International Journal of Innovative Computing, Information and Control
Volume **7**, Number **10**, October **2011**

ICIC International ©2011 ISSN 1349-4198
pp. **5947–5960**

FUNCTION AND SURFACE APPROXIMATION BASED ON ENHANCED KERNEL REGRESSION FOR SMALL SAMPLE SETS

Mohd Ibrahim Shapiai[1], Zuwairie Ibrahim[1], Marzuki Khalid[1]
Lee Wen Jau[2], Vladimir Pavlovic[3] and Junzo Watada[4]

[1]Centre of Artificial Intelligent and Robotics (CAIRO)
Universiti Teknologi Malaysia, International Campus
54100 Kuala Lumpur, Malaysia
{ ibrahim; zuwairie }@fke.utm.my; marzuki@utm.my

[2]Department of ATTD Automation (APAC) Pathfinding
Intel Technology Sdn. Bhd.
Kulim, Malaysia
{ wen.jau.lee; soon.chuan.ong }@intel.com

[3]Department of Computer Science
Rutgers University
NJ 08854, United States
vladimir@cs.rutgers.edu

[4]Graduate School of Information and Systems
Waseda University
2-7 Hibikino, Wakamatsu, Kita-Kyushu 808-0135, Japan
junzow@osb.att.ne.jp

Received June 2010; revised January 2011

Abstract. *The function approximation problem is to find the appropriate relationship between a dependent and independent variable(s). Function approximation algorithms generally require sufficient samples to approximate a function. Insufficient samples may cause any function approximation algorithm to result in unsatisfactory predictions. To solve this problem, a function approximation algorithm called Weighted Kernel Regression (WKR), which is based on Nadaraya-Watson kernel regression (NWKR), is proposed. In the proposed framework, the original NWKR algorithm is enhanced by expressing the observed samples in a square kernel matrix. The WKR is trained to estimate the weight for the testing phase. The weight is estimated iteratively and governed by the error function to find a good approximation model. Four experiments are conducted to show the capability of the WKR. The results show that the proposed WKR model is effective in cases where the target function is non-linear and the given training sample is small. The performance of the WKR is also compared with other existing function approximation algorithms, such as artificial neural networks (ANN).*
Keywords: Weighted kernel regression, Small samples, Non-linear function, Artificial neural network

1. **Introduction.** The need for function approximation arises in many fields of applied mathematics. There are numerous function-approximation techniques available in the machine learning community. The modelling of function approximations using ANN has received significant attention from a number of researchers [1-3]. For example, the hybrid model of ANN with PSO has been proposed by [4,5] for function approximation. Genetic programming [6,7], evolutionary algorithms [8] and fuzzy systems [9,10] are other well-known techniques that can be found in the literature. However, most existing function

approximation algorithms perform well given sufficiently large samples. The performance of those function approximation algorithms degrades as the size of samples decreases.

Kernel regression, which is based on non-parametric statistics, explicitly utilises the available samples for function estimation. To find a non-linear relationship between input(s) X and output Y, kernel regression has been employed in many applications, such as pattern recognition, handwriting recognition, finance [11] and robotics [12].

Typical methods for solving the small sample function approximation problems rely on artificial sample approaches. Assuming that the artificial samples are relevant, with enough samples, the hypothesis will be a sufficiently close approximation to the actual value. Generating artificial samples is one way to incorporate prior information in machine learning [13]. Different methods of generating artificial samples have been proposed by Tsai and Li [14] and Huang and Moraga [15]. Tsai and Li proposed an algorithm to improve learning accuracy by combining the segmentation technique with an artificial sample generation method. In each segment, a simple linear regression line is calculated, and the boundary points (extremal points) are defined. The artificial samples are generated in each segment based on the estimated regression coefficients. The original samples and the artificial samples are used to train ANN with back propagation algorithm (ANNBP). In a different approach, Huang and Moraga proposed a Diffusion Neural Network (DNN) where the artificial samples are generated based on the principle of information diffusion. The original samples, artificial samples, and the corresponding probability values are used to train the ANNBP. However, these existing techniques were demonstrated only for two-dimensional problems.

The application of learning from small samples has gained increasing attention in many fields, such as semiconductor manufacturing for the assembly process, sparse prediction modelling [16], engine control simulation [17] and in the paper industry [18]. This study shows that the original NWKR is unable to approximate a function with small sample sets accurately. Hence, the modified kernel regression for function approximation is proposed to enhance and improve the original NWKR. Whereas the existing techniques to solve small samples rely on the ANNBP, a non-deterministic prediction model [19] that tends to produce inconsistent predictions, the proposed model produces consistent predictions due to the convexity of the weight estimation during training. The proposed model also does not require an artificial sample generation method to be incorporated in the model development. Finally, the ease of tuning the hyper-parameter model resembles the capability of the proposed technique when dealing with small samples.

The remainder of this paper is organised as follows: a brief review of existing function approximation algorithms is given in Section 2; the proposed approach is presented in Section 3; Section 4 includes a discussion of the experimental results; finally, the conclusion of this paper appears in Section 5.

2. **Function Approximation Algorithms.**

2.1. **Artificial neural network.**

The most popular ANN algorithm to approximate a function is the ANNBP [20,21]. The traditional ANNBP consists of an input layer, an output layer, and a set of one or more hidden layers. The numbers of nodes in the input and output layers correspond to the numbers of independent and dependent variable(s), respectively. However, there are no rules of thumb to determine the number of hidden layers and the numbers of nodes in the hidden layers. Traditional ANNBP suffers from instability and inaccuracy when the number of training samples is small. However, these problems can be overcome by the generalised regression neural network (GRNN) [22]. Thus, GRNN is often used for function approximation [23,24].

