# Approximation of Multidimensional Functions by Radon Ridge Feedforward Neural Networks

[1]Prof. Dr. Reyadh S. Naoum , [2]Najla'a M. Hussein

[1]Department of Mathematics, College of Science, University of Baghdad, Baghdad, Iraq
[2]Department of Computer Science, College of Science, University of Baghdad. Baghdad, Iraq

## Abstract

The main result of this paper is to present a new method to approximate multidimensional function by using feedforward ridge neural network with application of Radon Transform, and its inverse, to reduce the dimension of the space. This method consist of four stages: First, by using the Radon Transform, the multidimensional function can be reduced to several simpler one dimensional functions. Second, each of the one dimensional functions is approximated by using neural network technique into neural subnetworks. Third, these neural subnetworks are combined together to form the final approximation neural network. Four, using the inverse of Radon Transform to this final approximation neural network to get the approximation to the given function. Also, in this paper presenting an upper bound of the $L_\infty$ approximate error. Also, apply the above method to an example and compare the results with those in [2], and our numerical results are superior to those in [2].

## Introduction

Approximations of multidimensional function have been studied by many researchers such as Ciesielski and Sachia [2], Ellacott [4] and Pinkus [5]. Ciesielski and Sachia, [2], focused on a development of a constructive formula for the upper bound of $L_\infty$ error approximation. Ellacott, [4], proved that a semilinear feedforward network with one hidden layer can uniformly approximate any continuous function in C(K) where K is a compact set in $R^s$ and s is a positive integer. Pinkus, [5], presented an algorithms for approximating the multidimensional function by using ridge feedforward neural network.

Johann Radon, [6], showed that if f is continuous function and has a compact support, then the Radon Transform of f is uniquely determined by integrating along all lines in the domain $X \subset C(K)$.

The main result of this paper is the construction of a new method for approximating a Multidimensional function by feedforward network with one hidden layer of sigmoidal units and a linear output. Also, presenting an upper bound of $L_\infty$ error approximation. The method consists of four stages: First, with the use of Radon transform, [6], the problem of multidimensional approximation is replaced by, several simpler, one dimensional problems. Second, in each of the one dimensional problems the approximation subnetworks is found. Third, the subnetworks are combined together to form the final approximation network. Fourth, using inverse Radon Transform to this final approximation network to get the approximation to the given problem.

## Approximation of 1-D Functions

This section discusses the approximation of one dimensional function $f \in X$ (X is a normed linear space).

**Definition (1)**

A ridge function is a multivariate function $F: R^s \to R$ of the form

$$F(x_1, x_2, \ldots, x_s) = g(a_1x_1 + a_2x_2 + \ldots + a_sx_s) = g(\mathbf{a}.\mathbf{x}) \quad \ldots(1)$$

where $g: R \to R$, $\mathbf{a} = (a_1, a_2, \ldots, a_s) \in R^s \setminus \{0\}$ and s is positive integer.

In other words, it is a multivariate function constant on the parallel hyperplanes $\mathbf{a}.\mathbf{x}=c$, $c \in R$. The vector $\mathbf{a} \in R^s \setminus \{0\}$ is generally called the direction.

**Definition (2)**

Let f be a continuous function on a normed linear space X which can be well approximated by a linear combination of ridge functions where $g_i \in C(R)$ and $\phi_i \in X^*$ ($X^*$ is the set of all linear continuous functional on X), [1].

$$f \approx \sum_{i=1}^{m} c_i\, g_i \circ \phi_i \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2)$$

For some applications, such as in neural networks technique, it is very desirable to employ a single


ISSN:1814-5922

function g in the ridge functions.

**One Hidden Layered Neural Networks (3)**

An artificial neural network is a mathematical model of the human brain. In many literature different types of neural network models had been studied, but this paper take the case of a feedforward network with input layer, one hidden layer and output layer. This network consists of a large number of computing units arranged schematically in three layers as shown in figure (1):

