Hindawi
International Journal of Mathematics and Mathematical Sciences
Volume 2018, Article ID 9457578, 4 pages
https://doi.org/10.1155/2018/9457578

Research Article

Efficient Solving of Boundary Value Problems Using Radial Basis Function Networks Learned by Trust Region Method

Mohie Mortadha Alqezweeni, Vladimir Ivanovich Gorbachenko, Maxim Valerievich Zhukov, and Mustafa Sadeq Jaafar

Penza State University, Penza, Russia

Correspondence should be addressed to Vladimir Ivanovich Gorbachenko; gorvi@mail.ru

Received 7 March 2017; Revised 3 October 2017; Accepted 6 November 2017; Published 3 June 2018

Academic Editor: Irena Lasiecka

A method using radial basis function networks (RBFNs) to solve boundary value problems of mathematical physics is presented in this paper. The main advantages of mesh-free methods based on RBFN are explained here. To learn RBFNs, the Trust Region Method (TRM) is proposed, which simplifies the process of network structure selection and reduces time expenses to adjust their parameters. Application of the proposed algorithm is illustrated by solving two-dimensional Poisson equation.

1. Introduction

Mesh-free methods of solving boundary value problems have been widely studied in different researches in the last decade [1]. They belong to the class of projection methods [2–6]. In this paper, we are studying one of the most promising mesh-free methods using the radial basis functions (RBFs). In the case of their application, the approximation of the decision is presented as the weighted sum of RBFs where weights are selected in such a way that the approximated solution satisfies the boundary value problem in the selected sample points. The major difficulty of using the mesh-free methods based on RBFs is the nonformalizable selection of basis function parameters. This issue can be overcome with radial basis function neural networks (RBFNs) [7–11]. The process of boundary value problems' solution using RBFNs comes down to the RBFNs learning. The main difference between the RBFN method and other mesh-free methods is that it adjusts not only the weight of the basis functions but also their parameters [7–11]. In [12], it was suggested to use the Trust Region Method (TRM) for training RBNFs [13]. There are new results to improve the Trust Region Method performance in the submitted article, in particular the use of the Hessian matrix approximation to reduce the computation cost. Two approaches were considered for the initial allocation of radial basis function centers, and more optimal values of Trust Region Method hyperparameters were found. In Section 2, the RBFNs are discussed in order to solve the boundary value problems. Section 3 presents the proposed methodology of learning RBFNs using TRM. Section 4 presents the experimental analysis of the proposed solution to solve the boundary value problems. Finally, Section 5 presents the conclusion of the paper.

2. Radial Basis Function Networks

RBFN is a network consisting of two layers [14]:

(i) The first layer realizes nonlinear conversion of an input vector $\mathbf{x} = (x_1, x_2, \ldots, x_d) \in R^d$.

(ii) The second layer does the linear summation. RBF is used as a conversion function.

The output of a network is described by the expression

$$u(\mathbf{x}) = \sum_{m=1}^{M} w_m \varphi_m(\mathbf{x}; \mathbf{p}_m), \quad (1)$$

where M is the quantity of RBFs, w_m is the weight of RBF, and $\mathbf{p}_m$ is the RBF parameter vector F. Gauss's functions, multiquadrics, and the reverse multiquadrics, among others, are examples of RBFs [14].

Consider the process of solving boundary value problems using RBFNs using an example of the boundary value problem, given in the following operator form:

$$\begin{aligned} Lu(\mathbf{x}) &= f(\mathbf{x}), \quad \mathbf{x} \in \Omega, \\ Bu(\mathbf{x}) &= p(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega, \end{aligned} \tag{2}$$

where u is a desired solution; L is a differential operator; B is an operator of the boundary conditions; Ω is an area of solutions; $\partial\Omega$ is a border of the area; f and p are known functions. The solution is approximated by RBFN:

(1) Select N domestic and K boundary sample points from the sets Ω and $\partial\Omega$.

(2) Determine the structure of the network: the network type, the number of RBFs M, and the RBFs type.

Specify initial values: $\mathbf{w} = (w_1, w_2, \ldots, w_M)$ are the weights of RBFs and $\mathbf{p} = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_M)$ are parameters of RBFs (the structure of the vector $\mathbf{p}$ elements depends on the type of the functions; e.g., for a two-dimensional Gaussian function, it can be defined as $\mathbf{p}_i = (a_i, c_i^1, c_i^2)$, where $i = \overline{1 \cdots M}$, a_i is the shape parameter (width), and (c_i^1, c_i^2) is the position of its center).