In general, the architecture of GRNN consists of 4 layers, as shown in Figure 1. The numbers of nodes in the input and output layers correspond to the numbers of independent and dependent variable(s), respectively. In the pattern layer, the number of nodes is defined based on the number of training samples. The summation layer consists of two types of neurons, S-summation neurons and a single D-summation neuron. As in ANN, the accuracy of the prediction is degraded if the number of samples is insufficient [15].

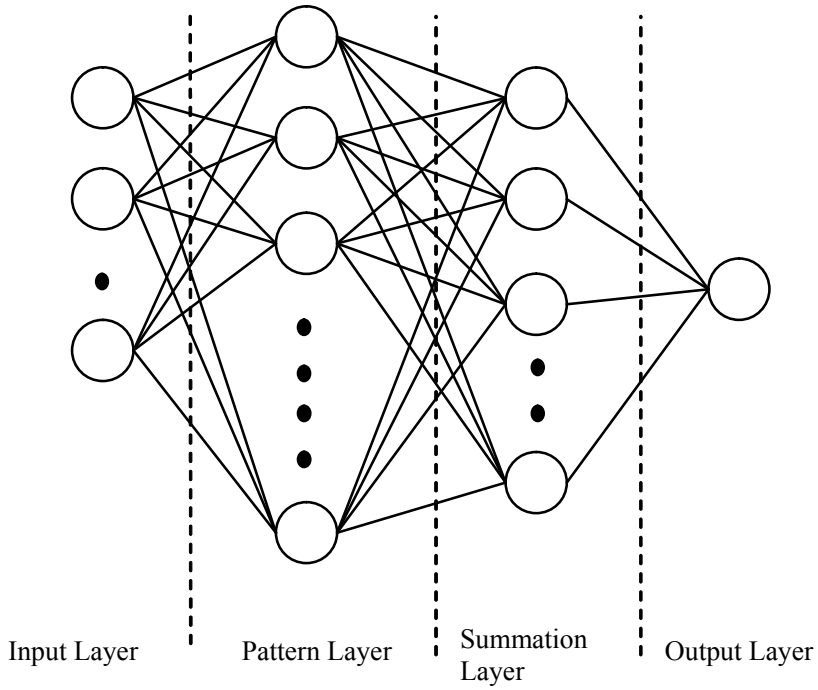

FIGURE 1. The architecture of generalized regression neural network (GRNN)

2.2. **Diffusion neural network.** DNN [15] is a modified ANNBP, which is based on the principle of information diffusion [25]. The principle is to create more samples artificially to fill some of the information gaps between the original training samples. DNN was derived to solve the small-sample problem by applying fuzzy set theory. Two artificial samples were artificially created for each of the original training samples. The original samples and the artificial samples were assigned associated possibility values for the input and the output values. The possibility value of 'one' was assigned to the original samples, and the possibility value for each of the artificial sample was based on the correlation coefficient of the given training samples. All samples with the associated possibility values were then used for the training step of the ANNBP. The architecture of DNN for function approximation of one independent variable and one dependent variable is shown in Figure 2. As DNN relies on ANNBP, its predictions still suffer from the same inaccuracy.

2.3. **Kernel regression.** Kernel regression, particularly the NWKR [12,26], is a non-parametric statistical technique to estimate the conditional expectation of a random variable. A research work has been carried out by [27] to approximate a non-linear function from small samples using NWKR. The kernel regression represents this estimate using a weighted combination of dependent variable samples, with weights determined by the proximity of the query input to the set of given input samples. This allows accurate interpolation and approximation in the vicinity of training samples. Kernels assign weights

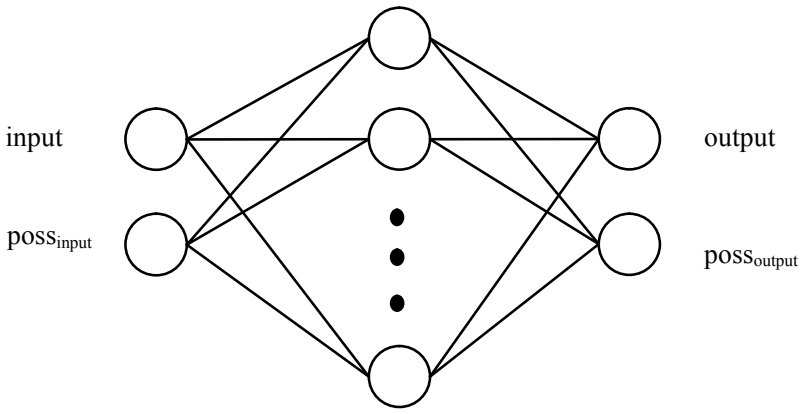

FIGURE 2. The architechture of DNN to approximate 2 dimensional non linear equation, 2-K-2 ANNBP network

to arbitrary samples based on their distance from the given samples, which is calculated using Equation (1) for the Gaussian kernel:

$$K_\sigma(x, x_i) = \frac{1}{\sqrt{2\pi}} \exp\left(\frac{-(x - x_i)^2}{\sigma_s}\right) \tag{1}$$

where σ_s denotes the smoothing parameter, and K_σ is a Gaussian kernel that is used to assign a weight, based on the Euclidean distance, to any arbitrary sample. The closer the arbitrary sample is to any given sample, the higher the weight that will be assigned on it. Several other types of kernel functions are commonly used, such as the uniform, triangle, epanechnikov, quartic (biweight), tricube (triweight) and cosine functions. Here, x_i is a list of observed independent variables, and x is an arbitrary point to be estimated.