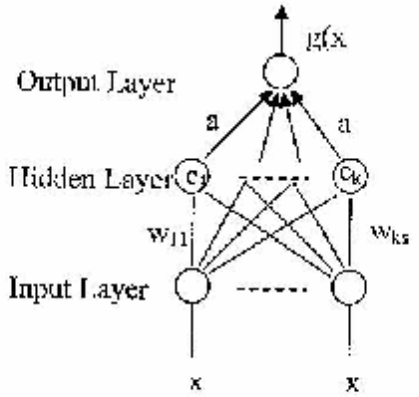


**Figure (1) Layers in a neural network**

Each unit of the input layer can be connected to each unit of the hidden layer. This connection has associated with it a weight, which is a real number. The weight attached to the link from input unit j to unit i on the hidden layer is denoted by $w_{ij}$. In a typical operation, each unit on the input layer will contain a real number. Let the $j^{th}$ unit contain the real number $x_j$. Then unit i on the hidden layer will receive from unit j on the input layer the quantity $w_{ij}x_j$. The total input that unit i receives from all the input units is then $c_i = \sum_{j=1}^{s} w_{ij}x_j$. Unit i on the hidden layer now processes this input with a continuous sigmoidal function $\theta : R \to R$ which is a given fixed univariate function ( called the activation function) but it shift to the argument by a real scalar $b_i$ (which is called the bias) and outputs the real number $\theta\left(\sum_{j=1}^{s} w_{ij}x_j + b_i\right)$. This output is then transmitted, with a weight $a_i$, to the output unit. The total output is then

$$g(x) = \sum_{i=1}^{n} a_i \theta\left(\sum_{j=1}^{s} w_{ij}x_j + b_i\right) \quad \ldots\ldots\ldots\ldots(3)$$

By definition (2), any continuous function of s variables can be approximated by a function of the form g in equation (3). Thus by suitable adjusting of the parameters k, $a_j, b_j$ and $w_{ij}$, this reproducing approximately any desired output with the artificial neural network.

Note that for each $b_j \in R$ and $w \in R^s \setminus \{0\}$ the activation function $\theta(w \cdot x + b_j)$ is a ridge function. Thus a lower bound on the degree of approximation by such function is given by ridge functions; that is, approximating a complicated function by simple functions. So for the simplification of computational complexity the feedforward network take the simplest case (one input, one hidden layer with n processing units and one output).

Then from definition (2) and the above paragraph to approximate a continuous function $\varphi : R \to R$ is

by considering the approximation form of the neural network $N : R \to R$ as, [1]:

$$N(x) = \sum_{i=1}^{n} a_i \theta(w_i x + b_i) \quad \ldots\ldots\ldots\ldots(4)$$

Activation functions $\theta : R \to R$ used in practice have the property of being monotonic increasing, bounded and sigmoidal, which means that $\lim_{t \to \infty} \theta(t) = 1$ and $\lim_{t \to -\infty} \theta(t) = 0$, they are also continuous and smooth. The most popular choice of the activation function is the logistic function which has the form

$$y = \theta(x) = \frac{1}{1 + e^{-x}} \quad \ldots(5)$$

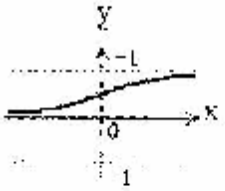


**Figure (2) Logistic Function**

Let us assume that the sigmoid (activation) function of our network is the hyperbolic tangent function which has the form

$$y = \theta(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad \ldots\ldots(6)$$

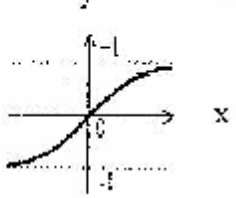


**Figure (3) Hyperbolic Tangent Function**

**Definition(4)**

The modulus of continuity of a function $\varphi$ is defined by

$$\omega(\varphi,\delta)= \sup_{\substack{x,y\in K \\ |x-y|<\delta}} |\varphi(x)-\varphi(y)| \quad \ldots\ldots\ldots\ldots\ldots(7)$$

Notice that, if $\varphi$ is a continuous real-valued function defined on a compact set then its necessarily uniformly continuous, so have $\omega$ is finite and $\omega \to 0$ as $\delta > 0$, [7].

**Definition(5)**

Let X be any closed and bounded (compact) set in $R^s$, and C(X) denote the set of all continuous real valued functions on X , C(X) is normed space with respect to the norm

$$\|\varphi\|_\infty = \sup_{x\in X} |\varphi(x)| \ , \varphi \in C(X) \quad \ldots\ldots\ldots\ldots\ldots(8)$$

The norm $\|.\|_\infty$ is called uniform norm.

The following theorem from [4] presents an upper bound for the error but this paper gives a different proof so as to be suitable for our problem.

**Theorem(6)**

Let the function $\varphi : R \to R$ be continuous on the compact set K in R. Then the upper bound of the approximate error of the function $\varphi$ by the network $\aleph$ on K is

$$\|\varphi - \aleph_n\|_\infty \le C\, \omega\left(\varphi, \frac{1}{n}\right) \quad \ldots\ldots\ldots\ldots\ldots(9)$$

where C is a constant which is independent of $\varphi$, suppose $\theta$ is continuous function (on R) and sigmoidal. Then we may choose $C=4\div 2S$ where $S = \sup_x |\theta(x)|$ for $x\in R$.

