Random Fourier Features and Spectral Bias

Daniel Kyte-Zable

Introduction

Recent research in computer vision has examined the consequences of replacing traditional means of representing images (such as voxel grids) with coordinate-based multilayer perceptrons (MLPs), fully-connected networks trained on a single image that predict target values from pixel coordinates. Coordinate-based MLPs, which offer fundamentally continuous representations of images, are well-suited to novel view synthesis [6] and image upscaling [8]; further, the cost of storing a trained network is typically cheaper than that of discrete representations.

However, coordinate-based MLPs often fail to capture high-frequency components in a target image [9,10]. This phenomenon, known as spectral bias, has been widely-observed in other networks trained to complete different tasks. Spectral bias has been put forward as a probable explanation for deep networks' ability to generalize to unseen data [3]; in high-dimensional learning domains, spectral bias prevents deep networks from learning noisy components in data and thereby overfitting.

Tancik et al. 2020 [9] demonstrated that positionally encoding data prior to training overcomes spectral bias in low-dimensional settings, such as pixel regression. To this end, [9] draws on the theory of Random Fourier Feature (RFF) embeddings, first proposed as a means of scaling kernel machines to large quantities of data [7]. Further research has extended [9]'s approach to overcoming spectral bias in other settings, such as training physics-informed neural networks (PINNs) [10].

We discuss both the theoretical and empirical links between spectral bias and RFF embeddings. To this end, we divide the report into three over-arching segments. We first detail how spectral bias can be ascribed to the eigendecomposition of the neural tangent kernel (NTK); we then explore experimental results highlighting the utility of RFF embeddings; finally, we discuss how tuning RFF embeddings changes the spectrum of the NTK.

Background

We introduce several pieces of notation and background to aid our discussion. We consider some arbitrary dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$ where $\mathbf{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}$ with small d (e.g. $d = 2$ for pixel regression). We denote the set of training samples as $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N \in \mathbb{R}^{N\times d}$ and the set of training labels as $\mathcal{Y} = \{y_i\}_{i=1}^N \in \mathbb{R}^N$. In our later experiments on images, we take $\mathcal{X}$ as a set of pixel coordinates and $\mathcal{Y}$ as the corresponding set of pixel intensities. We let $f_\theta : \mathbb{R}^d \to \mathbb{R}$ be a trained, fully-connected network with L hidden layers of widths $n_1, \ldots, n_L$ and non-linear activation function σ. We let $h_l : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$ be the pre-activation function given at each layer and let $h_{L+1} : \mathbb{R}^L \to \mathbb{R}$ be a single output unit. We write f_θ as

$$f_\theta(\mathbf{x}) = (h_{L+1} \circ \sigma \circ h_L \circ \ldots \sigma \circ h_1)(\mathbf{x})$$

where θ denotes the vector of weights and biases. Further, we let $f_{\theta,0}$ denote the network at initialization and let $f_{\theta,t}$ denote the network after t training epochs. We assume that $f_{\theta,t}$ is

trained via an MSE loss function $\mathcal{L}$:

$$\mathcal{L} = \frac{1}{N}||f(\mathcal{X}) - \mathcal{Y}||_2^2$$

In addition to notation on networks, we introduce notation on kernels. Generically, we define a kernel k as a function $k : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$. Given some dataset $\mathcal{D}$, we typically associate k with two objects: its Gram matrix $\mathbf{K}$, defined as $\mathbf{K}_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$, and its associated integral operator T_k:

$$T_k(f)(\mathbf{x}) = \int_D f(\mathbf{x})k(\mathbf{x}, \mathbf{x}')d\mathbf{x}'$$

where $f(\mathbf{x})$ is a continuous, square-integrable function over some domain D. In this report, we typically assert that k is a Mercer kernel. This means that k is continuous, symmetric and $\mathbf{K}$ is positive semi-definite, given as

$$\mathbf{v}^T\mathbf{K}\mathbf{v} \geq 0 \text{ for all } \mathbf{v} \neq \mathbf{0}$$

When k is a Mercer kernel, k admits a representation as a sum of countably many eigenfunctions $\phi_1, \phi_2, \ldots$ of T_k:

$$k(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x})\phi_i(\mathbf{x}') \text{ where } \lambda_i\phi_i(\mathbf{x}) = T_k(\phi_i)(\mathbf{x})$$

We also typically consider shift-invariant kernels, which depend only on the difference between two inputs and thus satisfy $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}')$, and rotation-invariant kernels, which depend only on the angle between two inputs. In our experiments, we specialize to the RBF kernel k_{RBF} given as

$$k_{\text{RBF}}(\mathbf{x}, \mathbf{x}') = e^{-\frac{||\mathbf{x}-\mathbf{x}'||_2^2}{2\sigma^2}}$$

where $\sigma \in \mathbb{R}$ is called the kernel's length-scale parameter. k_{RBF} is both shift-invariant and a Mercer kernel.