The dependent variable y corresponding to any arbitrary x values can be estimated by using Equation (2).

$$\hat{y}_i(x, x_i) = \frac{\sum_{i=1}^{n} y_i K_\sigma(x, x_i)}{\sum_{i=1}^{n} K_\sigma(x, x_i)}, \quad i = 1, 2, \ldots, n \tag{2}$$

where n is the number of observed samples. For a higher d-dimensional estimate, the ith observation for each of the d independent variables is given in the vector X_i in Equation (3).

$$X_i = \begin{bmatrix} X_i^1 \\ \vdots \\ X_i^p \\ \vdots \\ X_i^d \end{bmatrix}, \quad i = 1, 2, \ldots, n \tag{3}$$

The estimated value of $\hat{y}$ can be calculated using Equation (4).

$$\hat{y}(X, X_i) = \frac{\sum_{i=1}^{n} y_i \left(\prod_{p=1}^{d} K_\sigma(X^p, X_i^p)\right)}{\sum_{i=1}^{n} \left(\prod_{p=1}^{d} K_\sigma(X^p, X_i^p)\right)} \tag{4}$$

3. **Weighted Kernel Regression.** An overview of the proposed technique is given in Figure 3. The proposed technique requires a series of steps to develop the prediction model. The details of each of the two phases will be explained in the following subsections.

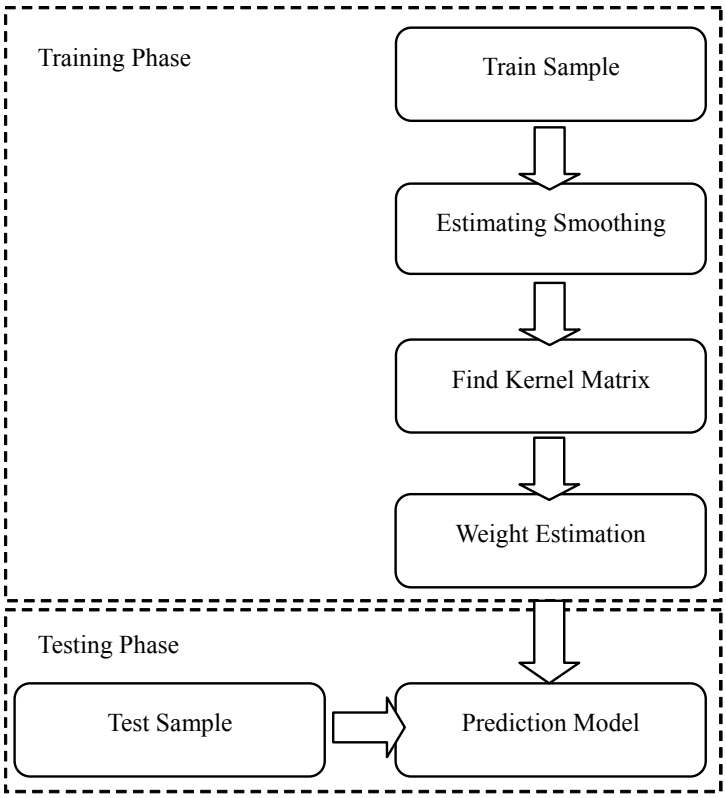

FIGURE 3. Overview of the proposed technique, WKR

3.1. **Training phase.** With an insufficient number of samples, popular model selection methods such as cross validation cannot be used [28,29]. As for NWKR, it is important to compromise between smoothness and fitness in selecting the smoothing parameter σ_s [30]. The proposed technique, based on NWKR theory, provides an easy method of tuning the hyper-parameters of the proposed model when dealing with small samples. The smoothing parameter for the proposed technique can be estimated using Equation (5).

$$\sigma_s = \max\left(\|X_{k+1}\|^2 - \|X_k\|^2\right), \text{ where } 1 < k < n-1 \text{ and } \|X_{k+1}\|^2 > \|X_k\|^2 \quad (5)$$

Initially, all the inputs of the available samples are arranged in ascending order of L_2-norm values. This setting is used in all of our experiments to estimate the smoothing parameter.

The kernel matrix $A = [a_{ij}]$, where $i = j = 1, \ldots, n$, with the generalised kernel matrix notation based on the Gaussian kernel, is given in Equation (6). The matrix A transforms the linear observed samples to non-linear problems by mapping the data into a higher dimensional feature space.

$$a_{ij} = \begin{cases} \dfrac{\prod_{p=1}^{d} K\left(X_i^p, X_j^p\right)}{\sum_{l=1}^{n}\left[\prod_{p=1}^{d} K\left(X_{i\vee j}^p, X_l^p\right)\right]}, & \text{if} \quad i \neq j \\ \dfrac{1}{\sum_{l=1}^{n}\left[\prod_{p=1}^{d} K\left(X_{i\vee j}^p, X_l^p\right)\right]}, & \text{if} \quad i = j \end{cases} \tag{6}$$

Once the kernel matrix is found, it is necessary to introduce the estimated weight. The estimated weight is determined based on the kernel matrix. The weight is updated iteratively by comparing the estimated values $\hat{y}_i$ to the actual value y_i. As the difference converges to a minimum value, after reaching the predefined iteration value, the training to estimate the weight will be stopped. Initially, arbitrary values are assigned to the weights. The weight is defined in the column vector, as shown in Equation (7).