**Proof**

For simplicity but without much loss of generality, choose K to be the interval [0,1] ,and let $n\in N$. Consider the step function $h_n(x)$ which takes the value $\varphi\left(\frac{v}{n}\right)$ in the interval $\frac{v}{n} \le x \le \frac{(v-1)}{n}$.

Thus from definition (4) get the inequality

$$|\varphi - h_n|_\infty \le \omega\left(\varphi, \frac{1}{n}\right) \quad \ldots\ldots\ldots\ldots\ldots(10)$$

It is convenient to write $h_n$ such that to satisfy (10) as

$$h_n(x) = \varphi(0) + \sum_{v=1}^{\mu} \left\{ \varphi\left(\frac{v}{n}\right) - \varphi\left(\frac{(v-1)}{n}\right) \right\}, \quad \ldots\ldots\ldots(11)$$

where $\mu$ is the largest integer which does not exceed nx.

Since $\|\varphi - h_n\|_\infty = \sup_{x\in K} |\varphi(x) - h_n(x)| \quad \ldots(12)$

$$\therefore |\varphi(x)-h_n(x)| = \left|\varphi(x) - \varphi(0) - \sum_{v=1}^{\mu} \{\varphi(x_v)-\varphi(x_{v-1})\}\right|$$
$$= |\{\varphi(x)-\varphi(0)-[\varphi(x_1)-\varphi(0)+\varphi(x_2)-\varphi(x_1)+\ldots-\varphi(x_\mu)]\}$$
$$- \varphi(x) - \varphi(x_\mu)|$$

$\ldots\ldots\ldots(13)$

Now consider a continuous sigmoidal function $\theta$ such as logistic function, and replace x by ax for some $a > 1$, this implies that steeping $\theta$ in the transitional region around $x = 0$.

In fact as $a \to \infty$, $\theta(ax) \to 0$ as $x < 0$, $\theta(ax) > 1$ as $x > 0$ $\quad \ldots(14)$

In other words $\theta$ converges pointwise to a simple threshold function $\Psi$ with the value $\theta(0)$ at 0. Where

$$\Psi(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \ge 0 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots(15)$$

Thus $\theta(ax) - \theta(a(x-1)) \to \Psi_1(x)$, where $\Psi_1$ is the unit step function

$$\Psi_1(x) = \begin{cases} 1 & \text{if } x \in (0,1) \\ 0 & \text{otherwise} \end{cases} \quad \ldots\ldots\ldots\ldots\ldots(16)$$

Since $\Psi_1(x)$ is discontinuous at 0 and 1, the convergence cannot be uniform.

So from (10) and the above explanation beyond it, to get uniform approximation of $\varphi$ by $\aleph_n$, define $A_n$

to be the smallest positive integer such that $|\theta(x)| \le n^{-1}$ for $x \le -A_n$ and $(1-n^{-1}) \le \theta(x) \le (1+n^{-1})$ for $x \ge A_n$. The quasi interpolate $\aleph_n$ can define as

$$\aleph_n(x) = \varphi(0) + \sum_{v=1}^{n} \left\{ \varphi\left(\frac{v}{n}\right) - \varphi\left(\frac{(v-1)}{n}\right) \right\} \theta(A_n(nx-v)) \quad \ldots(17)$$

for $x\in[0,1]$. Now,

$$\|\varphi - \aleph_n\|_\infty = \|\varphi - h_n + h_n - \aleph_n\|_\infty \le \|\varphi - h_n\|_\infty + \|h_n - \aleph_n\|_\infty$$

$\ldots\ldots\ldots(18)$

Since $\|\varphi - h_n\|_\infty \le \omega\left(\varphi, \frac{1}{n}\right)$ then only need the second term in the above equation to be considered.

Now for any $x\in[0,1]$ with $\mu$ defined as in (11), have the following

$$h_n(x) - \aleph_n(x) = \varphi(0) + \sum_{v=1}^{\mu} \left\{ \varphi\left(\frac{v}{n}\right) - \varphi\left(\frac{(v-1)}{n}\right) \right\} \{1 - \theta(A_n(nx - v))\}$$



ISSN:1814-5922

$$\sum_{v=\mu+1}^{\infty}\left\{\varphi\left(\frac{v}{n}\right)-\varphi\left(\frac{(v-1)}{n}\right)\right\}\theta(A_n(nx-v)).....(19)$$

Now $v \le \mu-1$ implies $nx-v \ge 1$ so $\left|1-\theta(A_n(nx-v))\right| \le \frac{1}{n}$ (by the definition of $A_n$). Similarly $v > \mu+2$ implies $\theta(A_n(nx-v))\le \frac{1}{n}$. Thus

$$|h_n(x)-K_n(x)| \le \omega\left(\varphi,\frac{1}{n}\right)+\left|\left\{\varphi\left(\frac{\mu}{n}\right)-\varphi\left(\frac{\mu-1}{n}\right)\right\}+\left\{\varphi\left(\frac{\mu-1}{n}\right)\right\}\theta(A_n(nx-\mu-1))\right|(20)$$

The second term on the right-hand side is bounded by $2(1+S)\,\omega\left(\varphi,\frac{1}{n}\right)$, which completes the proof.

