An Analysis of the Frequency Principle for Deep Ritz Method*

Han Liu

NYU SHANGHAI UNDERGRADUATE THESIS
CLASS OF 2021

Abstract

There have been many active works indicating a universal Frequency Principle (F-Principle) in the training process of the overparameterized neural networks. Given mild conditions, it has been repeatedly observed that neural networks fit the training data from low frequencies to high frequencies. It is the opposite of the spectral property when we directly apply some of the classical stationary iterative methods, such as the Jacobi method, to solve partial differential equations (PDEs). In this thesis, we will first have review the basic iterative methods, their smoothing property, and the multigrid method proposed to mitigate the slow convergence directly caused by the smoothing property. We then introduce the neural tangent kernel (NTK) and then give a mathematical explanation for the F-Principle. We will compare the dynamics of the solution convergence based on the classical iterative methods and the one derived from the overparameterized neural network. Finally, we will apply our analysis of the

*The author would like to express very great appreciation to Dr. Yunan Yang for her valuable and constructive suggestions during the planning and development of this thesis. Her willingness to give her time so generously has been very much appreciated.

F-Principle to the Deep Ritz method, which was proposed to solve boundary value problems and explain heuristically why the Deep Ritz method could be more efficient than the traditional Ritz method as a variational approach to solve for solutions to elliptic PDEs.

Key Words: *machine learning, neural tangent kernel, spectral analysis, Deep Ritz method, boundary value problem*

MSC Classification Numbers: *68T07, 65F08, 65K10*

Advisor Signature: Yuanzhen Yang

May. 7, 2021

1 Introduction

Understanding the training process of Deep Neural Networks (DNNs) is a fundamental problem in deep learning. Recently, there have been active research efforts that point out that there is a common implicit bias in the gradient-based training process of DNNs. That is, DNNs often fit target functions from low to high frequencies during the training process [2, 11]. The so-called F-principle is an interesting phenomenon since it is the opposite of the traditional iterative methods. In this thesis, We will first discuss the multigrid method, an algorithm for solving differential equations using a hierarchy of discretizations. The traditional iterative methods applied to solving PDEs often have the property that high-frequency components converge first, which is referred to as the smoothing property [9]. We can view the multigrid method as an approach to mitigating the smoothing property to achieve faster convergence. Afterward, we will move to neural networks and discuss the observed frequency principle (F-principle). We will also rederive the formulation for the neural tangent kernel (NTK) and show how this implies the frequency bias of the neural network. The frequency bias here is from lower frequencies to higher frequencies, which is the opposite of the behavior from classical iterative methods applied to solve PDEs. Finally, we will perform a similar analysis for the Deep Ritz method and explain why this could be a more efficient algorithm for solving variational problems than the traditional Ritz method [3, 6, 1, 5].

2 Multigrid Method

The multigrid method was originally proposed to solve boundary value problems. In this section, we are going to present the basic idea of the multigrid method. We will first describe the general setup of the problem. Then, we will use a basic iterative method, the Jacobi method, to solve a type of boundary value problem. Furthermore, in the following part, we will apply mathematical analysis to the weighted Jacobi

method to solve the 1D Poisson equation and discuss the frequency bias of the iterative method. Finally, we show how the multigrid method utilizes this particular property and improves the speed of convergence. The material in this section is based on [3].

2.1 Problem Description

We want to solve a simple boundary value problem that arises in many physical applications. We set up the general second-order boundary value problem as follows.

$$-u''(x) + \sigma u(x) = f(x),\ 0 < x < 1,\ \sigma \geq 0,\ u(1) = u(1) = 0. \tag{1}$$

We can partition the domain into n subintervals by introducing the grid points $x_j = jh$, where $h = 1/n$ is the constant width of the n subintervals. Based on such discretization, denoting $u(x_j) = v_j$ and $f(x_j) = f_j$, Equation (1) becomes

$$\frac{-v_{j-1} + 2v_j - v_{j+1}}{h^2} + \sigma v_j = f_j, \quad 1 \leq j \leq n-1, \quad v_0 = v_n = 0. \tag{2}$$

Alternatively, we can use the matrix form to represent the equations:

$$\begin{bmatrix} 2+\sigma h^2 & -1 & & & & & \\ -1 & 2+\sigma h^2 & -1 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -1 & 2+\sigma h^2 & -1 & \\ & & & & -1 & 2+\sigma h^2 & \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ \vdots \\ v_{n-2} \\ v_{n-1} \end{bmatrix} = h^2 \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_{n-2} \\ f_{n-1} \end{bmatrix}. \tag{3}$$

2.2 Basic Iterative Methods

Let $\mathbf{u}$ represents the exact solution, $\mathbf{v}$ represents our approximation to the solution and $\mathbf{e}$ represents the residual. Then we have the relation

$$\mathbf{u} = \mathbf{v} + \mathbf{e}. \tag{4}$$

We denote Equation (3) as $A\mathbf{v} = \mathbf{f}$. Then we split the matrix A as

$$A = D - L - U, \tag{5}$$

where D is the diagonal of A, and $-L$ and $-U$ are the strictly lower and upper triangular parts of A, respectively. Then $A\mathbf{u} = \mathbf{f}$ becomes

$$(D - L - U)\mathbf{u} = \mathbf{f}. \tag{6}$$

Isolating the diagonal terms of A, we have

$$D\mathbf{u} = (L + U)\mathbf{u} + \mathbf{f}, \tag{7}$$

which leads to

$$\mathbf{u} = D^{-1}(L + U)\mathbf{u} + D^{-1}\mathbf{f}. \tag{8}$$

Multiplying by D^{-1} corresponds exactly to solving the j-th equation for u_j, for $1 \leq j \leq n - 1$. This gives us the true solution $\mathbf{u}$. If we define the Jacobi iteration matrix by

$$R_J = D^{-1}(L + U), \tag{9}$$

then the Jacobi method appears in matrix form as

$$\mathbf{v}^{(1)} = R_J\mathbf{v}^{(0)} + D^{-1}\mathbf{f}, \tag{10}$$

where $\mathbf{v}^{(0)}$ represents the the initial guess on the first iteration. We can apply a simple modification as follows to the Jacobi method, which gives us the so-called weighted Jacobi method,

$$\mathbf{v}^{(1)} = [(1-\omega)I + \omega R_J]\,\mathbf{v}^{(0)} + \omega D^{-1}\mathbf{f}. \tag{11}$$