$$W = \begin{bmatrix} w_1 & w_2 & \cdots & w_n \end{bmatrix}^T \tag{7}$$

The estimated $\hat{y}_i$, the error equation and the estimated weight equation are given by Equations (8)-(10), respectively.

$$\hat{y}_i = \sum_{i=1}^{n} w_j a_{ij} \tag{8}$$

$$E\left(W\right) = \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right) \tag{9}$$

$$\hat{W}\left(X\right) = \arg\min_{W} E\left(W\right) \tag{10}$$

3.2. **Testing phase.** Once the optimum weight is obtained, the model is ready to predict any unseen samples (test samples). The test samples can be predicted by using Equation (11).

$$\hat{y}\left(X, \hat{W}\right) = \frac{\sum_{i=1}^{n} \hat{W}_i \left(\prod_{p=1}^{d} K_\sigma\left(X^p, X_i^p\right)\right)}{\sum_{i=1}^{n}\left(\prod_{p=1}^{d} K_\sigma\left(X^p, X_i^p\right)\right)} \tag{11}$$

4. Experimental Results and Discussions.

4.1. **Experiment I.** The first experiment is conducted to demonstrate the advantage of WKR over the DNN model and the other techniques for small sample problems. The following test functions from [15] are used:

$$y = x^2, \qquad x \in [0, 1] \tag{12}$$

$$y = 0.01x + 0.02x^2 + 0.9x^3, \qquad x \in [0, 1] \tag{13}$$

$$y = 1 - \exp\left(-2x^4\right), \qquad x \in [0, 1] \tag{14}$$

Initially, all the parameter settings for each function approximation algorithms are predefined. The parameter settings are summarised in Table 1.

Test samples, were also generated with step $t = 0.01$ in the domain $[0, 1]$ to approximate all test functions.

4.1.1. *Approximating* $y = x^2$. In this experiment setting, the same training samples employed by Huang and Moraga are used for model validation, with x are equally spaced as shown in Table 2. The main purpose of this experiment setting is to highlight the capability of the proposed technique as compared to DNN; therefore, we only report the result from DNN for this experiment. Hence, for the rest of the experiment settings, we do not report the performance result using DNN technique due to the non-deterministic nature of ANN [19]. However, this result shows that the proposed technique possesses a better generalisation error than the error value of DNN from the literature [15]. A further validation will be carried out to approximate $y = x^2$ with randomly generated samples.

TABLE 1. Parameter settings for each function approximation algorithm

Technique	Parameter Settings
DNN	Input Layer (2 nodes), One Hidden Layer (15 nodes with sigmoid function), Output Layer (1 node with a linear function), momentum rate = 0.9, learning rate = 0.7 and stopping criterion when iteration = 6 000 000
WKR	$\sigma_s = \max\left(\lVert X_{k+1}\rVert^2 - \lVert X_k\rVert^2\right)$, iteration = 2000
NWKR	$\sigma_s = \max\left(\lVert X_{k+1}\rVert^2 - \lVert X_k\rVert^2\right)$
ANNBP	Input Layer (1 node), One Hidden Layer (15 nodes with sigmoid function), Output Layer (2 nodes with a linear function), momentum rate = 0.9, learning rate = 0.7 and stopping criteria when either training error MSE < 10e-6 or iteration = 1000 (whichever one is reached first)
GRNN	Two layer network, first layer (radial basis neurons), second layer (linear basis neurons) $spread = \max\left(\lVert X_{k+1}\rVert^2 - \lVert X_k\rVert^2\right)$

TABLE 2. The small training samples as taken from Huang and Moraga

	S1	S2	S3	S4	S5
x	0	0.25	0.5	0.75	1
y	0	0.0625	0.25	0.5625	1

The Mean Square Error (MSE), given in Equation (15), is used as the performance criterion to measure the error between the actual value and the predicted value.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_{actual} - \hat{y}_{predict})^2 \tag{15}$$

The performance of each of the techniques with x equally spaced are summarised in Table 3. The WKR is found to be the best technique, with the smallest measured MSE. It is interesting to note that the WKR has improved on the prediction performance of DNN for equally spaced samples by 99.96%.

For a further evaluation of the proposed technique, the experiment is repeated 10 times, and the average of MSE in Equation (15) is measured as the performance indicator. In

TABLE 3. Comparison of the performance of all function approximation algorithms with x equally spaced

Technique	MSE
DNN [15]	0.002358
WKR	0.000001
NWKR	0.016546
ANNBP	0.082472
GRNN	0.002980

each run of the experiment, only five randomly generated samples are used for training. The same test samples are used to validate the performance of each algorithm. The computational results are summarised in Table 4. WKR obviously outperforms the other techniques, as shown by the smallest average MSE, which is 0.000597. Figure 4 shows the approximation function of $y = x^2$ with the minimum MSE from each technique. Note that the proposed WKR approximation overlaps the real function, in which it shows the highest accuracy.

TABLE 4. Results of 10 experiments to approximate $y = x^2$, with each data set consisting of 5 randomly generated samples

Technique	Average MSE	Standard Deviation	Min MSE	Max MSE
WKR	0.000597	0.001185	0.000010	0.003862
NWKR	0.053949	0.011657	0.032627	0.077177
ANNBP	0.090062	0.072029	0.017829	0.255362
GRNN	0.030940	0.014552	0.013625	0.061304

Figure 5 shows the graph of the average MSE as a function of the number of training samples. Based on the experimental result, the average MSE gradually decreases as the number of training samples increases. Obviously, the performance of the ANNBP is far behind that of the other techniques when the samples are small. WKR requires the least number of samples to approximate the non-linear function with high accuracy. Meanwhile, ANNBP requires at least 30 samples to achieve the accuracy of the WKR. However, when the training samples are not well distributed within the predefined domain, or in other words, when the training samples are close together, the performance of the WKR degrades and causes a large MSE.