## Decompose of multidimensional functions

This section describe briefly how to reduce the dimension of the multidimensional function by using the Radon Transform and select the function $f$ from a class D of $C^{\infty}$ functions with compact support. The functions in class D has a nice properties: the Radon transform of f may differentiated as often as desired, and changes in the order of various integrations may be made with full confidence.

Let $\Omega$ be an open subset of $R^S$ with a smooth bounded boundary $\Gamma$. Then the compact support of a function $f$ is defined as follows:

**Definition(7)**

Let $f:\Omega \to R$. Let $\overline{K}_f$ be the closure of the set $K_f$, where $K_f=\{x \in \Omega \ f(x)\neq 0\}$ is called the support of f, denoted by supp f. f is said to have compact support if it is zero outside a compact subset of $\Omega$, i.e. $\overline{K}_f$ is compact.

**Remark (8)**

$D(\Omega)$ is the space of infinitely differentiable functions with compact support in $\Omega$. $D(\Omega)\neq\Phi$.

**Definition(9)**

The Dirac mass concentrated at the point $\alpha$ or the Dirac delta function concentrated at the point $\alpha$, which is denoted by $\delta(x-\alpha)$ or $\delta_{\alpha}$, is defined by

$$(\delta_{\alpha},f)=\int_{\Omega}\delta_{\alpha}(x)f(x)dx=f(\alpha) \quad f\in D(\Omega).....(21)$$

In particular, for $\alpha=0$, the Dirac mass concentrated at the origin denoted by $\delta$ is defined as

$$(\delta,f)=\int_{\Omega}\delta(x)f(x)dx=f(0) \quad f\in D(\Omega).....(22)$$

Actually delta mass concentrated at $\alpha$ is not a function but a distribution.

**Definition(10)**

Let $f:R^S\to R$ be $C^{\infty}$ function with compact support. The continuous Radon Transform of the function f represents an image of a collection of projections along various directions (angels) in $R^S$. In general case, given a function f defined on $R^S$, the Radon Transform of f, designated by $\hat{f}$, is determined by integrating over each hyperplane in the space $R^S$ and is defined by

$$\hat{f}(p,\xi)=Rf=\int_{R^S} f(x)\,\delta(p-\xi,x)\,dx .....(23)$$

where $dx=dx_1dx_2\ldots dx_S$.

$\delta$ is the Dirac delta function.

$\xi$ is a unit vector in $R^S$ that defines the orientation of a hyperplane with equation

$p=\xi,x=\xi_1x_1+\xi_2x_2+\cdots+\xi_Sx_S$ (p is the orientation of the hyperplane).

Thus $\hat{f}(p,\xi)$ must be known for all p and $\xi$. [3].

**Remark(11)**

The discretisation is a major difficulty in applying Radon Transform in general. The simplest form of discrete Radon Transform is to select finite number on the angular variable of projection to produce the unit vector $\xi_j$ ( j = 1, 2, ..., L where L $\in$ N, L is the number of projection), then take the summation on the discrete data $x_i$ ( i =1,2, ..., m where $m\in N$ and $x_i\in R^S$) of the function $f\in C(R^S)$.

Then the discrete Radon Transform is defined by

$$\hat{g}_j(p,\xi_j)=\sum_{i=1}^{m} f(x_i)\,\delta(p-\xi_j\cdot x_i) .....(24).$$

**Example(12)**

Let $f(x,y)=e^{-x^2-y^2}$. Then

$$\hat{f}=Rf=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-x^2-y^2}\,\delta(p-\xi_1x-\xi_2y)dxdy...(25)$$

Now make the orthogonal linear transformation (in matrix notation)

$$\begin{pmatrix}u\\v\end{pmatrix}=\begin{pmatrix}\xi_1 & \xi_2\\ -\xi_2 & \xi_1\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix} .....(26)$$

The vector $\xi = (\xi_1, \xi_2)$ is still a unit vector. Following the change of variables,

$$\hat{f}(p, \xi) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-u^2 - v^2} \delta(p - u)\, du\, dv$$

$$= e^{-p^2} \int_{-\infty}^{\infty} e^{-v^2}\, dv = \sqrt{\pi}\, e^{-p^2} \quad \ldots\ldots\ldots\ldots(27)$$

