APPROXIMATION OF MULTIDIMENSIONAL FUNCTIONS BY RADON RADIAL BASIS NEURAL NETWORKS

[*]Prof. Dr. Reyadh S. Naoum and [**]Najla'a M. Hussein
[*]Department of Mathematics, College of Science, University of Baghdad, Iraq
[**]Department of Computer Science, College of Science, University of Baghdad, Iraq

Abstract

The main result of this paper is to present a new method to approximate multidimensional function by using Radial Basis 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 a suitable adjusting to the parameters of the method to reduce the L_2 approximate error. Also, we apply the above method to an example and a comparison is made 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 Baxter (et al.) [1], Ciesielski and Sacha [2], Ellacott [4], Niyogi and Girosi [6] and Orr [7].

Baxter (et al.), [1], it is known that the interpolation matrix $A = \left(h(x_j - x_k) \right)_{j,k=1}^{n}$ is invertible (where h is a radial function), they computed an upper bounds for the $\left\| A^{-1} \right\|_p$, $p \geq 1$, and they show that when A is symmetric and positive definite then h decays sufficiently quickly. Ciesielski and Sacha, [2], focused on a development of a constructive formula for the upper bound of L_∞ 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. Niyogi and Girosi [6], they derived a bound to generalization error for radial basis functions which apply to any approximation technique.

Orr, [7], gave an introduction to radial basis function (RBF) neural networks with least squares bound.

Johann Radon, [8], showed that if f is continuous 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 new method for approximating a multidimensional function by using radial basis neural network with one hidden layer with linear output. Also, we present an upper bound of the L_2 error approximation. The method consists of four stages: First: By the use of discrete Radon transform, [8], the problem of multidimensional approximation is replaced by, several simpler, one dimensional problems. Second: For 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 of Radon Transform to the final approximation network to get the approximation of the given problem.

Approximation of 1-D Functions:

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

Definition

Let X be a normed linear space. A function $f : X \to R$ is said to be radial if there exists a function $h : R^+ \to R$ such that $f(x) = h(\| x \|)$ for all $x \in X$, R^+ is a set of all positive real numbers.

A radial basis function is any translate of f ; that is, a function of the form

$$\mu(x) = f(x-\xi) = h(\| x-\xi \|), \quad \ldots\ldots\ldots\ldots (1)$$

where ξ is any prescribed point of X, [5]. Such a function depends on the distance $\| x-\xi \|$, $\| . \|$ usually is taken to be Euclidean norm, and this function is symmetric with respect to a center point ξ. Some examples of radial basis functions in one dimensional space are shown in Fig.(1).

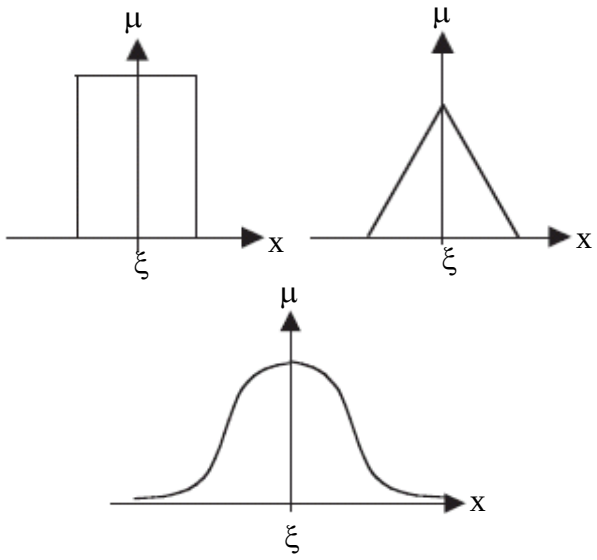

Fig.(1) : Three examples of one dimensional radial basis functions

Radial basis functions are a special class of functions that has the following characteristic feature: Their response decrease (or increase) monotonically with distance from a central point. A common use of such functions is for interpolation (to approximate a given function). In this context, one usually has data prescribed at points $\xi_1, \xi_2, \ldots, \xi_n$ in X and attempts to interpolate these data by function of the form

$$x \mapsto \sum_{j=1}^{n} c_j \, h\left(\| x-\xi_j \| \right) \quad x \in X, \ldots\ldots\ldots (2)$$

Radial basis functions can be employed in the neural computation for approximating continuous functions.

