PERGAMON

AVAILABLE AT
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Neural Networks 16 (2003) 729–734

Neural Networks

www.elsevier.com/locate/neunet

2003 Special issue

Numerical solution of elliptic partial differential equation using radial basis function neural networks

Li Jianyu[a,b,*], Luo Siwei[a], Qi Yingjian[a], Huang Yaping[a]

[a]*Computer Science Department, Northern Jiaotong University, Beijing 100044, People's Republic of China*
[b]*Information and Engineering College, Beijing Broadcasting Institute, Beijing, People's Republic of China*

Abstract

In this paper a neural network for solving partial differential equations is described. The activation functions of the hidden nodes are the radial basis functions (RBF) whose parameters are learnt by a two-stage gradient descent strategy. A new growing RBF-node insertion strategy with different RBF is used in order to improve the net performances. The learning strategy is able to save computational time and memory space because of the selective growing of nodes whose activation functions consist of different RBFs. An analysis of the learning capabilities and a comparison of the net performances with other approaches have been performed. It is shown that the resulting network improves the approximation results.

Keywords: Radial basis function neural networks; Partial differential equation; Multi-quadrics

1. Introduction

Many problems in science and engineering are reduced to a set of differential equations (DEs) through a process of mathematical modeling. It is not easy to obtain their exact solutions, so numerical methods must be resorted to. There are a lot of techniques available such as the finite difference method (FDM) (Smith, 1978) and the finite element method (FEM) (Hughes, 1987). These methods require the definition of a mesh (domain discretisation) where the functions are approximated locally. The construction of a mesh in two or more dimensions is a non-trivial problem. Usually, in practice, only low-order approximations are employed resulting in a continuous approximation of the function across the mesh but not its partial derivatives. The discontinuity of the approximation of the derivative can adversely affect the stability of the solution. While higher-order schemes are necessary for more accurate approximations of the spatial derivatives, they usually involve additional computational cost. To increase the accuracy of the low-order scheme, it is required that the computational mesh be refined with a higher density of elements in the regions near the contours. This, however, is also achieved at the expense of increased computational cost.

Another method to find an approximate particular solution is achieved by using radial basis function (RBF). Since Kansa (1990a, b) derived a modified RBFs method suitable for solving parabolic, hyperbolic, and elliptic partial differential equations (PDE), the method has been successfully applied for solving PDEs of various types. Later, Franke and Schaback (1997, 1998), Wendland (1999), Wu (1998) and Wu and Schaback (1994) contributed some theories on the solvability and error bounds in applying the RBFs to solve the PDEs. The advantage of RSFs is that they involve a single independent variable regardless of the dimension of the problem. They prove particularly attractive when the domain cannot be expressed as product domains of lower dimensions. In all the interpolation methods for scattered data sets, RBFs outperforms all the other methods regarding accuracy, stability, efficiency, memory requirement, and simplicity of the implementation. In a similar study, Stead (1984) examined the accuracy of partial derivative approximations over scattered data sets, also concluding that RBFs performed more accurately compared to other considered methods. Of the RBFs tested by Franke, Hardy's multi-quadrics (MQ) were ranked the best in accuracy, followed by Duchon's thin plate spline (TPS).

Even though the TPS has been considered as a good interpolating multi-variate function, they do, however, only converge linearly (Powell, 1994). On the other hand, the MQ

* Corresponding author. Present address: Computer Science Department, Northern Jiaotong University, Beijing 100044, People's Republic of China.

0893-6080/03/$ - see front matter © 2003 Published by Elsevier Science Ltd.
doi:10.1016/S0893-6080(03)00083-2

functions converge exponentially and always produce a minimal semi-norm error as proved by Madych and Nelson (1990). However, despite MQs excellent performance, it contains a free parameter, c^2, often referred to as the shape parameter. When c is small, the resulting interpolating surface is pulled tightly to the data points, forming a cone-like basis functions. As c increases, the peak of the cone gradually flattens. Moreover, when the problems to be solved are different, the choice of the optimal RBFs will change. So the network performance is decided not only by the number of the nodes, but also by the RBFs' type. Although, much work has been done in the past, there is one feature that the activation functions of the net are of the same RBF type. In this paper, we focus on the research of how to choose the activation function and how to increase the number of nodes as small as possible.

