Solving differential equation by Neural Network

October 21, 2018

We formulate ordinary differential equation(ODE) and partial differential equation(PDE) as learning algorithm for neural network. The neural network is trained to satisfy the differential operator and boundary conditions using stochastic gradient descent.

Contents

1 Experiment

This section presents the detail of implementation and the accuracy of empirical result for numerical examples.

1.1 Examples for ODE

Problem 1.1 Given $x_i \in [0,1]$, $1 \leq i \leq N$ where i is the number of discretization points. We would like to find the solution of $\Psi(x)$ for the following equation.

$$\frac{d}{dx}\Psi + (x + \frac{1+3x^2}{1+x+x^3})\Psi = x^3 + 2x + x^2\frac{1+3x^2}{1+x+x^3}$$

with $IC = \Psi(0)$ and $BC = \Psi(N)$. The analytic solution is

$$\Psi_a(x) = \frac{e^{-x^2/2}}{1+x+x^3} + x^2$$

In consideration of (5) below, the form of trial solution is taken to be:

$$\Psi_t(x) = \Psi(0) + xO(x,p)$$

with $IC = \Psi(0)$ and $BC = \Psi(1)$. We look for the solution Ψ_t by minimizing the following error quantity:

$$E_{\text{error}} = \sum_i (\frac{d\Psi_t(x_i)}{dx} - f(x_i, \Psi_t(x_i)))^2$$

where

$$f(x_i, \Psi_t(x_i)) = x^3 + 2x + x^2\frac{1+3x^2}{1+x+x^3} - (x + \frac{1+3x^2}{1+x+x^3}\Psi)$$

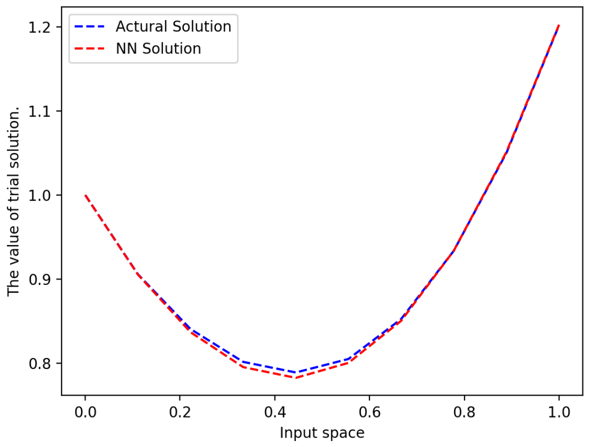

Figure 1: The actual and computed solution in problem 1.1.

In each iteration, we update the weight to be

$$
\begin{aligned}
w_{ij}^{(r+1)} &= w_{ij}^{(r)} - \gamma_r * \frac{\partial E_{\text{error}}}{\partial w_{ij}} \\
v_j^{(r+1)} &= v_j^{(r)} - \gamma_r * \frac{\partial E_{\text{error}}}{\partial v_j}
\end{aligned}
\tag{1}
$$

where γ_r is the learning rate.

Figure 1 displays the actual and computed solution of $\Psi_t(x_i)$ at the grid points.

Problem 1.2 Given $x_i \in [0, 2]$, $1 \leq i \leq N$ where i is the number of discretization points. We would like to find the solution of $\Psi(x)$ for the following equation.

$$\frac{d}{dx}\Psi + \frac{1}{5}\Psi = e^{\frac{x}{5}} cos(x)$$

The analytic solution is:

$$\Psi_a(x) = e^{\frac{1}{5}} sin(x)$$

In consideration of (5) below, the form of trial solution is taken to be:

$$\Psi_t(x) = \Psi(0) + xO(x, p)$$

with $IC = \Psi(0) = 0$ and $BC = \Psi(N)$.
Figure 2 displays the actual and computed solution of $\Psi_t(x_i)$ corresponding to the domain.

