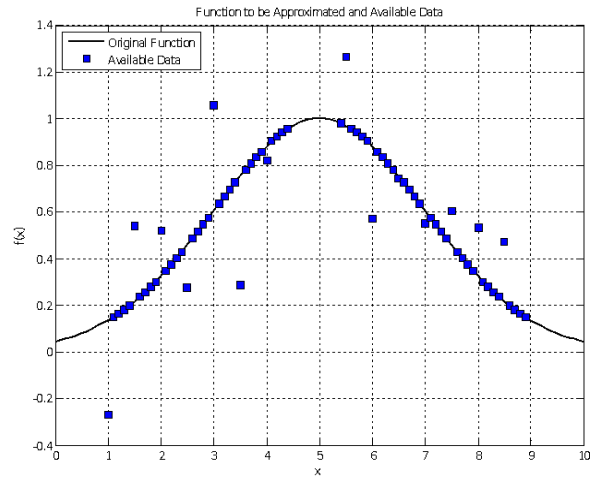
Figure 8. Output signal for the third test function in (a) training process and (b) checking process.

Table 1. RMSEs for test functions without outliers

Example Function	Training			Checking		
	RBFNN	Fuzzy	ANFIS	RBFNN	Fuzzy	ANFIS
Function 1	0.00833	1.45791	0.00972	1.11630	6.63749	0.31789
Function 2	0.00167	0.00066	0.00013	0.15542	0.00623	0.00393
Function 3	0.00232	0.04792	0.02496	4.08864	0.13285	0.05020

From Table 1, in many simulation cases, the RBFNN can be used to obtain lower RMSEs on training processes than in the other methods used. However, it does not perform as well on testing processes. For the second test function, although all the methods perform well in the training case, the ANFIS method has the best performance based on its lower value of RMSE.

More than 20 independent runs on Gaussian function were used as an example, with different numbers of outlier samples being tested. The proportions of outlier samples used were 5%, 10%, 15%, and 20%, with data taken from the training data set as shown in Figures 9 to 12. The relevant RMSE values are presented in Table 2 and the output signals are shown in Figures 13 to 16. These results show that ANFIS is less sensitive to the outlier samples than NN and FIS methods. The RMSEs of ANFIS are also lower, as shown in Table 2, and the output signals are closer to the tested original functions, as clearly seen in Figures 13 to 16.

**Figure 9.** 5% outlier samples.**Figure 10 .** 10% outlier samples.**Figure 11.** 15% outlier samples.**Figure 12.** 20% outlier samples.**Table 2.** RMSE results of outliers test for Gaussian function.

Outliers	RMSE in Training			RMSE in Checking		
	RBFNN	Fuzzy	ANFIS	RBFNN	Fuzzy	ANFIS
0	0.00167	0.00066	0.00013	0.15542	0.00623	0.00393
5%	0.04658	0.08326	0.07655	0.20845	0.05226	0.06369
10%	0.04712	0.08341	0.07923	0.18124	0.10870	0.10539
15%	0.05199	0.09977	0.08968	0.21279	0.06081	0.05430
20%	0.07021	0.11520	0.10910	0.25655	0.18535	0.07683

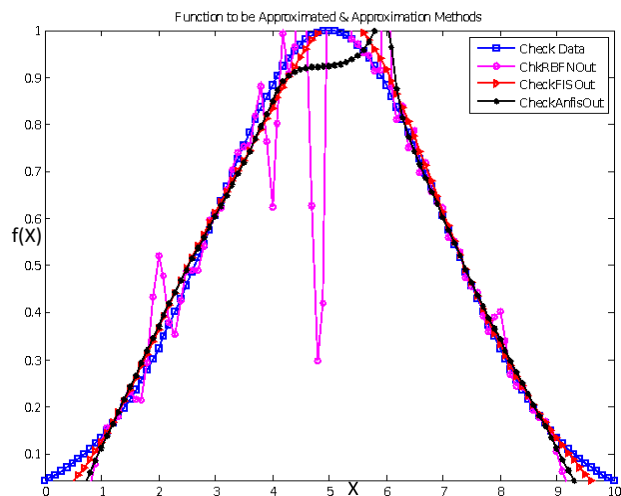


Figure 13. Output signals for 5% outliers test.