Neural Tangent Kernel

We seek to generalize our study of spectral bias to neural networks of arbitrary depth and arbitrary choice of activation function. Thus, we primarily utilize the neural tangent kernel first outlined in Jacot et al. 2018 [4]. Under certain asymptotic conditions, the NTK provides a general framework for analyzing the outputs and training dynamics of fully-connected networks. Given a network $f_{\theta,t}$ and data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ where $\mathbf{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}$, we define the epoch-indexed empirical NTK $\hat{\Theta}_t$ as follows:

$$\hat{\Theta}_t(\mathbf{x}_i, \mathbf{x}_j) = \nabla_\theta f_{\theta,t}(\mathbf{x}_i)^T \nabla_\theta f_{\theta,t}(\mathbf{x}_j)$$

We may then explicitly describe the evolution of θ_t and $f_{\theta,t}$ under training in terms of $\hat{\Theta}_t$, drawing on results from Lee et al. 2019 [5]:

$$\frac{d\theta_t}{dt} = -\eta \nabla_\theta f_{\theta,t}(\mathcal{X})^T \nabla_{f_{\theta,t}(\mathcal{X})}\mathcal{L}$$

$$\frac{df_{\theta,t}(\mathcal{X})}{dt} = -\eta \hat{\Theta}(\mathcal{X}, \mathcal{X}) \nabla_{f_{\theta,t}(\mathcal{X})}\mathcal{L}$$

where η is the learning rate. Remarkably, if θ is initialized according to a Gaussian distribution, $f_{\theta,t}$ is trained under gradient flow, and the width of each layer of f_θ goes to infinity, then $\hat{\Theta}_t$ converges to the deterministic NTK Θ, given as

$$\Theta(\mathbf{x}_i, \mathbf{x}_j) = \mathbb{E}_{\theta \sim \mathcal{N}}\Big[\nabla_\theta f_\theta(\mathbf{x}_i)^T \nabla_\theta f_\theta(\mathbf{x}_j)\Big]$$

In this infinite-width regime, training $f_{\theta,t}$ on $\mathcal{D}$ is then equivalent to performing kernel regression with Θ. Using this new description of the training dynamics of $f_{\theta,t}$, we may easily tie spectral bias to the eigendecomposition of Θ. Letting $\mathbf{K} \in \mathbb{R}^{N\times N}$ denote Θ's Gram matrix and rescaling $\mathcal{L}$ by a constant, we have

$$\frac{df_{\theta,t}(\mathcal{X})}{dt} \approx -\eta\mathbf{K}(f_{\theta,t}(\mathcal{X}) - \mathcal{Y})$$

Recognizing this as a first-order ordinary differential equation, we may explicitly solve for $f_{\theta,t}(\mathcal{X})$:

$$f_{\theta,t}(\mathcal{X}) \approx (\mathbf{I} - e^{-\eta\mathbf{K}t})\mathcal{Y}$$

In this setting, $\mathbf{K}$ is positive semi-definite, so we then take its spectral decomposition. We let $\mathbf{K} = \mathbf{Q}^T\mathbf{\Lambda}\mathbf{Q}$ where $\mathbf{Q} \in \mathbb{R}^{N\times N}$ is an orthogonal matrix of eigenvectors and $\mathbf{\Lambda} \in \mathbb{R}^{N\times N}$ is a diagonal matrix of eigenvalues. Using the properties of the matrix exponential, we then explicitly define the training error as

$$f_{\theta,t}(\mathcal{X}) - \mathcal{Y} \approx (\mathbf{I} - \mathbf{Q}e^{-\eta\mathbf{\Lambda}t}\mathbf{Q}^T)\mathcal{Y} - \mathcal{Y} = -\mathbf{Q}e^{-\eta\mathbf{\Lambda}t}\mathbf{Q}^T\mathcal{Y}$$

$$\mathbf{Q}^T(f_{\theta,t}(\mathcal{X}) - \mathcal{Y}) \approx -e^{-\eta\mathbf{\Lambda}t}\mathbf{Q}^T\mathcal{Y}$$