If we define the weighted Jacobi iteration matrix by

$$R_\omega = (1-\omega)I + \omega R_J, \tag{12}$$

then the method will be expressed as

$$\mathbf{v}^{(1)} = R_\omega \mathbf{v}^{(0)} + \omega D^{-1}\mathbf{f}. \tag{13}$$

Both of these two methods can be written in the form as

$$\mathbf{v}^{(1)} = R\mathbf{v}^{(0)} + \mathbf{g}, \tag{14}$$

where R is one of the iteration matrices derived earlier, and $\mathbf{g} = D^{-1}\mathbf{f}$ or $\omega D^{-1}\mathbf{f}$. Furthermore, these methods are designed such that the exact solution, $\mathbf{u}$, is the fixed-point solution of the fixed-point iteration. This means that iteration does not change the exact solution:

$$\mathbf{u} = R\mathbf{u} + \mathbf{g}. \tag{15}$$

2.3 Error Analysis

Next, we investigate the error incurred by such basic iterative methods. By subtracting the last two equations, we get

$$\mathbf{e}^{(1)} = \mathrm{Re}^{(0)}\,. \tag{16}$$

Repeating this argument, it follows that after m relaxation sweeps, the error in the m-th approximation is given by

$$\mathbf{e}^{(m)} = R^m \mathbf{e}^{(0)}. \tag{17}$$

If we now choose a particular vector norm and its associated matrix norm, it is possible to bound the error after m iterations by

$$\left\|\mathbf{e}^{(m)}\right\| \leq \|R\|^m \left\|\mathbf{e}^{(0)}\right\|. \tag{18}$$

This leads us to conclude that if $\|R\| < 1$, then the error is forced to zero as the iteration proceeds. It is shown in many standard texts such as [9] that $\lim_{m\to\infty} R^m = 0$ if and only if $\rho(R) < 1$. Therefore, it follows that the iteration associated with the matrix R converges independent of the initial guess as long as $\rho(R) < 1$.

The spectral radius $\rho(R)$ is also called the asymptotic convergence factor when it appears in the context of iterative methods [9]. It has some useful interpretations. First, it is roughly the worst factor by which the error is reduced with each relaxation sweep. The following argument also shows how many iterations are required to reduce the error by a factor of 10^{-d}. Let m be the smallest integer that satisfies

$$\frac{\left\|\mathbf{e}^{(m)}\right\|}{\|\mathbf{e}^{(0)}\|} \leq 10^{-d}. \tag{19}$$

This condition will be approximately satisfied if

$$(\rho(R))^m \leq 10^{-d}. \tag{20}$$

Solving for m, we have

$$m \geq -\frac{d}{\log_{10}(\rho(R))}. \tag{21}$$

The quantity $-\log_{10}(\rho(R))$ is called the asymptotic convergence rate. Its reciprocal

gives the approximate number of iterations required to reduce the error by one decimal digit. We see that as $\rho(R)$ approaches 1, the convergence rate decreases. Small values of $\rho(R)$ (that is, $\rho(R)$ positive and near-zero) give a high convergence rate.

We have established the importance of the spectral radius of the iteration matrix in analyzing the convergence properties of relaxation methods. Now it is time to compute some spectral radii. Consider the weighted Jacobi iteration applied to the one-dimensional model problem described by Equation (1). Without loss of generality, we set $\sigma = 0$. Recall that $R_\omega = (1-\omega)I + \omega R_J$, and thus

$$R_\omega = I - \frac{\omega}{2}\begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{bmatrix}. \tag{22}$$

Written in this form, it follows that the eigenvalues of R_ω and A are related by

$$\lambda\left(R_\omega\right) = 1 - \frac{\omega}{2}\lambda(A). \tag{23}$$

The eigenvalues of the original matrix A are

$$\lambda_k(A) = 4\sin^2\left(\frac{k\pi}{2n}\right), \quad 1 \le k \le n-1. \tag{24}$$

Also of interest are the corresponding eigenvectors of A. In all that follows within this section, we let $w_{k,j}$ be the j-th component of the k-th eigenvector, $\mathbf{w}_k$. The eigenvectors of A are then given by

$$w_{k,j} = \sin\left(\frac{jk\pi}{n}\right), \quad 1 \le k \le n-1, \quad 0 \le j \le n. \tag{25}$$

We see that the eigenvectors of A are simply the Fourier modes. With these results, we find that the eigenvalues of R_ω are

$$\lambda_k\left(R_\omega\right) = 1 - 2\omega \sin^2\left(\frac{k\pi}{2n}\right), \quad 1 \leq k \leq n-1, \tag{26}$$

while the eigenvectors of R_ω are the same as the eigenvectors of A.

The eigenvectors of the matrix A are important in much of the following discussion. They correspond very closely to the eigenfunctions of the continuous model problem. Just as we can expand fairly arbitrary functions using this set of eigenfunctions, it is also possible to expand arbitrary vectors in terms of a set of eigenvectors. Let $\mathbf{e}^{(0)}$ be the error in an initial guess used in the weighted Jacobi method. Then it is possible to represent $\mathbf{e}^{(0)}$ using the eigenvectors of A in the form

$$\mathbf{e}^{(0)} = \sum_{k=1}^{n-1} c_k \mathbf{w}_k, \tag{27}$$

where the coefficients $c_k \in \mathbb{R}$ give the amount of each mode in the error. We have seen that after m sweeps of the iteration, the error is given by

$$\mathbf{e}^{(m)} = R_\omega^m \mathbf{e}^{(0)}. \tag{28}$$

Using the eigenvector expansion for $\mathbf{e}^{(0)}$, we have

$$\mathbf{e}^{(m)} = R_\omega^m \mathbf{e}^{(0)} = \sum_{k=1}^{n-1} c_k R_\omega^m \mathbf{w}_k = \sum_{k=1}^{n-1} c_k \lambda_k^m\left(R_\omega\right) \mathbf{w}_k. \tag{29}$$

