SSRG International Journal of Electrical and Electronics Engineering
ISSN: 2348-8379/ https://doi.org/10.14445/23488379/IJEEE-V11I2P101

Volume 11 Issue 2, 1-10, February 2024

Original Article

Optimized Fuzzy Model in Piecewise Interval for Function Approximation

Anup Kumar Mallick[1], Sumantra Chakraborty[2], Kabita Purkait[3], Angsuman Sarkar[4]

[1,3,4]*Department of Electronics & Communication Engineering, Kalyani Government Engineering College, West Bengal, India.*
[2]*Department of Electronics & Telecommunication Engineering, Gaighata Government Polytechnic, West Bengal, India.*

[1]*Corresponding Author : anup.mallick@kgec.edu.in*

Received: 10 November 2023 Revised: 29 November 2023 Accepted: 11 January 2024 Published: 16 February 2024

Abstract *- Function approximation is a technique for estimating an unknown underlying function from input-output instances or examples. Researchers have proposed different methods of function approximations, such as the neural network method, the support vector regression method, the reinforced learning method, the clustering method, the neuro-fuzzy method, etc. This paper introduces a novel data-driven function approximation scheme where the input-output data set is first segmented into multiple pieces. A Mamdani-type fuzzy submodel is constructed for each piece or portion, and the membership functions' parameters for antecedent and consequent are optimally selected through the differential evolution algorithm. The efficacy of the suggested model is verified on three nonlinear functions, viz., a piecewise polynomial function, an exponentially decreasing sinusoidal function, and an exponentially increasing sinusoidal function. A comparative analysis is done based on the simulation results from the proposed model and the results obtained through the two state-of-the-art function approximation techniques, viz., the support vector regression model and the radial basis function network. The simulation results show that the proposed function approximator has satisfactorily approximated the three functions examined here, surpasses the two state-of-the-art techniques in approximating the two sinusoidal functions, and performs the near-best performance for the piecewise polynomial function. The proposed function approximator is expected to be applied as a new state-of-the-art method for function approximation.*

Keywords *- Differential evolution, Function approximation techniques, Membership function generation, Optimal fuzzy model, Piecewise function.*

1. Introduction

Function approximation reveals the underlying relationship between input and output variables in a given data set [1]. Function approximation may also be considered a mapping from the examples of input to the examples of output. The intention here is to find a relationship between the input and output. When the relation is approximated using some function, it is called a function approximation. The fitness of a function approximation technique for a given data set (X, Y) is estimated by the error function. The most frequently employed error function is given by Equation (1).

$$E = \frac{1}{2}\sum_{i=1}^{n}(y(i) - f(i))^2 \quad (1)$$

Where, E denotes the cost function, n represents the data point size, and f is the approximated function.

The function approximation techniques target to minimize this error function to enhance accuracy in estimation. There are multiple approaches to function approximation, such as the polynomial approximation approach [2-4], the artificial neural network-based approach [5], the support vector regression approach [6, 7], the reinforced learning approach [8], the clustering approach [9, 10], etc.

The polynomial function approximation is one of the direct and most straightforward models for function approximations. In polynomial function approximation, a polynomial of a certain degree is considered, and the polynomial coefficients are selected to minimize the error in the approximation. The higher the degree of the polynomial, the better the approximation accuracy.

However, complexity and computing performance are sacrificed to achieve accuracy. Therefore, efforts are made to obtain roughly the same performance with a polynomial of a lesser degree. Different polynomials, such as Chebyshev polynomials [11, 12], Weierstrass polynomials [13], and Bernstein polynomials [14], have been used for function

approximation. A chronology of function approximation using polynomial methods has been outlined by Trefethen [15]. Selecting the polynomial, determining the order of the polynomial, and choosing the coefficients of the polynomial are crucial tasks in function approximation, as they greatly influence the performance of the approximators.

Another widely applied method in Machine Learning for function approximation is Artificial Neural Network-based approximation. Cybenko [16], and Hornik et al. [17] believed that a 3-layer neural network can accurately estimate nonlinear functions. Researchers have used various artificial neural networks for function approximation.

Ferrari and Stengel [18] presented an algebraic approach for smooth function representation using a feed-forward neural network. Yang et al. [5] investigated the performance of Radial Basis Function Network (RBFN), backpropagation, and regression neural network for approximating Sphere, Rastrigin, and Griewank functions. Zainuddian and Pauline employed RBFN and wavelet neural networks to compare continuous functions. DeVore et al. have presented a detailed survey of different neural networks for approximation [19].

The significant drawback of the Artificial Neural Network-based method is its requirement for a considerable number of neurons in the hidden layer. A neural network requires many hidden neurons to approximate a function properly. As hidden neurons increase, memory and computational time requirements increase.

Another method of function approximation is Support Vector Regression (SVR). SVR is a Support Vector Machine (SVM) extension for regression. The data is transferred into a higher-dimensional space called the kernel space to obtain greater accuracy with nonlinear functions. Different kernels are used in SVR, such as linear, Gaussian, polynomial, etc. Although not as popular as SVM, SVR has also been proven effective in function approximation [6].

Fernando et al. [20] proposed a Multi-dimensional Support Vector Regression (MSVR). It employs a cost function with a hyper-spherical insensitive zone and can perform better than an SVM used separately for each feature. This paper uses an iterative process to the Karush-Kuhn-Tucker criteria to resolve the MSVR. Chuang et al. [7] have recommended a robust SVR network for function approximation, including outliers. Another robust SVR model is suggested in [21], where the rough set is used to tackle imprecise information in the support vector regression model.