TABLE 5. Results of 10 experiments to approximate $y = 0.01x + 0.02x^2 + 0.9x^3$, with each data set consisting of 5 randomly generated samples

Technique	Average MSE	Standard Deviation	Min MSE	Max MSE
WKR	0.000229	0.000260	0.000002	0.000734
NWKR	0.037149	0.011053	0.021916	0.054326
ANNBP	0.073572	0.065629	0.028958	0.249295
GRNN	0.021822	0.012239	0.006135	0.039757

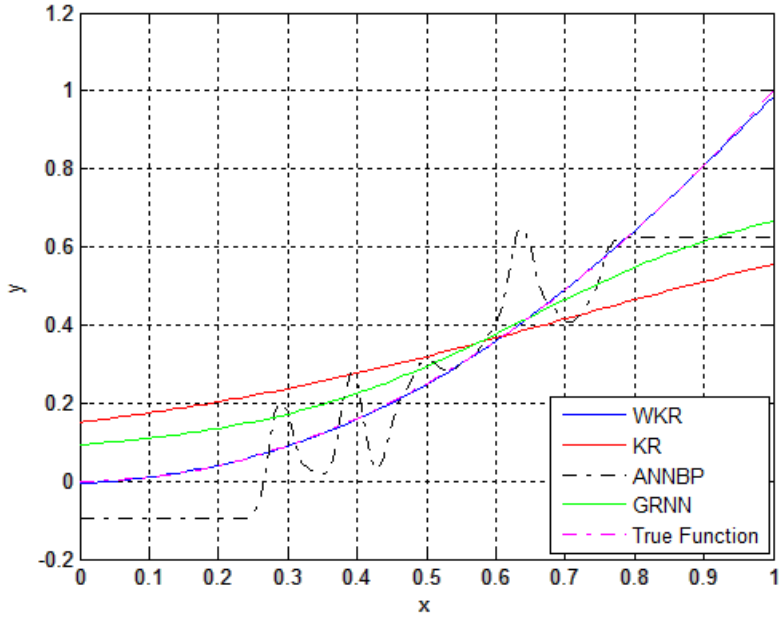

FIGURE 4. Comparison of the 4 minimum MSE of each of the function approximation algorithm with only 5 random samples and the real function; the WKR approximation overlaps the real function

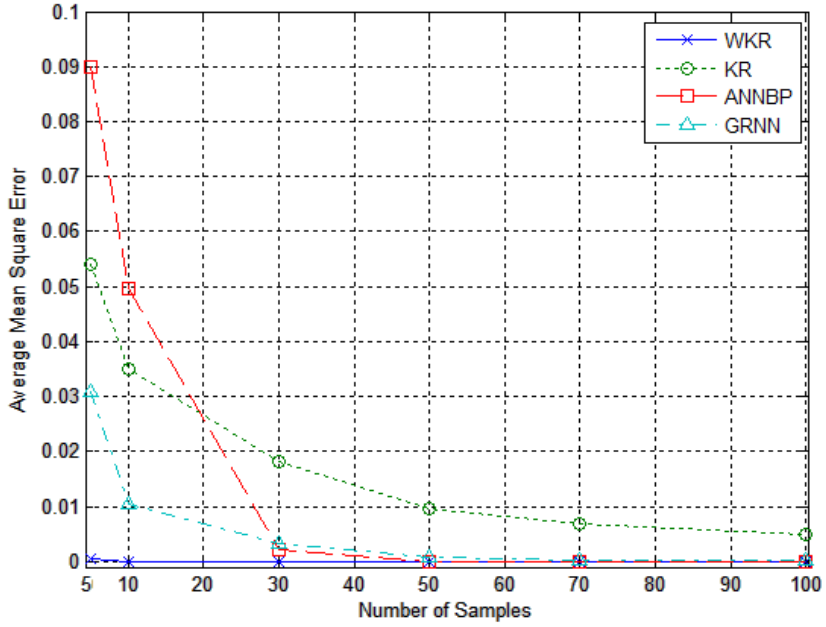

FIGURE 5. The improved approximation performance, average MSE, with increasing numbers of samples for each technique

4.1.2. *Approximating* $y = 0.01x + 0.02x^2 + 0.9x^3$. We use the same experimental setting from the previous section. Table 5 shows the result of the simulation experiment with 5 randomly generated samples. For this experiment, WKR produces the highest accuracy and improves on the prediction performance of DNN by 80.89%.

TABLE 6. Results of 10 experiments to approximate $y = 1 - exp(-2x^4)$, with each data set consisting of 5 randomly generated

Technique	Average MSE	Standard Deviation	Min MSE	Max MSE
WKR	0.002079	0.002224	2.94E-04	0.007501
NWKR	0.054276	0.016177	0.034817	0.082906
ANNBP	0.062496	0.035381	0.027976	0.134150
GRNN	0.037584	0.020203	0.012419	0.073868

4.1.3. *Approximating* $y = 1 - \exp(-2x^4)$. We use the same experimental setting from the previous section. Table 6 shows the result of the simulation experiment with 5 randomly generated samples. For this experiment, WKR produces the highest accuracy and improves on the prediction performance of DNN by 51.56%.