The last equality follows because the eigenvectors of A and R_ω are the same; therefore, $R_\omega \mathbf{w}_k = \lambda_k\left(R_\omega\right) \mathbf{w}_k$ This expansion for $\mathbf{e}^{(m)}$ shows that after m iterations, the k-th mode of the initial error has been reduced by a factor of $\lambda_k^m\left(R_\omega\right)$. It should also be noted that the weighted Jacobi method does not mix modes: when applied to a single mode, the iteration can change the amplitude of that mode, but it cannot

convert that mode into different modes. In other words, the Fourier modes are also eigenvectors of the iteration matrix.

From Equation (26), we see that the eigenvalues corresponding to the high-frequency modes are much smaller than the eigenvalues corresponding to the low-frequency modes. So, combine this property with Equation (29), we can conclude that the error corresponding to the high-frequency modes decays much faster than the error corresponding to the low-frequency modes. This result is demonstrated in Figure 1 below.

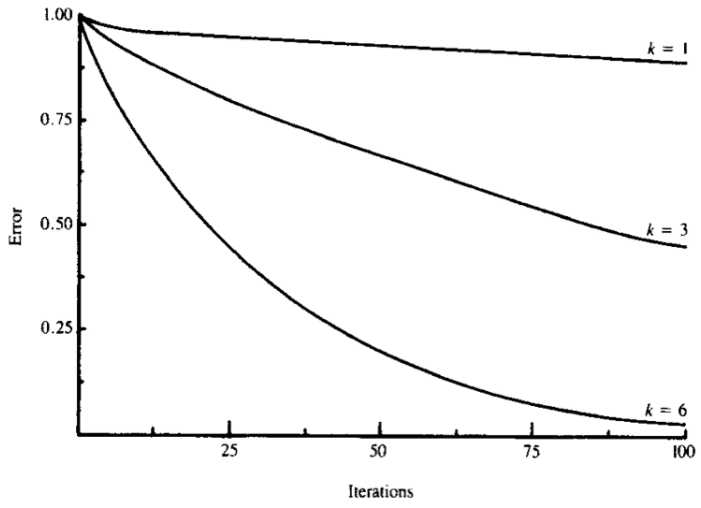


Figure 1: Weighted Jacobi iteration error decay with different mode k. We can see that the error corresponding to the high frequencies decays faster, which means the high-frequency components converges faster. Image source: *A Multigrid Tutorial* [3, page 14].

2.4 The multigrid method

So far, we have reviewed some standard iterative methods that possess the smoothing property. In other words, the smooth components, corresponding to the eigenvectors with small eigenvalues, decays slower, while the oscillatory components, corresponding to the eigenvectors with large eigenvalues, decays faster. This property makes these methods very effective at eliminating the high-frequency or oscillatory components of

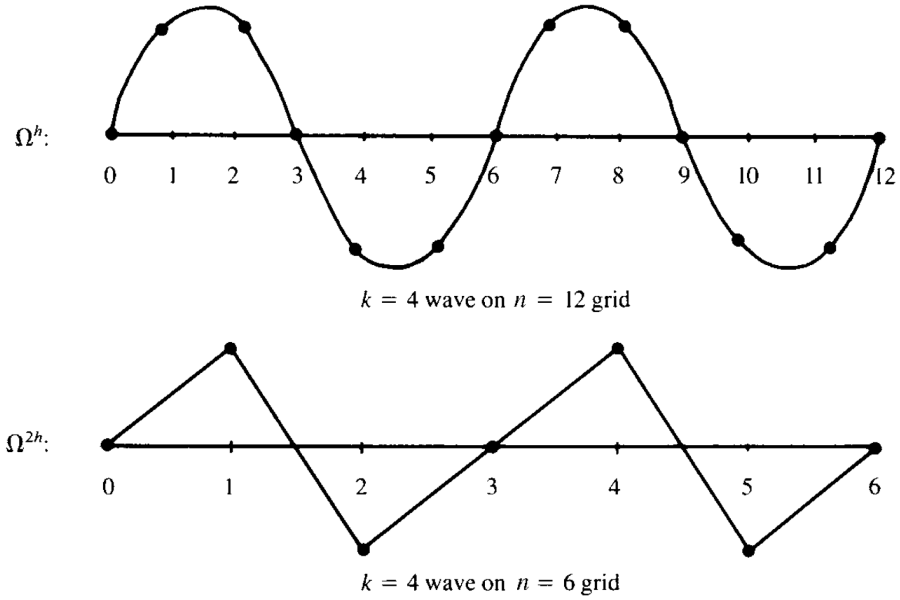


Figure 2: When we move from $n = 12$ grid to $n = 6$ gird, the $k = 4$ mode becomes more oscillatory. Image source: *A Multigrid Tutorial* [3, page 32].

the error while leaving the low-frequency or smooth components relatively unchanged. So, one way to speed up the convergence is to make the low-frequency component more oscillatory. This is one of the main goals achieved by the multigrid method.

Figure 2 shows very clearly how the smooth components look like on a coarser grid. A smooth wave (the sin function) at mode $k = 4$ on a grid Ω^h with $n = 12$ points has been projected directly to the grid Ω^{2h} with $n = 6$ points. On this coarse grid, the original wave still has a wavenumber of $k = 4$. We see a smooth wave on Ω^h looks more oscillatory on Ω^{2h}.