As $\mathbf{\Lambda}$ is diagonal, we may decompose this approximation according to the eigenvectors of $\mathbf{K}$:

$$\mathbf{q}_i^T(f_{\theta,t}(\mathcal{X}) - \mathcal{Y}) \approx -e^{-\eta\lambda_i t}\mathbf{q}_i^T\mathcal{Y}$$

for $i \in \{1, \ldots, N\}$. Taking $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$, we note that the speed of convergence along the i-th eigendirection of $\mathbf{K}$ is proportional to the size of λ_i. Here lies spectral bias: $f_{\theta,t}$ learns components of the target function lying along eigendirections with large corresponding eigenvalues faster than those components lying along eigendirections with small corresponding eigenvalues. As high-frequency components correspond to smaller eigenvalues, which happen to decay quickly during training [1], $f_{\theta,t}$ struggles to learn them.

We may partially generalize this result to finite-width networks, following the work of Cao et al. 2019 [2]. We let f_θ be a trained, two-layer ReLU network given as follows:

$$f_\theta(\mathbf{x}) = \sqrt{m} \cdot \mathbf{W}_2 \cdot \max\{0, \mathbf{W}_1\mathbf{x}\}$$

where $\mathbf{W}_1 \in \mathbb{R}^{m\times(d+1)}, \mathbf{W}_2 \in \mathbb{R}^{1\times m}$ represent the weights and biases of the first and second layers, respectively. In this set-up, Θ takes the form of a weighted sum of positive semi-definite arc-cosine kernels, and is thus a positive semi-definite kernel itself. As Θ is also symmetric and continuous, it is a Mercer kernel. Accordingly, Θ admits a representation as a sum of countably many eigenfunctions, which form the eigensystem of its associated integral operator T_Θ:

$$\Theta(\mathbf{x}, \mathbf{x}') = \sum_{j=1}^{\infty} \lambda_i\phi_i(\mathbf{x})\phi_i(\mathbf{x}') \text{ where } \lambda_i\phi_i(\mathbf{x}) = T_\Theta(\phi_i)(\mathbf{x}) = \int_D \Theta(\mathbf{x}, \mathbf{x}')\phi_i(\mathbf{x})'d\mathbf{x}'$$

This result allows the eigendecomposition of Θ to be studied directly, rather than numerically approximated. We further suppose that $\mathcal{X}$ is uniformly distributed over $\mathbb{S}^d$, the unit hypersphere in $\mathbb{R}^d$. We denote the eigensystem of Θ as $\phi_1, \ldots, \phi_N$ where $1 \geq \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$. Letting r_k be the sum of the multiplicity of the first k eigenvalues of Θ, we define $\mathbf{V}_{r_k} \in \mathbb{R}^{N\times r_k}$ where

$$\mathbf{V}_{r_k} = \frac{1}{\sqrt{N}} \begin{bmatrix} \phi_1(\mathbf{x}_1) & \ldots & \phi_{r_k}(\mathbf{x}_1) \\ \vdots & \vdots & \vdots \\ \phi_1(\mathbf{x}_N) & \ldots & \phi_{r_k}(\mathbf{x}_N) \end{bmatrix}$$

More explicitly, $\mathbf{V}_{r_k}$ represents the first r_k eigenfunctions of Θ evaluated on $\mathcal{X}$. Then $||\mathbf{V}_{r_k}^T(\mathcal{Y} - f_\theta(\mathcal{X}))||_2$ approximately represents how well f_θ has learned the components corresponding to

the first r_k eigenvalues of Θ. Under some asymptotic conditions on N, m, we have for any $\varepsilon, \delta > 0$,

$$||\mathbf{V}_{r_k}^T(\mathcal{Y} - f_\theta(\mathcal{X}))||_2 \leq 2(1-\lambda_{r_k})||\mathbf{V}_{r_k}^T\mathcal{Y}||_2 + \varepsilon\sqrt{N}$$

with probability at least $1-\delta$. As $k \to 1$, we note that the upper bound on $||\mathbf{V}_{r_k}^T(\mathcal{Y} - f_\theta(\mathcal{X}))||_2$ tightens, indicating that f_θ has learned the first r_k eigenfunctions well. As $k \to N$, said upper bound loosens, indicating that f_θ hasn't necessarily learned the first r_k eigenfunctions well. Thus, as in the infinite-width setting, f_θ is biased towards learning the target functions' components corresponding to larger eigenvalues.

