International Journal of Mathematics and Statistics Invention (IJMSI)
E-ISSN: 2321- 4767 P-ISSN: 2321-4759
www.ijmsi.org Volume 10 Issue 1 || January, 2022 || PP 22-27

Research for Solving Partial Differential Equations With High Accuracy

Hong Huang
Nanjing Normal University Zhongbei College,Jiangsu-212300,P.R.China

***ABSTRACT**: Construct a composite multi-layer radial basis function neural network to improve the Based on the real function approximation performance and operation accuracy, the high-precision composite multilayer radial basis function neural network is used to solve partial differential equations.*

***KEYWORDS**: deep neural network; high-precision solution; partial differential equation.*

Date of Submission: 28-03-2022 Date of Acceptance: 19-04-2022

I. INTRODUCTION

Numerical solution of partial differential equations (PDE) is one of the most computationally intensive fields in engineering and scientific applications [1]. The deep neural network (DNN) method has been applied in many research fields [2].There are many research,such as [3,4,5,6,7,8,9,10,11,12,16,17].But the solution accuracy is slightly worse [13]; The solution process takes a little longer [14]. By this,we construct a composite multi-layer radial basis function neural network which can provide a new and effective way with high computational accuracy, is better than that of the algorithm [15].

II. Classical BP neural network high-precision partial differential equation solving algorithm

2.1 Composite multilayer radial basis function neural network

The structure of the multi-layer radial basis function neural network includes several single-layer radial function networks [18,19].Denote the first layer network as $f_1(x) = w_0^1 + \sum_{k=1}^{n_1} w_k^1 \phi_k^1(x)$, where the Gaussian function is represented by $\phi_k^1(x)$, and w represents the weight, that is, $\phi(x) = \exp(-\|x - x_0\|^2 / c)$, where the center of the Gaussian function is represented by x_0, and the width coefficient and n -dimensional input samples are represented by c and x respectively. Divide the augmented sample $x_i' = (x_i, ay_i)$ into m clusters by the K-mean method, where $(a > 0)$, the sample mean in each cluster is regarded as the center of the Gaussian function, and the optimization algorithm is used to obtain Width coefficient, and apply the regular least square method to obtain the weight of the network $w_i = (w_0, w_1, \ldots, w_m)^T$. The output of the first layer of the network is denoted as $f_1(x)$, the true value of the objective function is denoted as y_i, and the residual is denoted as $e_i^1 = y_i - f_1(x_i)$. The objective function of the second layer network is e_i^1, and the fitting is implemented by $f_2(x) = \sum_{k=1}^{m_i} w_k^2 \phi_k^2(x)$. The fitting method is similar to that of the first layer, and the output is $f_2(x)$, $e_i^2 = e_i^1 - f_2(x_i)$. By analogy, the multi-layer radial basis function neural network can be obtained:

$$f(x) = f_1(x) + f_2(x) + \ldots + f_k(x) \quad (1)$$

Determine the number of layers and give a sufficiently small positive number ε to first record the generalized cross rate of the k -th layer as GCV_k. If $(GCV_k - GCV_{k-1}) / GCV_{k-1} > \varepsilon$, then the fitting error

$e_i^k\ (1 \le i \le N)$ to carry out calculations and continue to construct the $k+1$ -th layer network; if $(GCV_k - GCV_{k-1})/GCV_{k-1} \le \varepsilon$, discard the layer network and keep the $k-1$ -th layer network. The calculation equation of the generalized cross rate GCV is:

$$GCV_k = \frac{1}{N}\sum_{a=1}^{N}\left[e_a^{k-1} - f_k(x_a)\right]^2 / \left[1 - \frac{1}{N}tr(H)\right]^2 \quad (2)$$

In the equation: $H = B(B^T B + \lambda K)^{-1} B^T$, where $K = D^T D$. Then there are:

$$D = \begin{bmatrix} 1 & -2 & 1 & 0 & \dots & 0 \\ 0 & 1 & -2 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 1 & -2 & \dots & 1 \end{bmatrix} \quad (3)$$

The weight is determined by:

$$w = (B^T B + \lambda K)^{-1} B^T y \quad (4)$$

In order to further improve the performance of the real function approximation , each sample in the cluster is regarded as the center of a radial basis function [20], and each sample is related to a radial basis function. Correspondingly, these radial basis functions are simultaneously approximated to the real functions on the cluster, thereby further improving the accuracy of the multilayer radial basis function neural network.