To be more precise, note that the grid points of the coarse grid Ω^{2h} are the even-numbered grid points of the fine grid Ω^h. Consider the k-th mode on the fine grid evaluated at the even-numbered grid points. If $1 \leq k < \frac{n}{2}$, its components may be written as

$$w^h_{k,2j} = \sin\left(\frac{2jk\pi}{n}\right) = \sin\left(\frac{jk\pi}{n/2}\right) = w^{2h}_{k,j}, \quad 1 \leq k < \frac{n}{2}. \tag{30}$$

We see that the k-th mode on Ω^h becomes the k-th mode on Ω^{2h}. So, passing from the fine grid to the coarse grid, a smooth mode becomes *relatively* more oscillatory. To take advantage of such property, we could relax on a fine grid to eliminate the error corresponding to the high-frequency modes first. Then, we move to a coarser and to reduce the low-frequency error. At last, we use interpolation to go back to the fine grid. These are the main components of the multigrid method.

3 Frequency Principle

The frequency principle (F-principle) describes a common implicit bias in the overparameterized gradient-based optimization algorithm for Deep Neural Networks (DNNs) [11]. That is, DNNs often fit target functions from low to high frequencies during the training process. In this section, we will first give a brief introduction to the gradient-based optimization algorithm to train neural networks. Afterward, we will give a concrete example to demonstrate the frequency principle.

3.1 Gradient-Based Algorithm Training Neural Networks

A neural network consists of neurons organized in layers. Each neuron has inputs and produces a single output which can be sent to multiple other neurons. The activation function computes the input to a neuron from the outputs of its predecessor neurons and their connections as a weighted sum. Figure 3 shows the structure of a typical neural network with two hidden layers [10].

One way of updating the parameters in the neural network is backpropagation [7]. Backpropagation computes the gradient (derivatives) in the weight space of a feed-forward neural network with respect to a loss function. The (stochastic) gradient descent algorithm is widely used in the training of neural networks.

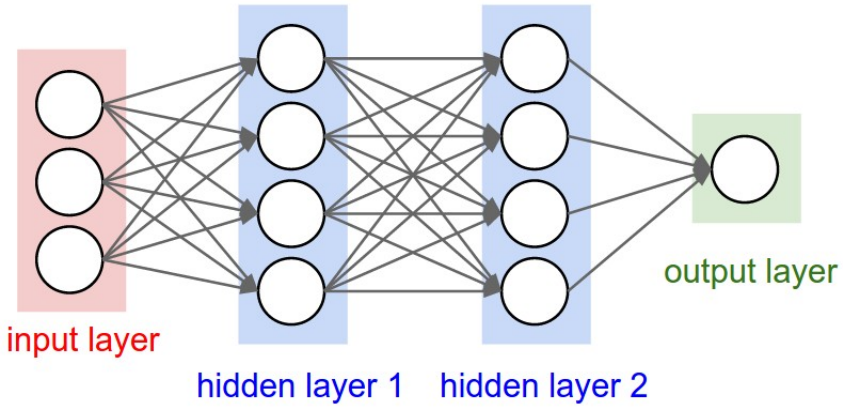


Figure 3: A neural network with two hidden layers. Image source: `https://github.com/karanvivekbhargava/vanilla-neural-network`.

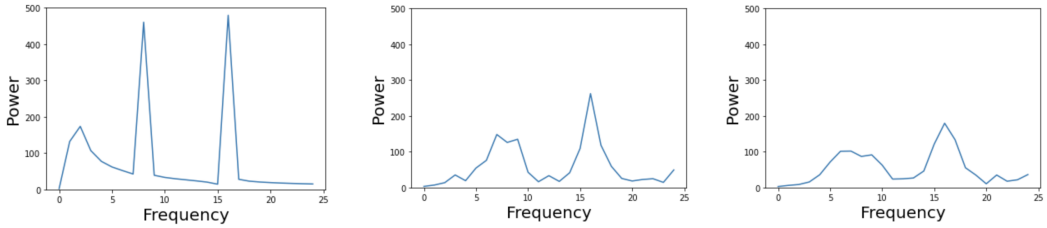
3.2 An Experiment in 1D case

In the experiment, we built up a simple neural network with one hidden layer and 10^4 neurons. We tried to let this neural network to learn the function

$$f(x) = \sin(x) + \sin(8x) + \sin(16x). \tag{31}$$

If we apply Fourier transform to this function, there will be three peaks (sum of three Dirac delta functions) corresponding to the components. The size of our training set is 201. That is, we have 201 pairs of training data (x_i, y_i) where $\{x_i\}$ are discretizations on an interval $[-2\pi, 2\pi]$. Therefore, the neural network is heavily overparameterized. We use the limited-memory Broyden–Fletcher–Goldfarb–Shanno (LBFGS) algorithm to minimize the L_2 loss [8]. We define the residual as $y_i - u(x_i)$ where u is the function represented by the neural network. We plot the power of the residuals in the frequency domain after the different number of iterations; see Figure 4.

We can see three peaks in all the plots of Figure 4. They correspond to the three frequency mode of the function $f(x)$: $k = 1, 8, 16$. From the three plots, it shows that the mode corresponding to the low frequency decays first. In other words, the neural network learns the low-frequency component first, which coincides with the



(a) Initial residual (b) After 1000 iterations (c) After 5000 iterations

Figure 4: The x-axis is the frequency (Hz) and the y-axis is the power of the residuals at a particular frequency.

F-principle.

4 Mathematical Analysis

In this section, we will apply mathematical analysis to the overparameterized neural network and review the spectral analysis of the Neural Tangent Kernel. We attempt to give a mathematical explanation for the frequency principle discussed in the previous section.