Radial Basis Neural Network

An artificial neural network is a mathematical model of the human brain. Many different types of neural network models are studied, but this paper describes a radial basis neural network with input layer, one hidden layer and output layer. A radial basis neural network is a feedforward neural network with the radial basis function as an activation function. The idea of radial basis function (RBF) network derive from the theory of function approximation, and this network consists of a large number of computing units arranged schematically in three layers as shown in Fig.(2). 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}, and is known as the radial basis function (RBF) center. In a typical operation, each unit on the input layer will contain a real number. Let the j^{th} input 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 $(x_j - w_{ij})^2$. The total input that unit i receives from all the input units is then $c_i = \| x_j - w_{ij} \| = \sum_{j=1}^{s} (x_j - w_{ij})^2$. Unit i on the hidden layer now processes this input with a radial basis function $\theta : R^+ \to R$ which is a given fixed univariate function (called the activation function) and the outputs are the real numbers $\theta_{ij}\left(\| x_j - w_{ij} \| \right)$ where (i=1, 2, …, k and j=1, 2, …, s). This output θ_{ij} is then transmitted, with a weight a_i, to the output unit. The total output is then have the form

$$g(x) = \sum_{i=1}^{k} a_i \, \theta_i\left(\| x - w_i \| \right). \quad \ldots\ldots\ldots (3)$$

It is know from definition (2.1), any continuous function of s variables can be approximated by a function of the form g given in equation (3). For our radial basis function (RBF) neural network the function θ_i and the centers w_i are assumed to have been fixed. Thus by suitable adjusting of the parameters a_i, this can be reproduce approximately, to any accuracy, any desired output with the use of artificial neural network.

From above analysis we can approximate any complicated type functions by simple

functions. For the simplification of the computational complexity take the case (one input, one hidden layer with n processing units and one output).

Thus to approximate a continuous function $\varphi: R \to R$ is by considering the approximation form of the neural network $\aleph: R \to R$ as:

$$\aleph(x) = \sum_{i=1}^{n} a_i \, \theta_i(\| x - w_i \|), \quad \text{.....(4)}$$

$\aleph$ is a linear combination of radial basis functions (RBF) and the corresponding network is known as a radial basis function network (RBF). Let us assume that the activation (transfer) function is the Gaussian function

$$\theta(x) = \exp\left(\frac{-x^2}{2\sigma^2}\right), \quad \text{.....(5)}$$

where $\sigma > 0$ is a parameter whose value controls the smoothness properties of the interpolating function.

The following theorem, from [7], presents an adjusting to the parameters a_i but this paper gives a different proof so as to be suitable for our problem.

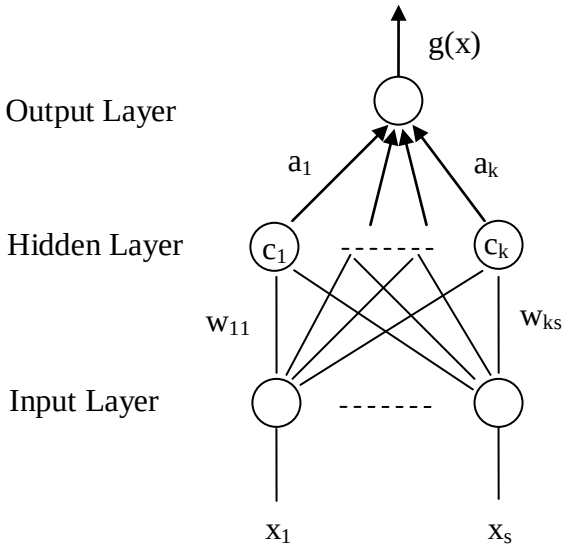

Fig.(2) : Layers in a neural network

Theorem

Let the function $\varphi: R \to R$ be continuous on the compact set K in R. The approximation error of the function φ by the network $\aleph$ given in (3) is defined as

$$E(x_j) = \sum_{j=1}^{s} \left(\varphi(x_j) - \aleph(x_j)\right)^2, \quad \text{.....(6)}$$

The error E is minimum if and only if

$$\underline{a} = A^{-1} H^T \varphi, \quad \text{.....(7)}$$

where A is the variance matrix and H is the neural net matrix.

Proof:

It is well known from calculus that to find an extreme point of a function we need

1- differentiate the function with respect to the free variable(s).
2- equate the result(s) with zero,
3- solve the resulting equation(s).

want to minimize the error $E(x_j) = \sum_{j=1}^{s} \left(\varphi(x_j) - \aleph(x_j)\right)^2$,

Where $\aleph(x) = \sum_{i=1}^{n} a_i \, \theta_i(\| x - w_i \|)$, ...(8)

and the free variables are the weights $\{a_i\}_{i=1}^{n}$.