(1) Construct the first layer of composite network

Through the augmented sample and K-mean method, the sample is divided into m1 clusters $C_{1p}(1 \le i \le n_{1p})$, and the division method is equivalent to the first layer of the multi-layer radial basis function neural network. $x_i^{1p}(1 \le i \le n_{1p})$ represents the samples in the p -th cluster C_{1p} of the first level, and $\tilde{\phi}_i^{1p}(x) = \exp(-\left\|x - x_i^{1p}\right\|^2 / c_i^{1p})$ represents the p -th child of the first level The radial basis function of the network, and $1 \le i \le n_{1p}$, where the width coefficient is expressed by c_i^{kp} . $f_{1p}(x) = \tilde{w}_0^{1p} + \sum_{i=1}^{n_{1p}} \tilde{w}_i^{1p}\tilde{\phi}_i^{1p}(x)$ represents the sub-radial basis function neural network of cluster C_{1p} . Taking all samples for x , the residual sum of squares can be expressed as:

$$RSS_{1p} = \sum_{j=1}^{N}(y_j - \tilde{f}_{1p}(x_j))^2 + \lambda_{1p} w_{1p}^T K_{1p} \tilde{w}_{1p} \quad (5)$$

It is possible to find $\tilde{w}_{1p}$ by the regular least square method. Let GCV_{1p} be the smallest and get the width coefficient c_i^{kp} to obtain the m_1 sub-radial basis function neural network $\tilde{f}_{11}(x), \tilde{f}_{12}(x), \dots, \tilde{f}_{1m_1}(x)$. By taking all the samples for x , the N equations $\hat{y} = \tilde{w}_1 \tilde{f}_{11}(x) + \tilde{w}_2 \tilde{f}_{12}(x) + \dots + \tilde{w}_{m_1} \tilde{f}_{1m_1}(x)$ are obtained, where $1 \le i \le N$; let $RSS = \sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \to \min$, the least square solution of $w_1 = (w_1, w_2, \dots, w_{m_1})$ can be obtained, then the regression model can be expressed as:

$$\hat{y}_1(x) = w_{11}\tilde{f}_{11}(x) + w_{12}\tilde{f}_{12}(x) + \dots + w_{1m_1}\tilde{f}_{1m_1}(x) \quad (6)$$

(2) Construct a composite network of the k -th ($k \ge 2$) layer

Perform calculation on the fitting error e_i^{k-1} of the $k-1$ -th layer, where $1 \le i \le N$, and treat it as the augmented sample of the k -th ($k \ge 2$) layer $\tilde{x}_i^k = (x_i, a_k e_i^{k-1})$, dividing the augmented sample $\tilde{x}_i^k$ by the K-mean method into m_k augmented clusters, which can be expressed as

$\tilde{C}_{kp} = \{\tilde{x}^{k}_{p_i} = (x_{p_i}, a_k e^{k-1}_{p_i}); 1 \le i \le n_{kp}\}$, where k -th ($k \ge 2$) and the p -th augmented cluster of the layer is represented by $\tilde{C}_{kp}$, and $1 \le p \le m_k$. The new cluster can be expressed as:

$$C_{kp} = \{x_{p_i} : 1 \le i \le n^k_p\} \quad (7)$$

where $1 \le p \le m_k$. Expressing the samples in the p -th cluster C_{kp} of the k -th layer by x^{kp}_i, and $1 \le i \le n^{kp}_p$, then the sub-radial basis function neural network opposite to the cluster C_{kp} can be expressed as:

$$\tilde{f}_{kp}(x) = \tilde{w}^{kp}_0 + \sum_{i=1}^{n_{kp}} \tilde{w}^{kp}_i \tilde{\phi}^{kp}_i(x) \quad (8)$$

where the radial basis function of the sub-network is expressed by $\tilde{\phi}^{kp}_i(x)(1 \le i \le n_{kp})$, that is, $\tilde{\phi}^{kp}_i(x) = \exp(-\left\|x - x^{kp}_i\right\|^2 / c^{kp}_i)$, and $1 \le i \le n_{kp}$, the width coefficient is expressed by c^{kp}_i. The weight vector of the sub-network is obtained by the regular least square method, which can be expressed as:

$$\hat{w}_{kp} = (B^T_{kp} B_{kp} + \lambda_{kp} K_{kp})^{-1} B^T_{kp} e^{k-1}_p \quad (9)$$

where the fitting error vector of the $k-1$ -th layer on the cluster C_{kp} is represented by $e^{k-1}_p = (e^{k-1}_{p_1}, e^{k-1}_{p_2}, \ldots, e^{k-1}_{p_{n_{kp}}})$. $B_{kp} = [1, B_{kp}]$, $B_{kp} = \{\phi_j(x_i)\}_{N \times n_{kp}}$ to obtain the p -th subnet of the k -th ($k \ge 2$) layer. In this way, a total of m_k sub-networks of the k -th layer are obtained, which can be expressed as $\tilde{f}_{k1}(x_i), \tilde{f}_{k2}(x_i), \ldots, \tilde{f}_{km_k}(x_i)$. By taking all the samples for x, N equations are obtained as $\hat{y}_i = w_1 \tilde{f}_{k1}(x_i) + w_2 \tilde{f}_{k2}(x_i) + \ldots + w_{m_k} \tilde{f}_{km_k}(x_i)$, where $1 \le i \le N$; let $RSS = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \to \min$, the least square solution of $w_k = (w_{k1}, w_{k2}, \ldots, w_{km_k})$ can be obtained, expressed as:

$$w_k = B_k (B^T_k B_k)^{-1} B^T_k e_{k-1} \quad (10)$$

where $e_k = [e^{k-1}_1, e^{k-1}_2, \ldots, e^{k-1}_N]^T$, and $B_k = \{y^j_k(x_i)\}_{N \times m_k}$. Then the regression model can be expressed as:

$$\hat{y}_k(x) = w_{k1} \tilde{f}_{k1}(x) + w_{k2} \tilde{f}_{k2}(x) + \ldots + w_{km_k} \tilde{f}_{km_k}(x) \quad (11)$$

It is the k -th layer network, and its generalized cross rate GCV_k can be expressed as:

$$GCV_k = \frac{1}{N} \sum_{j=1}^{N} \left[e^{k-1}_j - \tilde{y}_k(x_j) \right]^2 / \left[1 - \frac{1}{N} tr(H_k) \right]^2 \quad (12)$$

where $H_k = B_k (B^T_k B_k)^{-1} B^T_k$. This completes the construction of the k -th ($k \ge 2$) layer composite network. The number of network layers is determined in the same way as the multilayer radial basis function neural network.

2.2 PDE solving of composite multilayer radial basis function neural network

The partial differential operation unit is a composite multilayer radial basis function neural network. The composite multilayer radial basis function neural network and the composite multilayer radial basis function are denoted by $f(x)$ and $\phi(x)$ respectively. The specific partial derivative expression can be expressed as:

$$f_{j \ldots g}(x) = \frac{\partial^s f}{\partial x_j \ldots \partial x_g} = \sum_{i=1}^{m} w^{(i)} \frac{\partial^s \phi^{(i)}}{\partial x_j \ldots \partial x_g} \quad (13)$$

where the sum square error is expressed by g ; s is some discrete points; the first-order partial derivative and the second-order partial derivative of $f_{j \ldots g}(x)$ are expressed as $f_j(x)$ and $f_{jj}(x)$ respectively, the operation of the two The equation is like (14):

$$\begin{cases} f_j(x) = \sum_{i=1}^{m} w^{(i)} h^i(x) \\ f_{jj}(x) = \sum_{i=1}^{m} w^{(i)} \overline{h}^i(x) \end{cases} \quad (14)$$

where $h^i(x)$ and $\overline{h}^i(x)$ are both given known functions, and their calculation equations are as equation (15):

$$\begin{cases} h^i(x) = \dfrac{\partial \phi^{(i)}}{\partial x_j} = \dfrac{x_j - x_0}{(r^2 + c^2)^{0.5}} \\ \overline{h}^i(x) = \dfrac{\partial h^{(i)}}{\partial x_j} = \dfrac{\partial^2 \phi^{(i)}}{\partial x_j \partial x_j} = \dfrac{r^2 + c^2 - (x_j - x_0)^2}{(r^2 + c^2)^{1.5}} \end{cases} \quad (15)$$

where the center, width and radius of the composite multilayer radial base are represented by x_0 , c and r , respectively.