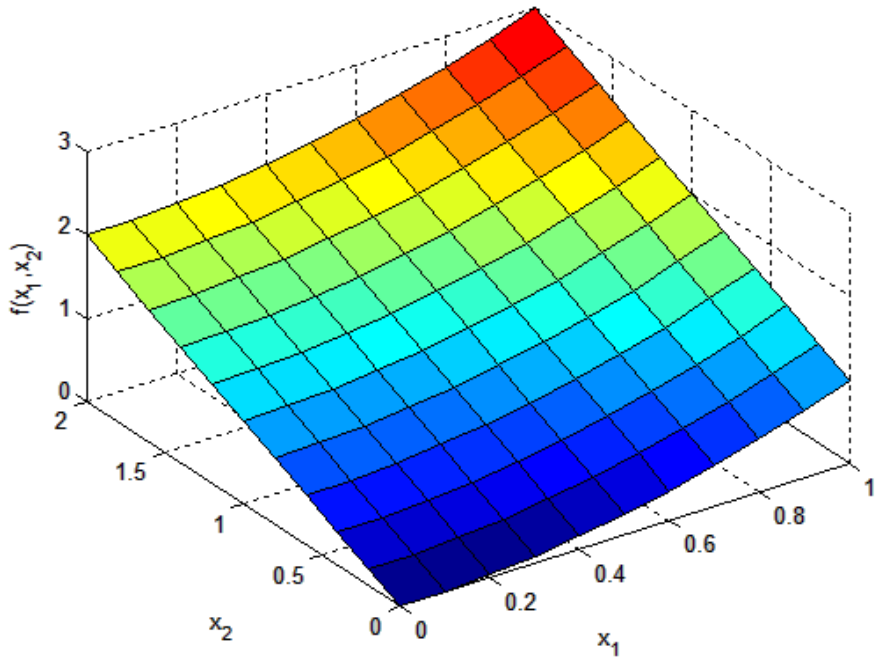

FIGURE 6. The real surface of the function $y = x_1^2 + x_2$

4.2. **Experiment II.** This experiment is conducted to evaluate the performance of WKR in approximating a non-linear surface, given in Equation (16). This equation consists of two independent variables and one dependent variable, so its graph is a surface in $R^2 \times R$. Figure 6 shows the real surface of $y = x_1^2 + x_2$.

$$y = x_1^2 + x_2 \tag{16}$$

In order to demonstrate the surface approximation problem with an insufficient number of samples, the sparse and small training samples were randomly generated within the predefined range in every experiment. As in the first experiment, the experiment was repeated 10 times with 5 randomly generated samples. Again, the average MSE was chosen as the performance measure. The range of x_1 is set between 0 and 1 and the range

TABLE 7. Parameter settings for each of the function approximation algorithms

Technique	Parameter Settings
WKR	$\sigma_s = \max\left(\|X_{k+1}\|^2 - \|X_k\|^2\right)$, iteration = 2000
NWKR	$\sigma_s = \max\left(\|X_{k+1}\|^2 - \|X_k\|^2\right)$
ANNBP	Input Layer (1 node), One Hidden Layer (15 nodes with sigmoid function), Output Layer (2 nodes with a linear function), momentum rate = 0.9, learning rate = 0.7 and stopping criteria when either training error MSE < 10e-6 or iteration = 1000 (whichever one is reached first)
GRNN	Two layer network, first layer (radial basis neurons), second layer (linear basis neurons) $spread = \max\left(\|X_{k+1}\|^2 - \|X_k\|^2\right)$

of x_2 is set between 0 and 2. The predefined range is set particularly to emphasise the sparseness of the training samples.

Initially, all the parameter settings for each of the function approximation algorithms were predefined. Those parameter settings are summarised in Table 7. To evaluate the learning accuracy quantitatively, all techniques were tested with all possible combinations of the following generated test samples, $\left[T_{s_j}^{x_1}, T_{s_j}^{x_2}\right]$

- $T_s^{x_1} = \left\{ T_{s_j}^{x_1} | j = 1, 2, \cdots, 11 \right\} = \{0, 0.1, \cdots, 1\}$
- $T_s^{x_2} = \left\{ T_{s_j}^{x_2} | j = 1, 2, \cdots, 11 \right\} = \{0, 0.1, \cdots, 2\}$

The results of the experiments are summarised in Table 8. In addition, Figure 7 shows the approximated surfaces with the minimum MSE, which is calculated after 10 runs of each technique. Figure 7 shows that the WKR successfully approximates the surfaces with the highest accuracy.

Figure 8 shows that the performance of all techniques depends on the number of training samples. Generally, the approximation improves with the number of available training samples. As the number of samples increases, the average MSE decreases. The proposed WKR requires the least number of samples before the average MSE become constant when the number of samples is 30. ANNBP struggles to improve the performance of the approximation with the limited number of samples. Again, when the training samples are not well-distributed within the predefined domain, or, in other words, when the training samples are close together, the performance of the WKR is degraded and causes a large MSE.