The approach reported in the present paper uses a growing-network paradigm. Our main goal is to use only as few knots as necessary (selected according to some quality criterion) to achieve a desired accuracy for the fit. We will describe the algorithm, which is based on adaptively inserting knots with different RBFs. The main algorithm includes two more algorithms: one for adapting the networks parameters and the other for inserting new neurons. The approach different from others' (Esposito, Marinaro, Oricchio, & Scarpetta, 2000) is that all parameters are trained and the activation functions of the added nodes can be different RBFs.

For the parameter adaptation we follow the idea of two-stage iterative gradient method (Jianyu, Siwei, & Yingjian, 2002). In this sense, the parameters of centers and width and the parameters of the output weights are trained, respectively, and interactively. New neurons are inserted when the error does not decrease any more by adjusting the parameters.

The outline of this paper is as follows. In Section 2 we introduce RBF Networks method for approximation of function and its derivatives. In Section 3 we formulate the two-stage gradient learning algorithm. Section 4 discusses node insertion. Section 5 reports the net performance on Dirichlet and Neumann problems and the comparisons of results obtained to those already reported in literature. We close the paper with conclusions and discussion in Section 6.

2. Methods for approximation of function and its derivatives

Some of the common types of RBF are as follows:

Linear function: $\phi_k = 1 + r_k$.
Duchon radial cubics: $\phi_k = r_k^3$.
Radial quadratic plus cubic: $\phi_k = 1 + r_k^2 + r_k^3$.
TPS: $\phi_k = r_k^2 \ln r_k$
Hardy MQ: $\phi_k = (r_k^2 + C^2)^{n/2}$ with C being a user specified constant and n being a positive integer.
Inverse MQ: $\phi_k = 1/\sqrt{(r_k^2 + C^2)}$.
Gaussian: $\exp(-r_k^2/C^2)$.

In this paper, many kinds of RBFs will be used to construct the optimal network. In order to express our algorithm clearly and concisely, we will only use the MQ RBFs.

The chosen MQ is given by

$$\phi^{(i)}(\|x - c^{(i)}\|) == \sqrt{(r^2 + a^{(i)2})}, \tag{1}$$

where, $a^{(i)} > 0$, and the parameters $c^{(i)}$, $a^{(i)}$ are the ith RBFs' center and width. Let us consider an RBF-network as a function approximator:

$$u(x) = \sum_{i=1}^{m} w^{(i)} \phi^{(i)}(x) = \sum_{i=1}^{m} w^{(i)} \sqrt{(r^2 + a^{(i)2})}, \tag{2}$$

where m is the number of the RBFs (the hidden neurons).

Given in Eq. (2), the derivatives of the function $u(x)$ are calculated by

$$u_{j\ldots l}(x) = \frac{\partial^k u}{\partial x_j \cdots \partial x_l} = \sum_{i=1}^{m} w^{(i)} \frac{\partial^k \phi^{(i)}}{\partial x_j \cdots \partial x_l}. \tag{3}$$

The derivatives (e.g. up to second order with respect to x_j) are calculated by

$$u_j(x) = \sum_{i=1}^{m} w^{(i)} h^i(x), \tag{4}$$

$$u_{jj}(x) = \sum_{i=1}^{m} w^{(i)} \bar{h}^i(x), \tag{5}$$

where

$$h^i(x) = \frac{\partial \phi^{(i)}}{\partial x_j} = \frac{(x_j - c_j^{(i)})}{(r^2 + a_j^{(i)2})^{0.5}}, \tag{6}$$

$$\bar{h}^i(x) = \frac{\partial h^{(i)}}{\partial x_j} = \frac{r^2 + a^{(i)^2} - (x_j - c_j^{(i)^2})}{(r^2 + a_j^{(i)2})^{1.5}}. \tag{7}$$