Suppose the two-dimensional Poisson equation in Ω space is:

$$\nabla^2 u = q(x), x \in \Omega \quad (16)$$

where Laplace change and spatial position are represented by ∇^2 and x respectively; the known function and unknown function related to x are represented by q and u respectively. The boundary conditions that define the equation (17) are:

$$\begin{cases} u = q_1(x), x \in \Gamma_1 \\ n \times \nabla u = q_2(x), x \in \Gamma_2 \end{cases} \quad (17)$$

where the unit normal vector and gradient operator are represented by n and ∇ respectively; the function of known x is represented by q_1 and q_2; the domain boundary is represented by Γ_1 and Γ_2, and $\Gamma_1 \cup \Gamma_2 = \Gamma$, and $\Gamma_1 \cap \Gamma_2 = \varphi$. The compound multilayer radial basis function neural network equations (1) and (15) will approximately replace the partial differential numerical solutions of equations (16) and (17). That is, the N -order derivative of the composite multilayer radial basis function is directly approximated to the N -order derivative of the initial function f . The N -order derivative of the radial basis function is replaced by the composite multilayer radial basis initial function, and the network approximates the initial function the result is obtained [21]. Based on this, this type of approximate neural network architecture constructed by partial differential equations and boundary conditions can be decomposed into m basis functions for the model u, and the unknown parameters of the composite multilayer radial basis function neural network can be determined by the least two Find out by multiplication [22], such unknown parameters include w_i , x_0 , c , where $i = 1, 2, \ldots, m$, then:

$$g = \sum_{x^{(i)} \in \Omega} [u_{11}(x^{(i)}) + u_{22}(x^{(i)}) - q(x^{(i)})]^2 + \sum_{x^{(i)} \in \Gamma_1} [u(x^{(i)}) + q_1(x^{(i)})]^2 + \sum_{x^{(i)} \in \Gamma_2} [n_1 u_1(x^{(i)}) + n_2 u_2(x^{(i)}) - q_2(x^{(i)})]^2 \quad (18)$$

where $i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, s$, where s represents some discrete points; $w = (w_0, w_1, \ldots, w_m)^T$, $c = (c_1, c_2, \ldots, c_m)$, $x = (x_1, x_2, \ldots, x_m)$, $x^{(i)} \in \Gamma_1$ and $x^{(j)} \in \Gamma_2$. After determining the center x_0 , width c

and weight w_i of the composite multilayer radial basis, the composite multilayer radial basis function neural network structure of l-th ($1 \le l \le K$) layer is obtained, and the composite multilayer radial basis function neural network structure is obtained through the number of composite layers. The layered radial basis function neural network solves partial differential equations [23], the process is as follows:

(1) After initialization, extract the training sample $X = \{x^{(1)}, x^{(2)}, \ldots, x^{(n)}\}$ and the expected target output y_i of the problem, where $i = 1, 2, \ldots, n$. Initialize ε, if the sample point $x^{(i)} \in \Omega$, y_i is determined by $q(x^{(i)})$. If $x^{(i)} \notin \Omega$, y_i is determined by $q_1(x^{(i)})$ or $q_2(x^{(i)})$.

(2) Construct a l-th -layer composite multi-layer radial basis function neural network structure, set m_0 hidden layer neurons at the same time, and implement random assignment to the relevant connection weights.

(3) Perform calculations on the value of the sum squared error g of the constructed network, which is the value of equation (18).

(4) Fix the radial basis center x_0 and the width c, and use the sum square error to optimize the weight w_i.

(5) Fix the weight w_i, and use the sum square error to optimize the center x_0 and width c of the radial basis.

(6) Perform judgment on whether the sum squared error g is higher than the initial constant ε in each calculation. If it is higher than ε, go to step (7), otherwise go to step (8).

(7) When calculating the sum square error of each data sample point $x^{(i)}$, pass $[u(x^{(i)}) + q_1(x^{(i)})]^2$, $[u_{11}(x^{(i)}) + u_{22}(x^{(i)}) - q(x^{(i)})]^2$ or $[n_1 u_1(x^{(i)}) + n_2 u_2(x^{(i)}) - q_2(x^{(i)})]^2$, respectively, for the relevant weight, center and width value is adjusted, go to step (3).

(8) Output the parameter values of the entire composite multilayer radial basis function neural network structure, that is, achieve high-precision solving of partial differential equations through the high-precision composite multilayer radial basis function neural network structure solution model.

