A Brief Report of Deep Ritz Method

Zeyu Jia
School of Mathematical Science
Peking University
1600010603

Dinghuai Zhang
School of Mathematical Science
Peking University

Zhengming Zou
School of Mathematical Science
Peking University

Abstract

1 The Ritz Method

2 Deep Ritz Method with neural networks

Using the Ritz Method mentioned above, it is natural to think of solving pde with deep neural networks. Considering the pde

$$\Delta u(x) = f(x), x \in \Omega \tag{1}$$

According to Ritz Method, what we need to do is

$$\min_{u \in H} I(u), \tag{2}$$

where

$$I(u) = \int_{\Omega} (\frac{1}{2}|\nabla u(x)|^2 - f(x)u(x))dx, \tag{3}$$

and H is the set of admissible function. Our main idea is to facilitate the multi-layer neural network approximation function $u(x)$ and use the gradient descent algorithm to get the final result.

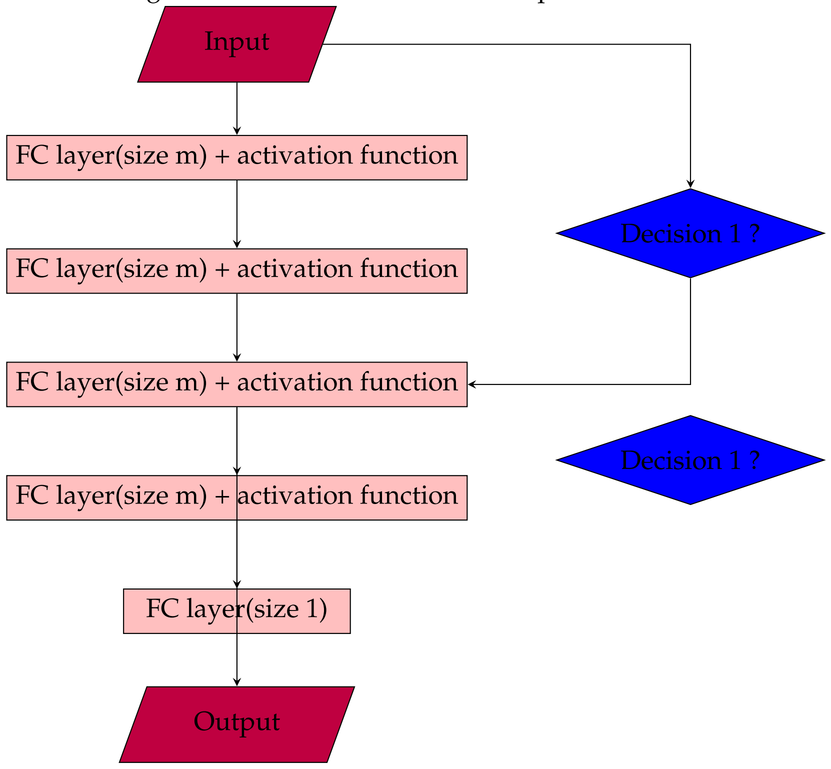
2.1 Building trail function

We mainly use a nonlinear transformation $x \to u_\theta(x)$ defined by deep neural networks to approximate function $u(x)$. Here θ denotes all parameters in our model. Similar to ResNet structure, we use several blocks to construct our networks, each block consists of two linear transformation, two activation functions and a residual connection. The i-th block can be expressed as

$$s_i = \phi(W_{i2}\phi(W_{i1}s_{i_1} + b_{i1}) + b_{i2}) + s_{i-1}. \tag{4}$$

where s_i is the output of the i-th layer. $W_{i1}, W_{i2} \in R^{m \times m}, b_{i1}, b_{i2} \in R^m$ are defined by linear transformation. ϕ is the activation function.

Figure 1: Network Structure of Deep Ritz Method


Input
FC layer(size m) + activation function
FC layer(size m) + activation function
FC layer(size m) + activation function
FC layer(size m) + activation function
FC layer(size 1)
Output
Decision 1 ?
Decision 1 ?

Because our pde involves the Laplace transform, we naturally hope that the second derivative of the function $u(x)$ is not a constant. So we need to pick a proper activation function to ensure the networks' nonlinearity. In our model, we have decided to use

$$\phi(x) = \max\{x^3, 0\} \tag{5}$$

The residual connection in 4 helps to avoid the gradient vanishing problems. After several blocks, we use a linear transform to get the final result. The whole network can be expressed as

$$u_\theta(x) = a \cdot f_n(x) \circ \ldots \circ f_1(x) + b \tag{6}$$

$f_i(x)$ is the i-th block. $a \in R^m, b \in R$ is defined by the final linear transform. Note that the input vector x is not necessarily m-dimensional. In order to handle this mismatch, we can pad x by a zero vector. In our model, we always assume $d < m$.
After building our trail function, we are left to minimize the $I(u)$ in (3)

2.2 Euler numerical integration method

The first problem we need to solve is to calculate the integral in (3). For simplicity, define:

$$g(x, \theta) = |\nabla u(x)|^2 - f(x)u(x) \tag{7}$$

then the $I(u)$ can be expressed as

$$I(u) = \int_\Omega g(x, \theta)dx \tag{8}$$

Obviously, it's impossible to calculate this integral directly. We use Euler numerical integration method to approximate the integral.

$$I(u) = L(x, \theta) = \frac{1}{N}\sum_{i=1}^{N} g(x_i, \theta) \tag{9}$$

Where each x_i corresponds to a data point. Each data point is taken from the grid $[-1, 1] \times [-1, 1]$ in steps of 0.001.

2.3 The stochastic gradient descent algorithm

In deep learning, stochastic gradient descent (SGD) is a common method to minimize the loss function. In this problem, we also choose SGD method to minimize $I(u)$, which can be expressed as:

$$\theta^{k+1} = \theta^k - \eta \nabla_\theta \frac{1}{N} \sum_{i=1}^{N} g(x_{i,k}, \theta^k) \tag{10}$$

where k is the number of iterations.$\{x_{i,k}\}$ is the randomly selected data from the grid. In SGD, we only select a small number of data points from the grid at a time. Due to our limited computing capiticy, we should make a compromise between computing the true gradient and the gradient at a single example. So we choose SGD to compute the gradient against more than one training example (also called a "mini-batch") at each step. In order to optimize our training process, we use the Adam version of SGD.

3 Our improvements

Along with the footsteps of professor E, we have achieved very good results. Based on the results already available, we want to make further improvements.

3.1 L2 regularization

3.2 Self adaptive

4 Numerical Results

4.1 The Poisson Equation

Considering the Poisson equation:

$$\begin{cases} \Delta u = 1, x \in \Omega \\ \quad u = 0, x \in \partial\Omega \end{cases} \tag{11}$$

Here $\Omega = \{(x, y) | x^2 + y^2 < 1\}$.

The exact solution to this problem is

$$u = \frac{1}{4}(x^2 + y^2 - 1) \tag{12}$$

As described above, we use three blocks(six fully connected layers) and a final linear transform with $m = 10$ to build our networks. There is a total of 671 parameters in our model. Considering the boundary condition, we need to make some modifications to our model. We have decided to use a penalty method and the modified function is:

$$I(u) = \int_\Omega (\frac{1}{2}|\nabla u(x)|^2 - u(x))dx + \gamma \int_\Omega |\Delta u(x)|^2 dx + \beta \int_{\partial\Omega} u(x)^2 dx \tag{13}$$