Lin et al. [22] introduced a hybrid model, i.e., Support Vector Regression-based Fuzzy Neural Network (SVRFNN), to integrate the reasoning efficiency of Fuzzy Neural Network (FNN) with high accuracy and robustness of SVR for function approximation. One of the significant problems in SVR lies in selecting an appropriate kernel, as no single kernel is best suited for all types of nonlinear functions.

Researchers have also used the clustering technique for function approximation. Clustering is an unsupervised learning tool that segregates data elements into different categories. Hence, modifications to conventional clustering techniques, namely the Alternative Cluster Estimation (ACE) algorithm, have been proposed in [9, 10].

The clustering technique has also been coupled with other function approximation techniques, such as enhanced clustering function approximation for RBFN, which is proposed in [23]. However, no fixed or standard rule exists to select the number of clusters that best approximate the function. Some other notable methods employed for function approximation are the gradient boosting method [24], reinforced learning method [8], neuro-fuzzy method [25], etc.

From the previous discussion, it appears that the performance of different methods for function approximations is influenced by the parameter selection, the architecture, or the type of approximator used, such as the order of polynomials in polynomial function approximation, the kind or architecture, and number of hidden layers for an artificial neural network, the types of kernel functions in support vector regression, and the number of clusters in alternative cluster estimation.

Thus, expert knowledge is required while selecting the parameters, architecture, or types of existing methods; otherwise, the approximators may fail to approximate the given function properly. With those flaws, this paper proposes a novel function approximation model, discussed in the next section.

The proposed work aims to present a new technique of function approximation in which less or no prior or expert knowledge is required. In the proposed model for function approximation, the envelope of the given data set function is divided into multiple segments. For each segment, a Mamdani-type fuzzy model is designed.

The membership functions' parameters of the fuzzy models are optimally selected by using the differential evolutional algorithm. To check the efficacy of the proposed model, the proposed algorithm is applied to approximate three nonlinear functions. It is compared with two well-known function approximation methods: radial basis function network and support vector regression. The remainder of this paper is arranged as given.

Section 2 illustrates the proposed method. The simulation results are reported and discussed in sections 3 and 4. Finally, the paper concludes with a future work scope in section 5.

2. Proposed Method

The proposed fuzzy model for function approximation is developed broadly in two stages: dividing the data set or function envelope into multiple pieces and generating a fuzzy submodel for each piece. Figure 1 presents the proposed framework. A detailed description of the proposed method is illustrated in the following subsections.

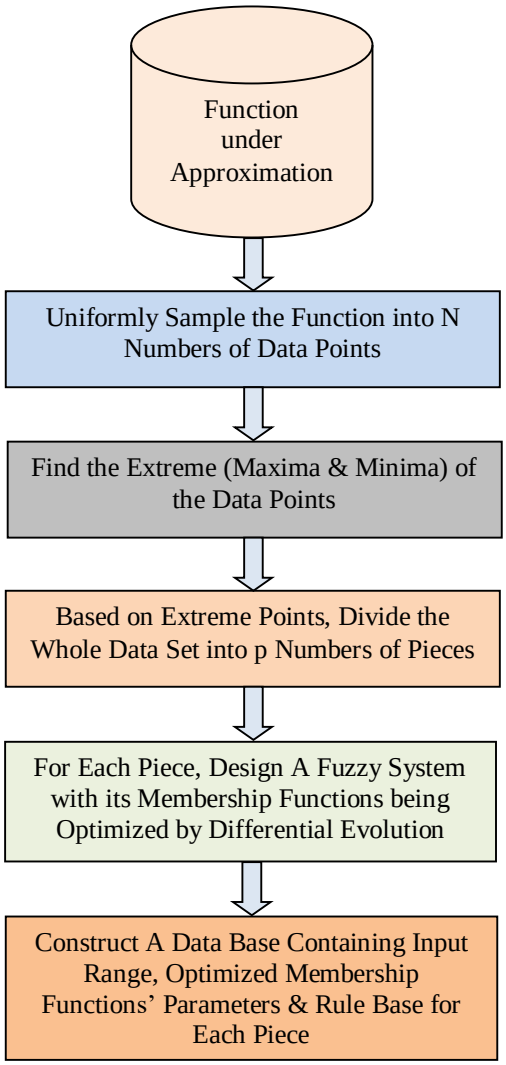


Fig. 1 Proposed framework

2.1. Division of Data Set Envelope into Pieces

At first, the function considered for approximation is uniformly sampled into N number of discrete points to generate a data set of discrete points. Then, the data set is divided into multiple pieces. To find the ranges of different pieces or portions, the extreme points, i.e., the maxima and the minima points, are first estimated. The following two conditions are used to find the extreme points of the data set.

Condition 1: A point of x, say x_i, is one of the maxima points, if

$$y_i > y_{i-1} \quad \text{and} \quad y_i > y_{i+1} \quad \text{for} \quad i \in [in, f] \tag{2}$$

Condition 2: Similarly, x_i is one of the minima points, if

$$y_i < y_{i-1} \quad \text{and} \quad y_i < y_{i+1} \quad \text{for} \quad i \in [in, f] \tag{3}$$

Here, y_i is the corresponding output of x_i. For illustration of this step, one example data set (X, Y) is considered in Figure 2.

