Available online at www.isr-publications.com/jmcs
J. Math. Computer Sci., 17 (2017), 355–364
Research Article

Online: ISSN 2008-949x

Journal of Mathematics and Computer Science

Journal Homepage: www.tjmcs.com - www.isr-publications.com/jmcs

A combination of curve fitting algorithms to collect a few training samples for function approximation

Saeed Parsa*, Mohammad Hadi Alaeiyan

Department of Computer Engineering, Iran University of Science and Technology, Narmak, Tehran, 16844, Iran.

Abstract

The aim of this paper is to approximate the numerical result of executing a program/function with a number of input parameters and a single output value with a small number of training points. Curve fitting methods are preferred to non-deterministic methods such as neural network and fuzzing system methods, because they can provide relatively more accurate results with the less amount of member in the training dataset. However, curve fitting methods themselves are most often function specific and do not provide a general solution to the problem. These methods are most often targeted at fitting specific functions to their training dataset. To provide a general curve fitting method, in this paper, the use of a combination of Lagrange, Spline, and trigonometric interpolation methods are suggested. The Lagrange method fits polynomial functions of degree N to its training values. In order to improve the resultant fitted polynomial our combinatorial method combines Lagrange with the polynomial resulted from the Spline method. If the absolute error of the actual value and the predicted value of a function are not desired, the trigonometric interpolation methods that fit trigonometric functions can be applied. Our experiments with a number of benchmark examples demonstrate the relatively high accuracy of our combinational fitting method.

Keywords: Output function approximation, black box approximation, curve fitting, linear function approximation, nonlinear function approximation.
2010 MSC: 68W25, 68W30, 12Y05, 14Q05.

1. Introduction

The aim is to propose a new method to approximate both linear and nonlinear real-valued executing functions, with minimum approximation error. Function approximation is a method to map a given input vector to an output vector. It is simply the mapping of a domain(x) to a range(y). In this case, the domain is represented by a real-valued vector and range is a single real-valued output. Function approximation is a known technique for analysis and reverse engineering for constrained fitting [4]. In addition, function approximation is of great use in software testing to verify the correctness of mathematical code, also it can be used as an oracle in black box software testing. Moreover, in certain situations, such as predicting a robot action and estimating the value of the lost pixels of an image, function approximation is also quite beneficial [22].

*Corresponding author
Email addresses: Parsa@iust.ac.ir (Saeed Parsa), hadi_alaeiyan@comp.iust.ac.ir (Mohammad Hadi Alaeiyan)

doi:10.22436/jmcs.017.03.02

Received 2015-07-11

There are several non-deterministic and deterministic approaches for function approximation, all of which are aimed at minimizing the approximation error with a small number of a training points. For instance, in order to reduce approximation error, it is observed that some function approximation methods, using neural networks [10], have modified their learning engine and its weight of neurons by changing or modifying their sigmoidal function [7] or feed-forward network and back-propagation learning algorithm [10]. To promote the accuracy of neural network fuzzing methods have been applied to the approximations provided by neural networks [20]. Also in certain situations, the approximation provided by a fuzzy system is used as input to neural networks [8]. A fuzzy system may include a set of fuzzy rules to map inputs to outputs according to the semantics of a function $f : X \to Y$. The rules have if-then style such as "if X is A, then Y is B." where A and B are multivalued or fuzzy sets that contain members to some degree. Also, A and B are subsets of X and Y, respectively. Moreover, genetic programming and algorithms have great use to approximate simple functions [15].

A major difficulty with function approximation based on genetics is that along with increasing the number of decisions in the approximated functions, the accuracy of the results decreases [15]. In general, the drawback of non-deterministic approaches such as neural networks, fuzzy and genetics is the relatively large size of the training points and approximation of more than one result for a given input [14]. The performance of non-deterministic function approximation algorithms degrades as the size of samples decreases. To resolve the difficulty, artificial sample generation techniques are used [12, 21]. However, these techniques were demonstrated only for functions with at most two input parameters [19]. On the other hand, deterministic methods tend to generate a single result for a given function in different execution of their underlying algorithms. However, yet the accuracy of these methods highly relies on the increasing number of the training points. Weighted kernel regression methods incorporate prior knowledge of model to improve the accuracy [16]. In general, there are several techniques to compensate the approximation error caused by the small number of training points such as finely tune the model parameter [19], pre-data processing [2, 17], and incorporation of prior knowledge [6, 13, 23]. However, as described in [2], there are not universally optimal solution to this problem yet.

