On the Generalizability of Two-Layer ReLU-activated Neural Networks with a Fourier Feature Embedding

Nithin Raghavan[1] **Richard Hu**[1] **Alberto L. Checcone**[1]

[1]University of California, Berkeley

1 Problem Statement

We would like to analyze the generalizability bounds of binary coordinate-based multi-layer perceptrons (MLPs) with an input Fourier Feature encoding using Rademacher analysis. Coordinate-based MLPs are a form of artificial neural network (ANN) that take in as input a low-dimensional input (usually a 2D or 3D coordinate), and they can be increasingly found in many problems in computer graphics and computer vision, such as 3D shape regression, 2D image regression, CT, MRI and more. However, MLPs with a raw coordinate input usually do not perform well in practice, as they fail to learn high-frequency features. Recent work by Tancik, *et al.* [19] introduced the Fourier Feature embedding, which is a generalization of the sinusoidal positional encoding found in many papers [21, 13, 20] that maps an input coordinate $v \in \mathbb{R}^d$ onto the surface of a hypersphere as follows:

$$\gamma(v) = \left[\alpha_1 \cos(2\pi b_1^\top v), \alpha_1 \sin(2\pi b_1^\top v), \ldots, \alpha_r \cos(2\pi b_r^\top v), \alpha_r \sin(2\pi b_r^\top v)\right]^\top$$

where for all $i \in [r]$, $\alpha_i \in \mathbb{R}$ are weighting factors, and $b_i \in \mathbb{R}^d$ are vectors of the same length as v that control the mapping of the input points onto the hypersphere. Hereafter, we shall refer to the b_i vectors as the frequency factors. We do not assume any frequency prior on the type of task or shape we are attempting to perform regression on, so in practice, we will derive the frequency factors b_i from an isotropic Gaussian distribution.

Tancik, *et al.* additionally performed further analysis using neural tangent kernel theory to show that coordinate-based MLPs with such an encoding are able to learn high frequency features. We intend to continue this line of research by using neural tangent kernel theory and Rademacher analysis to produce generalizability bounds on binary coordinate-based MLPs with a Fourier Feature embedding in order to produce bounds in terms of the number of frequency components r and the values of the frequency factors b_i. This has real-world applications for understanding the training dynamics of similar classes of networks with a Fourier Feature encoding. An example of such classes are occupancy networks [12] whose weights' ℓ_2 norms are bounded above by a constant, which [19] additionally demonstrated were significantly aided by a Fourier Feature embedding. Further experiments with occupancy networks can therefore help to verify our generalization bounds. Additionally, as the bounds we derive are in terms of the number of frequency components and frequency factors, we thereby have control over the hyperparameters in the Fourier Feature encoding. This allows us to measure the relative impact that each of these components has over the generalizability, giving an idea as to which parameters to select to achieve the best performance.

There are several tools which we will consider for this purpose. The first is neural tangent kernel (NTK) theory. The NTK is a type of kernel which describes the evolution of sufficiently-wide ANNs. In particular, the inference performed by infinite-width artificial neural networks is in expectation equal to kernel ridge regression with no ridge [9]. Previous work such as [2, 3] developed generalizability bounds on such networks using Rademacher complexity. Another tool we will consider is Reproducing Kernel Hilbert Space (RKHS) Theory. An RKHS is a Hilbert space induced by a kernel k that obeys the reproducing property (namely, that function evaluation on a point $f(x)$ is equivalent to an inner product with the kernel k in this space: $\langle k(x, \cdot), f \rangle$), among other axioms [1]. We have used a combination of these tools to obtain a bound similar in nature to [2], but more suited

to our purposes, which allow us to obtain a thorough, yet unwieldy, generalization bound in terms of the frequency factors and the number of frequency components.

2 Introduction

2.1 Related Work and Assumptions

It is well-known that MLPs have a spectral bias that prevents them from learning high-frequency functions [14, 5]. [21, 13] empirically found that a sinusoidal mapping of input coordinates is able to overcome this issue to an extent, and allows MLPs to learn higher-frequency features. Generalizing this, Tancik, *et al*. (2020) demonstrates that Fourier Feature embeddings for coordinate-based MLPs not only enable them to learn any higher-frequency features, but also improve their convergence rate. Fourier Features project the low-dimensional coordinate inputs onto a hypersphere. For our purposes, we set all $\alpha_i = 1$ in the Fourier Feature encoding as a simplification.

In our research, we expand on the work of Tancik, *et al*. (2020) by providing specific bounds on the generalizability of two-layer ReLU networks trained on inputs preprocessed with a Fourier Feature embedding. Because generalizability can be viewed as a complexity measure of data that one can use to predict the test accuracy of the learned neural network, we can thereby give a clear bound on the evolution of certain classes of neural networks so that we can see how they evolve over time. Furthermore, this also measures the richness of the class of functions that a neural network can learn.