Problem 1.3 Given $x_i \in [0, 2]$, $1 \leq i \leq N$ where i is the number of discretization points. We would like to find the solution of $\Psi(x)$ for the following equation.

$$\frac{d^2}{dx^2}\Psi + \frac{1}{5}\frac{d}{dx}\Psi + \Psi = -\frac{1}{5}e^{\frac{x}{5}} cos(x)$$

The analytic solution is:

$$\Psi_a(x) = e^{\frac{1}{5}} sin(x)$$

In consideration of (5) below, the form of trial solution is taken to be:

$$\Psi_t(x) = \Psi(0)(1 - x) + \Psi(1)x + x(1 - x)O(x, p)$$

with $IC = \Psi(0) = 0$ and $BC = \Psi(N)$.
Figure 3 displays the actual and computed solution of $\Psi_t(x_i)$ corresponding to the domain.

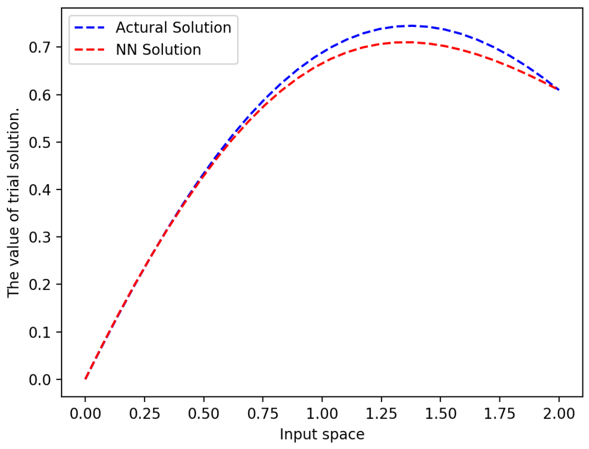

Figure 2: The actual and computed solution in problem 1.2.

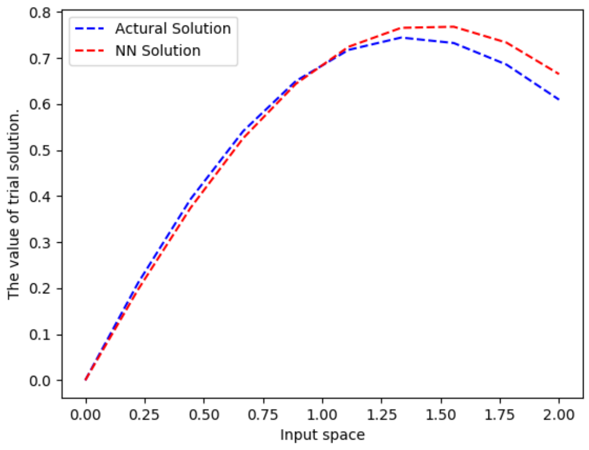

Figure 3: The actual and computed solution in problem 1.3.

2 Schematic of the Algorithm

To make the differential equation solvable by NN, the trial solution and original equation have to be transformed into the specific form. Therefore, this section provides the formulation for differential equation to adjust in order to solve by NN. The approach is based on Lagaris[1] and Chiaramonte[2].

2.1 Formulation

The proposed approach is illustrated in terms of the following general differential equation:

$$G(x, \Psi(x), \bigtriangledown\Psi(x), \bigtriangledown\Psi(x)^2) = 0, x \in D \tag{2}$$

subject to certain boundary conditions (B.Cs), where $x = (x_1, \ldots, x_n) \in \mathbf{R}^n, D \subset \mathbf{R}$ and Ψ is the trial solution employs a neural network and D denotes the definition of domain.

$$\Psi_t(x, p) = \hat{\Psi}(x) + F(x)N(x, p) \tag{3}$$

The parameter p is adjusted based on the weights and bias of neural network. $\hat{\Psi}(x)$ is the initial conditions(I.C.) which is set to be $\hat{\Psi}(0) = \Psi(0) = 0$ and contains no adjustable parameters. The scalar-value function $F(x)$ is chosen so as not to contribute to BC. $N(x, p)$ is the single-output forward neural network(NN). In order to be solved by NN, we transform (2) to the following system of equations:

$$E_{\text{error}}(p) = \min \sum_{i=1}^{m} G(x_i, \Psi(x_i), \bigtriangledown\Psi(x_i), \bigtriangledown\Psi(x_i)^2)^2, \forall x_i \in D, \forall i = 1 \ldots \tag{4}$$

subject to the constraints imposed by the B.Cs. If $\Psi_t(x, p)$ denotes the trial solution, $\Psi_t(x, p)$ will minimize the related error of (4). The general form of the trial solution Ψ_t for the first order ODE can be written as:

$$\Psi_t(x_i) = \Psi(0) + x_i O(x_i, p) \forall x_i \in D, \forall i = 1 \ldots \tag{5}$$

3 Stochastic Gradient Descent

Lagaris[1] had proved that BroydenFletcherGoldfarbShanno (BFGS) algorithm method is quadraticly convergent and has demonstrated excellent performance at computing the gradient of the error. Therefore, quasi-Newton BFGS algorithm was used to minimize the loss function in our model.