One might argue that this result outlined in [2], which only bounds the convergence from above, is not especially strong. In restricting $\mathcal{X}$ to $\mathbb{S}^d$, we cannot extend this result to any dataset in $\mathbb{R}^d$; nor can we extend it to networks of arbitrary depth. Critically, however, this upper bound indicates that the link between spectral bias and the NTK extends beyond the infinite-width setting. In order to better document this connection for networks of arbitrary depth, we later turn to numerical computation of the NTK's spectrum.

Random Fourier Features

Before exploring the newfound applications of Random Fourier Feature embeddings outlined in [9], we discuss both their original purpose, as first presented in [7]. RFF embeddings were formulated as a means of scaling up kernel machines – such as support vector machines or kernel ridge regressors – to large quantities of data. They utilize Bochner's Theorem, a classical result in harmonic analysis that enables approximation of continuous, shift-invariant kernels.

Bochner's Theorem. *Let k be a continuous, shift-invariant kernel on $\mathbb{R}^d$. Take $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$ and let $\delta = \mathbf{x} - \mathbf{x}'$. Then k is positive-definite if and only if $k(\mathbf{x}, \mathbf{x}') = k(\delta)$ is the Fourier transform of a non-negative measure ω for all pairs $\mathbf{x}, \mathbf{x}'$.*

Bochner's Theorem thus associates $k(\delta)$ with $p(\omega)$, typically referred to as $k(\delta)$'s spectral distribution or Fourier dual. Defining $\zeta_\omega(\delta) = e^{-i\omega^T\delta}$, we have

$$k(\delta) = \int_{\mathbb{R}^d} \zeta_\omega(\delta)p(\omega)d\omega \qquad p(\omega) = \int_{\mathbb{R}^d} \zeta_\omega^*(\delta)k(\delta)d\delta$$

When $k(\delta)$ is properly scaled, we can assert that $p(\omega)$ is a true probability distribution on $\mathbb{R}^d$. We may then express $k(\delta)$ as an expectation in terms of ω:

$$k(\mathbf{x}, \mathbf{x}') = k(\delta) = \mathbb{E}_{\omega\sim p(\omega)}\Big[\zeta_\omega(\delta)\Big] = \mathbb{E}_{\omega\sim p(\omega)}\Big[\zeta_\omega(\mathbf{x})\zeta_\omega^*(\mathbf{x}')\Big]$$

However, we note that $k(\delta)$ and $p(\omega)$ are both real-valued, whereas $\zeta_\omega(\delta)$ is complex-valued; for our purposes, it is sufficient to consider the real part of $\zeta_\omega(\delta)$. Using some trigonometric identities, we define $r_\omega(\mathbf{x})$ such that $r_\omega(\mathbf{x})r_\omega(\mathbf{x}') = \text{Re}(\zeta_\omega(\mathbf{x})\zeta_\omega^*(\mathbf{x}')) = \cos(\omega^T(\mathbf{x}-\mathbf{x}'))$:

$$k(\mathbf{x}, \mathbf{x}') = k(\delta) = \mathbb{E}_{\omega\sim p(\omega)}\Big[r_\omega(\mathbf{x})r_\omega(\mathbf{x}')\Big]$$

We may then approximate $k(\mathbf{x}, \mathbf{x}') = k(\delta)$ by drawing m random samples from $p(\omega)$:

$$k(\mathbf{x}, \mathbf{x}') \approx \frac{1}{m}\sum_{j=1}^{m} r_{\omega_j}(\mathbf{x})r_{\omega_j}(\mathbf{x}')$$