Empirical Rademacher complexity directly gives an upper bound on generalization error and Rademacher complexity can give us an easily verifiable measure that can differentiate between true labels and random labels [2]. We therefore conduct our analysis by utilizing the NTK. [9] proved that the as the width of an MLP with a ReLU activation increases to infinity, its inference converges to kernel ridge regression using the NTK with the ridge parameter λ approaching zero, with the NTK itself defined as follows, where f is the neural network prediction function and the weights θ are initialized as Gaussian [19, 3]:

$$k_{\mathrm{NTK}}(x_i, x_j) = \mathbb{E}_{\theta \sim \mathcal{N}} \left(\left\langle \frac{\partial f(x_i;\theta)}{\partial \theta}, \frac{\partial f(x_j;\theta)}{\partial \theta} \right\rangle \right)$$

As such, the NTK is a dot product kernel, which only depends on the dot product between two sample points $x_i^\top x_j$ (i.e. $k_{\mathrm{NTK}}(x_i, x_j) = h_{\mathrm{NTK}}(x_i^\top x_j)$, a function of a single variable). For a two-layer neural network of the kind we will conduct our analysis on, [2, 6] describe its NTK as:

$$h_{\mathrm{NTK}}(x_i^\top x_j) = \frac{x_i^\top x_j (\pi - \cos^{-1}(x_i^\top x_j))}{2\pi}$$

The NTK for larger networks can be defined recursively. When dealing with a sample of size n, the matrix that solves the kernel ridge regression problem (with no ridge) is the kernel matrix (or Gram matrix) $K_{\mathrm{train}} \in \mathbb{R}^{n \times n}$ such that $K_{\mathrm{train},ij} = h_{\mathrm{NTK}}(x_i^\top x_j)$. [2, 3] additionally provide bounds that indicate that even when the network width is not infinite, the NTK nonetheless is able to closely approximate the actual training dynamics. Thus, assuming a sufficiently overparameterized network, the NTK enables us to observe and track the behavior of our Fourier Feature embedded neutral networks in the limit without needing to work directly with the structure of the MLP, which greatly simplifies computations. Arora, *et al.* , which used Rademacher complexity to obtain a generalization bound for a general two-layer neural network with a normalized input x, additionally found that the relative values of the projection of the input labels onto the eigenvectors of the NTK Gram matrix was necessary for improved generalizability as well [2]. As a Fourier Feature embedding additionally results in the NTK becoming shift-invariant:

$$\begin{aligned} h_{\mathrm{NTK}}(\gamma(x_i)^\top \gamma(x_j)) &= h_{\mathrm{NTK}} \left(\sum_{i=1}^{r} \alpha_i^2 \cos(b_i \pi x_i) \cos(b_i \pi x_j) + \alpha_i^2 \sin(b_i \pi x_i) \sin(b_i \pi x_j) \right) \\ &= h_{\mathrm{NTK}} \left(\sum_{i=1}^{r} \alpha_i^2 \cos(b_i \pi (x_i - x_j)) \right) \end{aligned}$$

We can then see that a Fourier Feature mapping enables us to manipulate the eigenvalues of the NTK Gram matrix, thereby allowing the frequency factors b_i to control generalizability. Such a

mapping additionally turns the NTK Gram matrix into a convolution if the training data is sampled on a regular grid. Intuitively, increasing the number of frequencies greatly will result in weaker generalizability due to overfitting, whereas tuning the frequency factors could potentially result in stronger generalizability, which is confirmed by our results. When conducting our analysis, we derived inspiration from [2, 8, 17, 10].

Our principal result is that with high probability, the generalization error of two-layer ReLU networks with a Fourier Feature embedding is upper bounded by an expression that contains both the frequency factors b_i and the number of frequencies r. Broadly speaking, this means that it would be possible to numerically optimize the hyperparameters of the Fourier Feature embedding to minimize this error and produce a model that can generalize the training data best. In order to arrive at this result, we derived a very basic Rademacher bound in Section 3.1 that expresses the generalizability only in terms of the r hyperparameter, by assuming an upper bound on the cosine terms. Subsequently in Section 3.2, we strengthened that bound by considering the loss class of kernel ridge regression with the RKHS of the NTK. In Section 3.3, we make a first attempt at proving a rather weak bound on the generalization error involving both the number of frequency components r and frequency factors b_i using results from Arora, *et al.* (2019). Finally in Section 3.3, we again utilize the Reproducing Kernel Hilbert Space to derive a stronger bound on the generalization error involving both r and b_i, and in Section 3.4, we attempt to intuitively connect this bound back to the original bound in Section 3.1.