3. Two-stage gradient learning algorithm

First, let us begin to consider the 2D Poisson's equation over the domain Ω

$$\Delta u = p(x), \qquad x \in \Omega, \tag{8}$$

where Δ is the Laplace operator, x is the spatial position, p is a known function of x and u is the unknown function of x to be found. Eq. (8) is subject to Dirichlet and/or Neumann boundary conditions over the boundary $\partial\Omega$

$$u = p_1(x), \qquad x \in \partial\Omega_1, \tag{9}$$

$$n \times \nabla u = p_2(x), \qquad x \in \partial\Omega_2, \tag{10}$$

where n is the outward unit normal; ∇ is the gradient operator; $\partial\Omega_1$ and $\partial\Omega_2$ are the boundary of the domain such as $\partial\Omega_1 \cup \partial\Omega_2 = \partial\Omega$ and $\partial\Omega_1 \cap \partial\Omega_2 = \phi$; p_1 and p_2 are known functions of x.

Numerical solutions of PDEs such as Eqs. (8)–(10) is intimately connected with approximating function and its derivatives. It is proposed here that the solution u and its derivatives can be approximated in terms of basis functions (2), (4), and (5). The design of networks is based on the information provided by the given PDE and its boundary conditions.

Now we begin to introduce our method. With the model u being decomposed into m basis functions in a given family, the unknown parameters $w^{(i)}$, $c^{(i)}$, $b^{(i)}$, $i = 1, 2\cdots m$ are to be found by minimizing the following sum square error (SSE)

$$l = F(w, c, a) = \sum_{i=1}^{n} [u_{11}(x^{(i)}) + u_{22}(x^{(i)}) - p(x^{(i)})]^2 + \sum_{j=1}^{k} [u(x^{(j)}) - p_1(x^{(j)})]^2 + \sum_{j=1}^{k} [n_1 u_1(x^{(j)}) + n_2 u_2(x^{(j)}) - p_2(x^{(j)})]^2, \tag{11}$$

where $w = (w^{(1)}, w^{(2)}, \ldots, w^{(m)})$, $c = (c^{(1)}, c^{(2)}, \ldots, c^{(m)})$, $a = (a^{(1)}, a^{(2)}, \ldots, a^{(m)})$ and $x^{(i)} \in \Omega$, $x^{(j)} \in \partial\Omega$, $i = 1, 2, \ldots n$, $j = 1, 2, \ldots, k$ are some given discrete points.

The algorithm is a gradient descent type which is calculated in two stages, once to optimize the weights $w^{(i)}$ and once to optimize the centers and the widths $c^{(i)}$, $a^{(i)}$ as follows

Step 1: fix $c^{(i)}$, $a^{(i)}$, the purpose is to find $w^{(i)}$ that minimizes the SSE by the following formula

$$w_t^{(i)} = w_{t-1}^{(i)} - \eta_{t-1} \frac{\partial l(c_{t-1}^{(i)}, a_{t-1}^{(i)}, w_{t-1}^{(i)})}{\partial w_{t-1}^{(i)}}. \tag{12}$$

Step 2: fix $w_t^{(i)}$, the purpose is to find $c^{(i)}$, $a^{(i)}$ that minimizes the SSE by

$$c_t^{(i)} = c_{t-1}^{(i)} - \beta_{t-1} \frac{\partial l(c_{t-1}^{(i)}, a_{t-1}^{(i)}, w_t^{(i)})}{\partial c_{t-1}^{(i)}}, \tag{13}$$

$$a_t^{(i)} = a_{t-1}^{(i)} - \alpha_{t-1} \frac{\partial l(c_t^{(i)}, a_{t-1}^{(i)}, w_t^{(i)})}{\partial a_{t-1}^{(i)}}, \tag{14}$$

where η_{t-1}, β_{t-1}, α_{t-1} are the learning rates at time $t-1$. The aforementioned steps are carried on in turn. We call it interactive gradient algorithm.