Curve fitting is a well-known deterministic approach to function approximation in mathematical fields. Curve fitting methods such as Lagrange, Spline, Newton, and trigonometric interpolations have been largely applied to estimate both linear and nonlinear functions. These approximation methods themselves could be linear or nonlinear dependent on the number of training points. Lagrange attempts to find a polynomial, passing through all the points in the training points. However, a major drawback is the oscillation of the resultant high degree polynomials amongst the adjacent training points [3]. To damp the oscillation, we combine the interpolation polynomial resulted from the Lagrange with the one produced by the Spline method on the same set of training points. However, in certain situations where the function to be approximated, is itself inherently oscillating the damping appears to affect adversely and worsens the approximated results. To overcome the problem, we apply trigonometric interpolation to approximate mathematical functions that describe repetitive oscillations. These performances help us to propose a fewer among of training points with better accuracy in function approximation.

The remainder of this paper is organized as follows: in Section 2, the proposed approach is presented and its algorithm is presented in Section 3, and Section 4 includes a discussion of the experimental results.

2. Combinatorial curve fitting method

In order to approximate a given function $F(p1, p2, \ldots, pn)$, the domain $[pi\_min, pi\_max]$ of each input parameter pi, is subdivided into a sequence of intervals $< [pi\_min, pi\_min + L], [pi\_min + L, pi\_min + 2 * L], \ldots, [pi\_max - L, pi\_max] >$. The width of the intervals, L, is determined in such a way that the function oscillations are divided amongst the intervals in such a way that the approximation error is minimized. If the behavior of the function to be approximated, is trigonometric within an interval, then the trigonometric interpolation method is applied, otherwise, the combinatorial method is applied.

Many types of research and approximation methods have utilized the Lagrange interpolation [3, 5] as one of the standard approximation tools to the problem of nonlinear function approximation because, Lagrange interpolates a training points, collected for a given function, into a polynomial function that passes through all the points within the training points. The degree of the resultant polynomial could be maximally the number of training points minus one, which could be very high. Therefore, the polynomial may suffer from unsolved problems such as Runge's and Gibbs phenomenon [3]. On the other hand, the Spline interpolation of degree k avoids the problem of Runge's phenomenon, by limiting the degree of the interpolating function to $\leqslant k$. However, the interpolating function cannot cover all the training points collected from a function of degree greater than k. Therefore, to provide a full coverage of the training points while avoiding the problems of Runge's phenomenon, a combination of both Lagrange and Spline of degree k could be promising. However, neither Lagrange nor the Spline could not prevent the Gibbs phenomenon. To smooth down the oscillations provided by the Gibbs phenomenon, we pass a linear approximation through the training points by connecting each two consecutive points with a line. Therefore, to provide a convenient approximation of a nonlinear function, a finite linear combination of Lagrange, Spline and Linear approximation interpolations of the function could be used. Considering the experimental results for approximation of several programming functions the following relation for combining the interpolating functions is suggested:

$$\begin{aligned}\text{Combinatorial Interpolation} = {} & 0.5 * \text{Spline Degree } 3 + 0.25 * \text{Linear Approximation} \\ & + 0.25 * \text{Lagrange Interpolation.}\end{aligned}$$

The accuracy of the interpolating function depends on the comprehensive coverage of the fluctuations and oscillations of the function, to be approximated. The coverage itself depends on the number of training points, collected for the function. Therefore, in the case of fast oscillations, a relatively high number of training points would be required. To resolve the difficulty, we apply trigonometric approximation for the sub-intervals for which the combinatorial method cannot provide an acceptable approximation.

3. Combinatorial algorithm

We have applied a combination of three different curve-fitting methods, namely Lagrange, Spline, and trigonometric interpolations, to approximate both nonlinear and linear functions of more than one input parameter. To treat functions with more than one input parameter we alternatively, alter only one of the inputs and keep the rest as constant for a number of executions. In this way, the program execution space traversed in a stepwise manner. The difficulty is to determine the suitable number of executions, in such a way that the estimation error is minimal, the range of each of the input parameter is covered uniformly and the execution time is acceptable.