Now, for the i^{th} weight, differentiating the error function in (6) with respect to a_i

$$\frac{\partial E}{\partial a_i} = 2\sum_{j=1}^{s} (\aleph(x_j) - \varphi(x_j)) \frac{\partial \aleph}{\partial a_i}(x_j), \quad \text{.....(9)}$$

and from (8) get that

$$\frac{\partial \aleph}{\partial a_i}(x_j) = \theta_i(x_j) \quad \text{..... (10)}$$

(for simplicity, write $\theta_{ij}\left(\| x_j - w_{ij} \|\right)$ as $\theta_i(x_j)$), substitute (10) in (9) and equating the result to zero leads to the equation

$$\sum_{j=1}^{s} \aleph(x_j)\, \theta_i(x_j) = \sum_{j=1}^{s} \varphi(x_j)\, \theta_i(x_j) \qquad i = 1, 2, \ldots, n \quad \text{.....(11)}$$

There are n such equations, each representing one constraint on the solution. Since there are exactly as many constraints as there are unknowns, the system of equations has a unique solution.

In matrix form equation (11) becomes:

$$\theta_i^T \aleph = \theta_i^T \varphi \quad , \quad i = 1, 2, \ldots, n \quad \text{.....(12)}$$

Where

$$\boldsymbol{\theta}_i = \begin{bmatrix} \theta_i(x_1) \\ \theta_i(x_2) \\ \vdots \\ \theta_i(x_s) \end{bmatrix}, \aleph = \begin{bmatrix} \aleph(x_1) \\ \aleph(x_2) \\ \vdots \\ \aleph(x_s) \end{bmatrix}, \varphi = \begin{bmatrix} \varphi(x_1) \\ \varphi(x_2) \\ \vdots \\ \varphi(x_s) \end{bmatrix}$$

.....(13)

The above system, of n equations, can be written as

$$H^T \aleph = H^T \varphi \ , \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (14)$$

Where H is called the design matrix of the neural net and has the form

$$H = \begin{bmatrix} \theta_1(x_1) & \theta_2(x_1) & \ldots & \theta_n(x_1) \\ \theta_1(x_2) & \theta_2(x_2) & \ldots & \theta_n(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \theta_1(x_s) & \theta_2(x_s) & \ldots & \theta_n(x_s) \end{bmatrix}, \ldots\ldots\ldots (15)$$

Now, back to equation (8), the j^{th} component of $\aleph$, when the weights are at their optimal values, has the form

$$\aleph_j = \aleph(x_j)$$

$$= \sum_{i=1}^{n} a_i \, \theta_i(x_j) \ , \quad j = 1, 2, \ldots, s \ \ldots\ldots\ldots (16)$$

$$= \bar{\boldsymbol{\theta}}_j^{\,T} \, \underline{a}, \quad j = 1, 2, \ldots, s$$

Where $\bar{\boldsymbol{\theta}}_j = \begin{bmatrix} \theta_1(x_j) \\ \theta_2(x_j) \\ \vdots \\ \theta_n(x_j) \end{bmatrix}$, ... (17)

Thus $\boldsymbol{\theta}_i$ is one of the columns of H and $\bar{\boldsymbol{\theta}}_j$ is one of its rows. Thus equation (16) becomes

$$\aleph = \begin{bmatrix} \aleph_1 \\ \aleph_2 \\ \vdots \\ \aleph_s \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{\theta}}_1^{\,T} \, \underline{a} \\ \bar{\boldsymbol{\theta}}_2^{\,T} \, \underline{a} \\ \vdots \\ \bar{\boldsymbol{\theta}}_s^{\,T} \, \underline{a} \end{bmatrix} = H \, \underline{a} \ \ldots\ldots\ldots\ldots\ldots\ldots (18)$$

substitute (18) in equation (14), get

$$H^T \varphi = H^T \aleph = H^T H \, \underline{a} \ \ldots\ldots\ldots\ldots\ldots\ldots (19)$$

Since $H^T H$ is positive definite and thus its non-singular then $\underline{a}$ is solution of normal equation (20)

$$\underline{a} = (H^T H)^{-1} H^T \varphi = A^{-1} H^T \varphi \ \ldots\ldots\ldots (20).$$

Numerically to compute $\underline{a}$ just need to find the inverse of the matrix $H^T H$. Now since $K(H^T H) = K^2(H)$, K is the condition number , which is defined to be $K(H) = \| H \| \| H^{-1} \|$, and thus the computation of $(H^T H)^{-1}$ could be sensitive to the rounding error. Therefore, use the least square method, assume $H^T H$ is of full rank, to find $\underline{a}$.

Our numerical results shows that as increasing the dimension of $H^T H$ the condition number become bigger and thus will have an ill-condition system. However, this is not the case when use least square method or conjugate gradient method.

If the matrix arise from using the above method, neural radial basis function, is not of full rank that is rank($H^T H$) < n then we use the singular value decomposition technique.