REFERENCES

[1]. Sirignano J, MacArt J F, Freund J B. DPM: A deep learning PDE augmentation method with application to large-eddy simulation[J]. Journal of Computational Physics, 2020, 423: 109811.

[2]. Opschoor J A A, Schwab C, Zech J. Deep learning in high dimension: ReLU network Expression Rates for Bayesian PDE inversion[J]. SAM Research Report, 2020, 2020: OSZ20_920.

[3]. Wu K, Xiu D. Data-driven deep learning of partial differential equations in modal space[J]. Journal of Computational Physics, 2020, 408: 109307.

[4]. Long Z, Lu Y, Ma X, et al. Pde-net: Learning pdes from data[C]/International Conference on Machine Learning. PMLR, 2018: 3208-3216.

[5]. Long Z, Lu Y, Dong B. PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network[J]. Journal of Computational Physics, 2019, 399: 108925.

[6]. Xu H, Chang H, Zhang D. DLGA-PDE: Discovery of PDEs with incomplete candidate library via combination of deep learning and genetic algorithm[J]. Journal of Computational Physics, 2020, 418: 109584.

[7]. Pannekoucke O, Fablet R. PDE-NetGen 1.0: from symbolic partial differential equation (PDE) representations of physical processes to trainable neural network representations[J]. Geoscientific Model Development, 2020, 13(7): 3373-3382.

[8]. Belbute-Peres F A, Economon T, Kolter Z. Combining differentiable PDE solvers and graph neural networks for fluid flow prediction[C]//International Conference on Machine Learning. PMLR, 2020: 2402-2411.

[9]. Geneva N, Zabaras N. Modeling the dynamics of PDE systems with physics-constrained deep auto-regressive networks[J]. Journal of Computational Physics, 2020, 403: 109056.

[10]. Pannekoucke O, Fablet R. PDE-NetGen 1.0: from symbolic partial differential equation (PDE) representations of physical processes to trainable neural network representations[J]. Geoscientific Model Development, 2020, 13(7): 3373-3382.

[11]. Lye K O, Mishra S, Ray D, et al. Iterative surrogate model optimization (ISMO): An active learning algorithm for PDE constrained optimization with deep neural networks[J]. Computer Methods in Applied Mechanics and Engineering, 2021, 374: 113575.

[12]. Holl P, Koltun V, Um K, et al. phiflow: A Differentiable PDE Solving Framework for Deep Learning via Physical Simulations[C]//NeurIPS Workshop. 2020.

[13]. Yao H, Ren Y, Liu Y. Fea-net: A deep convolutional neural network with physicsprior for efficient data driven pde learning[C]//AIAA Scitech 2019 Forum. 2019: 0680.

[14]. Huang J, Wang H, Yang H. Int-deep: A deep learning initialized iterative method for nonlinear problems[J]. Journal of Computational Physics, 2020, 419: 109675.

[15]. Greenfeld D, Galun M, Basri R, et al. Learning to optimize multigrid PDE solvers[C]//International Conference on Machine Learning. PMLR, 2019: 2415-2423.

[16]. Zhu Y, Zabaras N, Koutsourelakis P S, et al. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data[J]. Journal of Computational Physics, 2019, 394: 56-81.
[17]. Yang L, Meng X, Karniadakis G E. B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data[J]. Journal of Computational Physics, 2021, 425: 109913.
[18]. Pang G, Yang L, Karniadakis G E. Neural-net-induced Gaussian process regression for function approximation and PDE solution[J]. Journal of Computational Physics, 2019, 384: 270-288.
[19]. Haehnel P, Mareček J, Monteil J, et al. Using deep learning to extend the range of air pollution monitoring and forecasting[J]. Journal of Computational Physics, 2020, 408: 109278.
[20]. Lu L, Meng X, Mao Z, et al. DeepXDE: A deep learning library for solving differential equations[J]. SIAM Review, 2021, 63(1): 208-228.
[21]. Meng X, Karniadakis G E. A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems[J]. Journal of Computational Physics, 2020, 401: 109020.
[22]. Li W, Xiang X, Xu Y. Deep domain decomposition method: Elliptic problems[C]//Mathematical and Scientific Machine Learning. PMLR, 2020: 269-286.
[23]. Wang Z, Zhang Z. A mesh-free method for interface problems using the deep learning approach[J]. Journal of Computational Physics, 2020, 400: 108963.