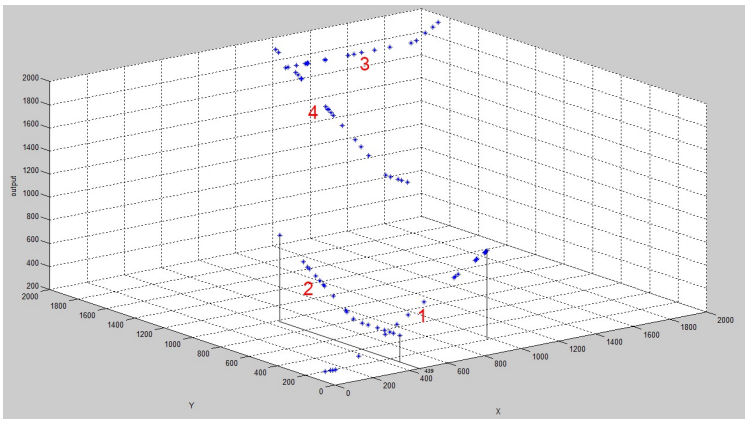

Figure 1: the space of training points of function $O = \max(x, y)$ and its intervals.

As an example, consider the program input-output space shown in Figure 1. In this example, there are two inputs $0 \leqslant x \leqslant 2000$ and $0 \leqslant y \leqslant 2000$ and one output, O for a function, $\max(x, y)$. In order to avoid the Runge's phenomenon which in here is a problem of the oscillation at the edges of the input intervals, the interval $[0, 2000]$ of the input variables x and y are broken down into $[0, 999]$ and $[1000, 2000]$. Our experimental result shows that the width 1000 is well suited to prevent the Runge's phenomenon. There are four training points shown with dotted lines and labeled 1 to 4 in Figure 1. The dotted lines labeled 1 and 2 represent input spaces within the interval $[0, 999]$ and the input spaces for the interval $[1000, 2000]$ are labeled 3 and 4. Each set of input forms a straight line in the program input space because it is produced by altering one of the input parameters while keeping the other ones fixed. For instance, the line labeled 1 represents the inputs produced by altering the value of the parameter x from 0 to 999 and keeping the value of y equal to 200.

3.1. A new combinatorial curve-fitting algorithm

Our proposed combinatorial curve-fitting algorithm is presented in Tables 1-3. The algorithm accepts a function F and the size L, of the intervals in which the varying input parameter of the function $F(X1, X2, \ldots, Xn)$ is alternatively selected. Also, there is a third input parameter, max _error, which defines the maximum tolerable error of the approximations provided by the algorithm for the function F. The value of max _error, by our experiments is found to be a real number between 10^{15} and 10^{13}. The output of the combinatorial-curve-fitting algorithm is an approximation of the value returned by the function F. Consequently, the training points that used for approximation are able to approximate function F.

The algorithm loops over the input parameters $Xi, 1 \leqslant i \leqslant n$, of the function $F(X1, X2, \ldots, Xn)$ and each time selects a parameter Xi. Then, it breaks down the range of the values of the selected parameter, Xi, into intervals of the width L and subsequently generates $((\alpha/100) * L)$ number of training points as random numbers within each of the resulted intervals. It also generates $((\alpha/100) * L)^2$ number of the testing points as random numbers within each of the intervals. Experimental results show that the appropriate number of training points is achieved by setting the value of equal to 2. Also, its range is in [1,100]. Each training and testing point is represented as a tuple $(X1, X2, \ldots, Xi\_k, \ldots, Xn, y)$ where Xi_k is a random number within an interval of the Xi values and y is equal to $F(X1, X2, \ldots, Xi\_k, \ldots, Xn)$. The resultant training and testing points are passed as input parameters to the Function_Approximation function. Also the maximum acceptable approximation error, max _error is sent to the function as its third parameter. The function adjusts the number of training points in such a way that the approximation error is minimized. The function also determines whether polynomial or trigonometric curves are preferred to be fitted with the training points. The output of this function is the adjusted set of training points and the appropriate curve fitting method as either trigonometric or polynomial.