(3) Learn the network; that is, find such values of $\mathbf{w}$ and $\mathbf{p}$ that the error functional representing the sum of squares of residuals in sample points reaches the minimum value:

$$J(\mathbf{w}, \mathbf{p}) = \sum_{i=1}^{N} \left[Lu(\mathbf{x}_i; \mathbf{w}, \mathbf{p}) - f(\mathbf{x}_i)\right]^2 + \lambda \sum_{i=N+1}^{N+K} \left[Bu(\mathbf{x}_i; \mathbf{w}, \mathbf{p}) - p(\mathbf{x}_i)\right]^2 \longrightarrow \min, \tag{3}$$

where $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \in \Omega$, $\mathbf{x}_{N+1}, \mathbf{x}_{N+2}, \ldots, \mathbf{x}_{N+K} \in \partial\Omega$ λ is the selectable penalty factor, which takes into account the residuals in the boundary sample points. The trained network provides the solution at any arbitrary point when its coordinate comes to the input layer of the network.

3. Learning RBFN Using TRM

The efficiency of neural network method for solving boundary value problems depends on the efficiency of the method of solving the problem of the error functional minimization (3). TRM is one of the best approaches to solve problem (3) [12, 15]. The key features of this method are as follows: it has the possibility of simultaneous optimization of a large number of parameters; it has a high level of efficiency and convergence even for bad-conditioned problems; it can overcome local minimums; it has the capability of minimization of concave functions, that is, functions with negative definite Hessian matrix; and finally, when the second-order Taylor expansion is used as an approximating function, the problem of the objective function minimization reduces to the problem of minimizing a quadratic functional [13]. All these make TRM ideal for solution of the problem of minimizing the error functional, which often has a large number of parameters to be optimized and many local minima, and it is an ill-conditioned problem.

The basic idea of TRM is that, at each iteration k of the function $f(\mathbf{x})$ minimization, where $\mathbf{x} \in \Omega \subseteq R^n$ (Ω is the domain of definition and R is a set of real numbers), the function f is replaced by an approximating function m_k in the trust region $B_k \subseteq \Omega$ and the minimum of m_k is calculated in B_k, which becomes a new minimum of f.

Depending on how the decrease predicted by the model is confirmed by the objective function, the decision on the expansion or contraction of the trust region is taken. There is a formal description of the algorithm.

Algorithm 1 (TRM).

Step 1 (*initialization*). Specify an initial value of $\mathbf{x}_0$, the radius of the trust region Δ_0, the threshold accuracy of the estimated models μ_1 and μ_2 such that $0 < \mu_1 \leq \mu_2 < 1$, the transformation coefficients γ_1 and γ_2 of the trust region ($0 < \gamma_1 \leq \gamma_2 < 1$), and the number of the iterations $k = 0$.

Step 2 (*approximation of* f). Select the norm $\|\cdot\|$ and construct a function m_k, approximating function f in the area B_k.

Step 3 (*minimization of* m_k). Select the method of conditional minimization of m_k, which allows finding such a step $\mathbf{s}_k$ that point $\mathbf{x}_k + \mathbf{s}_k$ is a global minimum of m_k in B_k.

Step 4. This step consists in evaluation of the model accuracy p_k that is calculated as follows:

$$p_k = \frac{f(\mathbf{x}_k) - f(\mathbf{x}_k + \mathbf{s}_k)}{m_k(\mathbf{x}_k) - m_k(\mathbf{x}_k + \mathbf{s}_k)}. \tag{4}$$

If $p_k \geq \mu_1$, then $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_k$; otherwise, $\mathbf{x}_{k+1} = \mathbf{x}_k$.

Step 5. Change the trust region radius

$$\Delta_{k+1} \in \begin{cases} [\Delta_k; \infty), & \text{if } p_k \geq \mu_2, \\ [\gamma_2 \Delta_k; \Delta_k), & \text{if } p_k \in [\mu_1, \mu_2), \\ [\gamma_1 \Delta_k; \gamma_2 \Delta_k), & \text{if } p_k < \mu_1. \end{cases} \tag{5}$$

Step 6 (*increasing the number of the iterations* $k = k + 1$). If the required accuracy of the solution has been reached, or k is equal to the maximum number of iterations, or the radius of the trust region is too small, then complete the learning; otherwise, go back to Step 2.