2.2 Rademacher Bound from Arora, *et al.* (2019)

Theorem 5.1 of [3] provides an upper bound on the generalization error of two-layer ReLU networks trained by gradient descent evaluated with any 1-Lipschitz loss function, which we summarize as follows. Arora, *et al.* combines the fact that with probability $1-\delta$, the class of two-layer ReLU networks whose first layer weights have bounded Frobenius norms of at most D and whose second layer weights have bounded ℓ_2-norms of at most C has Rademacher complexity given by the following (noting that since the Frobenius norm is greater than the ℓ_2-norm, we can use the same function class as before):

$$\mathrm{Rad}_S\left(\mathcal{H}\right) \leq \frac{D}{\sqrt{2n}}\left(1+\left(\frac{2\log\frac{2}{\delta}}{m}\right)^{1/4}\right)+\frac{2C^2\sqrt{m}}{\kappa}+C\sqrt{2\log\frac{2}{\delta}}$$

given κ such that $m \geq \kappa^{-2}\mathrm{poly}\left(n, \lambda_0^{-1}, \delta^{-1}\right)$ and that with probability $1-\delta$, the total movement of the first layer weights W at training epoch k (shorthand: $W(k)$) is bounded as follows:

$$\|W(k)-W(0)\|_F \leq \sqrt{y^\top K_{\mathrm{train}}^{-1} y}+O\left(\frac{n\kappa}{\lambda_0\delta}\right)+\frac{\mathrm{poly}\left(n,\lambda_0^{-1},\delta^{-1}\right)}{m^{1/4}\kappa^{1/2}}$$

They then derive the result that the population loss is upper bounded by

$$\mathcal{L}_{\mathcal{D}}(f_{W(k),a}) \leq \sqrt{\frac{2y^\top K_{\mathrm{train}}^{-1} y}{n}}+O\left(\sqrt{\frac{\log\frac{n}{\lambda_0\delta}}{n}}\right)$$

A similar bound also appears in [7]. We intend to use this bound as inspiration for the bounds that we derive for Fourier Feature-encoded networks. Arora, *et al.* does not explicitly make use of the Fourier Feature embedding (as it assumes that the input vector is normalized), but we derive bounds using the RKHS norm that are close, but better suited for our use case.

3 Original Research

3.1 Basic Rademacher Bound

We would like to get a basic idea of the generalizability of bounded two-layer neural network classes only in terms of the number of frequencies in the Fourier Feature embedding. This seemed like a logical start as the number of frequencies r is an easily-adjustable hyperparameter, and it would keep our final expression simple. Assume we have a set of training points $S = ((x_1, y_1), \cdots, (x_n, y_n))$

where $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$. Assume the width of the neural network is m, and we have a Fourier Feature embedding γ with fixed parameters b_i and r. Let ϕ define the rectified linear unit.

By definition, the network computes, for some input x,

$$f_{W,a}(x) = \frac{1}{\sqrt{m}} \sum_{j=1}^{m} a_j \phi(w_j^\top \gamma(x))$$

where the network prediction function f is parameterized by $w_1, \cdots, w_m \subset \mathbb{R}^{2r}$, the first-layer weights, and $a = (a_1, \cdots, a_m)^m \in \mathbb{R}^m$, which are the second-layer weights. Let $\mathcal{H}$ be the following class of two-layer neural networks whose weights' ℓ_2-norm are upper bounded by some constants $C, D \in \mathbb{R}$:

$$\mathcal{H} \doteq \{f_{W,a} : \|a\|_2 \leq C, \|w_j\|_2 \leq D \ \ \forall j \in [m]\}$$

In order to get an idea of how to proceed with this basic Rademacher bound, note that since the input vectors x_i are mapped onto the hypersphere by the Fourier Feature embedding, the ℓ_2-norm of the embedded input is therefore

$$\|\gamma(x_i)\|_2 = \sqrt{\sum_{i=1}^{r} \cos^2(2\pi b_i^\top x_i) + \sum_{i=1}^{r} \sin^2(2\pi b_i^\top x_i)} = \sqrt{r}$$

and we can use this to make a simplifying assumption. The empirical Rademacher complexity of a function class $\mathcal{F}$ with respect to a sample $S = (x_1, \cdots, x_n)$ of size n is defined as

$$\mathrm{Rad}_S(\mathcal{F}) = \mathbb{E}_{\sigma \in \{\pm 1\}} \left(\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i f(x_i) \right)$$

so, we proceed to derive our weak Rademacher bound as the following. Using Cauchy-Schwarz, we get the following:

$$\begin{aligned}
\mathrm{Rad}_S(\mathcal{H}) &= \mathbb{E}_{\sigma \in \{\pm 1\}} \left(\sup_{f_{W,a} \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i f_{W,a}(\gamma(x_i)) \right) \\
&= \mathbb{E}_{\sigma \in \{\pm 1\}} \left(\sup_{f_{W,a} \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \left(\frac{1}{\sqrt{m}} \sum_{j=1}^{m} a_j \phi(w_j^\top \gamma(x)) \right) \right) \\
&= \frac{1}{\sqrt{m}} \mathbb{E}_{\sigma \in \{\pm 1\}} \left(\sup_{f_{W,a} \in \mathcal{H}} \frac{1}{n} a^\top \sum_{i=1}^{n} \sigma_i \phi(W \gamma(x_i)) \right) \\
&\leq \frac{1}{\sqrt{m}} \mathbb{E}_{\sigma \in \{\pm 1\}} \left(\sup_{\|a\|_2 \leq C, \|w_j\|_2 \leq D \ \ \forall j \in [m]} \|a\|_2 \left\| \frac{1}{n} \sum_{i=1}^{n} \sigma_i \phi(W \gamma(x_i)) \right\|_2 \right)
\end{aligned}$$

Now, it remains to bound $\left\| \frac{1}{n} \sum_{i=1}^{n} \sigma_i \phi(W \gamma(x_i)) \right\|_2$. Note that

$$\|W \gamma(x_i)\|_2 = \sqrt{\sum_{j=1}^{m} (w_j^\top x_i)^2} \leq \sup_{j \in [m]} \sqrt{m (w_j^\top x_i)^2} = \sqrt{m} \sup_{j \in [m]} |w_j^\top x_i|$$

For such a j, define w to be the value of w_j that maximizes $|w_j^\top x_i|$, and continue:

$$\begin{aligned}
\mathrm{Rad}_S(\mathcal{H}) &\leq \frac{1}{\sqrt{m}} \mathbb{E}_{\sigma \in \{\pm 1\}} \left(\sup_{\|a\|_2 \leq C, \|w_j\|_2 \leq D \ \ \forall j \in [m]} \|a\|_2 \left\| \frac{1}{n} \sum_{i=1}^{n} \sigma_i \phi(W \gamma(x_i)) \right\|_2 \right) \\
&\leq C \mathbb{E}_{\sigma \in \{\pm 1\}} \left(\sup_{\|w_j\|_2 \leq D \ \ \forall j \in [m]} \sup_{j \in [m]} \left| \frac{1}{n} \sum_{i=1}^{n} \sigma_i \phi(w_j^\top \gamma(x_i)) \right| \right) \\
&= C \mathbb{E}_{\sigma \in \{\pm 1\}} \left(\sup_{\|w\|_2 \leq D} \left| \frac{1}{n} \sum_{i=1}^{n} \sigma_i \phi(w^\top \gamma(x_i)) \right| \right) \\
&= C \mathbb{E}_{\sigma \in \{\pm 1\}} \left(\sup_{\|w\|_2 \leq D} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \phi(w^\top \gamma(x_i)) \right)
\end{aligned}$$

Where the last step was able to be taken because the class $\mathcal{H}$ is symmetric. We now refer to Theorem 12 of [4], which states that for any K-Lipschitz function ϕ that satisfies $\phi(0) = 0$, then $\mathrm{Rad}_S(\phi \circ \mathcal{F}) = 2K\mathrm{Rad}_S(\mathcal{F})$. Since ReLU is 1-Lipschitz, we can go further using the same trick as before (and extract w in order to use Cauchy-Schwarz):

$$\begin{aligned}
\mathrm{Rad}_S(\mathcal{H}) &\leq C\mathbb{E}_{\sigma\in\{\pm 1\}}\left(\sup_{\|w\|_2\leq D}\frac{1}{n}\sum_{i=1}^{n}\sigma_i\phi(w^\top\gamma(x_i))\right)\\
&\leq 2C\mathbb{E}_{\sigma\in\{\pm 1\}}\left(\sup_{\|w\|_2\leq D}\frac{1}{n}\sum_{i=1}^{n}\sigma_i w^\top\gamma(x_i)\right)\\
&\leq \frac{2CD}{n}\mathbb{E}_{\sigma\in\{\pm 1\}}\left(\left\|\sum_{i=1}^{n}\sigma_i\gamma(x_i)\right\|_2\right)\\
&\leq \frac{2CD}{\sqrt{n}}\mathbb{E}_{\sigma\in\{\pm 1\}}\left(\|\gamma(x_i)\|_2\right)\\
&= 2CD\sqrt{\frac{r}{n}}
\end{aligned}$$

Assuming we hold the sample size n constant, this result indicates that increasing the number of frequencies greatly will result in weaker generalizability (due to overfitting), which matches with our previous intuition. It is clear that this bound does not give us much information, as reducing r to zero would result in a very small generalization error. This is trivially true, as the function class would have no features and so the training accuracy, along with the test accuracy, would be 1. Therefore, we need to produce a better bound with frequency factors b_i in order to see if there is a generalizability tradeoff between them and the number of frequencies r, which may give us more information.