For the data set shown in Figure 2, assume that y is defined in the discrete points: in, $in+1$, $in+2, \ldots, f$. Using the above two conditions given in Equation (2) and Equation (3), the extreme points of the data set envelope are first found.

Let us assume that the maxima points of the data set given in Figure 2 are denoted by (x_a, y_a), (x_c, y_c), and (x_e, y_e), and the minima points are represented by (x_b, y_b), (x_d, y_d). Then, the given data set is segmented into six piecewise intervals denoted by P1, P2, P3, P4, P5, and P6. The input and output ranges of each piece are given in Table 1.

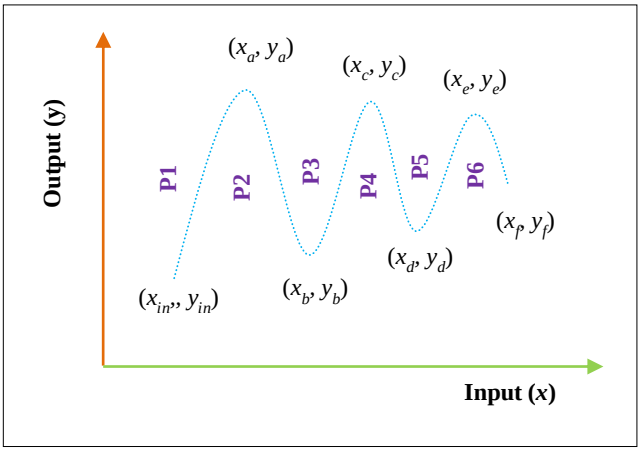


Fig. 2 An example data set

2.2. Generation of Fuzzy Submodel for Each Piece

Next, a fuzzy submodel is constructed for each piece of the data set envelope. In this paper, the fuzzy model is of the Mamdani type. The shapes of the membership functions are considered to be Gaussian.

Both the antecedent and consequent fuzzy sets consist of three fuzzy subsets: Low, Medium, and High. The low subset is regarded as a right-sided Gaussian, and the high subset is a left-sided Gaussian. The membership function parameters are optimally generated using the differential evolution algorithm.

Differential Evolution (DE) is a metaheuristic optimization tool that helps find the optimal parameter value in the search space [26-28].

In this paper, DE is used to select the parameters of membership functions for the antecedent and the consequent. Each individual in the optimization technique is represented by the following structure in Figure 3.

Table 1. Input and output ranges of each piece

Piece	Input Range	Output Range
P1	$[x_{in}, x_a]$	$[y_{in}, y_a]$
P2	$[x_{a+1}, x_b]$	$[y_{a+1}, y_b]$
P3	$[x_{b+1}, x_c]$	$[y_{b+1}, y_c]$
P4	$[x_{c+1}, x_d]$	$[y_{c+1}, y_d]$
P5	$[x_{d+1}, x_e]$	$[y_{d+1}, y_e]$
P6	$[x_{e+1}, x_f]$	$[y_{e+1}, y_f]$

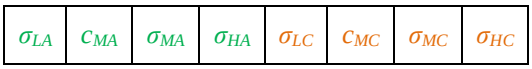


Fig. 3 Individual representing membership functions parameters

In Figure 3, the following notations are used.

σ_{LA}, σ_{MA}, σ_{HA} : Standard deviations of antecedent subsets, low, medium, and high, respectively.
c_{MA}, c_{MC} : Mean of subset medium for antecedent and consequent, respectively.
σ_{LC}, σ_{MC}, σ_{HC}: Standard deviations of consequent subsets, low, medium, and high, respectively.

In the optimization technique, the cost function for the j^{th} piece, say, f(j), is calculated by the sum squared errors of all kj data points belonging to the jth piece, as given in Equation (4).

$$f(j) = \sum_{m=1}^{k_j} (act(m) - fs(m))^2; \quad j \in (1, p) \qquad (4)$$

Here, *act(m)* is the actual or given output value of the m^{th} input data point, *fs(m)* is the output obtained through the proposed model for the *m-th* point, *p* denotes the total number of pieces of the given data set.

The optimization technique at each stage aims to reduce the above cost/objective function. The best individual found at the final iteration provides the membership functions' parameters for the antecedent and, consequently, the fuzzy submodel constructed for the jth piece data segment.

An algorithm for selecting the parameters of membership functions using differential evolution is given in Algorithm 1.

Algorithm 1: Membership function generation of fuzzy submodel for the jth piece
get no. of data points (k_j) in j-th piece, maxitr
generate initial population or target vectors
calculate cost functions of the target vectors (Equation 4)
set itr=0
while itr < maxitr
    set itr=itr+1
    for each individual
        perform mutation to generate donor vector
        perform recombination and create a trial vector
        calculate the cost function of the trial vector (Equation 4)
        select the target vector for the next iteration
    end for
end while
return the best individual of the last iteration

In general, the nonlinear functions contain maxima and minima points. As a result, a single rule base to design fuzzy function approximators will not apply to the whole data set. This is the primary motive for dividing the data set into multiple pieces. The extreme points create the different portions, so each piece represents either a monotonically increasing or a monotonically decreasing piece. Then, two sets of rule bases (Rule Base 1 for the monotonically increasing pieces and Rule Base 2 for the monotonically decreasing pieces) are used.