This approximation can be written as a random embedding of $\mathbf{x}, \mathbf{x}'$ into m-dimensional space such that. Given some set of random vectors $\omega_1, \ldots, \omega_m$ sampled from $p(\omega)$, we define $z(\mathbf{x}) \in \mathbb{R}^m$ as

$$z(\mathbf{x}) = \left[\frac{r_{\omega_1}(\mathbf{x})}{\sqrt{m}}, \ldots, \frac{r_{\omega_m}(\mathbf{x})}{\sqrt{m}}\right]^T \text{ such that } k(\mathbf{x}, \mathbf{x}') \approx z(\mathbf{x})z(\mathbf{x}')$$

For a data set $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, we construct matrix $\mathbf{Z} \in \mathbb{R}^{N\times m}$ defined as $\mathbf{Z}_{i,j} = z(\mathbf{x}_i)_j$. Then we may approximate the Gram matrix $\mathbf{K} \in \mathbb{R}^{N\times N}$ corresponding to k as $\mathbf{K} \approx \mathbf{Z}\mathbf{Z}^T$. We may view this procedure as finding a 'nice' random higher-dimensional embedding of $\mathcal{D}$ – specifically, one where evaluating a kernel in $\mathbb{R}^d$ at two points is roughly equivalent to dotting said points embedded in $\mathbb{R}^m$.

In RFF's original setting in [7], approximating $\mathbf{K}$ as $\mathbf{Z}\mathbf{Z}^T$ significantly reduces computation and storage costs when $N \gg 0$. Storing $\mathbf{K}$ requires $\mathcal{O}(N^2)$ space, whereas storing $\mathbf{Z}$ requires $\mathcal{O}(Nm)$ space; similarly, computing $\mathbf{K}$ directly requires $\mathcal{O}(N^3)$ computations whereas computing $\mathbf{Z}\mathbf{Z}^T$ requires $\mathcal{O}(Nm^2)$ computations. If we take $m > d$ but $m \ll N$, then this provides a significant improvement. Further, the approximation decreases exponentially in m, so it is usually sufficient to consider a 'small' number of random features.

Applications to Low-Dimensional Tasks

Closely following the work presented in [9], we repurpose RFF embeddings as a means of overcoming spectral bias, thereby allowing fully-connected networks fo 'perfectly' learn all components of a target function. Rather than passing our training samples $\mathcal{X} \in \mathbb{R}^{N\times d}$ directly into our network f_θ, we choose some $p(\omega)$, draw m samples $\omega_1, \ldots, \omega_m$ from $p(\omega)$, and pass $\gamma(\mathcal{X}) \in \mathbb{R}^{N\times 2m}$ into f_θ, where $\gamma(\mathcal{X})$ is defined as follows:

$$\gamma(\mathbf{x}) = \frac{1}{\sqrt{m}}\begin{bmatrix}\cos(\mathbf{B}\mathbf{x})\\ \sin(\mathbf{B}\mathbf{x})\end{bmatrix} \text{ and } \gamma(\mathcal{X}) = \frac{1}{\sqrt{m}}\begin{bmatrix}\cos(\mathbf{B}\mathcal{X})\\ \sin(\mathbf{B}\mathcal{X})\end{bmatrix}$$

where the rows of random matrix $\mathbf{B} \in \mathbb{R}^{m\times d}$ are $\omega_1, \ldots, \omega_m$. While not immediately obvious, we note that $\gamma(\mathbf{x})^T\gamma(\mathbf{x}')$ is an approximation for $k(\mathbf{x}, \mathbf{x}')$:

$$\begin{aligned}
\gamma(\mathbf{x})^T\gamma(\mathbf{x}') &= \frac{1}{m}\sum_{\ell=1}^{m} \cos(\omega_\ell^T\mathbf{x})\cos(\omega_\ell^T\mathbf{x}') + \sin(\omega_\ell^T\mathbf{x})\sin(\omega_\ell^T\mathbf{x}')\\
&= \frac{1}{m}\sum_{\ell=1}^{m} \cos(\omega_\ell^T(\mathbf{x}-\mathbf{x}'))\\
&= \frac{1}{m}\sum_{\ell=1}^{m} r_{\omega_\ell}(\mathbf{x})r_{\omega_\ell}(\mathbf{x}')\\
&\approx k(\mathbf{x}, \mathbf{x}')
\end{aligned}$$

Under the infinite-width regime, in which $\hat{\Theta}$ converges to Θ, training f_θ on the randomly-embedded dataset $\mathcal{D}_\gamma = \{\gamma(\mathcal{X}), \mathcal{Y}\}$ is approximately equivalent to composing Θ with k:

$$\Theta(\gamma(\mathbf{x})^T\gamma(\mathbf{x}')) \approx \Theta(k(\mathbf{x}, \mathbf{x}'))$$

This embedding yields two primary advantages. First, it allows us to explicitly tune the predictions of f_θ; in selecting both k and its parameters (such as length scale for the RBF kernel), we may probabilistically tune $\mathbf{B}$ and thus tune $\gamma(\mathcal{X})$.