Hence, have the important result

$$R\left\{ e^{-x^2 - y^2} \right\} = \sqrt{\pi}\, e^{-p^2} \quad \ldots\ldots\ldots\ldots(28)$$

From the above remark (11) use the discrete Radon Transform will decompose a multidimensional function into scalar functions for each p and $\xi$. These functions $f_{(\ell)} : R \to R$ ($\ell = 1, 2, \ldots, L$ where L is the number of projections) are approximated using the function $N(x)$ given in the preceding section, see equation (4). Now each function $f_{(\ell)} : R \to R$ is approximated by a subnetwork $N^{(\ell)} : R \to R$, given in (4), and these subnetworks $N^{(\ell)}$ are combines into the final approximation network which have the form

$$N(x) = \sum_{\ell=1}^{L} N^{(\ell)}(x, w^{(\ell)}) \quad (29)$$

which is an approximation to the discrete Radon Transform $\hat{g}(p, \xi)$ of the function f(x).

It is necessary to invert the Radon Transform, that is, to solve for f in terms of $\hat{f}$.

**Definition(13)**

The Radon Transform inversion formula has the form

$$f(x) = \begin{cases} \dfrac{1}{2(2\pi i)^{n-1}} \left(\dfrac{\partial}{\partial p}\right)^{n-1} \displaystyle\int_{|\xi|=1} \hat{f}(p, \xi)\, d\xi & \text{if } n \text{ is odd} \\ \dfrac{1}{2(2\pi i)^{n}} \displaystyle\int_{|\xi|=1} d\xi \int_{-\infty}^{\infty} \left(\dfrac{\partial}{\partial p}\right)^{n-1} \dfrac{\hat{f}(p, \xi)}{p - \xi x}\, dp & \text{if } n \text{ is even} \end{cases}$$

$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(30)$

where $i = \sqrt{-1}$, [3].

Thus often using the inverse Radon Transform to the network N(x) in equation (29) to get back to the dimension of the space that begin with and this will lead us to the approximation of the function f(x), where $x \in R^S$.

**The Algorithm:**

Step1: Input

a- The vector $x \in R^S$.

b- The network target $y \in R$.

c- The number of neurons n (where $n \in N$) in the hidden layer, the error goal, the maximum number of epochs(iteration).

d- The angles $\lambda$ of the projections.

Step2: Compute the discrete Radon Transform to the input vector x and the angles $\lambda$ to get the function $f_{(\ell)}$.

Step3: Applied a feedforward neural network $N^{(\ell)}$ for each of the Radon Transform $f_{(\ell)}$ using (4).

Step4: Combine these subnetworks $N^{(\ell)}$ which is found in Step3 into final network. N(x) by using (29).

Step5: Compute the discrete inverse Radon Transform to the N(x) which has been computed in Step4 and thus get the approximate to the given function.

**Example [2]**

Let us consider the following two dimensional function $f : R^2 \to R$.

$$z = f(x_1, x_2) = 3(1 - x_1)^2 \exp\left[-x_1^2 - (x_2 + 1)^2\right] - 10\left(\frac{1}{5}x_1 - x_1^3 - x_2^5\right)$$

$$\exp\left[-x_1^2 - x_2^2\right] - \frac{1}{3}\exp\left[-(x_1 + 1)^2 - x_1^2\right] \quad (31)$$

A three dimensional plot of the function f for $-4 \le x_1 \le 4$ and $-4 \le x_2 \le 4$ is shown in figure (4). A two dimensional problem is chosen so that to explain how the steps of the algorithm can be illustrated.

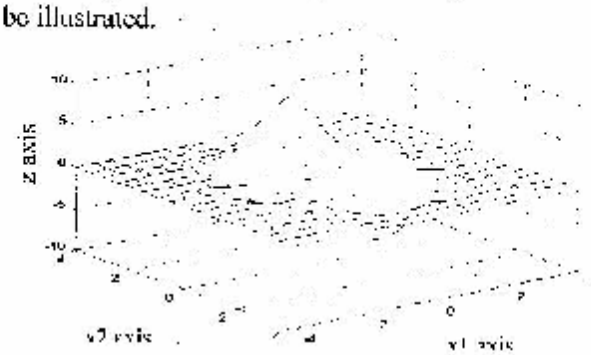

Figure (4) Function $f(x_1, x_2)$

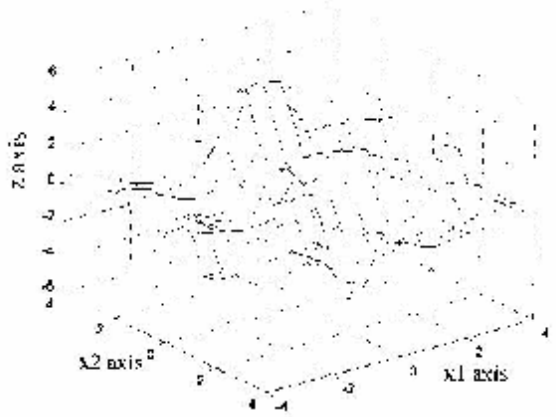


**Figure (5) approximation with L=6 and n=9**

The above algorithm was, numerically, implemented by using MATLAB version (7.0), the feedforward neural networks use the gradient descent algorithm, hyperbolic tangent function in the hidden layer and pureline function in the output layer.