3.1 ODE

3.1.1 First Order ODE

We consider the first order ODE:

$$\frac{d}{dx}\Psi(x) = f(x, \Psi)$$

We discretinize our domain $[0, 1]$ into n grids and obtain input vector x_i, $1 \leq i \leq N$ where i denotes the single input unit in discretenized domain. We apply activation function to obtain the output of hidden unit. A popular choice for choosing activation function is the sigmoid function.

$$\sigma(y) = \frac{1}{1 + e^y} \tag{6}$$

The output of hidden units given by activation function is mapped from 0 to 1.

$$H_h = \sum_{i=1}^{n=N} \sigma(w_{ih} * x_i) \tag{7}$$

The output forward propogation of NN is:

$$O = \sum_{i=1}^{N} \sum_{h=1}^{H} v_j \sigma(w_{ih} x_i) + u_i \tag{8}$$

Figure 4 expresses the diagram of NN with one hidden layer looking for the parameters for trial solution in ode and pde. This example only has one hidden layer. After we get the output for whole x_i, $i \in \{0, \ldots, n\}$ via NN, we obtain the sum of error quantity.

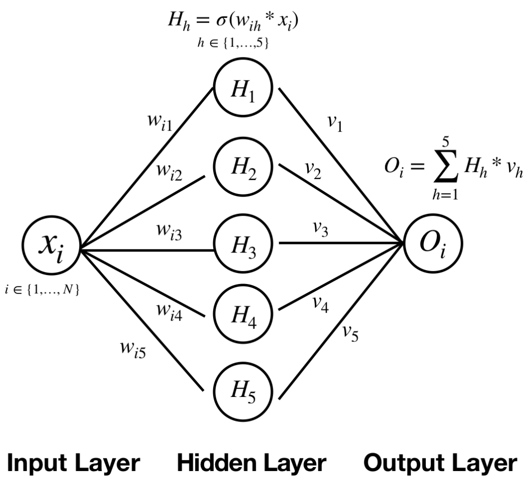

Figure 4: The diagram of Neural Network for computing the parameters of trial solution in ode $O_i(x_i, p)$ with one hidden layer.

$$\begin{aligned} E &= E_i[p] + \cdots + E_n[p] \\ E_{\mathrm{i}}[p] &= \sum_{i=1}^{N} \{\frac{d\Psi_t(x_i)}{dx} - f(x_i, \Psi_t(x_i))\}^2 \end{aligned} \tag{9}$$

where p is the parameters in NN and $f(x_i, \Psi_t(x_i))$ is the value of $\frac{d}{dx}\Psi$. In the case of one dimension ode, we obtain the value after we shift all the items except for $\frac{d}{dx}\Psi$ to the right side.

We optimize the parameters of NN by minimizing the total error quantity in (9) from $E_i, i \in \{1 \dots n\}, h \in \{1 \dots H\}$.

$$\frac{\partial E}{\partial v_h} = \sum_{i=1}^{N} \frac{\partial E_{\mathrm{i}}}{\partial O_i} \frac{\partial O_i}{\partial v_h} \tag{10}$$

$$\frac{\partial E}{\partial w_{ih}} = \sum_{i=1}^{N} \frac{\partial E_{\mathrm{i}}}{\partial O_i} \frac{\partial O_i}{\partial w_{ih}} \tag{11}$$

where h is the number of hidden unit in each hidden layer, $1 \leq h \leq H$

$$\frac{\partial E_i}{\partial O_i} = 2(\{\frac{\partial \Psi_t(x_i)}{\partial x} - f(x_i, \Psi_t(x_i))\}) \tag{12}$$