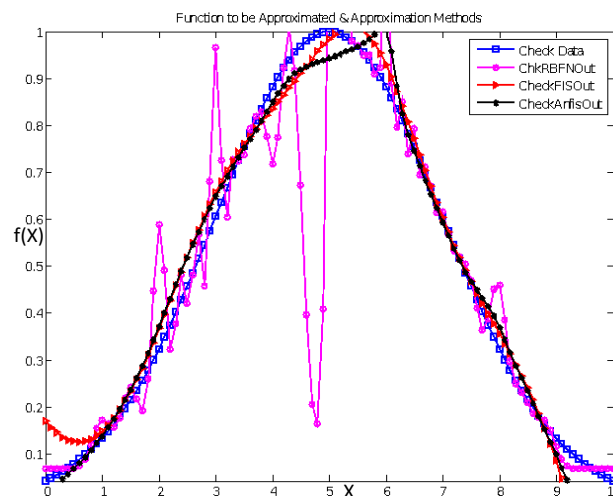


Figure 14. Output signals for 10% outliers test.

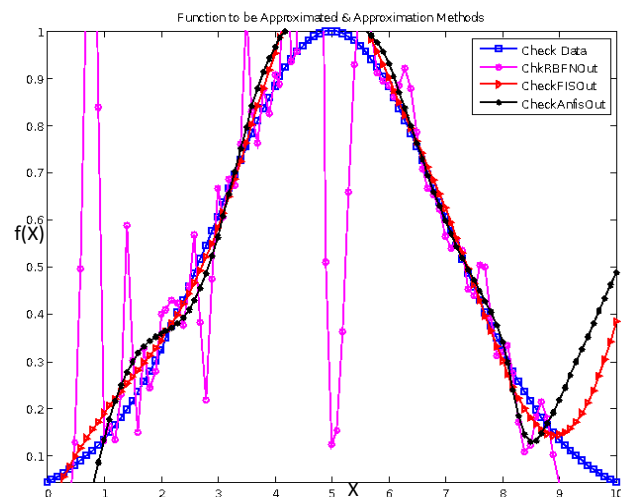


Figure 15. Output signals for 15% outliers test.

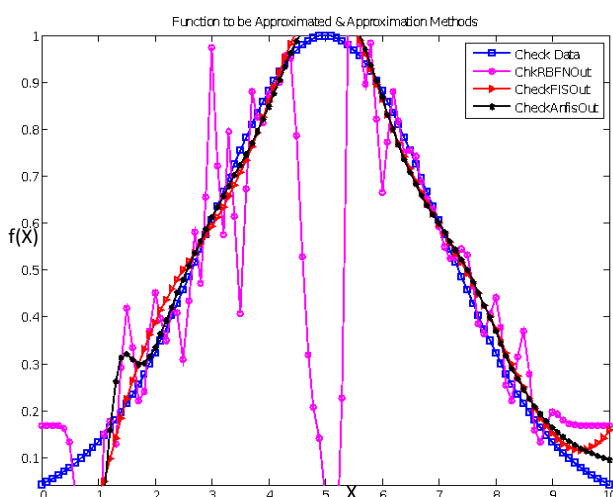


Figure 16. Output signals for 20% outliers test.

5. Conclusion

In this work, the ANFIS was used as a novel method to realise function approximation with appropriate membership function and good learning. The simulation results show that the proposed ANFIS is more efficient than other methods used previously for realising complex function approximation due to its ability to reduce the effects of outlier samples; this conclusion is based on testing with several experimental functions. The convergence of the proposed ANFIS is faster, reducing the required number of learning epochs. It is also clear that the RMSE values in the checking process are much larger than in the training process.