Rule Base 1:
Rule 1 : If the Antecedent is Low, Then the Consequent is Low.
Rule 2 : If the Antecedent is Medium, Then Consequent is Medium.
Rule 3 : If the Antecedent is High, Then the Consequent is High.

Rule Base 2:
Rule 1 : If the Antecedent is Low, Then the Consequent is High.
Rule 2 : If the Antecedent is Medium, Then the Consequent is Medium.
Rule 3 : If the Antecedent is High, Then the Consequent is Low.

For each input training data point, three rules of the corresponding Rule Base (either Rule Base 1 or Rule Base 2) are inferred with varying firing strengths, and the corresponding consequents are estimated. Outputs of all three rules are aggregated using the fuzzy MAX aggregation method [29].

The outcome of the inference engine is fuzzy, with the system being Mamdani-type. Hence, the output needs to be defuzzified to get a crisp output.

This paper uses one popular defuzzification technique, i.e., Center of Gravity (COG) [30] for defuzzification. The steps involved in generating the suggested fuzzy model are depicted in Algorithm 2.

Algorithm 2: Design of the proposed model
get function to be approximated
perform sampling the function uniformly into n number of data points
  for each data point
  check if it is an extreme point (maximum or minimum point)
  end for
perform division of data points in p numbers of pieces according to the extreme points
  for each piece
    design a fuzzy sub system
  end for
return database of the proposed model (input range, rule base, optimized membership functions' parameters for each piece)
function design of fuzzy subsystem for each piece
  if the piece is an increasing piece
    set Rule Base 1 for the fuzzy sub system
  else if
    set Rule Base 2 for the fuzzy sub system
  end if
perform optimization using differential evolution algorithm for membership functions' parameters
return fuzzy sub system parameters for each piece

An example database generated after constructing the proposed model for the function depicted in Figure 2 is given in Table 2.

After constructing the proposed fuzzy model, the uniformly sampled input data points are again used to check the performance of the designed model in function approximation. The steps for approximating the function using the designed model are depicted in Algorithm 3.

Algorithm 3: Function approximation using the proposed model
get the database generated after training
perform sampling the function uniformly into n numbers of data points
  for each input data point
    check in which piece the data point belongs to
    fetch the rule base and membership functions' parameter of that piece from the database
    use Mamdani fuzzy model with MAX aggression and COG defuzzification to estimate output
  end for
return predicted output for each input data point

By joining the test input-output data points, the function is approximated. The proposed model is named a piecewise optimum fuzzy model, or, in short, POFM.

3. Simulation Results

This section presents and analyzes the experimental results in approximating three nonlinear functions (a piecewise polynomial function, an exponentially decreasing sinusoidal, and an exponentially increasing sinusoidal function).

The Performance of the Proposed Model (POFM) is evaluated with the results obtained through the Radial Basis Function Network (RBFN) and the Support Vector Regression (SVR).

In RBFN, the goal of min squared error and the spread were set at their default values of 0 and 1, respectively, and the number of epochs was 500. For SVR, the kernel function is of the Gaussian type.

For DE in POFM, the coefficient F was generated using a Cauchy distribution. The crossover probability in DE was kept at 0.8, and the number of iterations was fixed at 500.

Table 2. Generated database after training

Piece	Input Range	Membership Parameters	Rule Base
P1	$[x_{in}, x_a]$	The value of the best individual (Figure 3) as obtained from the optimization technique for the respective piece	Rule Base 1
P2	$[x_{a+1}, x_b]$		Rule Base 2
P3	$[x_{b+1}, x_c]$		Rule Base 1
P4	$[x_{c+1}, x_d]$		Rule Base 2
P5	$[x_{d+1}, x_e]$		Rule Base 1
P6	$[x_{e+1}, x_f]$		Rule Base 2

The simulation results of the three models (RBFN, SVR, and POFM) were compared based on two error measures: average sum squared error and average error, as given in Equation (5), and in Equation (6), respectively.

Average Sum Squared Error (ASSE) =

$$\frac{1}{m}\sum_{i=1}^{m}(exact(i)-predicted(i))^2 \quad (5)$$

Average Error (AE) = $\frac{1}{m}\sum_{i=1}^{m}|exact(i)-predicted(i)|$ (6)

In Equation (5), and Equation (6), m denotes the test data set size. The simulations were done in MATLAB 2016a, and the results are given below in Examples 1-3.

Example 1: Piecewise Polynomial Function

The underlying function is a piecewise polynomial [9], as given in Equation (7), with x being the input and y being the output.

$$y=\begin{cases}\exp(0.5(x-4)) & \text{for} \quad 0\le x\le 4\\ \exp(-0.5(x-4)) & \text{for} \quad 4<x\le 8\end{cases} \quad (7)$$

The functions approximated by different models (RBFN, SVR, and POFM) are shown in Figures 4(b) - 4(d). A comparison of performance based on the errors of the models for the piecewise polynomial function is given in Table 3.

Example 2: Exponentially Decreasing Sinusoidal Function

The function given by Equation (8) comprises one sinusoidal part and another exponentially decreasing component [1]. The combined effect is shown in Figure 5(a).

$$y=\sin(4\pi x)\exp(-|5x|) \quad for \quad -1\le x\le 1 \quad (8)$$

The simulation results for approximating the function in Example 2 with three different models are depicted in Figures 5(b) - 5(d). The approximation errors of the models for the exponentially decreasing function are reported in Table 4.