It has been known that the learning rates are very important for the convergence of the network parameters. If it is small, the convergence is slight; if it is large, the parameters oscillate and do not converge. But how to choose the best learning rates is problem-dependent. η_{t-1} at time of $t-1$ is determined by minimizing the SSE (11) in which $w^{(i)}$ is replaced by:

$$w_{t-1}^{(i)} - \eta_{t-1} \frac{\partial l}{\partial w_{t-1}^{(i)}}, \tag{15}$$

and can be obtained directly.

β_{t-1}, α_{t-1} can be decided by a recurrent procedure. For example, for some positive number, $a > 0$, set, $\beta_{t-1} = a$, the proper learning rate β_{t-1} is determined by the following formula:

$$\beta_{t-1} = \begin{cases} \dfrac{a}{10}, & \text{if } l(c_t^{(i)}, a_t^{(i)}, w_t^{(i)}) > l(c_{t-1}^{(i)}, a_{t-1}^{(i)}, w_t^{(i)}) \\ 10a, & \text{if } l(c_t^{(i)}, a_t^{(i)}, w_t^{(i)}) < l(c_{t-1}^{(i)}, a_{t-1}^{(i)}, w_t^{(i)}) \end{cases} \tag{16}$$

There are two features in this algorithm compared with the conventional gradient descent algorithm: (a) the learning rates are decided properly; (b) in Eqs. (13) and (14), $c_t^{(i)}$ is determined by $c_{t-1}^{(i)}$, $a_{t-1}^{(i)}$, $w_t^{(i)}$, not by $c_{t-1}^{(i)}$, $a_{t-1}^{(i)}$, $w_{t-1}^{(i)}$; $a_t^{(i)}$ is determined by $c_t^{(i)}$, $a_{t-1}^{(i)}$, $w_t^{(i)}$, not by $c_{t-1}^{(i)}$, $a_{t-1}^{(i)}$, $w_{t-1}^{(i)}$.

Sometimes it is difficult to determine the centers and the widths by two-stage gradient method when the governing equation is non-linear. They are determined by the following formulae:

$$c_{t+1}^{(i)} = c_t^{(i)} + N(0, \sigma) c_t^{(i)}, \qquad a_{t+1}^{(i)} = a_t^{(i)} + N(0, \sigma) a_t^{(i)}, \tag{17}$$

where $N(0, \sigma)$ is a random number chosen from a Gaussian probability distribution with mean zero and variance σ. If the SSE of the net is less than the previous one, then the centers and the widths changes according to Eq. (17). Otherwise, the old values are kept and the value σ is initialized to a fix value and then modified according to the following rule:

$$\sigma(t+1) = \begin{cases} b_1 \sigma(t), & \text{if the SSE } l(t+1) \le l(t),\ b_1 > 1; \\ b_2 \sigma(t), & \text{if SSE } l(t+1) \ge l(t),\ b_2 < 1. \end{cases} \tag{18}$$

4. Node insertion

Let us assume that we are given a large number of data and need to fit them with different RBFs within a given tolerance. The idea is to start with a very few nodes of set N in the domain, and then, as long as the SSE exceeds the tolerance, repeatedly insert a node at that data location whose SSE component is the largest. Here is the Algorithm in detail.

Algorithm. Node insertion

1. Let data points $X = \{x^{(1)}, x^{(2)}, \ldots, x^{(n)}\}$, and object output data $y^{(i)}$, $i = 1, 2, \ldots, n$ and a tolerance ε (TOL) be given. If $x^{(i)}$ lies in the domain Ω, $y^{(i)}$ will be $p(x^{(i)})$. If $x^{(i)}$ belongs to the boundary, $y^{(i)}$ will be $p_1(x^{(i)})$ or $p_2(x^{(i)})$.
2. Choose m_0 initial nodes whose centers are some subset of X and let $m = m_0$, and calculate the SSE.