Secondly, it ensures that the composed kernel $\Theta \circ k$ is shift-invariant as $k(\mathbf{x}, \mathbf{x}')$ is a function of $\mathbf{x} - \mathbf{x}'$. Θ is itself a dot-product kernel, ensuring that it is rotation-invariant but not necessarily shift-invariant. However, for the sake of pixel regression, we would like to model all components of an image regardless of position; thus shift-invariance is a key property.

Our complete procedure is this: we pick some parameterized shift-invariant kernel k, find its

spectral density $p(\omega)$, draw m samples $\omega_1, \ldots, \omega_m$, construct $\mathcal{D}_\gamma = \{\gamma(\mathcal{X}), \mathcal{Y}\}$ and then train f_θ on $\mathcal{D}_\gamma$. In our following experiments, we specialize to the RBF kernel k_{RBF} with length-scale σ: its smoothness ensures that f_θ's predictions are 'nice' and its corresponding spectral density $p(\omega)$ is $\mathcal{N}(0, \sigma^{-2}\mathbf{I})$, an isotropic Gaussian. This simplifies our sampling procedure, as the components of $\mathbf{B}$ are independent and identically distributed: $\mathbf{B}_{i,j} \sim \mathcal{N}(0, \sigma^{-2}\mathbf{I})$.

Experiments

We conduct two series of computational experiments to examine the efficacy of RFF embeddings in overcoming spectral bias in low-dimensional settings. We first train several networks to fit 20 training points drawn from a pre-determined, high-frequency function $f : \mathbb{R} \to \mathbb{R}$ given by

$$f(x) = \sin(2\pi x) + \cos(7\pi x) + \frac{\sin(12\pi x)}{3} + \frac{\cos(15\pi x)}{2}$$

To ensure meaningful comparisons between experiments, we fix the learning rate η, the number of random features m and the architecture of f_θ: we set $\eta = 0.01$, $m = 5$, and let f_θ be a fully-connected ReLU network with four hidden layers, each with 64 hidden units. We also fix the batch size as 1.

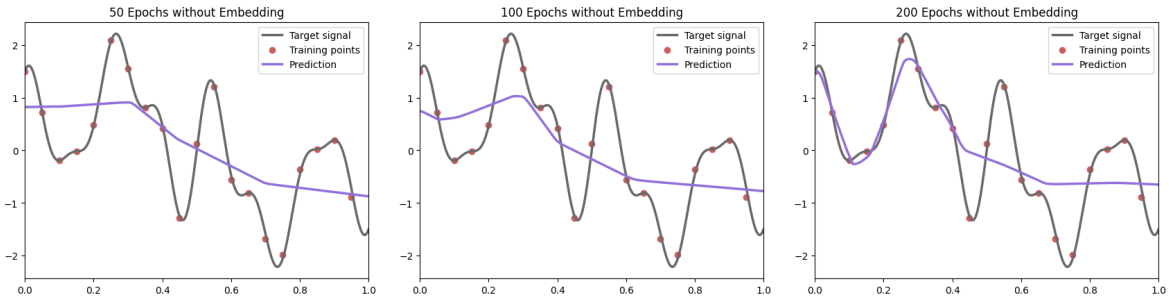

Figure 1: Embedding-free network's prediction at various training stages. The network fails to learn the high-frequency components of f, and struggles to interpolate even after 200 training epochs.