3.2 Rademacher Bound from the RKHS Induced by the Neural Tangent Kernel

Theorem 3.2 of [3] provides explicit equivalence between trained neutral networks and kernel regression. Assuming Gaussian initialization of the weights, and for m large enough (which it will attain in the limit), $f_{W,a}$ converges to a kernel regression predictor using the neural tangent kernel (NTK) with the regularization eventually going to 0 ($\lambda \to 0$), based on [9]. Additionally, the same paper provides bounds for the training dynamics of sufficiently wide neural networks using the NTK. [6] additionally describes how the corresponding function space induced by the NTK is an RKHS, and analyzing properties of functions associated with such an RKHS may give us some insight as to the dynamics of the original neural network function class. For generalization purposes, we intend to analyze the Rademacher complexity of kernel classes with the Fourier Feature-modified NTK by utilizing properties of the corresponding RKHS. This is sensible because we can make direct use of the reproducing property, which states that for all $f \in \mathcal{K}$, for a point x in the sample S, $f(x) = \langle h_{\mathrm{NTK}}(x,\cdot), f(\cdot)\rangle$ (where $\langle f, g\rangle = \int_S f(x)g(x)dx$).

Formally, define $\mathcal{K}$ to be the reproducing kernel Hilbert space with the NTK kernel $h_{\mathrm{NTK}}(\cdot,\cdot)$ and let $f(\cdot) = \sum_{j=1}^{n}\beta_i h_{\mathrm{NTK}}(\cdot,\gamma(x_i))$. The kernel ridge regression problem is then defined to be

$$f_{\mathrm{ker}} = \arg\min_{f\in\mathcal{K}}\frac{1}{2}\sum_{i=1}^{n}(y_i - f(\gamma(x_i)))^2 + \frac{\lambda}{2}\|f\|_{\mathcal{K}}^2$$

We need to find the empirical risk minimizer (ERM) to get the actual kernel prediction function that the neural network learns [18, 11]. The representer theorem states that the minimizer f_{ker} can be represented as a finite linear combination of kernel products evaluated on the training set points [15]. Therefore the solution is of the following form:

$$f_{\mathrm{ker}} = \sum_{i=1}^{n}\beta_i^* h_{\mathrm{NTK}}(\cdot,\gamma(x_i))$$

If f_{ker} is a kernel method with kernel function $h_{\mathrm{NTK}}(\cdot,\cdot)$ then the corresponding kernel Gram matrix $K_{\mathrm{train}} \in \mathbb{R}^{n\times n}$ is defined such that $(K_{\mathrm{train}})_{ij} = h_{\mathrm{NTK}}(x_i, x_j)$ for any two points in the sample. To evaluate this method on another set $S' = (x'_1, \cdots, x'_{n'})$ of n' testing data points, we construct another

kernel Gram matrix, $K_{\text{test}} \in \mathbb{R}^{n' \times n}$, defined as $(K_{\text{test}})_{ij} = h_{\text{NTK}}(x'_i, x_j)$ for $x_i \in S', x_j \in S$. Then our kernel method prediction function is $f(S'; S) = K_{\text{test}} K_{\text{train}}^{-1} y$, where y refers to the original labels.

If we want to measure the generalization error of this ERM, let $\epsilon, \delta \in (0, 1)$. We can use Theorem 26.12 from [17] to obtain a bound on the population loss, assuming that the loss function is ρ-Lipschitz, the ℓ_2 norm of each member of the class of functions is bounded by B, the ℓ_2 norm of the input is bounded by R, and that the output of the loss function differs from the real value by no more than c (a similar bound can be found in [16]):

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}\left(|f_{\text{ker}} - y|\right) \leq L_S(h) + \frac{2\rho BR}{\sqrt{n}} + c\sqrt{\frac{2\log(2/\delta)}{n}} = \frac{2\rho BR}{\sqrt{n}} + c\sqrt{\frac{2\log(2/\delta)}{n}}$$

since the empirical loss is (by definition) zero. In order to calculate this, note that for any function f in $\mathcal{K}$, $\|f\|_2 \leq \|f\|_{\mathcal{K}}$. (Proof: consider $|f(x)|^2$. By Cauchy-Schwarz, we have that $|f(x)|^2 = |\langle f, h_{\text{NTK}}(x, \cdot)\rangle| \leq |k(x, x)|\|f\|_{\mathcal{K}}^2$, and since $\int |h_{\text{NTK}}(x, x)|^2 \leq 1$, this implies that $\|f\|_2 \leq \|f\|_{\mathcal{K}}$.) Because of this, we focus on the balls of radius f bounded above by M:

$$\mathcal{M} = \{f \in \mathcal{K} : \|f\|_2 \leq \|f\|_{\mathcal{K}} \leq M\}$$