Example 3: Exponentially Increasing Sinusoidal Function

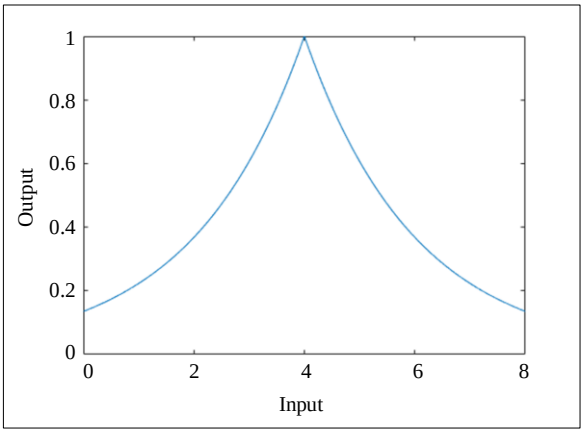
In contrast to the function considered in Example 2, the function in Example 3 is an exponentially increasing sinusoidal function, as given in Equation (9).

$$y=\sin(4\pi x)\exp(|5x|) \quad for \quad -1\le x\le 1 \quad (9)$$

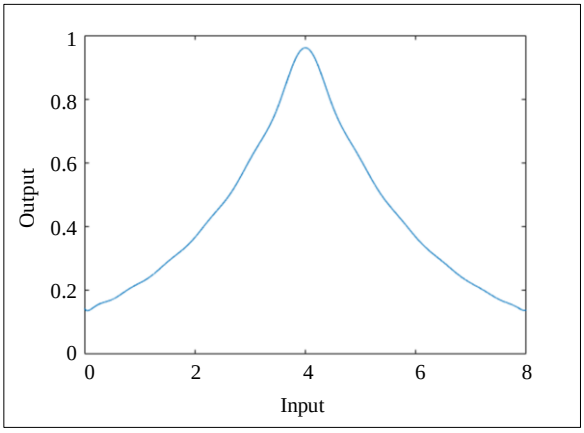
The functions approximated by RBFN, SVR, and POFM for the Example given in Equation (9) are shown in Figures 6(b) - 6(d). The errors in the approximation of the models are compared in Table 5.

Table 3. Errors of different approximators for the function in example 1

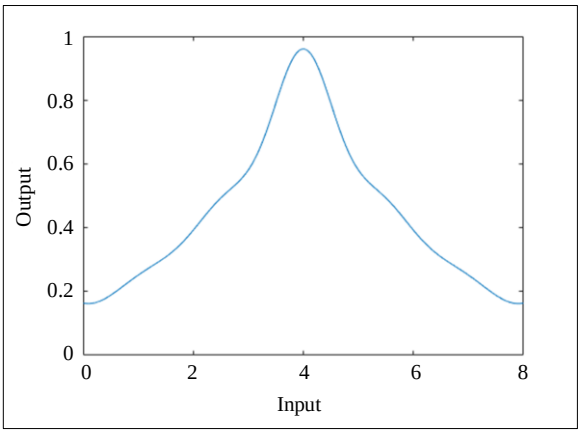
Approximator Used	Average Sum Squared Error	Average Error
RBFN	0.000025	0.002966
SVR	0.000467	0.020371
POFM	0.000051	0.005034



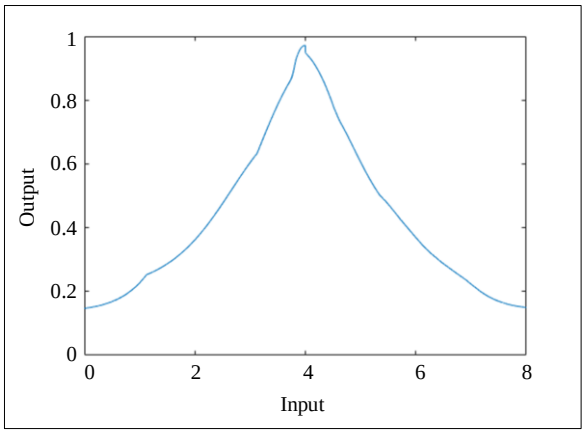
(a)



(b)

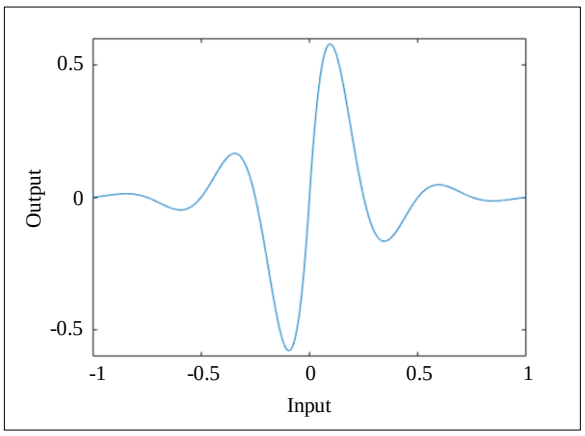


(c)