The function Function_approximation firstly, depends on whether the training points are collinear applies Lagrange interpolation method, or not applies a combination of Lagrange, Spline degree one and Spline degree three to approximate each testing point. In order to combine the approximations provided by these three methods for each testing point over a given set of training points, Table 4 is used. For example, in row 1 of Table 4, the combinatorial method returns zero provided that when applying the three methods to a testing point, and its associated training points all of them result in infinity (INF) or not a number (NAN). In this table, S, Linear, and L indicate the results of Spline degree three, Spline degree one and Lagrange method, respectively.

The approximation error is computed as the difference between the value resulted from applying the combinatorial method to a testing point and the actual value returned by the function, to be approximated. If the approximation is greater than max _error, then Function_approximation continues the processes to achieve better approximation by calling the functions Complimentary_Fitting, Find_intersections, Trigonometric_case, Two_extra_points, and Two_extra_points2, respectively, that all of them continue the process until errors value is greater than max _error. Hence, function Two_extra_points2 selects two points that they are very close to the testing points, its approximation value is able to be interpolated by spline methods.

Table 1: the combinatorial-curve-fitting algorithm.

```
1   Algorithm combinatorial-curve-fitting
2   Input
3     F: a function to be approximated;
4     L: the width of the intervals that splites an input parameter space into the
    intervals with length L.
5     max_error: the maximum value of the error, acceptable for the approximation.
6   Output:
7     Fitted curves: set of interpolated curves;
8   Begin
9     For each formal parameter, Xi, of the function F(X1, X2, ..., Xn) do
10    Begin
11      Assume Xi is a real number;
12      Partition the range of Xi into intervals, Li,j, of the width L defined by the user;
13      For each interval Li,j do
14      Begin
15        Compute no-points = %α * L;
16        Generate no-points training points within the Li,j interval, randomly;
17        Generate randomly, no-points*no-points testing points within the Li,j
            interval such that none of the training points is repeated in testing points;
18        Fitted_Curves = Function_Approximation (training_points, testing_points, max_error);
19      end {for each interval}
20    end {for each input}
21  End {Algorithm combinatory}

22  Set of interpolated Curves Function_Approximation(training_points, testing_points, max_error)
23  Begin
24    For each pi in testing_points
25    Begin
26        if (training_points_are_collinear(training_points)) then
27          estimation_error = Fit_a_Line(training_points, OUT set of interpolated_Curves);
28        else estimation_error =
    Combinatorial_method(training points, pi, OUT set of interpolated_Curves);
29        if estimation_error > max_error then
30          estimation_error = Complimentary_Fitting(training points, pi,
                                  OUT set of interpolated_Curves);
31        if estimation_error > max_error then
32          estimation_error = Find_intersections(training points, pi,
                                  OUT set of interpolated_Curves);
33        if estimation_error > max_error then
34          estimation_error = Trigonometric_case(training points, pi,
                                  OUT set of interpolated_Curves);
```

To put it in a nutshell, this combinatorial method uses $b * ((\alpha\%)L) + 7w + 2p$ training points, to fulfill function approximation. Where b is the number of intervals, L is the width of intervals, w is the number of testing points that entered to the function Complimentary_Fitting and p is the number of testing points that reached to Two_extra_points, as well as α, is the modules of L that shows the number of the training points in each interval. Moreover, this method does not propose a unique result based on different random inputs.