To calculate the (12). We need to differentiate the trial solution $\Psi_t(x_i)$:

$$\frac{\partial \Psi_t}{\partial x} = \frac{d\Psi(0)}{dx} + \frac{d}{dx} O_i(x_1, p) + x\frac{dO_i(x_1, p)}{dx} \tag{13}$$

These equations reduce to

$$\frac{\partial \Psi_t}{\partial x} = O_i(x, p) + \sum_{i=1}^{N} \sum_{h=1}^{H} v_h w_{ih} \sigma(w_{ih} x_i)_h \tag{14}$$

Using (14) and $f(x_i, \Psi_t(x_i))$, the gradient of O_i is:

$$\frac{\partial O_i}{\partial v_h} = \sum_{h}^{H} \sum_{i=1}^{N} \sigma(w_{ih} x_i) \tag{15}$$

We compute the gradient of E_i with respect to input to hidden weight by using (10)-(15) and get:

$$\frac{\partial O_i}{\partial w_{ij}} = \sum_{h}^{H} \sum_{i=1}^{N} (\sigma(x_i w_{ih}))' x_i \tag{16}$$

Finally, we obtain the gradient of total error quantity with respect to the network weights in (10) and (11). The network weight can be easily updated as:

$$\begin{aligned} w_{ij}^{(r+1)} &= w_{ij}^{(r)} - \gamma_r * \frac{\partial E_{\text{error}}}{\partial w_{ij}} \\ v_j^{(r+1)} &= v_j^{(r)} - \gamma_r * \frac{\partial E_{\text{error}}}{\partial v_j} \end{aligned} \tag{17}$$

3.1.2 Kth Order ODE

The different thing between the multiple order ODE and first order ODE is the total of error sum quantity and the trial solution. We use second order ODE for example to illustrate the way to find the solution of kth order ODE where k is larger than 1.

We consider the second order ODE:

$$\frac{d^2}{dx^2}\Psi(x) = f(x, \Psi, \frac{d}{dx}\Psi)$$

The sum of error quantity to be minimized becomes:

$$E_{\text{i}}[p] = \sum_{i=1}^{N}\{\frac{d^2\Psi_t(x_i)}{dx^2} - f(x_i, \Psi_t(x_i), \frac{d}{dx}\Psi_t(x_i))\}^2 \tag{18}$$

The trial solution is written as:

$$\Psi_t(x) = \Psi(0)(1-x) + \Psi(1)x + x(1-x)O(x,p) \tag{19}$$

To obtain the $\frac{d^2}{dx^2}\Psi(x_i)$ in 18, we differentiate the trial solution $\Psi_t(x_i)$ from (19) to obtain the one and second order of $\Psi_t(x_i)$.

$$\frac{d}{dx}\Psi = -\Psi(0) + \Psi(1) + O(x,p) + x\frac{\partial}{\partial x}O(x,p) - 2xO(x,p) - x^2\frac{\partial}{\partial x}O(x,p)\frac{d}{dx} \tag{20}$$

$$\frac{d^2}{dx^2}\Psi = -2O(x,p) + 2\frac{\partial}{\partial x}O(x,p) - 4x\frac{\partial}{\partial x}O(x,p) + x\frac{\partial^2}{\partial x^2}O(x,p) - x^2\frac{\partial^2}{\partial^2 x}O(x,p) \tag{21}$$

After we computing the derivative of sum of total error quantity with respect to $O(x,p)$, we can follow the formulation for computing gradient of the feedforward NN's $O(x,p)$ with respect to the network weights using (10)-(15) in section 3.1.1.

3.2 PDE

3.2.1 Two-Dimension PDE

Solving the kth order PDE by NN is similar with solving first order PDE. We consider the Poisson equation:

$$\frac{\partial^2\Psi(x,y)}{\partial x^2} + \frac{\partial^2\Psi(x,y)}{\partial y^2} = f(x,y). \tag{22}$$

The error quantity to be minimized is given by:

$$E[p] = \sum_i\{\frac{\partial^2\Psi(x,y)}{\partial x^2} + \frac{\partial^2\Psi(x,y)}{\partial y^2} - f(x,y)\}^2. \tag{23}$$

The form of trial solution is based on the types of Boundary conditions encountered in the solution of partial differential equations. For a Dirichlet BC, the trial solution is:

$$\Psi_t(x,y) = A(x,y) + x(1-x)y(1-y)O(x,y,p). \tag{24}$$

where $A(x,y)$ is chosen so as to satisfy the BC.

$$A(x,y) = (1-x)f_0(y) + xf_1(y) + (1y)g_0(x)[(1x)g_0(0) + xg_0(1)] + yg_1(x)[(1x)g_1(0) + xg_1(1)] \tag{25}$$

where $f_0(y) = \Psi_t(0,y)$, $f_1(y) = \Psi_t(1,y)$, $g_0(x) = \Psi_t(x,0)$ and $g_1(x) = \Psi_t(x,1)$.