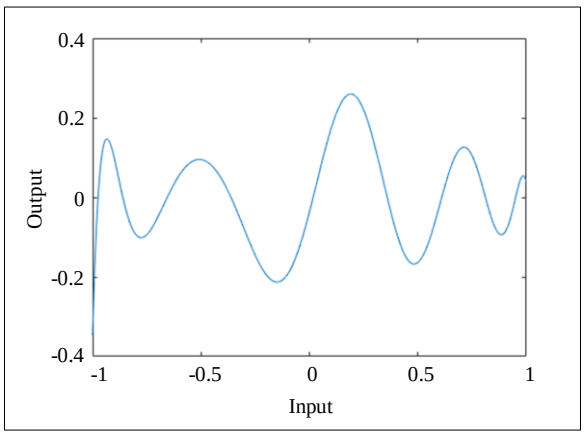


(d)

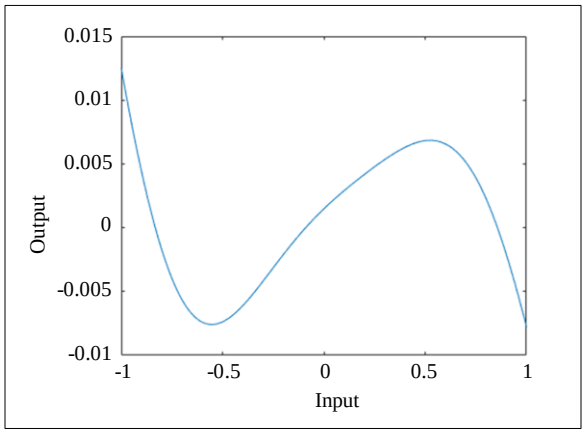
Fig. 4(a) Exact function and approximated functions by, (b) RBFN, (c) SVR, and (d) POFM for the function in example 1.



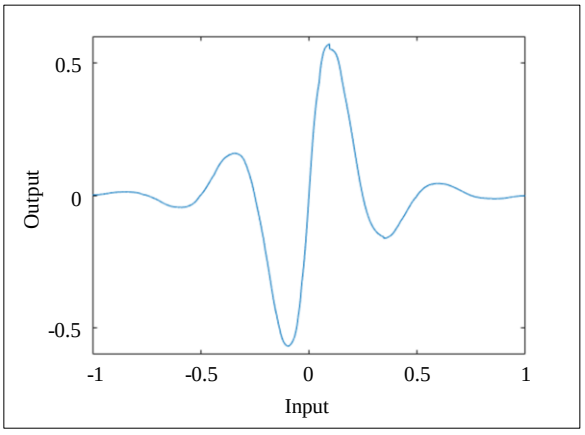
(a)



(b)



(c)



(d)

Fig. 5(a) Exact function and approximated functions by, (b) RBFN, (c) SVR, and (d) POFM for the function in example 2.

Table 4. Errors of different approximators for the function in example 2

Approximator Used	Average Sum Squared Error	Average Error
RBFN	0.028153	0.132018
SVR	0.043062	0.122560
POFM	0.000016	0.002276