3. While SSE > TOL do 'weight' each data point $x^{(i)}$ according to its error component, i.e. let

$$e^{(i)} := [u_{11}(x^{(i)}) + u_{22}(x^{(i)}) - p(x^{(i)})]^2, \text{ or } [u(x^{(j)}) - p_1(x^{(i)})]^2, \text{ or } [n_1u_1(x^{(j)}) + n_2u_2(x^{(j)}) - p_2(x^{(j)})]^2, \quad i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, k. \tag{19}$$

4. Find the data point $x^{(v)} \notin N$ with maximum weight $e^{(v)}$ and insert it as a node, i.e. $N = N \cup \{x^{(v)}\}$,

$$m = m + 1 \text{ and } w_0^{(v)} = 0 \tag{20}$$

5. Train the parameters of the new inserted node by two-stage gradient algorithm.
6. Train the parameters of all nodes by two-stage gradient algorithm.

5. Numerical examples

Although our technique is fit to solve differential equation (ODE and PDE), we focus on discussing the PDE problems.

A measure of the average error of the solution or the norm of the error of the solution, N_e, is defined as

$$N_e = \frac{\sqrt{\sum_i^q [u(x^{(i)}) - u_e(x^{(i)})]^2}}{\sqrt{\sum_{j=1}^q u_e(x^{(i)})^2}}, \tag{21}$$

where $u(x^{(i)})$ and $u_e(x^{(i)})$ are the calculated and exact solution at the point i, respectively, and q is the number of the training data.

5.1. Linear second order elliptic PDE

Example 1. The problem here is to determine a function $u(x_1, x_2)$ satisfying the following PDE

$$\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} = \sin(\pi x_1)\sin(\pi x_2), \tag{22}$$

on the rectangle $0 \le x_1 \le 1$, $0 \le x_2 \le 1$ subject to the Dirichlet condition $u = 0$ along the whole boundary of the domain. The exact solution is given by

$$u_e(x_1, x_2) = -\frac{1}{2\pi^2}\sin(\pi x_1)\sin(\pi x_2). \tag{23}$$

This problem is solved using RBF by Mai-Duy and Tran-Cong (2001). The authors used the following configure 5×5 (i.e. 5×5 hidden nodes in the hidden layer), 10×10 and 20×20 to solve the problem using the following data point densities 5×5, 10×10 and 20×20. In the present approach, with the configure of only 21 nodes and 25 data points, we get the accuracy $N_e = 5.32 \times 10^{-3}$. It is obviously less than the accurate 10^{-2} of Mai-Duy and Tran-Cong (2001) with 25 nodes. The results are shown in Fig. 1.

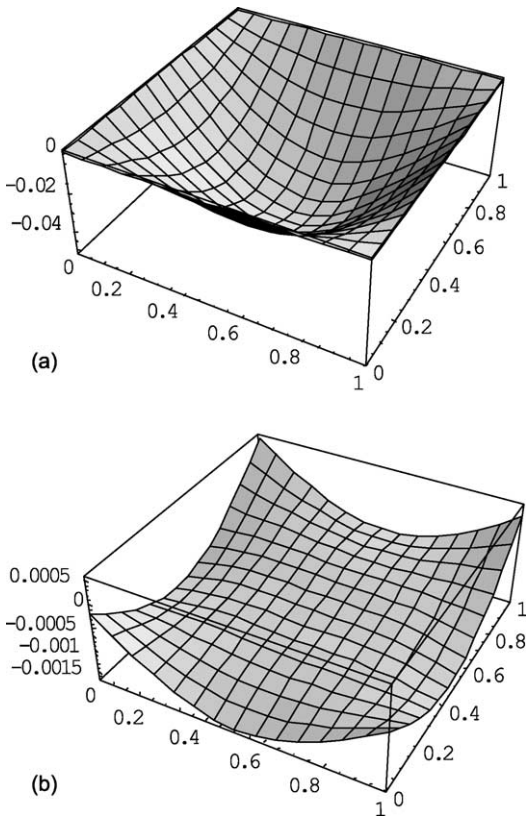

Fig. 1. Net approximation results for the Dirichlet problem of (22), (a) plot of the exact solution, (b) plot of the approximation error curve with 21 RBFs (9 MQ and 12 Gaussian) on 25 data points.