Table 2: the combinatorial-curve-fitting algorithm.

```
35        if estimation_error > max _error then
36          estimation_error = Two_extra_point(training points, pi,
                                  OUT set of interpolated_Curves);
37        if estimation_error > max _error then
38          estimation_error = Two_extra_points2(training points, pi,
                                  OUT set of interpolated_Curves);
39    end {for each testing point}
40  End {Algorithm Approximation}

41  Function training_points_are_collinear(training_points: set of points): Boolean;
42  {/*This function determines whether the training points, training_points,
    are collinear.*/}

43  Procedure Fit_a_Line(training_points: set of points, OUT set of interpolated_Curves);
44  {/*Uses the Lagrange interpolation method to fit a line to the training points,
    training_points. The training points are supposed to be collinear*/}

45  Function Error Combinatorial_method(training_points: set of points, pi:
    testing points, OUT set of interpolated_Curves)
46  {/*Uses table 4 to combine the approximations provided for the by:
47                                  1. Lagrange (training_points, pi),
48                                  2. Spline_degree_three (training_points, pi) and
49                                  3. Spline_degree_one (training_points, pi) */}

50  Function Error Complimentary_Fitting(training_points: set of points, pi:
    testing points, OUT set of interpolated_Curves)
51  {/*This Function is invoked whenever the approximation error of the
    Combinational_method is greater than a certain value, max _error, provided by the
    user or not. To reduce the approximation error, Complementary_Fitting adds 3
    random distinct points immediately before pi and 4 random distinct points
    immediately after pi to the training_points*/}

52  Function Error Find_intersections(training_points: set of points, pi: testing points,
    OUT set of interpolated_Curves)
53  {/* In order to reduce the approximation error at the testing point pi,
    Find_intersections is invoked. This function splits the set of training points,
    completed by Complimentary_Fitting, into two sets of points greater than or equal
    and less than the testing point pi */}

54  Function Error Trigonometric_case(training_points: set of points, pi:
    testing points, OUT set of interpolated_Curves)
55  {/*This function fits a trigonometric curve P(x) = a_0 + Σ_{j=1}^{n}(a_j cos(jt) + b_j sin(jt))
    [3] to training_points.*/}

56  Function Error Two_extra_points(training_points: set of points, pi:
    testing points, OUT set of interpolated_Curves)
```

Table 3: the combinatorial-curve-fitting algorithm.

57	{/*This function is invoked to reduce the approximation error at a testing point, pi. This function adds two extra points very close to pi to training_points and then invokes Combinatorial_method (training points, pi, interpolated_Curves) to fit a polynomial to training points. If the approximation error is still high, the function Two_extra_points2 (training_points, pi) will be invoked.*/}
58	Function Error Two_extra_points2(training_points: set of points, pi: testing points, OUT set of interpolated_Curves)
59	{/*This function fits a trigonometric curve to training_points and invokes Trigonometric_case function.*/}

Table 4: Combining the results from three methods of Lagrange, Spline degree one and Spline degree three.

Row	Return	Lagrange	Spline degree one	Spline degree three
1	0	NAN or INF	NAN or INF	NAN or INF
2	S	NAN or INF	NAN or INF	S
3	L	L	NAN or INF	NAN or INF
4	Linear	NAN or INF	Linear	NAN or INF
5	0.5*S+0.5*L	L	NAN or INF	S
6	0.5*S+0.5*Linear	NAN or INF	Linear	S
7	0.5*L+0.5*Linear	L	Linear	NAN or INF
8	0.5*S+0.25*Linear+0.25*L	L	Linear	S

4. Numerical experiments and results

This experiment is conducted to show the advantage of our proposed combinatorial method over a fuzzing system [20], a regression approximation method, NWKR, and two known neural network methods, ANNBP and DNN model [18], for small samples problem with a few number of training points. In Table 5, the results of our experiments with a function $y = x^2$ in the domain $[-32768, 32767]$, is presented. We have applied 490 training samples to approximate the function with our proposed combinatorial method. Figure 2 shows the real function $y = x^2$ in the dotted curve. The combinatorial method applied Lagrange, Spline degree 3, and Linear approximation methods to approximate the function for $y = x^2$ in the interval defined in Figure 2. The mean square approximation error (MSE) is $1.36 * 10^{10}$ and the curve for the resultant interpolating function is exactly the same as the one depicted in Figure 2. The function $y = x^2$ is also used in [18] to show the advantage of NWKR as compared to ANNBP and DNN model. The number of samples used in [18] is 5 in the interval $[0, 1.2]$ that is equal to 218450 samples in the domain $[-32768, 32767]$.

Table 5: MSE of approximated result with methods DNN, ANNHP, NWKR and curve fitting.