5. **Conclusions.** In general, given a limited number of samples, non-linear function and surface approximation are extremely difficult. Hence, an enhanced kernel regression called weighted kernel regression (WKR) is proposed to approximate non-linear functions and surfaces with small samples. By introducing iteratively computed weights to the kernel regression, the proposed WKR has successfully improved the performance of function and surface approximations. Four experiments are conducted to show the effectiveness and practicability of the WKR in solving small sample problems. It is shown that the proposed approach is superior to KR, ANNBP, GRNN and also DNN. However, if the training samples are not well distributed in the domain range, the performance of the

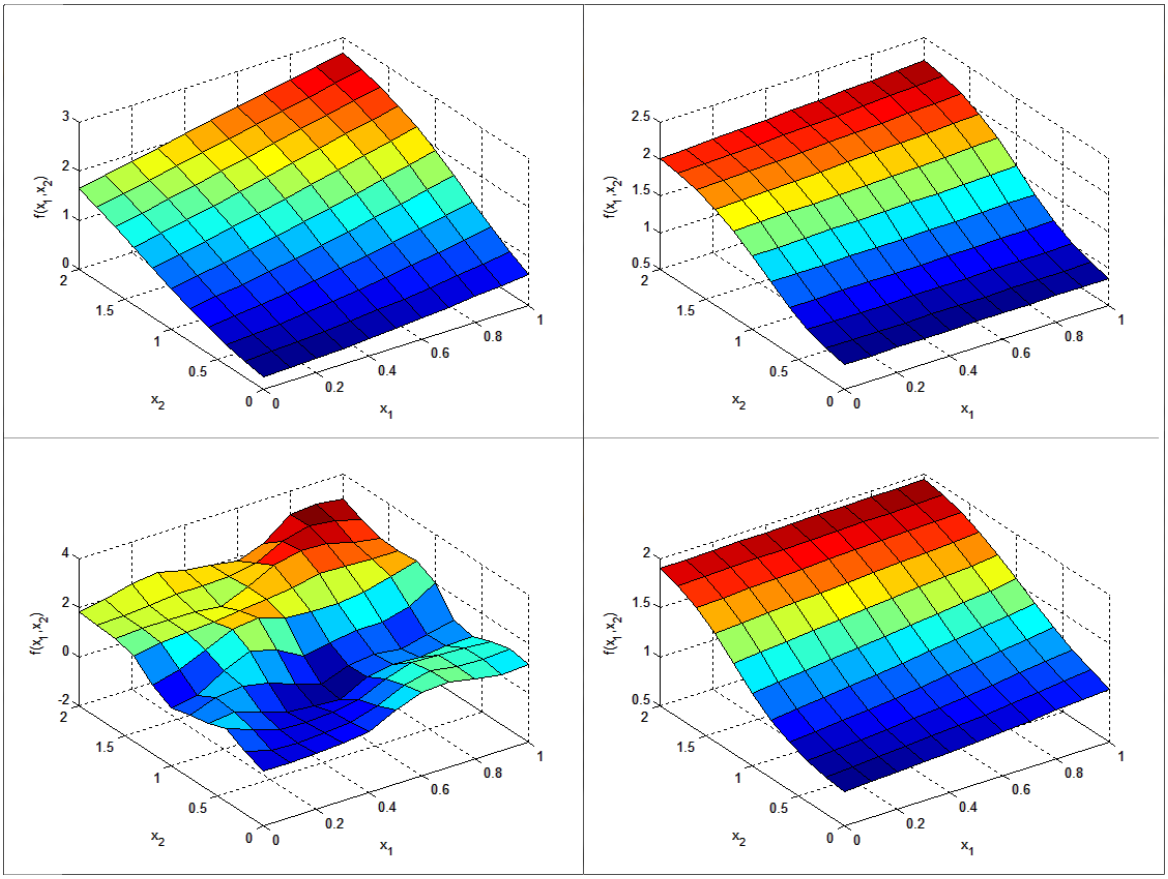

FIGURE 7. Surface approximation with the minimum MSE from every technique with only 5 random samples: WKR (top left), KR (top right), ANNBP (bottom left) and GRNN (bottom right)

TABLE 8. Results of 10 experiments with each data set consisting of 5 randomly generated samples

Technique	Average MSE	Standard Deviation	Min MSE	Max MSE
WKR	0.084271	0.055658	0.017717	0.190274
NWKR	0.246330	0.069160	0.151632	0.331272
ANNBP	0.691426	0.556066	0.182981	2.102804
GRNN	0.239673	0.053704	0.156651	0.303544

proposed technique is degraded and causes a large MSE. In the future, the proposed technique will be further improved by introducing artificial samples and an alternative approach to estimate the weight of the WKR.

Acknowledgment. This work was financially supported by Universiti Teknologi Malaysia and INTEL Corporation by the UTM-INTEL Research Collaboration Fund (Vote 73332).

REFERENCES

[1] G. Cybenko, Approximation by superpositions of a sigmoidal function, *Mathematics of Control, Signals, and Systems*, vol.2, pp.303-314, 1989.

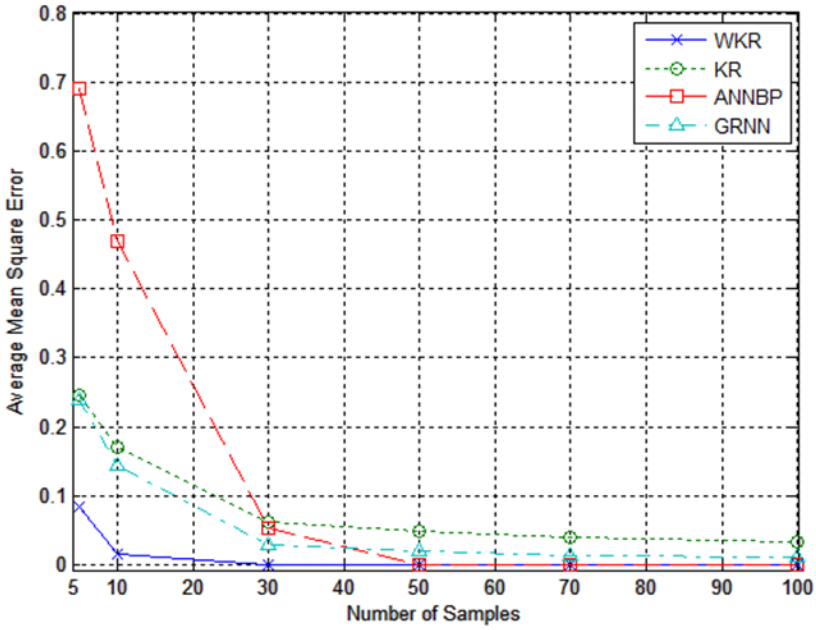

FIGURE 8. The improvement in surface approximation performance of each technique as indicated by the average MSE when the number of samples increases