We want to get as tight a bound on M as possible. For a generic function $f \in \mathcal{K}$, we have that

$$\begin{aligned}
\|f\|_{\mathcal{K}}^2 &= \left\| \sum_{i=1}^{n} \beta_i h_{\text{NTK}}(\cdot, \gamma(x_i)) \right\|_{\mathcal{K}}^2 \\
&= \left\langle \sum_{i=1}^{n} \beta_i h_{\text{NTK}}(\cdot, \gamma(x_i)), \sum_{j=1}^{n} \beta_j h_{\text{NTK}}(\cdot, \gamma(x_j)) \right\rangle \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_i \beta_j \left\langle h_{\text{NTK}}(\cdot, \gamma(x_i)), h_{\text{NTK}}(\cdot, \gamma(x_j)) \right\rangle \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_i \beta_j h_{\text{NTK}}(\gamma(x_i), \gamma(x_j)) \\
&= \beta^\top K_{\text{train}} \beta
\end{aligned}$$

To get the bound for f_{ker}, note that we can write it in a special form: $f_{\text{ker}}(S') = K_{\text{test}} K_{\text{train}}^{-1} y$. If K_{test} were actually just the training labels again (i.e. $S = S'$), then $K_{\text{test}} = y^\top$, and so $\|f_{\text{ker}}\|_{\mathcal{K}} = \sqrt{y^\top K_{\text{train}}^{-1} y}$.

Noting that the loss function is 1-Lipschitz, and that $\|\gamma(x_i)\|_2 = \sqrt{r}$ like before, we get that the generalization error is

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}\left(|f_{\text{ker}} - y|\right) \leq 2\sqrt{r \cdot \frac{y^\top K_{\text{train}}^{-1} y}{n}} + O\left(\sqrt{\frac{2\log(2/\delta)}{n}}\right)$$

where $(K_{\text{train}})_{ij} = h_{\text{NTK}}(x_i^\top x_j)$. This maps the generalization error in a form that contains both the number of frequencies r as well as the frequency factors b_i, as K_{train} is composed of both. In the next section, we will explore ways of representing this in terms of explicit reliance on the frequency factors in order to obtain a better intuition as to their role in the bound.

3.3 RKHS Bound with Explicit Reliance on Frequency Factors

Our next goal was to create a closed form bound depending on the frequency factors. The intention was to derive an (perhaps complicated) expression that involved every tune-able parameter. Ultimately, we derived that when the sample points are uniformly sampled (the inputs are a fixed distance apart from each other, specifically $x_i = \frac{i}{n}$), the NTK Gram matrix becomes circulant, and thus forms a convolution over the input space. Such an assumption about the inputs is reasonable because in practice one easily could take a uniform sample of points from a 2D or 3D image, and there is no particular reason why the points need to be sampled randomly.

In this case, because K_{train} represents a convolution, the eigenvectors of our kernel matrix are the columns of the Discrete Fourier Transform (DFT) matrix:

$$F = \frac{1}{\sqrt{n}} \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \dots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \dots & \omega^{2(n-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \dots & \omega^{3(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \omega^{2(3-1)} & \dots & \omega^{(n-1)(n-1)} \end{bmatrix}$$

Where $\omega = e^{\frac{-2\pi i}{n}}$ is the primitive nth root of unity.

Noting this fact, we can write out the spectral decomposition of the kernel matrix to come up with some expression for its inverse in terms of the $h_{\text{NTK}}(\gamma(x_i)^\top \gamma(x_j))$ kernel function.

$$\begin{aligned} K_{\text{train}} &= F \Lambda F^* \\ &= \sum_{i=1}^{n} \lambda_i v_i v_i^{*T} \\ K_{\text{train}}^{-1} &= F \Lambda^{-1} F^* \\ &= \sum_{i=1}^{n} \frac{1}{\lambda_i} v_i v_i^* \end{aligned}$$

Where v_i is the ith column of the DFT matrix and $\Lambda_{i,i}^{-1} = \frac{1}{\lambda_i}$ is the inverse diagonal eigenvalue matrix. The additional simplification uses the fact that the spectral decomposition can also be written as a weighted sum of dyads.

As the eigenvalues of circulant matrices have a closed form expression, we can then write each eigenvalue λ_i as a summation involving ω and elements from the first row of the original matrix: $\lambda_i = A_{1,1} + A_{1,2}\omega^j + \cdots + A_{1,N}\omega^{(n-1)j}$. In our case, this looks like:

$$\lambda_i = \sum_{j=1}^{n} \omega^{i(j-1)} K_{1,j}$$

However, we know that the first row of our kernel matrix is the h_{NTK} function applied to $\gamma(x_1)$ and each of the sample points:

$$\lambda_i = \sum_{j=1}^{n} \omega^{i(j-1)} h_{\text{NTK}}(\gamma(x_1)^\top \gamma(x_j))$$

$$h_{\text{NTK}}(\gamma(x_i)^\top \gamma(x_j)) := \sum_{j=1}^{r} a_j^2 \cos(2\pi b_j (x_i - x_j)) \left(\frac{\pi - \cos^{-1}\left(\sum_{j=1}^{r} a_j^2 \cos(2\pi b_j (x_i - x_j))\right)}{2\pi} \right)$$