The Singular Value Decomposition

The ideas that lead to the spectral decomposition can be extended to provide a decomposition for a rectangular, rather than a square, matrix. It can be decompose a matrix that is not square nor symmetric by first considering a matrix A that is of dimension m×n where m ≥ n. This assumption is made for convenience only; all the results will also hold if m < n. As it turns out, the vectors in the expansion of A are the eigenvectors of the square matrices AA^T and A^TA. The former is a outer product and results in a matrix that is spanned by the row space of A. The latter is a inner product and results in a matrix that is spanned by the column space (i.e., the range) of A.

The *singular values* are the nonzero square roots of the eigenvalues from AA^T and A^TA. The eigenvectors of AA^T are called the "left" singular vectors (U) while the eigenvectors of A^TA are the "right" singular vectors (V). By retaining the nonzero eigenvalues k = min(m, n), a *singular value decomposition* (SVD) can be constructed. That is

$$A = U \Lambda V^T, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (21)$$

where U is an m×m orthogonal matrix (U^TU =I), V is an n×n orthogonal matrix (V^TV = I), and Λ is an m×n matrix whose off-diagonal entries are all 0's and whose diagonal elements satisfy

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0 \quad \text{.....(22)}$$

It can be shown that the rank of A equals the number of nonzero singular values and that the magnitudes of the nonzero singular values provide a measure of how close A is to a matrix of lower rank. That is, if A is nearly rank deficient (singular), then the singular values will be small. In general, the SVD represents an expansion of the original data A in a coordinate system where the covariance matrix Σ_A is diagonal.

Remember, this is called the singular value decomposition because the factorization finds values or eigenvalues or characteristic roots (all the same) that make the the following characteristic equation true or singular. That is

$$|A - \lambda I| = 0 \text{ , ...(23)}$$

Using the determinant this way helps solve the linear system of equations thus generating an n^{th} degree polynomial in the variable λ. This polynomial, that yields n-roots, is called the characteristic polynomial.

Equation (23) actually comes from the more generalized eigenvalue equation which has the form

$$Ax = \lambda x \text{ , ...(24)}$$

which, when written in matrix form, is expressed as

$$AX = X\Lambda \text{ , ...(25)}$$

This implies

$$Ax - \lambda x = 0, \text{.....(26)}$$

$$\text{Or } (A - \lambda I)x = 0, \text{.....(27)}$$

The theory of simultaneous equations tells us that for this equation to be true it is necessary to have either x = 0 or $|A - \lambda I| = 0$. Thus the motivation to solve equation (23).

Solving A System of Linear Equations

A set of linear algebraic equations can be written as

$$Ax = b, \text{.....(28)}$$

where A is a matrix of coefficients (m×n), and b (m×1) is some form of a system output vector. The vector x is what usually solve for. If m = n then there are as many equations as unknowns, and there is a good chance of solving for x. That is

$$A^{-1}Ax = A^{-1}b, \text{.....(29)}$$

$$x = A^{-1}b, \text{.....(30)}$$

Here, simply compute the inverse of A. This can prove to be a challenging task, however, for there are many situations where the inverse of A does not exist. In these cases the approximating of the inverse via the SVD which can turn a singular problem into a non-singular one.

Vector x in equation (28) can also be solved for by using the transpose of A. That is

$$A^TAx = A^Tb \text{ ,.....(31)}$$

$$x = (A^TA)^{-1}A^Tb \text{ , ...(32)}$$

This is the form of the solution in a least-squares sense from standard multivariate regression theory where the inverse of A is express as

$$A^* = (A^TA)^{-1}A^T, \text{ .....(33)}$$

where A* is called the More-Penrose pseudoinverse. The use of the SVD can aid in the computation of the generalized pseudoinverse.

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^∞ 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 Ω be an open subset of R^s with a smooth bounded boundary Γ. Then the compact support of a function f is defined as follows:

Definition

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 Ω, i.e. $\overline{K}_f$ is compact.

Remark

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

Definition

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

$$(\delta_\alpha, f) = \int_\Omega \delta_\alpha(x)\, f(x)\, dx = f(\alpha),\ f \in D(\Omega) \quad \text{.....(34)}$$

where (. , .) is an inner product defined on $D(\Omega)$.

In particular, for α=0, the Dirac mass concentrated at the origin denoted by δ is defined as

$$(\delta, f) = \int_\Omega \delta(x)\, f(x)\, dx = f(0),\ f \in D(\Omega) \quad \text{.....(35)}$$

Actually delta mass concentrated at α is not a function but a distribution.

Definition

Let $f : R^s \to R$ be C^∞ 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

$$\hat{f}(p, \xi) = Rf = \int_{R^s} f(\mathbf{x})\, \delta(p - \xi.\mathbf{x})\, d\mathbf{x} \quad \text{.....(36)}$$

where $d\mathbf{x}=dx_1 dx_2 \ldots dx_s$.

δ is the Dirac delta function.