Step 7 (*stop*). The algorithm is a generalized algorithm of TRM, as it has no instructions on how to construct a function

m_k, select the norm $\|\cdot\|$, and what method to use for m_k minimization.

Since the error functional is a twice differentiable function, hence the second-order Taylor series can be used as m_k model of J; as the norm, we are using the Euclidean norm here. To get the second-order Taylor series, it is necessary to calculate the Hessian matrix, which has a large computation cost. Instead of the exact value of the Hessian, we are using its approximate value, which is a multiplication of the Jacobi matrices $\mathbf{H}(\mathbf{x}) \approx \mathbf{G}(\mathbf{x}) = [\mathbf{J}(\mathbf{x})]^T \mathbf{J}(\mathbf{x})$, where the Jacobian of the error function has the form

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} \frac{\partial r_1}{\partial q_1} & \frac{\partial r_1}{\partial q_2} & \cdots & \frac{\partial r_1}{\partial q_{M\cdot(1+d+l)}} \\ \frac{\partial r_2}{\partial q_1} & \frac{\partial r_2}{\partial q_2} & \cdots & \frac{\partial r_2}{\partial q_{M\cdot(1+d+l)}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r_{N+K}}{\partial q_1} & \frac{\partial r_{N+K}}{\partial q_2} & \cdots & \frac{\partial r_{N+K}}{\partial q_{M\cdot(1+d+l)}} \end{bmatrix}, \quad (6)$$

where

$$r_i(\mathbf{q}) = \begin{cases} Lu(\mathbf{x}_i; \mathbf{q}) - f(\mathbf{x}_i), & 1 \le i \le N, \\ \sqrt{\lambda}\left[Bu(\mathbf{x}_i; \mathbf{q}) - p(\mathbf{x}_i)\right], & N < i \le N+K, \end{cases} \quad (7)$$

$$\mathbf{q} = \left(w^1, p_1^1, \ldots, p_l^1, \ldots, w^M, p_1^M, \ldots, p_l^M\right).$$

Using the second-order Taylor series leads to the need for solving the conditional problem of a quadratic functional minimization. To solve it, Steihaug's method [16] is used here. This method is a modification of the method of preconditioned conjugate directions, taking into account the restrictions on the solution (the solution should lie in B_k) during m_k functional minimization and allowing working with negative definite Hessian matrix.

4. Experimental Study

As an example of solving boundary value problems using RBFN, learned by TRM, consider the boundary value problem for the two-dimensional Poisson equation, described in [8]

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = f(x,y), \quad (x,y) \in \Omega,$$
$$u(x,y) = p(x,y), \quad (x,y) \in \partial\Omega. \quad (8)$$

This problem often arises in thermodynamics, electrostatics, and image processing.

Let the solution domain be square bounded by the points $(0,0)$ and $(1,1)$, $f(x,y) = \sin(\pi x)\sin(\pi y)$, $p(x,y) = 0$. This problem has an analytical solution $u_a(x,y) = -(1/2\pi^2)\sin(\pi x)\sin(\pi y)$. We will use it to evaluate the accuracy of the resulting numerical solution. There is an error functional of this problem:

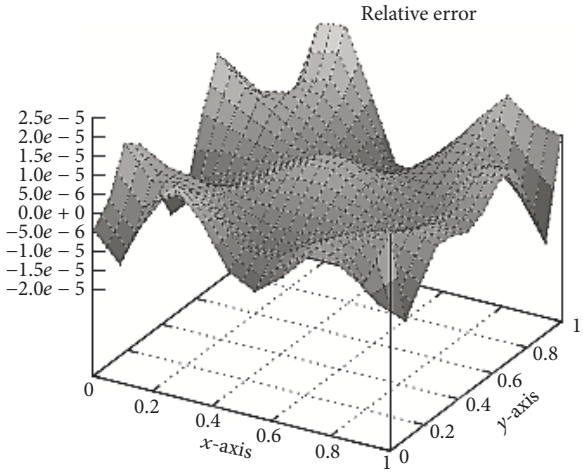

Figure 1: Experiment number 1, error solutions.