This yields the following expression for our kernel matrix:

$$\begin{aligned} K_{\text{train}} &= \sum_{i=1}^{n} v_i v_i^* \sum_{j=1}^{n} \omega^{i(j-1)} h_{\text{NTK}}(\gamma(x_1)^\top \gamma(x_j)) \\ &= \sum_{i=1}^{n} v_i v_i^* \sum_{j=1}^{n} \omega^{i(j-1)} \sum_{k=1}^{r} a_k^2 \cos(2\pi b_k (x_1 - x_j)) \left[\frac{\pi - \cos^{-1}(\sum_{q=1}^{r} a_q^2 \cos(2\pi b_q (x_1 - x_j)))}{2\pi} \right] \end{aligned}$$

And its inverse, which can be substituted into the RKHS bound:

$$\begin{aligned} K_{\text{train}}^{-1} &= \sum_{i=1}^{n} v_i v_i^* \frac{1}{\sum_{j=1}^{n} \omega^{i(j-1)} h_{\text{NTK}}(\gamma(x_1)^\top \gamma(x_j))} \\ &= \sum_{i=1}^{n} v_i v_i^* \frac{1}{\sum_{j=1}^{n} \omega^{i(j-1)} \sum_{k=1}^{r} a_k^2 \cos(2\pi b_k (x_1 - x_j)) \frac{\pi - \cos^{-1}(\sum_{q=1}^{r} a_q^2 \cos(2\pi b_q (x_1 - x_j)))}{2\pi}} \end{aligned}$$

This expression is quite large, and so we would like to get some intuition as to how precisely the RKHS bound (with the above expression substituted in) is better than the basic Rademacher bound. Intuitively, in the worst case, the RKHS bound should reduce to the basic one. As such, in the next section we make further assumptions in order to produce that connection.

3.4 Attempting to Convert the RKHS Bound to Basic Rademacher Bound

This section details the approaches we tried to reduce the RKHS bound to our basic Rademacher bound given these assumptions. Our goal was to draw a relation between the bounds as doing so would inspire increased confidence in their correctness.

Our first attempt was to plug in fixed frequency values to see if one simplified the above expression. Namely, let

$$z_j := \sum_{k=1}^{r} a_k^2 \cos(2\pi b_k (x_1 - x_j))$$

$$K_{\text{train}}^{-1} = \sum_{i=1}^{n} v_i v_i^* \frac{1}{\sum_{j=1}^{n} \omega^{i(j-1)} z_j \left(\frac{\pi - \cos^{-1}(z_j)}{2\pi}\right)}$$

If we know what n and r are, then it would be easy to choose frequency values b to achieve a given z_j. Thus, we tried to see if any given choice for z_j converted the RKHS bound to our original weak bound.

If $\forall j, z_j = 1$:

$$\begin{aligned}
\lambda_{\max} &= \frac{1}{\sum_{j=1}^{n} \omega^{i(j-1)} \left(\frac{\pi - \cos^{-1}(1)}{2\pi}\right)} \\
&= \frac{1}{\sum_{j=1}^{n} \omega^{i(j-1)} \left(\frac{1}{2}\right)} \\
&= \frac{2}{\sum_{j=1}^{n} \omega^{i(j-1)}}
\end{aligned}$$

It is worth noting that each term in the summation represents a rotation along a polar circle, so their norms are all identical. However, that does not lead to any particularly useful insight.

Another idea we had was by looking back at our original expression:

$$\sqrt{r \cdot \frac{y^\top K_{\text{train}}^{-1} y}{n}}$$

We note that with a given r and n, the rest of the bound depends only on $y^\top K_{\text{train}}^{-1} y$. For a fixed y with $y \in \{\pm 1\}^n$, the Rayleigh quotient $\frac{y^\top K^{-1} y}{n}$ has upper and lower bounds based on the eigenvalues of K_{train}^{-1}. Thus, we tried to see if we could use the largest eigenvalue as an upper bound for the inside of the square root.

$$\begin{aligned}
\lambda_{\max} &= \max_{z_j} \frac{1}{\sum_{j=1}^{n} \omega^{i(j-1)} z_j \left(\frac{\pi - \cos^{-1}(z_j)}{2\pi}\right)} \\
\infty &= \lim_{z_j \to 0^+} \frac{1}{\sum_{j=1}^{n} \omega^{i(j-1)} z_j \left(\frac{\pi - \cos^{-1}(z_j)}{2\pi}\right)}
\end{aligned}$$

However, by when substituting into the eigenvalue expression above, we realized that as $z_j \to 0$, the denominator of the fraction gets infinitesimally small, so the eigenvalue blows up arbitrarily close to infinity. This line of reasoning may yield useful bounds given a particular target z_j and K_{train}, but without such constraints, this upper bound is useless.