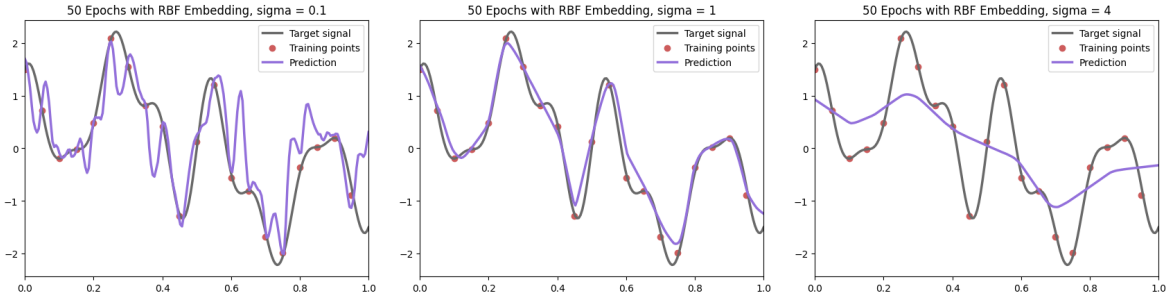

Figure 2: Predictions of networks with embeddings given by different σ. When σ is well-tuned, the network accurately reconstructs f. When σ is too high, it underfits with a low-frequency prediction; when σ is too low, it overfits with a high-frequency prediction.

We then perform pixel regression on a predetermined 512-by-512 image with a five different networks: one without an RFF embedding, and four with an embedding, where $\sigma \in \{5, 1, 0.3, 0.04\}$. We fix $\eta = 0.01$, $m = 10$, and f_θ to be a fully-connected ReLU network with four hidden layers,

each with 256 hidden units. We train each network for 100 epochs and increase the batch size to 1024, reflecting the size of $\mathcal{D}$ and $\mathcal{D}_\gamma$.

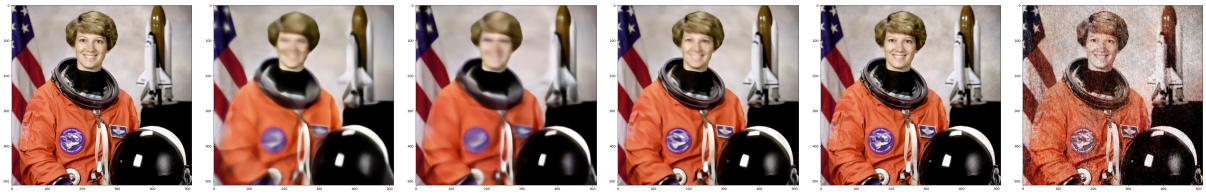

Figure 3: From left to right: (1) original 512 × 512 image, (2) prediction without embedding, (3) prediction with embedding, $\sigma = 5$, (4) $\sigma = 1$, (5) $\sigma = 0.3$, and (6) $\sigma = 0.04$.

Across both sets of experiments, we observe that the embedding-free networks fail to capture the high-frequency components in the target function. We also observe that the quality of the prediction given by the networks trained on RFF-embedded data rests on the choice of σ. When σ is too high, these networks underfit the data, yielding a prediction similar to that of the embedding-free networks. Conversely, when σ is too low, these networks appear to learn high-frequency noise in the data, thus overfitting the data. In the 'Goldilocks Zone' between these extremes, however, these networks appear to perfectly reconstruct the target function.

Spectral Analysis

We explore how RFF embeddings overcome spectral bias by examining the spectrum of the composed kernel $\Theta \circ k_{\text{RBF}}$ for different values of length-scale σ. Analytically representing the spectrum and eigenfunctions of a network of arbitrary depth is not straightforward; thus we restrict our analytical study to two-layer bias-free networks, and treat larger networks solely through empirical study. Our networks take the form

$$f_\theta(\mathbf{x}) = \frac{1}{\sqrt{m}} \mathbf{W} \begin{bmatrix} \cos(\mathbf{B}\mathbf{x}) \\ \sin(\mathbf{B}\mathbf{x}) \end{bmatrix}$$

where $\mathbf{W} \in \mathbb{R}^{1 \times 2m}$ is a matrix of weights and m is the number of random features. We may then explicitly describe the composed empirical NTK $\hat{\Theta} \circ k_{\text{RBF}}$'s associated Gram matrix $\hat{\mathbf{K}}$ as

$$\hat{\Theta} \circ k_{\text{RBF}}(\mathbf{x}_i, \mathbf{x}_j) \approx \hat{\mathbf{K}}_{i,j} = \frac{1}{m} \begin{bmatrix} \cos(\mathbf{B}\mathbf{x}_i) \\ \sin(\mathbf{B}\mathbf{x}_i) \end{bmatrix}^T \begin{bmatrix} \cos(\mathbf{B}\mathbf{x}_j) \\ \sin(\mathbf{B}\mathbf{x}_j) \end{bmatrix} = \frac{1}{m} \sum_{\ell=1}^{m} \cos(\mathbf{b}_\ell^T(\mathbf{x}_i - \mathbf{x}_j))$$