Step1: initially put the input angles $\lambda \in \{0^\circ, 30^\circ, 60^\circ, 90^\circ, 120^\circ, 150^\circ\}$ i.e. L=6, n = 9, with error goal $10^{-5}$, epochs=5000. Figure(5) show the approximation of the function $f(x_1, x_2)$ with the use of the above λ's and n.

Step2: Compute The discrete Radon Transform to the function $f(x_1, x_2)$ using λ's and input vector x and thus get the one dimensional functions $f_{(\ell)}$.

Step3: Each of the one dimensional functions $f_{(\ell)}$ are approximated by a single hidden layer feedforward neural networks. See the illustration in figures. (6) to (8), where star points represent functions $f_{(\ell)}$ and circle points represent their approximation with neural networks output $N^{(\ell)}$.

Step4: The networks approximating functions $f_{(\ell)}$ are combined together, using equation (29), to form the final neural network approximation N(x).

Step5: Compute the discrete inverse Radon Transform to N(x) and thus get the approximation to the function $f(x_1, x_2)$.

Examples for the approximations to the function $f(x_1, x_2)$ with (L=10, n=9), (L=15, n=9), (L=10, n=15) and (L=15, n=15) are shown in figures (9) to (12) respectively.

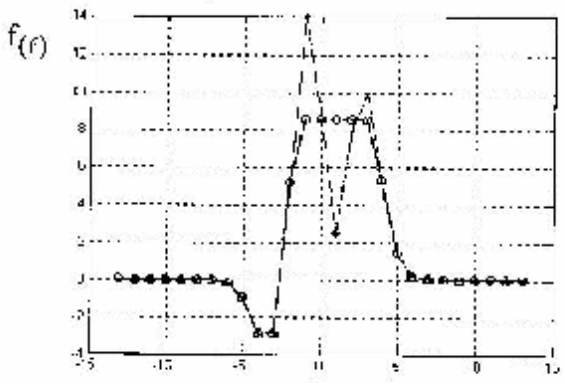


**Figure (6) Function $f_{(\ell)}$ at λ = 0°**

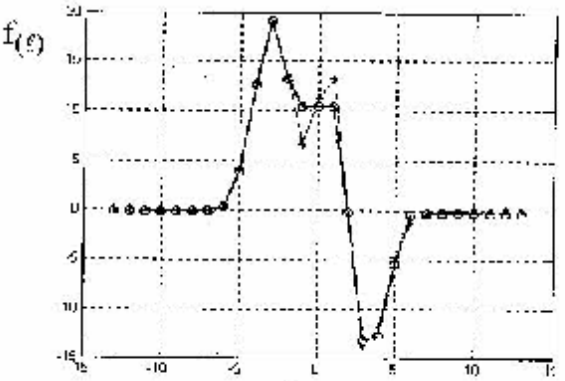


**Figure (7) Function $f_{(\ell)}$ at λ = 60°**

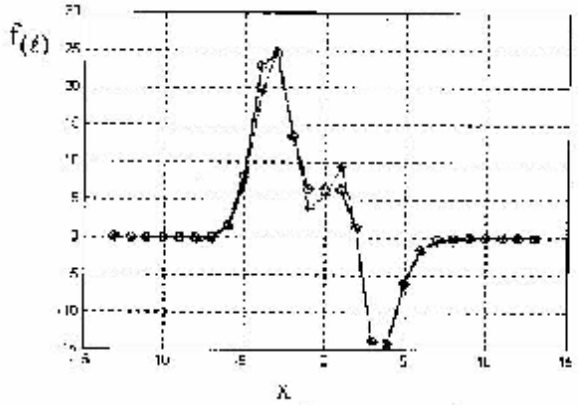