	DNN	ANNHP	NWKR	Combinatorial-curve-fitting
MSE	0.002358	0.004528	0.001182	$1.36 * 10^{10}$

However, NWKR and neural network approximation methods suffer from the over-fitting phenomenon. Neural networks have been widely used for function approximation due to their structural simplicity and fast learning capabilities [9]. The neural networks are able to handle continuous input data. The multilayer perceptron network is the most common network used today. Due to the powerful nonlinear function approximation and adaptive learning abilities, neural networks have drawn great attention in the field of function approximation. The advantage of neural network approach is their generalization capability, which lets them deal with partial or noisy inputs. However, the accuracy of approximated function

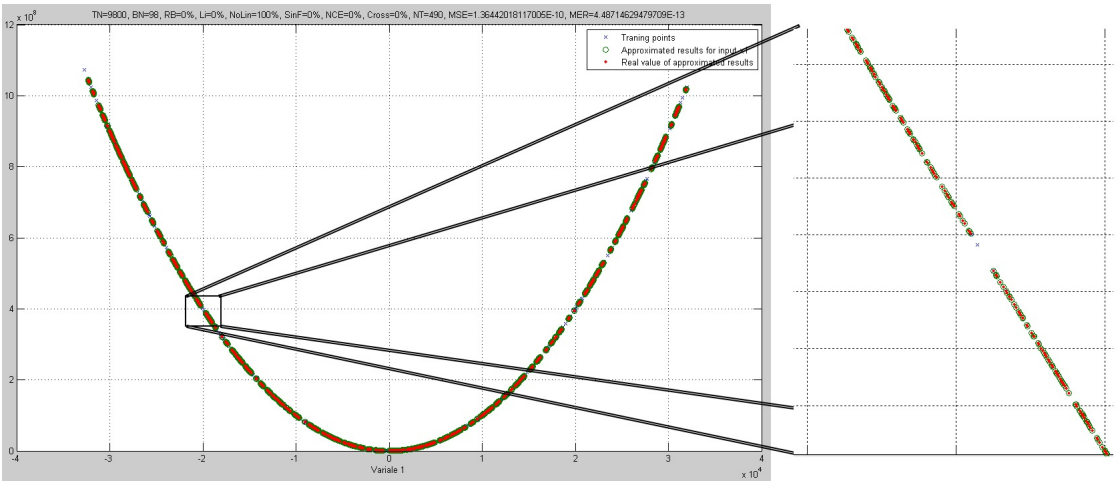

Figure 2: Approximated results for $y = x^2$.

obtained by a neural network is dependent on the number of training points and weights of neurons. If the number of training points is not adequate, the approximation provided by the neural networks will suffer from the over-fitting phenomenon. The proposed combinatorial method has resolved the overfitting and under-fitting phenomena by smoothing down the fluctuations caused by the differences in the degree of the real function and its approximations.

Table 6: RMSE of approximated result with methods Proposes in [20], LS-Based and curve fitting.

	Proposes method in [20]	LS-Based method	Combinatorial-curve-fitting
RMSE	0.0087	0.0843	0.0062

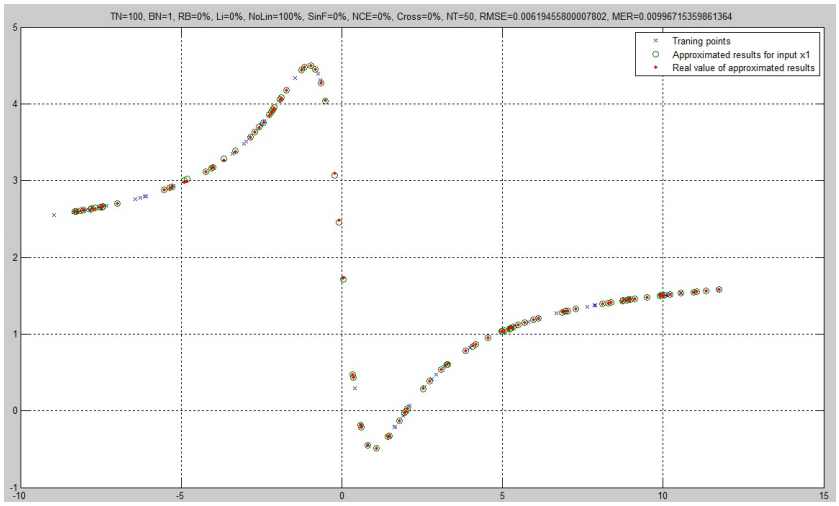

Figure 3: approximated results for $((x-2)(2x-1))/(1+x^2)$,"O" in this figure shows approximated value for testing points and "." shows the actual value of testing points and "×" shows 50 random training points.