$$J(\mathbf{w}, \mathbf{p}) = \sum_{i=1}^{N} \left[\frac{\partial^2 u_{\mathrm{RBF}}(x_i, y_i; \mathbf{w}, \mathbf{p})}{\partial x^2} + \frac{\partial^2 u_{\mathrm{RBF}}(x_i, y_i; \mathbf{w}, \mathbf{p})}{\partial y^2} - \sin(\pi x_i)\sin(\pi y_j) \right]^2 + \lambda \sum_{i=N+1}^{N+K} \left[u_{\mathrm{RBF}}(x_i, y_i; \mathbf{w}, \mathbf{p}) \right]^2. \quad (9)$$

Two experiments were done with different initial values of the network parameters. In both experiments, 144 randomly selected sample points were used to learn the networks. 100 of them were located in the area of solutions Ω, and 44 were located on the border $\partial\Omega$. Gaussian function was used as RBF. In the first experiment, RBFN consisted of 16 neurons, whose centers were randomly distributed in a square area bounded by points $(-0.2; -0.2)$ and $(1.2; 1.2)$. The width of the RBFs was randomly chosen from the interval $[0.3; 0.6]$. Initial values of the RBF weights have been chosen randomly from the interval $[-0.05; 0.05]$. Penalty factor $\lambda = 1000$. TRM parameters were equal to $\Delta_0 = 2$, $\mu_1 = 0.2$, $\mu_2 = 0.7$, $\gamma_1 = 0.5$, and $\gamma_2 = 0.7$. Note that the presented hyperparameters of both RBFN and TRM were selected manually, although more advanced techniques can be used in the future, like grid search, genetic algorithm, and others.

Learning of the network was completed in 8 iterations, with the value of the solution error being equal to $1.7 \cdot 10^{-4}$. The solution error was calculated according to the formula of the relative standard error:

$$\varepsilon = \frac{\sqrt{\sum_{i=1}^{N+K} \left(u_{\mathrm{RBF}}(x_i, y_i; \mathbf{w}, \mathbf{p}) - u_a(x_i, y_i)\right)^2}}{\sqrt{\sum_{i=1}^{N+K} u_a(x_i, y_i)^2}}. \quad (10)$$

The error graph is shown in Figure 1.

In the second experiment, the RBFN consisted of 64 neurons, whose centers were located in the nodes of a uniform square grid bounded by points $(-0.2; -0.2)$ and $(1.2; 1.2)$; the neurons' width was constant and equal to 0.5.

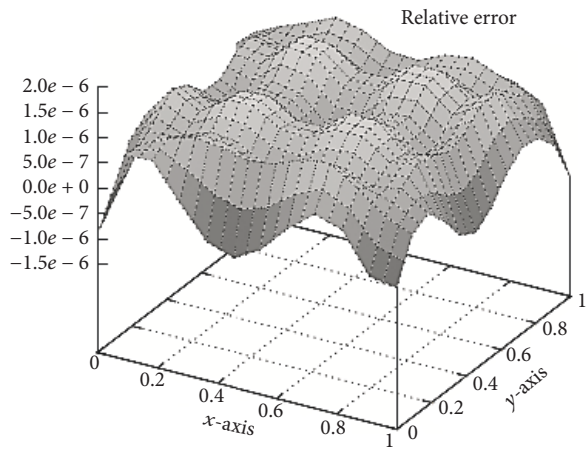

Figure 2: Experiment number 2, error solutions.

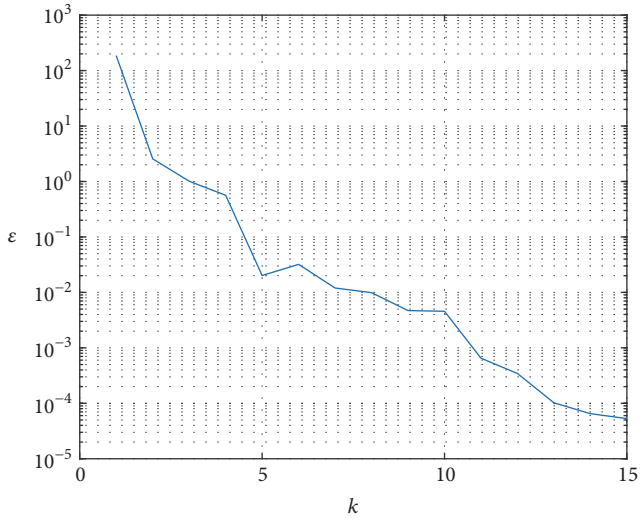

Figure 3: The dependency of the error ε (10) on the iteration number k.