4.1 Problem Setup

We consider a network with two layers, defined as below.

$$f(\mathbf{x}) = \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r \sigma\left(\mathbf{w}_r^T \mathbf{x}\right), \tag{32}$$

where $\mathbf{x} \in \mathbb{R}^{d+1}$ is the input and $W = [\mathbf{w}_1, \ldots, \mathbf{w}_m] \in \mathbb{R}^{(d+1)\times m}$ and $\mathbf{a} = [a_1, \ldots, a_m]^T \in \mathbb{R}^m$ respectively are the weights of the first and second layers. Here, σ denotes the Rectified Linear Unit (ReLU) function, $\sigma(x) = \max(x, 0)$. We remark that this model does not explicitly include bias. Let the training data consists of n pairs $\{\mathbf{x}_i, y_i\}_{i=1}^n, \mathbf{x}_i \in \mathbb{R}^{d+1}$ and $y_i \in \mathbb{R}$. And we further set $a_r \sim$ Uniform {-1,1} for $r = 1, \cdots, m$, and maintain it fixed throughout the training. We want to use the

neural network to approximate the relationship between $\mathbf{x}$ and y by optimizing over the set of weights to achieve the least-squares error. We apply the gradient descent (GD) method to minimize the L_2 loss defined as

$$\Phi(W) = \frac{1}{2}\sum_{i=1}^{n}(y_i - f(x))^2. \tag{33}$$

We initialize the network with $\mathbf{w}_r^{(0)} \sim \mathcal{N}(0, \kappa^2 I)$, where κ is a fixed constant.

4.2 Neural Tangent Kernel

The idea of Neural Tangent Kernel is that a properly randomly initialized overparameterized DNN, trained by gradient descent with infinitesimal step size (a.k.a. gradient flow), is equivalent to a kernel regression predictor with a deterministic kernel. This particular kernel is referred to as the neural tangent kernel (NTK) [4].

With a fixed stepsize η, the iterative GD update can be written as:

$$\begin{aligned}
& \mathbf{w}_r^{(k+1)} - \mathbf{w}_r^{(k)} && (34)\\
= \; & -\eta\left(\frac{\partial\Phi(\mathbf{W}^{(k)})}{\partial\mathbf{w}_r}\right)^{\top} && (35)\\
= \; & -\eta\frac{a_r}{\sqrt{m}}\left(\sum_{i=1}^{n}\left(f\left(\mathbf{x}_i\right) - y_i\right) I\left\{\left(\mathbf{w}_r^{(k)}\right)^{\top}\mathbf{x}_i \geq 0\right\}\mathbf{x}_i\right)^{\top} && (36)\\
= \; & -\eta\frac{a_r}{\sqrt{m}}\left(\sum_{i=1}^{n}\left(f(\mathbf{x}_i) - y_i\right) I_{ri}\mathbf{x}_i\right)^{\top} && (37)\\
= \; & -\eta(Z_r R)^{\top} = -\eta R^{\top} Z_r^{\top}, && (38)
\end{aligned}$$

where the residual vector

$$R = [f(\mathbf{x}_1) - y_1,\ f(\mathbf{x}_2) - y_2,\ \ldots,\ f(\mathbf{x}_n) - y_n]^{\top}, \tag{39}$$

and

$$Z_r = \frac{1}{\sqrt{m}}[a_r\mathbf{I}_{r1}\mathbf{x}_1 \; a_r\mathbf{I}_{12}\mathbf{x}_2 \; \ldots \; a_r\mathbf{I}_{ri}\mathbf{x}_i \; \ldots \; a_r\mathbf{I}_{rn}\mathbf{x}_n], \tag{40}$$

in which the indicator

$$\mathbf{I}_{ri} = \begin{cases} 1, & \mathbf{w}_r^\top \mathbf{x}_i \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{41}$$

We conclude

$$\begin{aligned} & W^{(k+1)} - W^{(k)} && (42)\\ = \; & [\mathbf{w}_1^{(k+1)} - \mathbf{w}_1^{(k)} \; \ldots \; \mathbf{w}_r^{(k+1)} - \mathbf{w}_r^{(k)} \; \ldots \; \mathbf{w}_m^{(k+1)} - \mathbf{w}_m^{(k)}] && (43)\\ = \; & -\eta[R^\top Z_1^\top \; \ldots \; R^\top Z_r^\top \; \ldots \; R^\top Z_m^\top] && (44)\\ = \; & -\eta R^\top Z^\top, && (45) \end{aligned}$$

where Z is the $(d+1)m \times n$ block matrix

$$Z = \frac{1}{\sqrt{m}} \begin{pmatrix} a_1\mathbf{I}_{11}\mathbf{x}_1 & a_1\mathbf{I}_{12}\mathbf{x}_2 & \ldots & a_1\mathbf{I}_{1n}\mathbf{x}_n \\ a_2\mathbf{I}_{21}\mathbf{x}_1 & a_2\mathbf{I}_{22}\mathbf{x}_2 & \ldots & a_2\mathbf{I}_{2n}\mathbf{x}_n \\ \vdots & & & \vdots \\ a_m\mathbf{I}_{m1}\mathbf{x}_1 & a_m\mathbf{I}_{m2}\mathbf{x}_2 & \ldots & a_m\mathbf{I}_{mn}\mathbf{x}_n \end{pmatrix} = \begin{bmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_m \end{bmatrix}, \tag{46}$$

where $Z_1, Z_2, \cdots, Z_m$ are defined in Equation (40). One should notice that Z is also the Jacobian matrix of f with respect to the weight W:

$$Z = \begin{bmatrix} \frac{\partial f_1}{\partial \mathbf{w}_1} & \cdots & \frac{\partial f_n}{\partial \mathbf{w}_1} \\ \vdots & & \vdots \\ \frac{\partial f_1}{\partial \mathbf{w}_r} & \cdots & \frac{\partial f_n}{\partial \mathbf{w}_r} \\ \vdots & & \vdots \\ \frac{\partial f_1}{\partial \mathbf{w}_m} & \cdots & \frac{\partial f_n}{\partial \mathbf{w}_m} \end{bmatrix}. \tag{47}$$