In Table 6, root minimum square error (RMSE) for approximations of a nonlinear function $((x-2)(2x-1))/(1+x^2)$, using fuzzy method presented in [20], LS-Based, and our combinational method. As shown in Figure 2, the fuzzy method outperforms the LS-Based function approximation method. However, a major difficulty with the fuzzy based approximation methods is their dependency on the experts knowledge [1]. Fuzzy systems generate IF-THEN rules to determine the approximation method dependent on a function behavior in different intervals of its domain [2, 8]. The basic idea behind the

fuzzy systems method is the subdivision of the input space into fuzzy regions and approximate the system in each subdivision by a simple model. The main concern underlying fuzzy system is to minimize the number of rules. In Table 6 the results of our experiments with a function $((x-2)(2x-1))/(1+x^2)$ in domain $(-8,12)$ is presented. We have applied 50 training samples to approximate the function, using our proposed combinatorial method. Figure 3 shows the real function $((x-2)(2x-1))/(1+x^2)$ in dotted curve. The combinatorial method applies Lagrange, Spline degree 3, and Linear approximation methods to approximate the function $((x-2)(2x-1))/(1+x^2)$ in the interval defined in Figure 2. The root mean square approximation error (RMSE) is 0.0062 and the curve depicting, the resultant interpolating function is exactly the same as the one shown in Figure 3. In Figure 3, white bullet points represent all the approximated points, provided by our combinatorial method. The function $((x-2)(2x-1))/(1+x^2)$ is also used in [20] to show the advantage of the fuzzy function approximation method in comparison with the LS-Based method. The number of samples used in [20] are 50 points within the interval $(-8,12)$ and the RMSE is 0.0087.

5. Conclusions

In order to fit a curve into a nonlinear function with a small number of training points, dependent on the behavior of the function in different intervals of its domain, different curve fitting methods could be applied. Our experiments suggest the combinatorial use of trigonometric, Lagrange, Spline degree 3, and linear interpolation methods to model a variety of behaviors shown by a nonlinear function in different intervals of its domain that this combinatorial method collects a suitable number of training points that these points are able to replace with the function. For instance, sinusoidal behaviors could be modeled appropriately, using trigonometric curve fitting methods and polynomial, power and logarithmic behaviors could be modeled appropriately, using Lagrange and Spline interpolation methods. In addition, linear behaviors could be better modeled by applying the well-known linear approximation method [11]. Moreover, multidisciplinary behaviors could be better modeled by finding the intersections of curves, depicted for the multidisciplinary function, and apply different curve fitting methods to fit suitable curves within consecutive intersecting points.

The width of the intervals and the number of training points should be determined carefully because the insufficient width of the intervals and the number of the training points may result in unsatisfactory predictions. For the time being, we are working on automatic detection of the width of the intervals and the suitable number of training points within different intervals.

Acknowledgment

We would like to give special thanks to Iran National Science Foundation (ISNF) for its financial supports of this research work.

References

[1] L. E. Aik, Y. Jayakumar, *A study of neuro-fuzzy system in approximation-based problems*, Mat., **24** (2008), 113–130. 4
[2] R. Andonie, L. Fabry-Asztalos, C. B. Abdul-Wahid, S. Abdul-Wahid, G. I. Barker, L. C. Magill, *Fuzzy ARTMAP prediction of biological activities for potential HIV-1 protease inhibitors using a small molecular data set*, IEEE/ACM Trans. Comput. Biol. Bioinf., **8** (2011), 80–93. 1, 4
[3] K. E. Atkinson, *An introduction to numerical analysis*, Second edition, John Wiley & Sons, Inc., New York, (1989). 1, 2, 2
[4] P. Benkö, G. Kós, T. Várady, L. Andor, R. Martin, *Constrained fitting in reverse engineering*, Comput. Aided Geom. Design, **19** (2002), 173–205. 1
[5] J. P. Berrut, L. N. Trefethen, *Barycentric Lagrange interpolation*, SIAM Rev., **46** (2004), 501–517. 2
[6] G. Bloch, F. Lauer, G. Colin, Y. Chamaillard, *Support vector regression from simulation data and few experimental samples*, Inform. Sci., **178** (2008), 3813–3827. 1
[7] G. Cybenko, *Approximation by superpositions of a sigmoidal function*, Math. Control Signals Systems, **2** (1989), 303–314. 1