Initial values of weights of the RBFs were equal to zero. The solution error was equal to $5.3 \cdot 10^{-5}$ and was reached for 15 iterations. The error graph is shown in Figure 2.

As expected, the more the neurons in the network, the more accurately it approximates the desired solution. However, the process of learning such a network requires much more time (for learning the network from the first experiment, it needed two times fewer iterations than training a network from the second experiment). The dependency of the error ε (10) on the iteration number k is shown in Figure 3.

5. Conclusion

In this paper, a method based on TRM was proposed for RBFNs learning. Approximate values of the Hessian matrix are used to improve its performance. This method allows reducing the time of network's learning.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was funded by the Russian Foundation for Basic Research, Project no. 16-08-00906A.

References

[1] G. R. Liu, *Mesh Free Methods: Moving Beyond the Finite Element Method*, CRC Press, Boca Raton, Fla, USA, 2002.

[2] G. E. Fasshauer, *Meshfree Approximation Methods with MATLAB*, World Scientific, Singapore, 2007.

[3] M. D. Buhmann, *Radial Basis Functions: Theory and Implementations*, vol. 12 of *Cambridge Monographs on Applied and Computational Mathematics*, Cambridge University Press, Cambridge, UK, 2003.

[4] W. Chen and Z. J. Fu, *Recent Advances in Radial Basis Function Collocation Methods*, Springer, 2013.

[5] G. Pang, W. Chen, and Z. Fu, "Space-fractional advection-dispersion equations by the Kansa method," *Journal of Computational Physics*, vol. 293, pp. 280–296, 2015.

[6] H. Sun, X. Liu, Y. Zhang, G. Pang, and R. Garrard, "A fast semi-discrete Kansa method to solve the two-dimensional spatiotemporal fractional diffusion equation," *Journal of Computational Physics*, vol. 345, pp. 74–90, 2017.

[7] N. Mai-Duy and T. Tran-Cong, "Numerical solution of differential equations using multiquadric radial basis function networks," *Neural Networks*, vol. 14, no. 2, pp. 185–199, 2001.

[8] L. Jianyu, L. Siwei, Q. Yingjian, and H. Yaping, "Numerical solution of elliptic partial differential equation using radial basis function neural networks," *Neural Networks*, vol. 16, no. 5-6, pp. 729–734, 2003.

[9] A. N. Vasiliev and D. A. Tarhov, *Principles and Techniques of Neural Network Modeling*, Nestor History, St. Petersburg, Russia, 2014.

[10] N. Yadav, A. Yadav, and M. Kumar, *An Introduction to Neural Network Methods for Differential Equations*, SpringerBriefs in Applied Sciences and Technology, Springer, 2015.

[11] C. S. Dash, A. K. Behera, S. Dehuri, and S. Cho, "Radial basis function neural networks: a topical state-of-the-art survey," *Open Computer Science*, vol. 6, no. 1, pp. 33–63, 2016.

[12] V. I. Gorbachenko and M. V. Zhukov, "Solving boundary value problems of mathematical physics using radial basis function networks," *Computational Mathematics and Mathematical Physics*, vol. 57, no. 1, pp. 145–155, 2017.

[13] A. R. Conn, N. I. M. Gould, and P. Toint, *Trust Region Method*, Society for Industrial and Applied Mathematics, Philadelphia, Pa, USA, 2000.

[14] S. O. Haykin, *Neural Networks and Learning Machines*, Pearson, 2008.

[15] V. I. Gorbachenko and M. V. Zhukov, "Approaches and methods of learning networks of radial basis functions for solving problems of mathematical physics," *Neurocomputers: Development, Application*, vol. 9, pp. 12–18, 2013.

[16] T. Steihaug, "The conjugate gradient method and trust regions in large scale optimization," *SIAM Journal on Numerical Analysis*, vol. 20, no. 3, pp. 626–637, 1983.