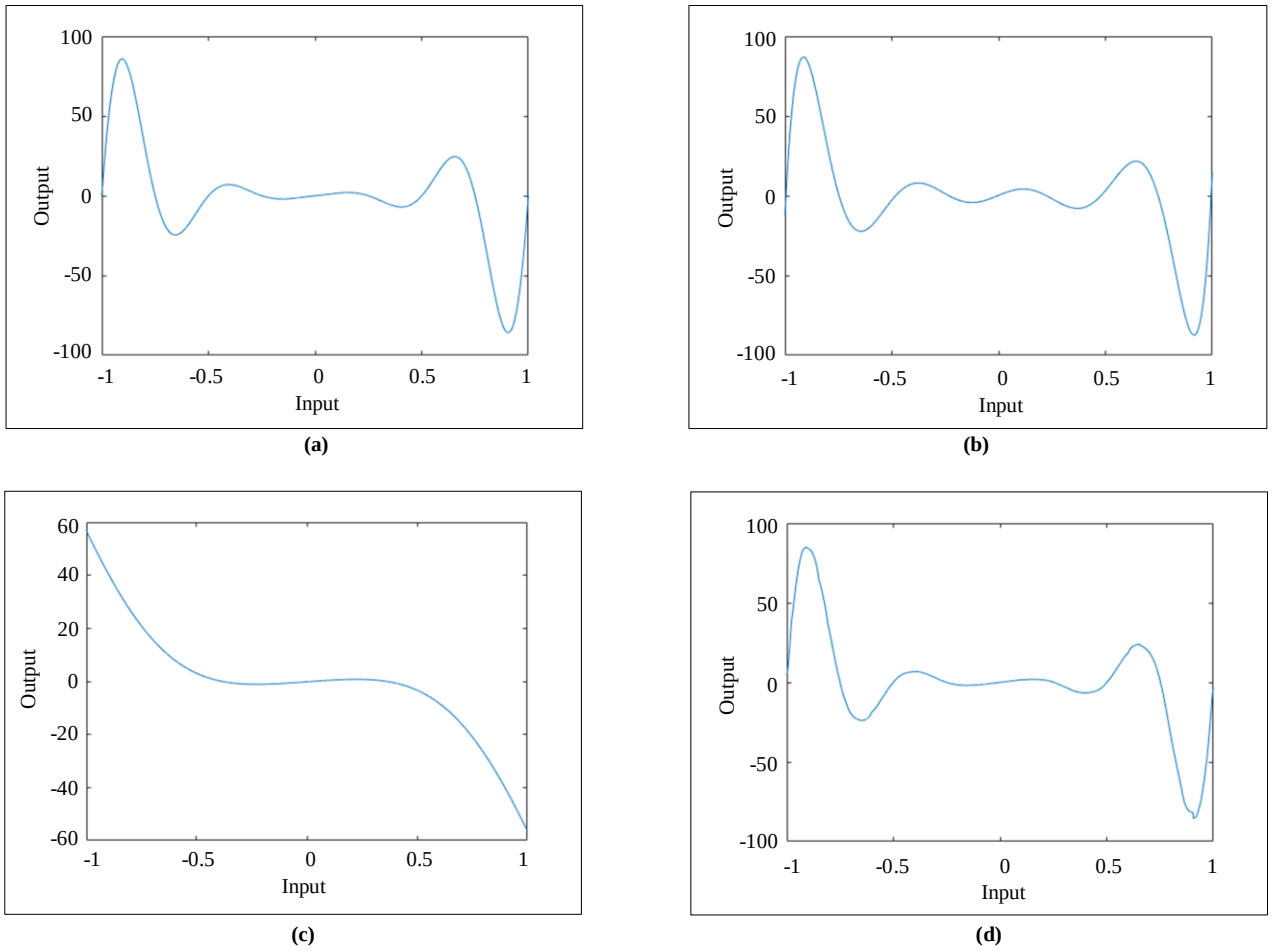


Fig. 6(a) Exact function and approximated functions by, (b) RBFN, (c) SVR, and (d) POFM for the function in example 3.

Table 5. Errors of different approximators for the function in example 3

Approximator Used	Average Sum Squared Error	Average Error
RBFN	6.790763	2.185588
SVR	417.123242	14.040023
POFM	0.355411	0.324149

4. Discussion

Figure 4 shows that all three models (POFM, RBFN, and SVR) can approximate the piecewise polynomial function with certain deviations at some points from the exact polynomial function. Table 3 indicates that the Proposed Model (POFM) for the piecewise polynomial function does not show the best performance, but its performance is much closer to the best approximator (RBFN). For the piecewise polynomial function, the Average Sum Squared Error and Average Error are 0.000025 and 0.002966, respectively, for RBFN; 0.000467 and 0.020371, respectively, for SVR; and 0.000051 and 0.005034, respectively, for POFM.

From Table 4, it is found that for the exponentially decreasing sinusoidal function approximation, the Proposed Model (POFM) yields the least Average Sum Square Error (0.000016) and the least Average Error (0.002276) in comparison to the average mean square error (0.028153) and average error (0.132018) for the RBFN model and the Average Sum Square Error (0.043062) and Average Error (0.122560) of the SVR model.

Figure 5 shows that RBFN and SVR fail to adequately approximate the exponentially decreasing sinusoidal function properly, whereas the Proposed Model (POFM) has satisfactorily approximated the function.

Regarding the exponentially increasing sinusoidal function, Figure 6(b) shows that RBFN performs well. Still, it yields a sizeable Average Sum Squared Error (6.790763) and

a significant average sum square error (2.185588), as reported in Table 5. Figure 6(c) and Table 5 indicate the failure of the SVR in approximating the exponentially increasing sinusoidal function. Figure 6(d) shows that the Proposed Model (POFM) has satisfactorily approximated the exponentially increasing sinusoidal function and has encountered the lowest average sum squared error and the lowest average error compared to RBFN and SVR.

5. Conclusion

In this paper, a new function approximation model is proposed. The proposed model consists of multiple fuzzy submodels, where each submodel is employed for an individual interval of the given data. The only parameters required to be designed in the proposed model are the membership functions' parameters, which have been selected using the optimization technique. Therefore, not much expert knowledge is required to choose the parameters of the suggested approximator.

From the investigation results, it can be seen that the suggested framework has satisfactorily approximated the three nonlinear functions considered here. Compared to two widely used function approximation models (Support Vector Regression & Radial Basis Function Network), the suggested model performs best for two nonlinear functions and the second best for one nonlinear function in terms of approximation errors. A similar piecewise function approximation technique may be developed, with each piece formulated by an optimized T-S-type fuzzy model.

References

[1] Zarita Zainuddin, and Ong Pauline, "Function Approximation Using Artificial Neural Networks," *International Journal of Systems Applications, Engineering & Development*, vol. 1, no. 4, pp. 173-178, 2007. [Google Scholar] [Publisher Link]

[2] Ivy Kidron, "Polynomial Approximation of Functions: Historical Perspective and New Tools," *International Journal of Computers for Mathematical Learning*, vol. 8, pp. 299-331, 2003. [CrossRef] [Google Scholar] [Publisher Link]

[3] Victor Zalizniak, *Essentials of Scientific Computing: Numerical Methods for Science and Engineering*, Horwood Publishing, England, 2008. [Google Scholar] [Publisher Link]

[4] Michael A. Cohen, and Can Ozan Tan, "A Polynomial Approximation for Arbitrary Functions," *Applied Mathematics Letters*, vol. 25, no. 11, pp. 1947-1952, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[5] Sibo Yang et al., "Investigation of Neural Networks for Function Approximation," *Procedia Computer Science*, vol. 17, pp. 586-594, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[6] Mariette Awad, and Rahul Khanna, *Efficient Learning Machines - Theories, Concepts, and Applications for Engineers and System Designers*, Apress Open, 2015. [Google Scholar] [Publisher Link]