Example 2. In this example, the boundary conditions of the problem include both Dirichlet and Neumann type. Consider the following PDE

$$\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} = (\lambda^2 + \mu^2)\exp(\lambda x_1 + \mu x_2) \tag{24}$$

on the rectangle $0 \le x_1 \le 1, 0 \le x_2 \le 1$ with the following boundary conditions

$$u = \exp(\lambda x_1 + \mu x_2), \quad x_2 = 0 \text{ and } x_2 = 1$$
$$u_1 = \lambda \exp(\lambda x_1 + \mu x_2), \quad x_1 = 0 \text{ and } x_1 = 1 \tag{25}$$

The exact solution is

$$u_e = (x_1, x_2) = \exp(\lambda x_1 + \mu x_2), \tag{26}$$

here λ and μ are chosen to be 2 and 3, respectively. This problem was solved using MQ approximation scheme by Kansa (1990b). The author used a total of 30 nodal points, including 23 scattered points in the interior and

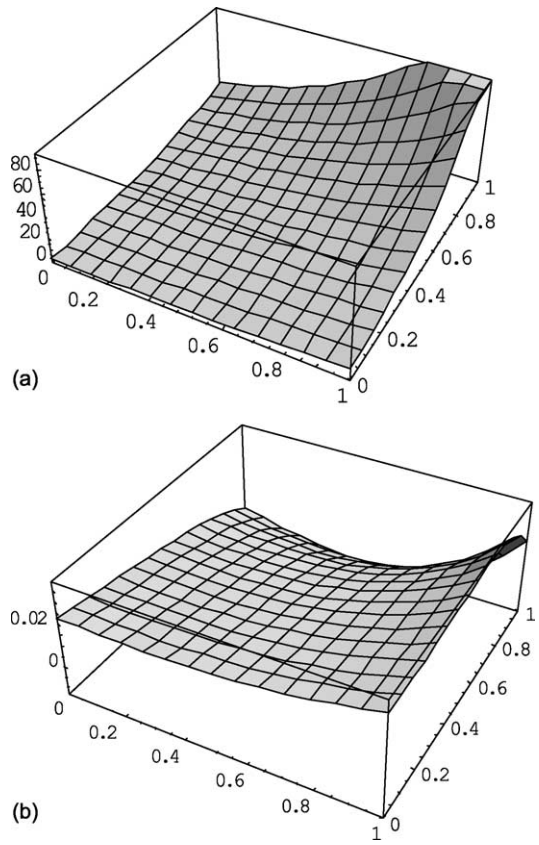

Fig. 2. Net approximation results for the Dirichlet problem of (24), (a) plot of the exact solution, (b) plot of the approximation error curve with 27 RBFs (13 MQ and 14 Gaussian) on 40 data points.

18 along the boundary. The reported results showed that the norm of error is 1.66×10^{-2}. In the present approach, with the configure of 27 nodes and 40 data points, we get a greater accuracy $N_e = 2.2 \times 10^{-4}$. The results are shown in Fig. 2.

5.2. Non-linear second order elliptic PDE

In recent years, many authors have studied the classical solution of the Dirichlet problem in a bounded convex domain Ω in R^n, $n \geq 2$ with smooth boundary $\partial\Omega$:

$$\begin{cases} f(\lambda(D^2u)) = \psi(x,u) & \text{in } \Omega \\ u = \phi(x) & \text{on } \partial\Omega \end{cases}, \tag{27}$$

where $D^2u = \{u_{ij}\}$ is the Hessian matrix, $\lambda(D^2u)$ are the eigenvalues $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ of the Hessian matrix, D^2u, and $f = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ is a smooth symmetric function of $\lambda_1, \lambda_2, \ldots, \lambda_n$. See for instance, Caffarelli, Nirenberg, and Spruck (1987), Cheng and Yau (1977), Lions, Trudinger, and Urbas (1986), and Nell and Trudiger (1995). The equation is assumed to be elliptic for the functions under consideration, i.e.

$$\frac{\partial f}{\partial \lambda_i} > 0, \quad \forall i \tag{28}$$

and to satisfy

f is a concave function. (29)

The function f will be required to satisfy other conditions. It is assumed to be defined in an open convex cone $\Gamma \subset R^n$, and $\Gamma \neq R^n$, with vertex at the origin, containing the positive cone:

$$\{\lambda \in R^n | \text{each component } \lambda_i > 0\}. \tag{30}$$

Eqs. (28)–(30) guarantees the Dirichlet problem has only one solution.