6. Suggestions for Future Work

To improve the results, the training could be repeated after reinitialising the network. In addition, the number of hidden layers and neurons could be increased, different membership functions tested, or the training data set expanded. The ANFIS demonstrates the ability to develop a more accurate approximation for many nonlinear functions than either lone NN or FIS by combining both networks' advantages. This makes it possible to use the ANFIS for modeling and controlling a wide variety of nonlinear complex systems.

7. References

- [1] Wei-Yen Wang, Tsu-Tian Lee, Ching-Lang Liu, and Chi-Hsu Wang, August 1997 Function approximation using fuzzy neural networks with robust learning algorithm *IEEE transactions on systems, man, and cybernetics, part B: cybernetics*, **27**, pp. 740-47.
- [2] Chi-Hsu Wang, Wei-Yen Wang, Tsu-Tian Lee, and Pao-Shun Tseng, May 1995 Fuzzy B-spline membership function (BMF) and its applications in fuzzy-neural control *IEEE transactions on systems, man, and cybernetics*, **25**, pp. 841-51.
- [3] Wang L 1992 Fuzzy systems are universal approximators *Proceeding of Second IEEE International Conference on Fuzzy Systems* San Diego, CA, USA, pp 1163-70.
- [4] Dickerson J and Kosko B 1996 Fuzzy function approximation with ellipsoidal rules *IEEE transactions on systems, man, and cybernetics, part B: cybernetics*, **26**, pp 542-60.
- [5] Lin Y, Zhou X, Gu S, and Wang S, 2012 The takagi-sugeno intuitionist fuzzy systems are universal approximators *Second International Conference on Consumer Electronics, Communications and Networks* pp 2214-17.
- [6] Ying H, Ding Y, Li S, and Shao S Comparison of necessary conditions for typical takagi-sugeno and mamdani fuzzy systems as universal approximators *IEEE transactions on systems, man, and cybernetics, part A: systems and humans* **29**, pp. 508-14.
- [7] Cabrera L, Elbuluk M, and Zinger D 1997 Learning techniques to train neural networks as a state selector for inverter-fed induction machines using direct torque control *IEEE transactions on power electronics*, **12**, pp. 788-99.
- [8] Tao K, 1995 Extended radial basis function (ERBF) networks-linear extension and connections *Integrated Systems, Inc. Asilomar Conference on Signals, Systems and Computers* pp 907-11.
- [9] Sun T, Tsai S, Tsai C, Huo C, and Liu C 2008 Nonlinear function approximation based on least wilcoxontakagi-sugeno fuzzy model *8th International Conference on Intelligent Systems Design and Applications*, pp 312-17.
- [10] Kasabov N, 1998 Foundations of neural networks, fuzzy systems, and knowledge engineering (Massachusetts Institute of Technology), 2nd printing.
- [11] L. P. J. Veenlenturf, 1995, "Analysis and applications of artificial neural network", (Prentice Hall International UK Ltd.).
- [12] Raheema M and Abdullah A 2016 Function approximation using neural and fuzzy methods *Communications on Applied Electronics (CAE), Foundation of Computer Science FCS*, **6**, pp 35-42.
- [13] Hu Y and Hwang J 2002 Handbook of neural network signal processing (CRC Press).
- [14] Jain L and. Martin N 1998 Fusion of neural networks, fuzzy systems and genetic algorithms: industrial applications (CRC Press).
- [15] Patan K 2008 Artificial neural networks for the modelling and fault diagnosis of technical processes (Springer-Verlag Berlin Heidelberg).
- [16] Beale M, Hagan M and Demuth H 2014 Neural network toolbox™ User's Guide R2014a", (MathWorks, Inc.).
- [17] Sumathi S and Surekha P 2010 Computational intelligence paradigms: theory and applications using MATLAB (CRC Press).
- [18] Jantzen J, 2007 Foundations of fuzzy control (John Wiley and Sons, Ltd.).
- [19] Ablameyko S, Goras L, Gori M, and Piuri V 2003 Neural networks for instrumentation, measurement and related industrial applications (IOS Press).