[7] Chen-Chia Chuang et al., "Robust Support Vector Regression Networks for Function Approximation with Outliers," *IEEE Transactions on Neural Networks*, vol. 13, no. 6, pp. 1322-1330, 2002. [CrossRef] [Google Scholar] [Publisher Link]

[8] Xin Xu, Lei Zuo, and Zhenhua Huang, "Reinforcement Learning Algorithms with Function Approximation: Recent Advances and Applications," *Information Sciences*, vol. 261, pp. 1-31, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[9] T.A. Runkler, and J.C. Bezdek, "Alternating Cluster Estimation: A New Tool for Clustering and Function Approximation," *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 4, pp. 377-393, 1999. [CrossRef] [Google Scholar] [Publisher Link]

[10] J. Gonzalez et al., "A New Clustering Technique for Function Approximation," *IEEE Transactions on Neural Networks*, vol. 13, no. 1, pp. 132-142, 2002. [CrossRef] [Google Scholar] [Publisher Link]

[11] S.Y. Reutskiy, and C.S. Chen, "Approximation of Multivariate Functions and Evaluation of Particular Solutions Using Chebyshev Polynomial and Trigonometric Basis Functions," *International Journal of Numerical Methods in Engineering*, vol. 67, no. 13, pp. 1811-1829, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[12] Theodore J. Rivlin, *Chebyshev Polynomials*, 2nd ed., Courier Dover Publications, 2020. [Google Scholar] [Publisher Link]

[13] Dilcia Perez, and Yamilet Quintana, "A Survey on the Weierstrass Approximation Theorem," *Arxiv*, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[14] Rida T. Farouki, "The Bernstein Polynomial Basis: A Centennial Retrospective," *Computer Aided Geometric Design*, vol. 29, no. 6, pp. 379-419, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[15] Lloyd N. Trefethen, *Approximation Theory and Approximation Practice*, Extended ed., Society for Industrial and Applied Mathematics (SIAM) Publications, 2019. [Google Scholar] [Publisher Link]

[16] G. Cybenko, "Approximation by Superposition of a Sigmoidal Function," *Mathematics Control, Signals and Systems*, vol. 2, pp. 303-314, 1989. [CrossRef] [Google Scholar] [Publisher Link]

[17] Kurt Hornik, Maxwell Stinchcombe, and Halbert White, "Multilayer Feedforward Networks are Universal Approximator," *Neural Networks*, vol. 2, no. 5, pp. 359-366, 1989. [CrossRef] [Google Scholar] [Publisher Link]

[18] S. Ferrari, and R.F. Stengel, "Smooth Function Approximation Using Neural Networks," *IEEE Transactions on Neural Networks*, vol. 16, no. 1, pp. 24-38, 2005. [CrossRef] [Google Scholar] [Publisher Link]

[19] Ronald DeVore, Boris Hanin, and Guergana Petrova, "Neural Network Approximation," *Acta Numerica*, vol. 30, pp. 327-444, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[20] Fernando Pérez-Cruz et al., "Multi-Dimensional Function Approximation and Regression Estimation," *International Conference on Artificial Neural Networks*, vol. 2415, pp. 757-762, 2002. [CrossRef] [Google Scholar] [Publisher Link]

[21] Chih-Ching Hsiao, Shun-Feng Su, and Chen-Chia Chuang, "A Rough-Based Robust Support Vector Regression Network for Function Approximation," *2011 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011)*, Taipei, Taiwan, pp. 2814-2818, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[22] Chin-Teng Lin et al., "Support-Vector-Based Fuzzy Neural Network for Pattern Classification," *IEEE Transactions on Fuzzy Systems*, vol. 14, no. 1, pp. 31-41, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[23] H. Pomares et al., "An Enhanced Clustering Function Approximation Technique for A Radial Basis Function Neural Network," *Mathematical and Computer Modelling*, vol. 55, no. 3-4, pp. 286-302, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[24] Jerome H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189-1232, 2001. [Google Scholar] [Publisher Link]

[25] Paulo Vitor de Campos Souza, "Fuzzy Neural Networks and Neuro-Fuzzy Networks: A Review the Main Techniques and Applications Used in the Literature," *Applied Soft Computing*, vol. 92, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[26] Rainer Storn, and Kenneth Price, "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, pp. 341-359, 1997. [CrossRef] [Google Scholar] [Publisher Link]

[27] Rainer Storn, "Differential Evolution Research -Trends and Open Questions," *Advances in Differential Evolution*, vol. 143, pp. 1-31, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[28] Swagatam Das, and Ponnuthurai Nagaratnam Suganthan, "Differential Evolution: A Survey of the State-of-the-Art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4-31, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[29] Samir Roy, and Udit Chakraborty, *Introduction to Soft Computing, Neuro-Fuzzy and Genetic Algorithms*, Pearson, India, 2013. [Google Scholar] [Publisher Link]

[30] Snehashish Chakraverty, Deepti Moyi Sahoo, and Nisha Rani Mahato, *Concepts of Soft Computing, Fuzzy and ANN with Programming*, Springer, Singapore, 2019. [CrossRef] [Google Scholar] [Publisher Link]