Figure 5 illustrate the way to compute the parameters by NN for trial solution in two-dimensional problem. We denote the feedforward function as $O(x,y,p)$.

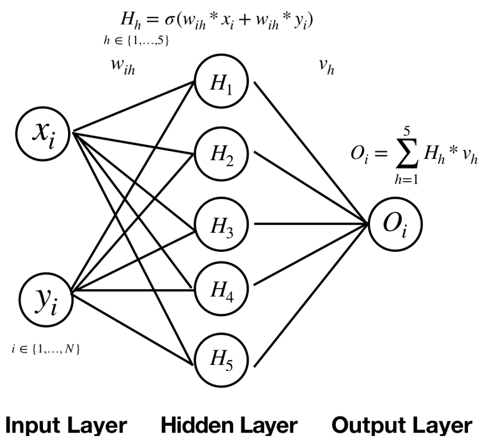

Figure 5: The diagram of Neural Network for computing the parameters of trial solution in pde $O_i(x_i, y_i, p)$ with one hidden layer.

For a mixed boundary condition, the trial solution is written as:

$$\Psi_t(x,y) = B(x,y) + x(1-x)y[O(x,y,p) - O(x,1,p) - \frac{\partial}{\partial y}O(x,1,p)] \tag{26}$$

$B(x,y)$ is again chosen so as to satisfy the BCs.

$$B(x,y) = (1-x)f_0(y) + xf_1(y) + g_0(x)[(1x)g_0(0) + xg_0(1)] + yg_1(x)[(1x)g_1(0) + xg_1(1)] \tag{27}$$

where $f_0(y) = \Psi_t(0,y)$, $f_1(y) = \Psi_t(1,y)$, $g_0(x) = \Psi_t(x,0)$ and $g_1(x) = \frac{\partial}{\partial x}\Psi_t(x,1)$.

4 Convergence Method

Because we did not apply activation function to the output unit of NN, the value of output does not converge. Therefore, in order to make the output stable, we set the condition that if

$$|\Psi_t(x_N) - BC| \leq \varepsilon \tag{28}$$

and the number of iteration is over 500, then the iteration stops. In (28), we set $BC = \Psi_a(x_N)$ and $\varepsilon = e^{-3}$.

5 Relative Work

Weinan[3] combines backward stochastic differential equation(BSDE) with NN to solve PDE. Trial solution is for the stochastic control problem. The stochastic process with continuous sample paths which satisfy that for all $t \in [0,T]$.

$$Y_t = g(\xi + W_T) + \int_t^T f(Y_s, Z_s)ds - \int_t^T \langle Z_s, dW_s \rangle \tag{29}$$

The nonlinear PDE is related to BSDE in a sense that for all $t \in [0,T]$ it holds that

$$Y_t = u(t, \xi + W_t) \in \mathbf{R}, \qquad Z_t = (\bigtriangledown_x u)(t, \xi + W_t) \in \mathbf{R} \tag{30}$$

To approximate the trial solution, they can be computed approximately by employing the policy Z.

References

[1] I. E. Lagaris, A. Likas and D. I. Fotiadis, *Artifial Neural Networks for Solving Ordinary and Partial Differential Equations*, IEEE Transaction on Neural Networks, vol. 9, No. 5, September 1998

[2] M. M. Chiaramonte and M. Kiener, *Solving differential equations using neural networks*

[3] E, Weinan, Han, Jiequn and Jentzen, Arnulf, *Deep Learning-Based Numerical Methods for High-Dimensional Parabolic Partial Differential Equations and Backward Stochastic Differential Equations*, A. Commun. Math. Stat. (2017) 5: 349.