ξ is a unit vector in R^s that defines the orientation of a hyperplane with equation

$p = \xi.\mathbf{x} = \xi_1 x_1 + \xi_2 x_2 + \cdots + \xi_s x_s$ (p is the orientation of the hyperplane).

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

Remark

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 ξ_j (j = 1, 2, …, L where L∈N, L is the number of projection), then take the summation on the discrete data $\mathbf{x_i}$ (i =1,2, …, m where m∈N and $\mathbf{x_i} \in R^s$) of the function $f \in C(R^s)$.

Then the discrete Radon Transform is defined by, [9]

$$\hat{g}_j(p, \xi_j) = \sum_{i=1}^{m} f(\mathbf{x_i})\, \delta(p - \xi_j \cdot \mathbf{x_i}) \quad \text{.....(37)}$$

Example 3:

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

$$\hat{f} = Rf = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-x^2 - y^2}\, \delta(p - \xi_1 x - \xi_2 y)\, dx\, dy \quad \text{.....(38)}$$

Now make the orthogonal linear transformation (in matrix notation)

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \xi_1 & \xi_2 \\ -\xi_2 & \xi_1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \quad \text{.....(39)}$$

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

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

$$= e^{-p^2} \int_{-\infty}^{\infty} e^{-v^2}\, dv \quad \text{.....(40)}$$

Hence, have the important result

$$R\left\{ e^{-x^2 - y^2} \right\} = \sqrt{\pi}\ e^{-p^2} \quad \text{.....(41)}$$

From the above remark (3.5) use the discrete Radon Transform will decompose a multidimensional function into scalar functions for each p and ξ. These functions $f_{(\ell)} : R \to R$ ($\ell = 1, 2, \ldots, L$ where L is the number of projections) are approximated using the function $\aleph(x)$ given in the preceding section, see equation (4). Now each function $f_{(\ell)} : R \to R$ is approximated by a subnetwork $\aleph^{(\ell)} : R \to R$, given in (4), and these subnetworks $\aleph^{(\ell)}$ are combines into the final approximation network which have the form

$$N(x) = \sum_{\ell=1}^{L} \aleph^{(\ell)}\left(\left\| x - w^{(\ell)} \right\| \right) \quad \text{.....(42)}$$

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

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

Definition

The Radon Transform inversion formula has the form

$$f(\mathbf{x}) = \begin{cases} \dfrac{1}{2(2\pi i)^{s-1}} \left(\dfrac{\partial}{\partial p}\right)^{\frac{s-1}{2}} \displaystyle\int_{|\xi|=1} \hat{f}(p,\xi)\, d\xi & \text{if } s \text{ is odd} \\ \dfrac{1}{2(2\pi i)^{s}} \displaystyle\int_{|\xi|=1} d\xi \int_{-\infty}^{\infty} \dfrac{\left(\dfrac{\partial}{\partial p}\right)^{s-1} \hat{f}(p,\xi)}{p-\xi.\mathbf{x}}\, dp & \text{if } s \text{ is even} \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (43)$$

where $i = \sqrt{-1}$.

Thus often using the inverse Radon Transform to the network N(x) in equation (42) 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 $\mathbf{x} \in R^s$.

The Algorithm:

Step1: Input a- The vector $x \in R^s$.

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

c- The error goal.

d- The angles λ of the projections.

Step 2: Compute the discrete Radon Transform to the input vector **x** and the angles λ to get the function $f_{(\ell)}$.

Step 3: Applied a radial basis neural network $\aleph^{(\ell)}$ for each of the Radon Transform $f_{(\ell)}$ using equation (4).

Step 4: Combine these subnetworks $\aleph^{(\ell)}$, which is found in Step3, into final neural network N(x) by using (42).

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

5- Example, [2]:

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

$$z = f(x_1, x_2) = 3(1-x_1)^2 \exp[-x_1^2 - (x_2+1)^2] - 10\left(\frac{1}{5}x_1 - x_1^3 - x_2^5\right)\exp[-x_1^2 - x_2^2] - \frac{1}{3}\exp[-(x_1+1)^2 - x_1^2] \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (44)$$

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 Fig.(3). A two dimensional problem is chosen so that to explain how the steps of the algorithm can be illustrated.