[8] J. A. Dickerson, B. Kosko, *Fuzzy function approximation with ellipsoidal rules*, IEEE Trans. Systems Man Cybernet., **26** (1996), 542–560. 1, 4
[9] M. Gori, F. Scarselli, *Are multilayer perceptrons adequate for pattern recognition and verification?*, IEEE Trans. Pattern Anal. Mach. Intell., **20** (1998), 1121–1132. 4
[10] J. W. Hines, *A logarithmic neural network architecture for unbounded non-linear function approximation*, IEEE International Conference on Neural Networks, Washington, DC, USA, **2** (1996), 1245–1250. 1
[11] https : //en.wikipedia.org/wiki/Linear_interpolation 5
[12] C.-F. Huang, C. Moraga, *A diffusion-neural-network for learning from small samples*, Internat. J. Approx. Reason., **35** (2004), 137–161. 1
[13] F. Lauer, G. Bloch, *Incorporating prior knowledge in support vector regression*, Mach. Learn., **70** (2008), 89–118. 1
[14] Y. Mizukami, Y. Wakasa, K. Tanaka, *A proposal of neural network architecture for non-linear function approximation*, Proceedings of the 17th International Conference on Pattern Recognition, Cambridge, UK, **4** (2004), 605–608. 1
[15] K. Rodríguez-Vázquez, C. Oliver-Morales, *Multi-branches genetic programming as a tool for function approximation*, Genetic and Evolutionary Computation Conference, Seattle, WA, USA, (2004), 719–721. 1
[16] M. I. Shapiai, Z. Ibrahim, M. Khalid, *Enhanced weighted Kernel regression with prior knowledge using robot manipulator problem as a case study*, Procedia Eng., **41** (2012), 82–89. 1
[17] M. I. Shapiai, Z. Ibrahim, M. Khalid, L. W. Jau, S.-C. Ong, V. Pavlovich, *Solving small sample recipe generation problem with hybrid WKRCF-PSO*, Int. J. New Comput. Archit. Appl., **1** (2011), 810–820. 1
[18] M. I. Shapiai, Z. Ibrahim, M. Khalid, L. W. Jau, V. Pavlovich, *A non-linear function approximation from small samples based on Nadaraya-Watson kernel regression*, 2nd International Conference on Computational Intelligence, Communication Systems and Networks, Liverpool, UK, (2010), 28–32. 4
[19] M. I. Shapiai, Z. Ibrahim, M. Khalid, L. W. Jau, V. Pavlovic, J. Watada, *Function and surface approximation based on enhanced kernel regression for small sample sets*, Int. J. Innov. Comput. I., **7** (2011), 5947–5960. 1
[20] T.-Y. Sun, S.-J. Tsai, C.-H. Tsai, C.-L. Huo, C.-C. Liu, *Nonlinear function approximation based on Least Wilcoxon Takagi-Sugeno fuzzy model*, Eighth International Conference on Intelligent Systems Design and Applications, Kaohsiung, Taiwan, **1** (2008), 312–317. 1, 4, 6, 4
[21] T.-I. Tsai, D.-C. Li, *Approximate modeling for high order non-linear functions using small sample sets*, Expert Syst. Appl., **34** (2008), 564–569. 1
[22] G. S. Watson, *Smooth regression analysis*, Sankhyā Ser. A, **26** 1964, 359–372. 1
[23] J. Yuan, C.-L. Liu, X.-M. Liu, K.-S. Wang, T. Yu, *Incorporating prior model into Gaussian processes regression for WEDM process modeling*, Expert Syst. Appl., **36** (2009), 8084–8092. 1