If we divide η on both sides of Equation (45), and then take $\eta \to 0$, this becomes

a dynamics of $W(t)$. We have the following gradient flow

$$\frac{dW(t)}{dt} = -R^\top Z^\top. \tag{48}$$

On the other hand, for the i-th element of the residual vector R

$$\frac{dR_i}{dt} = \frac{df_i}{dt} = \sum_{r=1}^{m} \frac{\partial f_i}{\partial \mathbf{w}_r} \frac{d\mathbf{w}_r}{dt} \tag{49}$$

$$= \left[\frac{d\mathbf{w}_1}{dt} \cdots \frac{d\mathbf{w}_r}{dt} \cdots \frac{d\mathbf{w}_m}{dt} \right] \begin{bmatrix} \frac{\partial f_i}{\partial \mathbf{w}_1} \\ \vdots \\ \frac{\partial f_i}{\partial \mathbf{w}_r} \\ \vdots \\ \frac{\partial f_i}{\partial \mathbf{w}_m} \end{bmatrix} = \frac{\partial W}{\partial t} \begin{bmatrix} \frac{\partial f_i}{\partial \mathbf{w}_1} \\ \vdots \\ \frac{\partial f_i}{\partial \mathbf{w}_r} \\ \vdots \\ \frac{\partial f_i}{\partial \mathbf{w}_m} \end{bmatrix}. \tag{50}$$

Therefore, we have

$$\left(\frac{\partial R}{\partial t} \right)^\top = \left[\frac{\partial R_1}{\partial t} \cdots \frac{\partial R_n}{\partial t} \right] \tag{51}$$

$$= \left(\frac{\partial W}{\partial t} \begin{bmatrix} \frac{\partial f_1}{\partial \mathbf{w}_1} \\ \vdots \\ \frac{\partial f_1}{\partial \mathbf{w}_m} \end{bmatrix} \cdots \frac{\partial W}{\partial t} \begin{bmatrix} \frac{\partial f_n}{\partial \mathbf{w}_1} \\ \vdots \\ \frac{\partial f_n}{\partial \mathbf{w}_m} \end{bmatrix} \right) \tag{52}$$

$$= \frac{\partial W}{\partial t} \begin{bmatrix} \frac{\partial f_1}{\partial \mathbf{w}_1} & \cdots & \frac{\partial f_n}{\partial \mathbf{w}_1} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ \frac{\partial f_1}{\partial \mathbf{w}_m} & \cdots & \frac{\partial f_n}{\partial \mathbf{w}_m} \end{bmatrix} \tag{53}$$

$$= -R^\top Z^\top Z \tag{54}$$

$$= -R^\top H, \tag{55}$$

where $H = Z^\top Z$. Finally, we obtain the dynamics for the residual vector

$$\frac{\partial R}{\partial t} = -H^\top R = -HR, \tag{56}$$

since H is a symmetric matrix. The elements in the matrix H are given by

$$H_{ij} = \frac{1}{m}\mathbf{x}_i^\top \mathbf{x}_j \sum_{r=1}^{m} \mathbf{I}_{ri}\mathbf{I}_{rj}. \tag{57}$$

We define a matrix $H^\infty \in \mathbb{R}^{n\times n}$ with its entity given by the following equation as the expectation of H over W, where we consider the weight matrix W as a random variable that follows a particular probability distribution such as the normal distribution. That is,

$$H_{ij}^\infty = \mathbb{E}_{\mathbf{w}\sim N(\mathbf{0},\mathbf{I})}\left[\mathbf{x}_i^\top \mathbf{x}_j \mathbb{I}\left\{\mathbf{w}^\top \mathbf{x}_i \geq 0, \mathbf{w}^\top \mathbf{x}_j \geq 0\right\}\right]. \tag{58}$$

The error (or variance) of considering H^∞ instead of H is small when m goes to infinity (the size of the neural network is infinitely large), so this argument is justified only under the overparameterized regime. The matrix H^∞ is also known as the neural tangent kernel (NTK).

4.3 Spectral Analysis

In this section, we will review the spectral analysis of H based on [2].

Given arbitrarily small constants ϵ, δ, consider a network with $m = \Omega\left(\frac{n^7}{\lambda_0^4 \kappa^2 e^{2\delta}}\right)$ units, $\kappa = O\left(\frac{e\delta}{\sqrt{n}}\right)$ and learning rate $\eta = O\left(\frac{\lambda_0}{n^2}\right)$ (λ_0 denotes the minimal eigenvalue of H^∞). Here, $f(n) = \Omega(g(n))$ means $\liminf_{n\to\infty} \frac{f(n)}{g(n)} > 0$ and $f(n) = O(g(n))$ means $\limsup_{n\to\infty} \frac{|f(n)|}{g(n)} < \infty$. Then with probability $1-\delta$ over the random initializations

$$\|\mathbf{y} - \mathbf{u}(t)\|_2 = \left(\sum_{k=1}^{n} (1-\eta\lambda_k)^{2t} \left(\mathbf{v}_k^T \mathbf{y}\right)^2\right)^{1/2} \pm \epsilon, \tag{59}$$

where $\mathbf{u}$ is the function represented by neural network, and $\mathbf{v}_1, \ldots, \mathbf{v}_n$ and $\lambda_1, \ldots, \lambda_n$ respectively are the eigenvectors and eigenvalues of H^∞ [2, section 4.1].

Let us consider the 1D case with $d = 0$, with and without the bias term in the two-layer neural network. The eigenvalues for the NTK in the case without bias are each given by

$$a_k = \begin{cases} \frac{1}{\pi^2} & k = 0 \\ \frac{1}{4} & k = 1 \\ \frac{2\left(k^2+1\right)}{\pi^2\left(k^2-1\right)^2} & k \geq 2 \text{ even} \\ 0 & k \geq 2 \text{ odd} \end{cases} \tag{60}$$

and the eigenvalues for the NTK where the neural network has bias are given by

$$c_k = \begin{cases} \frac{1}{2\pi^2} + \frac{1}{8} & k = 0 \\ \frac{1}{\pi^2} + \frac{1}{8} & k = 1 \\ \frac{k^2+1}{\pi^2(k^2-1)^2} & k \geq 2, \text{ even} \\ \frac{1}{\pi^2 k^2} & k \geq 2, \text{ odd} \end{cases} \tag{61}$$

each corresponding to the common k-th eigenvector $\cos(k\theta)$.