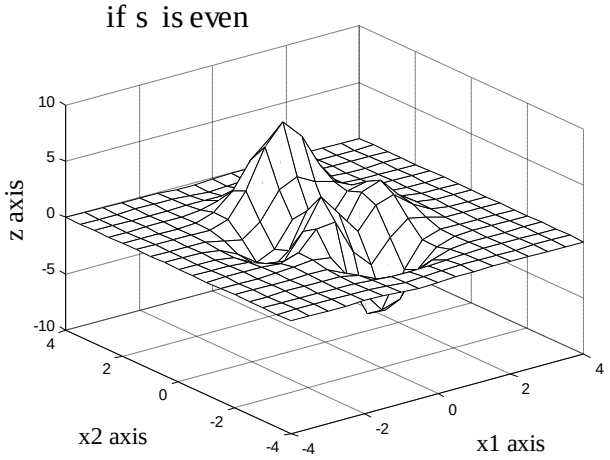

Fig. (3) : Function $f(x_1, x_2)$

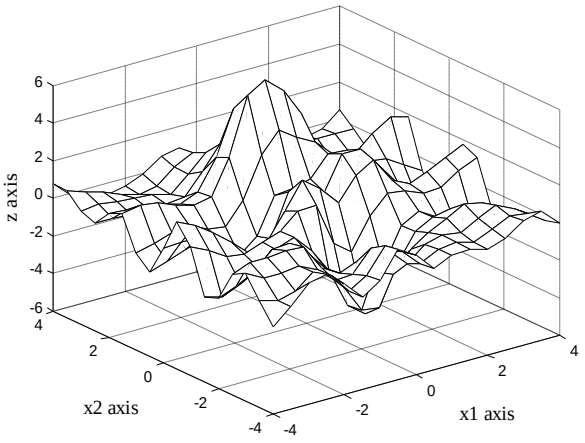

Fig. (4) : Approximation with L = 6

The above algorithm has been, numerically, implemented with the use of MATLAB version (7.0), the radial basis function neural network use the Gaussian function in the hidden layer and pureline function in the output layer. The numerical procedure as follows:

Step1: initially put the input angles $\lambda \in \{0^{\circ},30^{\circ},60^{\circ},90^{\circ},120^{\circ},150^{\circ}\}$ i.e L=6 , error goal 10^{-5}. Fig.(4) show approximation of function $f(x_1,x_2)$ using above λ's.

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 using a single hidden layer radial basis neural networks. See the illustrated in figures. (5) to (7), where star points represent functions $f_{(\ell)}$ and circle points represent their approximation with neural network output $\aleph^{(\ell)}$.

Step4: The networks approximating functions $f_{(\ell)}$ are combined together, using equation (42), 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 approximating of the function $f(x_1,x_2)$ with L=10 and L=15 are shown in figure(8) and figure(9) respectively.