**Figure (8) Function $f_{(\ell)}$ at λ = 90°**

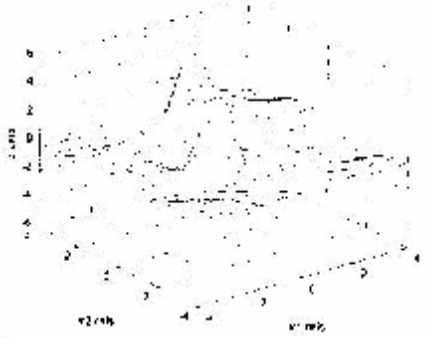

**Figure (9) approximation with L=10 and n=9**

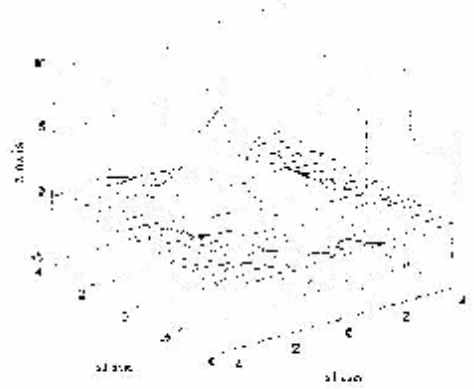

Figure (10) approximation with L=15 and n=9

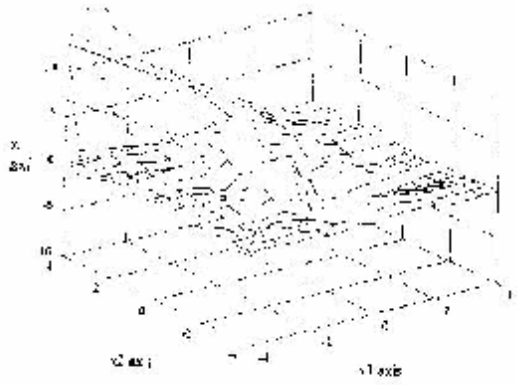

Figure (11) approximation with L=10 and n=15

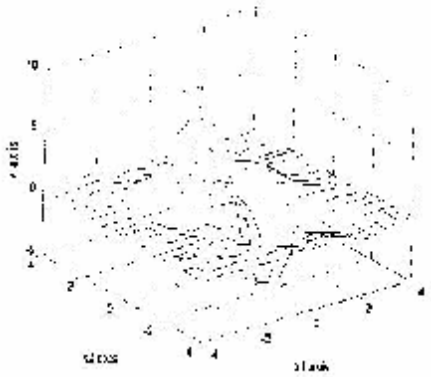

Figure (12) approximation with L=15 and n=15

## Conclusion

This paper develops a method, with the aid of neural network technique, to approximate function of several variables. Our numerical results are more accurate than those given in [2] and this can be concluded from figures (5) to (12) which represents the approximation functions to the function given in the example by the new method while figures (15) to (17) (see Appendix) represents the approximation functions to the same example by the method in [2]. Also, figures (13) and (14) show the compare error between the exact data of the example and the approximation data for both the new method and the method in [2] respectively. Computationally our method is more easy to be use with less flops than the method in [2].

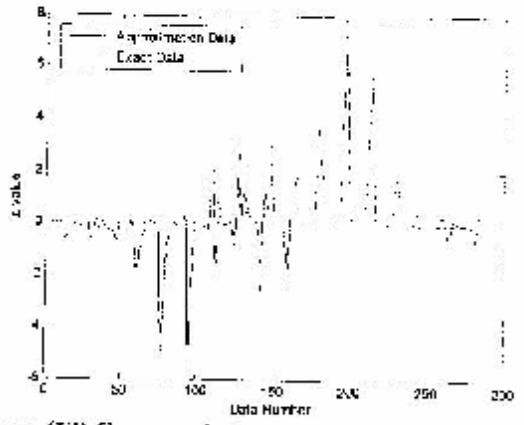


Figure (13) Compare between exact and approximate data of the new method

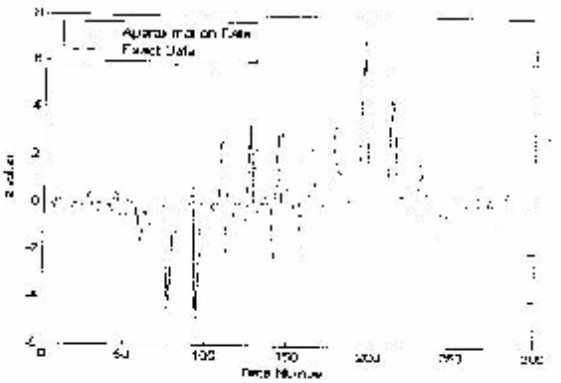


Figure (14) Compare between exact and approximate data of the method [2]

## Appendix

For comparison we present the results in [2]:

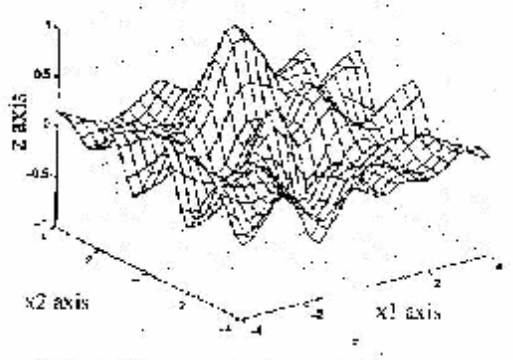


Figure (15) approximation with L=6 and n=9

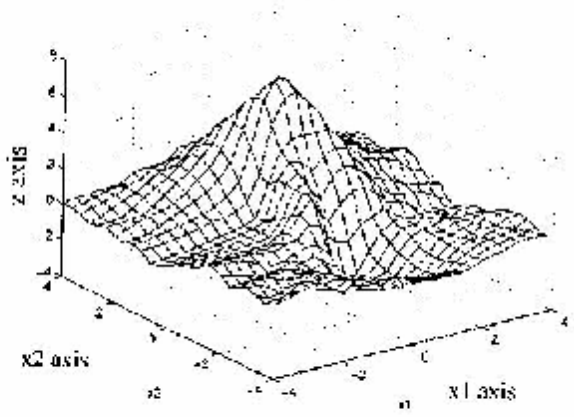


**Figure (16) approximation with L=12 and n=9**

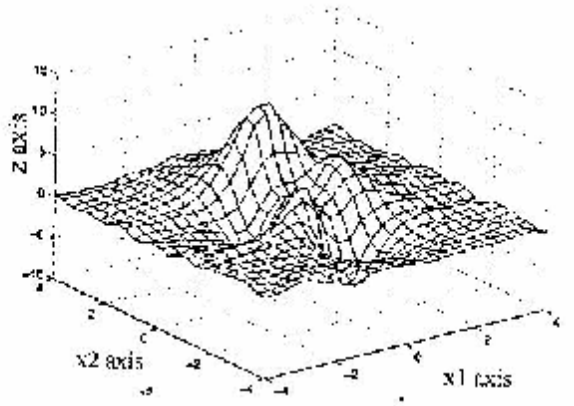


**Figure (17) approximation with L=12 and n=18**

## References


1- Cheney, W. and Light, W. "A Course in approximation theory", The Brooks/Cole series in advanced mathematics. 2000.

2- Ciesielski, K. and Sacha, J. P. "Synthesis of feedforward networks in supremum error bound", IEEE Transactions on neural networks, Vol. 11, No. 6, 2000.

3- Deans, S.R. "The Radon Transform and some of its applications", Publisher by John Wiley and Sons,1983.

4- Ellacot, S. W. "Aspects of the numerical analysis of neural networks", Acta Numerica, pp. 145-202, 1994.

5- Pinkus, A. "Approximation by ridge functions", surface fitting and multiresolution methods,A. Le Méhauté, C. Rabut, and L.L. Schumaker(eds.),Vanderbilt University Press. Nashville, TN, pp. 1-14, 1997.

6 Radon, J. "On the determination of functions from their integrals along certain manifolds", Berichte Sächsische Akademie der Wissenschaften, Leipzig, Math-Phys. Kl., Vol. 69, pp. 262-267, 1917.

7- Siddiqi, A. H. "Functional analysis with applications", McGraw-Hill Publishing company limited, 1986.



## الخلاصة

الهدف الرئيسي من هذا البحث هو تقديم طريقة جديدة لتقريب الدوال متعددة الابعاد باستخدام الشبكات العصبية التقدمية الخطية Ridge feedforward neural nerworks مع تطبيقات تحويل الرادون Radon transform ومعكوسه لتقليص حجم الفضاء. تتألف الطريقة من اربعة مراحل، اولاً: باستخدام تحويل الرادون يمكن تقليص حجم الدالة متعددة الابعاد الى عدة دوال ذات بعد واحد. ثانياً: كل من هذه الدوال ذات البعد الواحد تقرب باستخدام تقنية الشبكات العصبية الى شبكات جزئية subnetworks. ثالثاً: يتم جمع الشبكات العصبية الجزئية معاً لتشكيل الشبكة الرئيسية والتي تمثل شبكة التقريب النهائية. رابعاً: باستخدام معكوس تحويل الرادون للشبكة النهائية يعطينا التقريب للدالة المعطاة. ايضاً قدمنا حد اعلى لتقريب $L_{2}$ للخطأ.

وقد طبقنا الطريقة اعلاه بمثال وقارنا النتائج التي حصلنا عليها مع نتائج البحث [2] وايضا كانت حساباتنا افضل من الموجودة في [2].