But there are few results of numerical solution. In this work, we are going to study the numerical solution of Monge–Ampere equation which is a special case of Eq. (27).

The Dirirchlet problem for Monge–Appere equation is as follows:

$$\begin{cases} \det D^2u = \varphi & \text{in } \Omega, \\ u = \phi & \text{on } \partial\Omega \end{cases} \tag{31}$$

where Ω is a bounded convex domain in R^n, $n \geq 2$, D^2u is the Hessian matrix of u and det represents the determination of D^2u.

With the earlier assumptions and explanations, the following problem will be studied.

Table 1
Results for the Dirichlet problem of $\det D^2u = 4$

x	y	Exact	Numerical
0.090431	0.0051258	0.008203	0.068002
0.907687	−0.262936	0.893031	0.811486
−0.49403	0.415965	0.417092	0.460829
0.212206	0.213663	0.090683	0.168499
−0.550199	−0.708161	0.804211	0.72451
0.063556	−0.51718	0.271514	0.335523
0.523413	0.256127	0.339562	0.349226
−0.370998	−0.232061	0.191492	0.309568
−0.606749	−0.562723	0.684801	0.704808
−0.838913	0.171053	0.733034	0.696195
0.158338	−0.218039	0.072612	0.147352
−0.660485	−0.121274	0.450948	0.475388
0.992115	0.125333	1	0.891162
0.637424	0.770513	1	0.904439
0.187381	0.982287	1	1.09248
−0.309017	0.951057	1	1.06037
−0.728969	0.684547	1	1.06959
−0.992115	0.125333	1	1.10944
−0.929776	−0.368125	1	0.904333
−0.535827	−0.844328	1	0.931287
−0.062791	−0.998027	1	0.947677
0.309017	−0.951057	1	0.941198
0.728969	−0.684547	1	0.983203
0.968583	−0.24869	1	1.12118

Example 3.

$$\begin{cases} \det D^2 u = 4 & \text{in } \Omega \\ u = 1 & \text{on } \partial\Omega \end{cases}, \tag{32}$$

where Ω is a unit circular domain. The exact solution is

$$u_e(x_1, x_2) = x_1^2 + x_2^2. \tag{33}$$

It should be pointed out that Eq. (30) is necessary. Without it, Example 3 will have at least another solution

$$u_e(x_1, x_2) = 2 - (x_1^2 + x_2^2). \tag{34}$$

If we study the numerical solution of Eq. (32) satisfying Eqs. (28)–(30), we have to modify the SSE as follows:

$$l = F(w,c,a) = \sum_{i=1}^{n} [\det D^2 u(x^{(i)}) - 4]^2 + \sum_{j=1}^{k} (u(x^{(j)}) - 1)^2 + \omega \left(\sum_{i=1}^{n} [u_{11}(x^{(i)}) - 1]^2 + \sum_{i=1}^{n} [u_{22}(x^{(i)}) - 1]^2 \right) \tag{35}$$

where ω ($\omega = 0.01$) is a positive constant and $x^{(i)}$, $x^{(j)}$, $i = 1,\ldots,n, j = 1,\ldots,k$ are, respectively, points inside the domain and on the boundary. Numerical results are displayed in Table 1 with 52 nodes and 110 data points (60 in the domain and 50 on the boundary).

6. Conclusions

In this paper, a new incremental algorithm for growing RBF networks and a two-stage learning strategy for training the net parameters are reported. The novelty of the algorithm has different aspects. First of all, all learning procedure is accomplished by a two-stage gradient method, which not only optimizes the weights of the net, but also optimizes the centers and widths of the Neurons. Moreover, the activation function of the added neuron can be different RBF. The results show that the strategy is better than the algorithm whose neurons are the same RBF type.