Recall that in section 2, we discussed that how the error vector can be expressed as a linear combination of eigenvectors. Similarly, we can decompose the error here. It suffices to understand how fast $\sum_{k=1}^{n}(1- \eta\lambda_k)^{2\tau}\left(\mathbf{v}_k^\top \mathbf{y}\right)^2$ converges to 0 as the number of iteration τ grows. Define $\xi_k(\tau) = (1-\eta\lambda_k)^{2\tau}\left(\mathbf{v}_k^\top \mathbf{y}\right)^2$, and notice that each sequence $\{\xi_k(\tau)\}_{\tau=0}^{\infty}$ is a geometric sequence which starts at $\xi_k(0) = \left(\mathbf{v}_k^\top \mathbf{y}\right)^2$ and decreases at ratio $(1-\eta\lambda_k)^2$, for each eigenmode $k = 1, \ldots, n$.

In other words, we can think of decomposing the label vector $\mathbf{y}$ into its projections onto all eigenvectors $\mathbf{v}_k$ of $\mathbf{H}^\infty$: $\|\mathbf{y}\|_2^2 = \sum_{k=1}^{n}\left(\mathbf{v}_k^\top \mathbf{y}\right)^2 = \sum_{k=1}^{n}\xi_k(0)$, and the k-th coefficient shrinks exponentially at ratio $(1-\eta\lambda_k)^2$. From Equation (60) and (61), we can see that the smaller the k, the larger the eigenvalues λ_k. The larger is the λ_k, the faster $\{\xi_k(\tau)\}_{\tau=0}^{\infty}$ decreases to 0. This analysis is in the same vein with the observation in Figure (4).

5 Deep Ritz Method

Our motivation for investigating the Deep Ritz method [6] is to apply our understanding of the frequency principle to a particular DNN-based method and see if it helps understand the effectiveness of the proposed method. The Deep Ritz Method is a deep learning-based method used for solving variational problems, particularly ones that are formulated to find a numerical solution to PDEs. It works very efficiently when solving the elliptic PDEs [6]. In this section, we try to offer an aspect that was not provided in the original paper. That is why the Deep Ritz method works well for the elliptic equations based on the neural network structure.

5.1 Problem Setup

We consider the same problem described in Equation (1).

$$-u''(x) + \sigma u(x) = 0, \quad 0 < x < 1, \quad \sigma \geq 0, \quad u(0) = u(1) = 0. \tag{62}$$

Here, for simplicity, we set $\sigma = 0$. The Deep Ritz method uses the following objective function instead of the simple L_2 loss. We will still assume the gradient descent algorithm is applied to minimize the objective function

$$I(f) = \int_{\Omega} \frac{1}{2}|f'(x)|^2 dx + \beta \int_{\partial\Omega} f(x)^2 ds, \tag{63}$$

where β is a penalty constant to enforce the boundary condition and f is our neural network approximation of the PDE solution u above.

We define the input data vector as $x = [x_0, \ldots, x_n]^\top$, where $x_i = i/n$, with $i = 0, 1, \ldots, n$. The corresponding output training data is $f = [f_0, f_1, \ldots, f_n]^\top$ where $f_i = f(x_i) = u(x_i)$ for $i = 0, 1, \ldots, n$. Thus, we have $n + 1$ training data pairs $\{(x_i, f_i)\}_{i=0}^n$.

5.2 Mathematical Analysis

Again, we build up a simple two-layer neural network.

$$f(x) = \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r \sigma\left(w_r x\right). \tag{64}$$

Let m be the number of neurons, and $W = [w_1, w_2, \cdots, w_m]^\top$ is the weight vector of the neurons. Let $a_r \sim$ Uniform {-1,1} for $r = 1, \cdots, m$, and maintain it fixed throughout the training. We can approximate $f'(x)$ by the following first-order finite difference

$$f'(x) = n \cdot [0, f(x_2) - f(x_1), \cdots, f(x_n) - f(x_{n-1}), 0]^\top. \tag{65}$$

Then the discretized loss function $I(f)$ becomes

$$\begin{aligned} I(f) &= \frac{1}{2n} f'(x)^\top f'(x) + \frac{\beta}{2} \cdot \left(f\left(x_0\right)^2 + f\left(x_n\right)^2\right) && (66) \\ &= \frac{n}{2}\left(\left(f\left(x_2\right) - f\left(x_1\right)\right)^2 + \cdots + \left(f\left(x_n\right) - f\left(x_{n-1}\right)\right)^2\right) && (67) \\ &\quad + \frac{\beta}{2}\left(f(x_0)^2 + f\left(x_n\right)^2\right). && (68) \end{aligned}$$

We define g as

$$g = \begin{bmatrix} \sqrt{\frac{\beta}{2}} & 0 & & & \\ & -\sqrt{\frac{n}{2}} & \sqrt{\frac{n}{2}} & & \\ & & \ddots & \ddots & \\ & & & -\sqrt{\frac{n}{2}} & \sqrt{\frac{n}{2}} \\ & & & & \sqrt{\frac{\beta}{2}} \end{bmatrix} \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_{n-1}) \\ f(x_n) \end{bmatrix} = Bf. \tag{69}$$

Then, we can rewrite the objective function $I(f)$ as

$$I(f) = g^\top g = f^\top B^\top B f. \tag{70}$$

Let $A = B^\top B \in \mathbb{R}^{(n+1)\times(n+1)}$. Obviously, A is a symmetric matrix, i.e., $A^\top = A$. Recall that $W = [w_1, w_2, \cdots, w_m]^\top$ are the weights of the neurons. We want to find the dynamics describing how the weights change with the iterations during the training process. Since we use a fixed stepsize in the gradient descent algorithm to update W, we have

$$\frac{dW}{dt} = -\left(\frac{\partial I}{\partial W}\right)^\top. \tag{71}$$

On the other hand,

$$\left(\frac{\partial I}{\partial W}\right)^\top = \begin{bmatrix} \frac{\partial f_0}{\partial w_1} & \cdots & \frac{\partial f_n}{\partial w_1} \\ \vdots & & \vdots \\ \frac{\partial f_0}{\partial w_m} & \cdots & \frac{\partial f_n}{\partial w_m} \end{bmatrix} \begin{bmatrix} \frac{\partial I}{\partial f_0} \\ \vdots \\ \frac{\partial I}{\partial f_n} \end{bmatrix} = \bar{Z}\left(\frac{\partial I}{\partial f}\right)^\top = \bar{Z}(2Af), \tag{72}$$