[2] K. Funahashi, On the approximate realization of continuous mappings by neural networks, *Neural Networks*, vol.2, pp.183-192, 1989.
[3] K. Hornik, Approximation capabilities of multilayer feedforward networks, *Neural Networks*, vol.4, pp.251-257, 1991.
[4] T. Su, J. Jhang and C. Hou, A hybrid artificial neural networks and particle swarm optimization for function approximation, *International Journal of Innovative Computing, Information and Control*, vol.4, no.9, pp.2449-2457, 2008.
[5] J. Zhang et al., A hybrid particle swarm optimization-back-propagation algorithm for feedforward neural network training, *Applied Mathematics and Computation*, vol.185, pp.1026-1037, 2007.
[6] Y. Yeun et al., Function approximations by superimposing genetic programming trees: With applications to engineering problems, *Information Sciences*, vol.122, pp.259-280, 2000.
[7] K. Rodríguez-Vázquez and C. Oliver-Morales, Multi-branches genetic programming as a tool for function approximation, *Lecture Notes in Computer Science*, vol.3103, pp.719-721, 2004.
[8] A. Hamzeh and A. Rahmani, Approximating the environmental reinforcement signal with non-linear polynomials using learning classifier systems, *International Journal of Innovative Computing, Information and Control*, vol.4, no.7, pp.1797-1809, 2008.
[9] B. Kosko, Fuzzy systems as universal approximators, *IEEE Transactions on Computers*, vol.43, pp.1329-1333, 1994.
[10] L. Wang, Fuzzy systems are universal approximators, *International Conference Fuzzy Systems*, San Diego, CA, pp.1163-1170, 1992.
[11] B. Xu and B. Wu, On nonparametric estimation for the growth of total factor productivity: A study on china and its four eastern provinces, *International Journal of Innovative Computing, Information and Control*, vol.3, no.1, pp.141-150, 2007.
[12] G. Watson, Smooth regression analysis, *Sankhy: The Indian Journal of Statistics*, Series A, vol.26, pp.359-372, 1964.
[13] P. Niyogi et al., Incorporating prior information in machine learning by creating virtual examples, *Proc. of the IEEE*, vol.86, 1998.
[14] T. Tsai and D. Li, Approximate modeling for high order non-linear functions using small sample sets, *Expert Systems with Applications*, vol.34, pp.564-569, 2008.

[15] C. Huang and C. Moraga, A diffusion-neural-network for learning from small samples, *International Journal of Approximate Reasoning*, vol.35, pp.137-161, 2004.
[16] W. Lee and S. Ong, Learning from small data sets to improve assembly semiconductor manufacturing processes, *The 2nd ICCAE 2010*, Singapore, pp.50-54, 2010.
[17] G. Bloch et al., Support vector regression from simulation data and few experimental samples, *Information Sciences*, vol.178, pp.3813-3827, 2008.
[18] R. Lanouette et al., Process modeling with neural networks using small experimental datasets, *Computers & Chemical Engineering*, vol.23, pp.1167-1176, 1999.
[19] M. Graczyk et al., Nonparametric statistical analysis of machine learning algorithms for regression problems, *Knowledge-Based and Intelligent Information and Engineering Systems*, pp.111-120, 2010.
[20] P. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, Ph.D. Thesis, Harvard University, Cambridge, MA, 1974.
[21] D. E. Rumelhart et al., Learning representations by back-propagating errors, *Nature*, vol.323, pp.553-536, 1986.
[22] X. J. Guo et al., An empirical research for forecasting model based on the generalization regression neural network, *International Conference on Automation and Logistics*, Qungdao, China, 2008.
[23] D. Patterson, *Artificial Neural Networks*, Prentice Hall, Singapore, 1996.
[24] C. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, USA, 1995.
[25] C. Huang and Y. Shi, *Towards Efficient Fuzzy Information Processing: Using the Principle of Information Diffusion*, Physica Verlag, Heidelberg, Germany, 2002.
[26] È. Nadaraya, On estimating regression, *Theory of Probability and Its Applications*, vol.9, pp.157-159, 1964.
[27] M. I. Shapiai et al., A non-linear function approximation from small samples based on nadaraya-watson kernel regression, *The 2nd International Conference on Computational Intelligence, Communication Systems and Networks*, Liverpool, UK, pp.28-32, 2010.
[28] M. Jang and S. Cho, Observational learning algorithm for an ensemble of neural networks, *Pattern Analysis & Applications*, vol.5, pp.154-167, 2002.
[29] A. Isaksson et al., Cross-validation and bootstrapping are unreliable in small sample classification, *Pattern Recognition Letters*, vol.29, pp.1960-1965, 2008.
[30] J. Zhang et al., An improved kernel regression method based on Taylor expansion, *Applied Mathematics and Computation*, vol.193, pp.419-429, 2007.