The network here allows us to obtain a fast and accurate reconstruction of some solutions of DEs starting from randomly sampled data sets. The incremental architecture allows us to achieve good approximation results without wasting memory space (only a few neurons are needed) and computational time. This is due to the ability of the net to save training time by selectively growing the net structure and by keeping the action of the different activation functions localized.

A comparison to other incremental algorithms reported in literature has showed that the learning strategy proposed here is able to achieve better performance (compared to other approaches) both in terms of net size and computational time.

It should be pointed out that there is no mathematical theory to guarantee which RBF type should be chosen when adding a new node. The proper choice of such functions requires some skill and experience on the user's side. Further developments of the present algorithm and their application in numerical solutions of DEs are under way and the results will be reported in future (Definitions).

Acknowledgements

This work is supported by the Chinese Natural science fund (Grant No. 69973002).

References

Caffarelli, L., Nirenberg, L., & Spruck, J. (1987). The Dirichlet problem for non-linear second order elliptic equations (1): Monge–Ampere equations. *Communication on Pure and Applied Mathematics*, *37*, 369–402.

Cheng, S. Y., & Yau, S. Y. (1977). On the regularity of the Monge–Ampere equation. *Communication on Pure and Applied Mathematics*, *30*, 41–68.

Esposito, A., Marinaro, M., Oricchio, D., & Scarpetta, S. (2000). Approximation of continuous and discontinuous mappings by a growing neural RBF-based algorithm. *Neural Networks*, *13*, 651–665.

Franke, C., & Schaback, R. (1997). Convergence orders of meshless collocation methods using radial basis functions. *Advances in Computational Mathematics*, *8*(4), 381–399.

Franke, C., & Schaback, R. (1998). Solving partial differential equations by collocation using radial basis functions. *Applied Mathematics and Computation*, *93*(1), 73–82.

Hughes, T. J. R. (1987). *The finite element method*. New Jersey: Prentice Hall.

Jianyu, L., Siwei, L., & Yingjian, Q. (2002). Interactive gradient algorithm for radial basis function networks. *The Sixth International Conference on Signal Processing*, 1187–1190.

Kansa, E. J. (1990a). Multi-quadrics—a scattered data approximation scheme with applications to computational fluid dynamics—II. *Computers in Mathematical Applications*, *19*(8/9), 127–145.

Kansa, E. J. (1990b). Multi-quadrics—a scattered data approximation scheme with applications to computational fluid dynamics—II. *Computers in Mathematical Applications*, *19*(8/9), 147–161.

Lions, P.-L., Trudinger, N. S., & Urbas, J. I. E. (1986). The Neumann problem for equations of Monge–Ampere type. *Communication on Pure and Applied Mathematics*, *XXXIX*, 539–563.

Madych, W. R., & Nelson, S. A. (1990). Multi-variable interpolation and conditionally positive definite functions-II. *Mathematical Computations*, *54*, 211–230.

Mai-Duy, N., & Tran-Cong, T. (2001). Numerical solution of differential equations using multi-quadric radial basis function networks. *Neural Networks*, *14*, 185–199.

Powell, M. J. D. (1994). The uniform convergence of TPS interpolation in two dimensions. *Numerische Mathematik*, *68*(1), 107–128.

Smith, G. D. (1978). *Numerical solution of partial differential equations: Finite difference methods*. Oxford: Clarendon Press.

Stead, S. (1984). Estimation of gradients from scattered data. *Rocky Mountain Journal of Mathematics*, *14*, 265–279.

Trudiger, N. (1995). On the Dirichlet problem for Hessian equations. *Acta Mathematica*, *175*, 151–164.

Wendland, H. (1999). Meshless Galerkin methods using radial basis functions. *Mathematical Computations*, *68*(228), 1521–1531.

Wu, Z. (1998). In Z. Chen, C. A. Micchelli, & Y. Xu (Eds.), *Advances in computational mathematics: Guangzhou* (pp. 202*Lectures notes on pure and applied mathematics*, New York: Marcel Dekker.

Wu, Z., & Schaback, R. (1994). Shape preserving properties and convergence of univariate multi-quadic quasi-interpolation. *Acta Mathematice Applicatae Sinica*, *10*(4), 441–446.