where $\bar{Z} \in \mathbb{R}^{m\times(n+1)}$ is similar to Z defined in (46) as

$$\bar{Z} = \begin{bmatrix} \frac{\partial f_0}{\partial w_1} & \cdots & \frac{\partial f_n}{\partial w_1} \\ \vdots & & \vdots \\ \frac{\partial f_0}{\partial w_m} & \cdots & \frac{\partial f_n}{\partial w_m} \end{bmatrix} = \frac{1}{\sqrt{m}} \begin{bmatrix} a_1\mathbf{I}_{10}x_0 & \cdots & a_1\mathbf{I}_{1n}x_n \\ \vdots & & \vdots \\ a_m\mathbf{I}_{m0}x_0 & \cdots & a_m\mathbf{I}_{mn}x_n \end{bmatrix}. \tag{73}$$

Therefore, we have

$$\frac{dW}{dt} = -\left(\frac{\partial I}{\partial W}\right)^\top = -2\bar{Z}Af. \tag{74}$$

On the other hand,

$$\left(\frac{df}{dt}\right)^\top = \left[\frac{df(x_0)}{dt}, \cdots, \frac{df(x_n)}{dt}\right] \tag{75}$$

$$= \frac{dW}{dt}\begin{bmatrix} \frac{\partial f_0}{\partial w_1} & \cdots & \frac{\partial f_n}{\partial w_1} \\ \vdots & & \vdots \\ \frac{\partial f_0}{\partial w_m} & \cdots & \frac{\partial f_n}{\partial w_m} \end{bmatrix} \tag{76}$$

$$= -2f^\top A\bar{Z}^\top \bar{Z} \tag{77}$$

$$= -2f^\top A\bar{H}, \tag{78}$$

where $\bar{H} = \bar{Z}^\top \bar{Z}$. Finally, by taking the transpose, we get the dynamics for f as

$$\frac{df}{dt} = -2\bar{H}^\top A^\top f = -2\bar{H}Af, \tag{79}$$

as $\bar{H}^\top = \bar{H}$, and $A^\top = A$.

5.3 Discussion

Looking at the derivation of Equation (79), we observe that the matrix $\bar{H}$ in Equation (79) is precisely the same as the matrix H defined in Equation (57) except for the dimensions. Moreover, the matrix A in Equation (79) is also very similar to the matrix R derived from the iterative method discussed in section 2.2. We know from section 2.3 and section 4.3 that the spectral properties of this matrix HA determine which frequency component converges first. To be more precise, if we project the error onto the eigenvectors, those projected coefficients corresponding to the smaller eigenvalues of $\bar{H}A$ will decrease faster during the iterations. We know that in the limit of overparameterization, $\bar{H} \approx H^\infty$, and the latter has the property that low-frequency converges faster. On the other hand, the finite-difference matrix A is based on a differential operator. Unsurprisingly, A biases towards the high frequencies, which is the opposite of the behaviors of H^∞ (or $\bar{H}$). Heuristically, if we multiply them

together, it will neutralize the high-frequency bias of A and the low-frequency bias of H^∞ (or $\bar{H}$), and thus cancel out the biases, leading all components to converge at approximately the same speed. Since no component significantly drags down the overall convergence rate, we may observe faster error decay measured in any vector norm. This could be one main reason why the Deep Ritz method works well [6].

6 Conclusions

In section 2, we discussed the multigrid method, primarily how the iterative methods work, how the error decays with respect to the eigenvalue and the eigenvectors, and how the multigrid method tackles the downside of the "smoothing property" from the classical weighted Jacobi method in solving PDEs. It is one example of utilizing the spectral properties of the problem and coming up with a method to tackle its negative impacts. Counter-balancing the high-frequency bias was one primary motivation behind the multigrid method. In section 4, we review spectral analysis about the training process of the overparameterized neural networks. We understand that the iterative gradient-based method solving overparameterized neural network leads to the residual decay biases towards the low-frequency components of the residual. Thus the smooth modes converge the fastest. Based on our earlier understanding of the multigrid method, we realize this property could be a powerful tool for solving PDEs with a strong high-frequency bias. Therefore, if we apply overparameterized neural networks to a variational method for solving elliptic PDEs, it could work quite efficiently since the bias in the low frequencies from the overparameterized neural network and the bias towards the high frequencies from the PDE itself can be canceled out. The Deep Ritz method is precisely such a method belonging to this category. Our analysis also provides a new theoretical and mathematical insight into why the Deep Ritz method works better than the traditional methods [6].

References

[1] Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks, 2019.

[2] Ronen Basri, David Jacobs, Yoni Kasten, and Shira Kritchman. The convergence rate of neural networks for learned functions of different frequencies, 2019.

[3] William L. Briggs, Van Emden Henson, and Steve F. McCormick. *A Multigrid Tutorial (2nd Ed.)*. Society for Industrial and Applied Mathematics, USA, 2000.

[4] Simon Du. Ultra-Wide Deep Nets and the Neural Tangent Kernel (NTK), Jul 2020.

[5] Simon S. Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks, 2019.

[6] Weinan E and Bing Yu. The Deep Ritz method: A deep learning-based numerical algorithm for solving variational problems, 2017.

[7] Yann LeCun, D Touresky, G Hinton, and T Sejnowski. A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school*, volume 1, pages 21–28, 1988.

[8] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[9] Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

[10] Wikipedia contributors. Artificial neural network — Wikipedia, the free encyclopedia, 2021. [Online; accessed 4-May-2021].

[11] Zhi-Qin John Xu. Frequency principle: Fourier analysis sheds light on deep neural networks. *Communications in Computational Physics*, 28(5):1746–1767, Jun 2020.