One last similar approach we had was to produce a very weak bound. By noting that

$$y^\top K_{\text{train}}^{-1} y = \sum_i \sum_j y_i y_j K_{\text{train},ij}^{-1} \leq \sum_i \sum_j K_{\text{train},ij}^{-1} \leq n^2 \max_{i,j} K_{\text{train},ij} \leq n^2 \frac{1}{\lambda_{\min}\left(K_{\text{train}}\right)}$$

we realize that the largest element of the inverse of a PSD symmetric matrix K is upper bounded by the the inverse of the smallest eigenvalue of K (as the inverse of a PSD matrix is PSD, and the maximum can be computed directly from the orthogonal decomposition of the inverse). We could then obtain a weak bound by finding the smallest eigenvalue of K_{train}. Unfortunately, we ran into a similar result as the above case, and so could not find values for z_j that resulted in a useful simplification to the bound. As a result, given more time, we would run experiments to test empirical results of various frequency hyperparameters and use the results to guide future theoretical research.

4 Conclusion

In summary, we managed to utilize Rademacher complexity to prove progressively tighter bounds on the generalization error of coordinate-based, two-layer, ReLU MLPs with Fourier Feature encodings. We began by proving a basic bound that contained only the number of frequencies r in order to establish a baseline intuition as to how the hyperparameters would broadly affect the generalization. We then proceeded to use RKHS theory with the NTK as the reproducing kernel to construct a stronger bound using r and the frequency factors b_i in order to obtain possible intuition as to a tradeoff between the two. Finally, we briefly touched on our analysis of the bound produced by Arora *et al*. (2019) (which also included the frequency factors, but used a different derivation that was unsuitable for our use case). To derive the tight bound in Section 3.3, we used explicit characteristics of the Fourier Feature embedding—specifically, that the Gram matrix of the NTK with the Fourier Feature encoding represents a convolution—in order to arrive at a full expression of the upper bound of generalizability that contained both the frequency factors b and the number of frequencies r, and was also specific to the Fourier Feature embedding. After deriving this bound, we attempted to connect this bound back to the first basic bound in order to demonstrate that the intuition from the most basic bound would hold consistent as we tightened the bounds but we were unable to successfully draw this connection.

From this point, the most obvious direction of expansion would be to attempt to complete the derivation in Section 3.4 to check if the intuition of the weaker bounds actually holds. On the other hand, another direction we could go would be to empirically test our bound by numerically optimizing over r and b and testing the performance of the hyperparameters using occupancy networks to observe to what extent the optimal hyperparameters can improve test accuracy. As stated previously, occupancy networks [12] are binary classification networks upon which we could use the same assumptions to conduct similar experiments to [2] in order to achieve this.

References

[1] Christine Thomas-Agnan Alain Berlinet. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Springer, 2004.

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

[3] Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché Buc, Emily B. Fox, and Roman Garnett, editors, *NeurIPS*, pages 8139–8148, 2019.

[4] Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *J. Mach. Learn. Res.*, 3(null):463–482, March 2003.

[5] Ronen Basri, Meirav Galun, Amnon Geifman, David Jacobs, Yoni Kasten, and Shira Kritchman. Frequency bias in neural networks for input of non-uniform density. *arXiv preprint arXiv:2003.04560*, 2020.

[6] Alberto Bietti and Julien Mairal. On the inductive bias of neural tangent kernels. *NeurIPS*, 2019.

[7] Yuan Cao and Quanquan Gu. Generalization error bounds of gradient descent for learning over-parameterized deep relu networks, 2019.

[8] Wei Hu. Neural tangent kernel (ntk) made practical, 2020.

[9] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural Tangent Kernel: Convergence and generalization in neural networks. *NeurIPS*, 2018.

[10] Zhihan Li, Yuwei Fan, and Lexing Ying. Multi-level fine-tuning: Closing generalization gaps in approximation of solution maps under a limited budget for training data, 2021.

[11] Julien Mairal and Jean-Philippe Vert. Machine learning with kernel methods, 2020.

[12] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. *CVPR*, 2019.

[13] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. *arXiv preprint arXiv:2003.08934*, 2020.

[14] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. *ICML*, 2019.

[15] Bernhard Schoelkopf, Ralf Herbrich, and Alex J. Smola. A generalized representer theorem. In David Helmbold and Bob Williamson, editors, *Computational Learning Theory*, pages 416–426, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[16] Jonathan Shafer. Unit 6: Rademacher complexity, 2021.

[17] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, USA, 2014.

[18] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[19] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017.

[21] Ellen D. Zhong, Tristan Bepler, Joseph H. Davis, and Bonnie Berger. Reconstructing continuous distributions of 3D protein structure from cryo-EM images. *ICLR*, 2020.