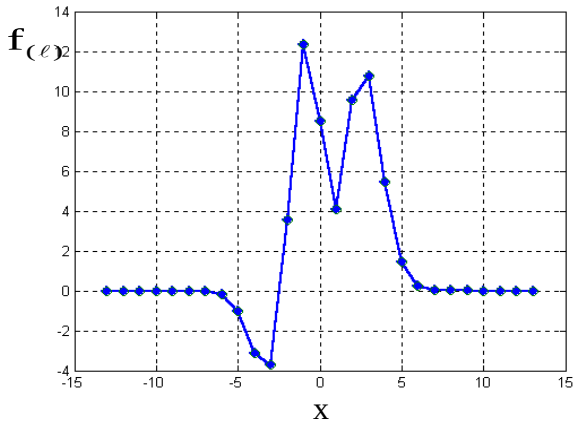

Fig. (5) : Function $f_{(\ell)}$ at λ = 0°

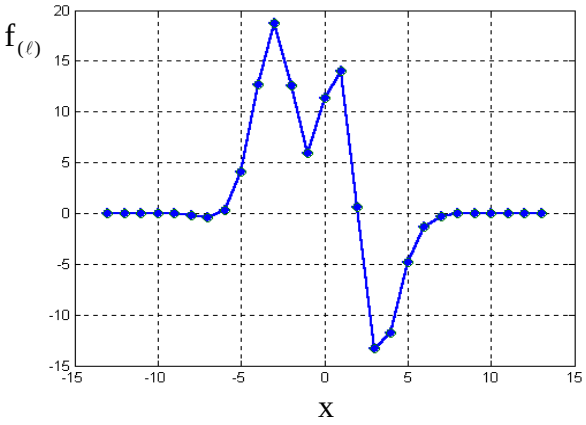

Fig.(6) : Function $f_{(\ell)}$ at λ = 60°

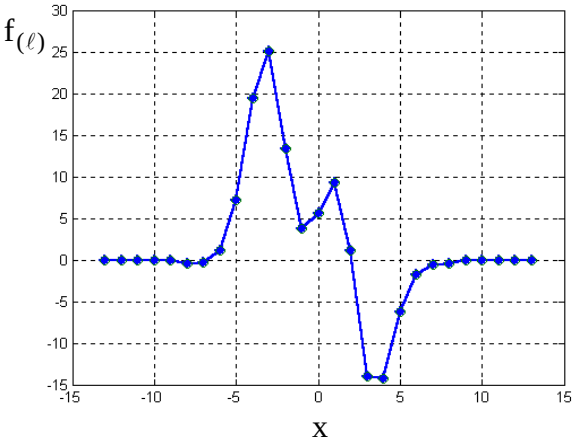

Fig.(7) : Function $f_{(\ell)}$ at λ = 90°

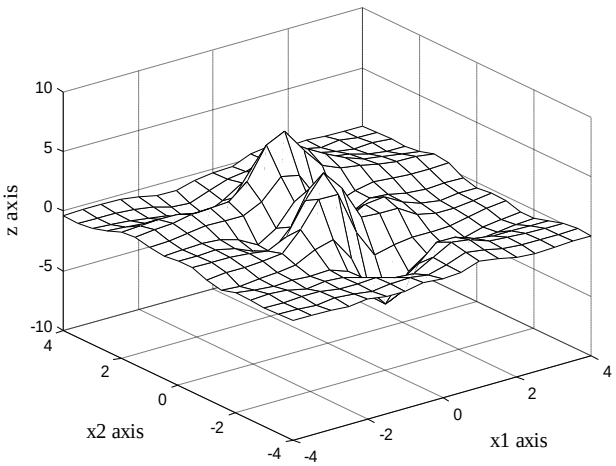

Fig. (8) : Approximation with L=10

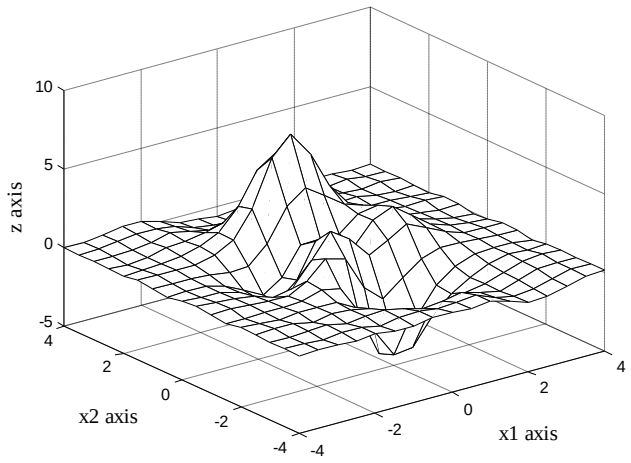

Fig. (9) :Approximation with L=15

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 Figs. (4) to (9) which represents the approximation functions to the function given in the example by the new method while Figs. (12) to (14) (see Appendix) represents the approximation functions to the same example by the method in [2]. Also, Figs. (10) and (11) 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].

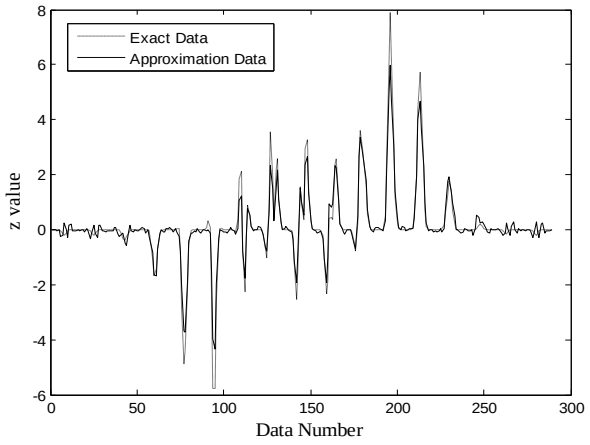

Fig. (10) : Compare between exact and approximation data of the new method

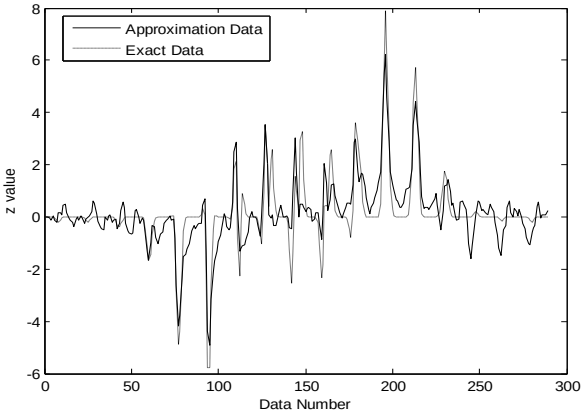

Fig. (11) : Compare between exact and approximation data of the method in [2]

Appendix

For comparison we present the results in [2]:

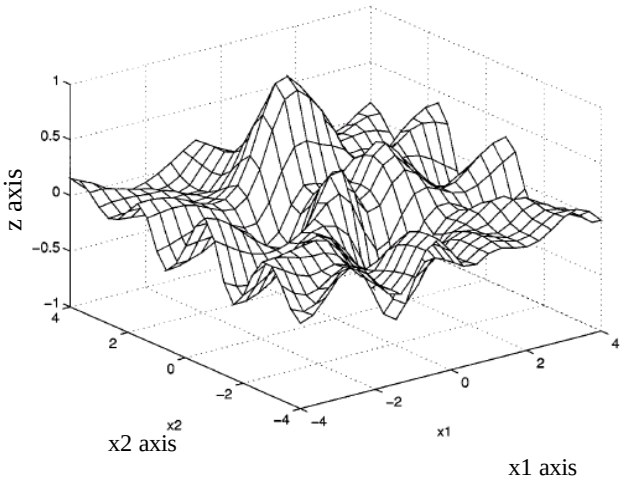

Fig. (12) : Approximation with L=6 and n=9

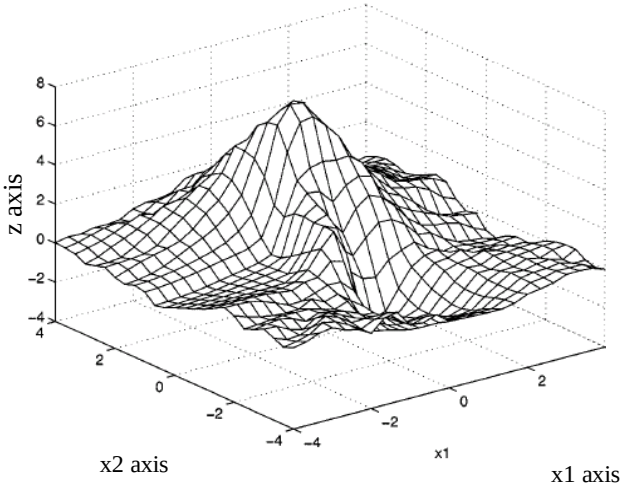

Fig. (13) : Approximation with L=12 and n=9

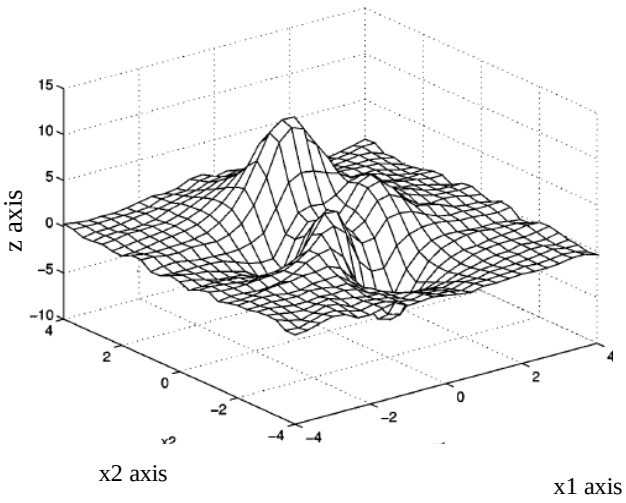

Fig.(14) : Approximation with L=12 and n=18

References

[1] Baxter, B.J.C., Sivakumar, N. and Ward, J.D. "Regarding the p-norms of Radial basis interpolation matrices", Constructive Approximation, Vol. 10, No. 4, pp. 451-468, 1994.

[2] Ciesielski, K. and Sacha, J. P. "Synthesis of feedforward networks in supremum error bound", IEEE Transactions on neural networks, Vol. 11, No. 6, pp. 1213-1227, 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, Cambridge University Press, Vol. 3, pp. 145-202, 1994.

[5] Light, W.A. and Cheney, E. W. "Interpolation by periodic Radial basis functions", Mathematical Analysis and Applications, Vol. 168, pp. 111-130, 1992.

[6] Niyogi, P. and Girosi , F. "On the relationship between generalization error, hypothesis complexity, and sample complexity for Radial basis functions", J. of Neural Computation, Vol. 8, pp. 819-842, 1996.

[7] Orr, M.J.L. "Introduction to Radial basis function networks", Center for Cognitive Science, University of Edinburgh, Scotland, 1996.

[8] 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.

[9] Sampat, M. P., Whitman, G. J., Markey, M. K. and Bovik, A. C. "Evidence Based Detection of Spiculated Masses and Architectural Distortions", Medical Imaging 2005: Image Processing 5747: pp. 26-37, 2005.

[10] 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 الشعاعية Radial basis neural networks بتطبيق تحويل الرادون Radon transform ومعكوسه لتقليص بعد 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. ايضا، قدمنا اسلوب لحساب معلمات الطريقة لتقليل الخطأ L_2.

وقد طبقنا الطريقة اعلاه بمثال وقارنا النتائج التي حصلنا عليها مع نتائج البحث [2] كما ان حساباتنا اجود من الموجودة في [2].