If we return to the infinite-width regime outlined in [5], under which $\hat{\Theta}$ converges to Θ, then the eigensystem of $\hat{\mathbf{K}}$ converges to that of $\mathbf{K}$, which is equivalent to the eigensystem of the familiar Hilbert-Schmidt operator T_Θ, whose eigenfunctions $g(\mathbf{x})$ satisfy $\lambda g(\mathbf{x}) = T_\Theta(g)(\mathbf{x})$. We then may re-write the eigenfunctions of $T_{\Theta \circ k_{\text{RBF}}}$ in terms of their Laplacian, using an identity defined in [10]:

Eigenfunctions of $T_{\Theta \circ k_{\text{RBF}}}$. *The eigenfunctions $g(\mathbf{x})$ of $T_{\Theta \circ k_{RBF}}$ corresponding to non-zero eigenvalues satisfy the following equation:*

$$\Delta g(\mathbf{x}) = -\frac{1}{m} ||\mathbf{B}||_F^2 g(\mathbf{x})$$

If we further restrict our study to $m = 1, d = 1$ and consider $x_1, \ldots, x_N$ drawn from [0, 1], we can explicitly describe the spectrum of $\Theta \circ k_{\text{RBF}}$. Recalling that $p(\omega) = \mathcal{N}(0, \sigma^{-2}\mathbf{I})$, we note

that $\mathbf{B} = b \in \mathbb{R}$ where $b \sim \mathcal{N}(0, \sigma^{-2})$ and $\Theta \circ k_{\text{RBF}}(x, x') = \cos(b(x - x'))$. The Laplacian expression reduces to

$$\frac{\partial^2 g(x)}{\partial x^2} = -b^2 g(x)$$

which enables straightforward calculation of the eigenfunctions $g(x)$ by solving both this second-order ordinary differential equation and the Laplacian eigenvalue equation given by $T_{\Theta \circ k_{\text{RBF}}}$. Specifically, the non-zero eigenvalues of $\Theta \circ k_{\text{RBF}}(x, x')$ are

$$\lambda_1 = \frac{1 + \frac{\sin b}{b}}{2} \text{ and } \lambda_2 = \frac{1 - \frac{\sin b}{b}}{2}$$

Here, we explicitly study the effects of changing σ. As $\sigma \to 0$, the variance of $p(\omega)$ increases. Then we increase the probability that $\frac{\sin b}{b}$ is close to 0, thereby increasing the probability that λ_1, λ_2 are very close to $\frac{1}{2}$. Thus, by decreasing σ, we probabilistically compress the spectrum of $\Theta \circ k_{\text{RBF}}$. As noted in our earlier discussion of the infinite-width case, f_θ learns the eigendirections of a target function at a rate proportional to their corresponding eigenvalues. By compressing the spectrum of $\Theta \circ k_{\text{RBF}}$, we appear to ensure that all eigendirections are learned at approximately the same rate, thereby overcoming spectral bias.

This theoretical result appears to validate the empirical results from our first set of experiments with $f : \mathbb{R} \to \mathbb{R}$. We observed that, as $\sigma \to 0$, the frequency of the network f_θ's prediction on $[0, 1]$ increased, indicating that σ tuned f_θ's ability to learn the high-frequency components of f.

To examine the effect of varying σ, we turn to numerical estimation of the spectrum of $\hat{\Theta} \circ k_{\text{RBF}}$ in the finite-width setting. We sample fifty equally-spaced points from the target function $f : \mathbb{R} \mapsto \mathbb{R}$ used in our first round of experiments. We fix our network f_θ as a fully-connected ReLU-based network with four hidden layers, each with 64 units; we then compute $\hat{\mathbf{K}} \in \mathbb{R}^{50 \times 50}$ and take its eigendecomposition. We do this procedure once for f_θ with no RFF embedding (i.e. f_θ is trained directly on $\mathcal{D}$) and three times for f_θ with an RFF embedding where $\sigma \in \{5, 1, 0.1\}$ and $m = 5$.

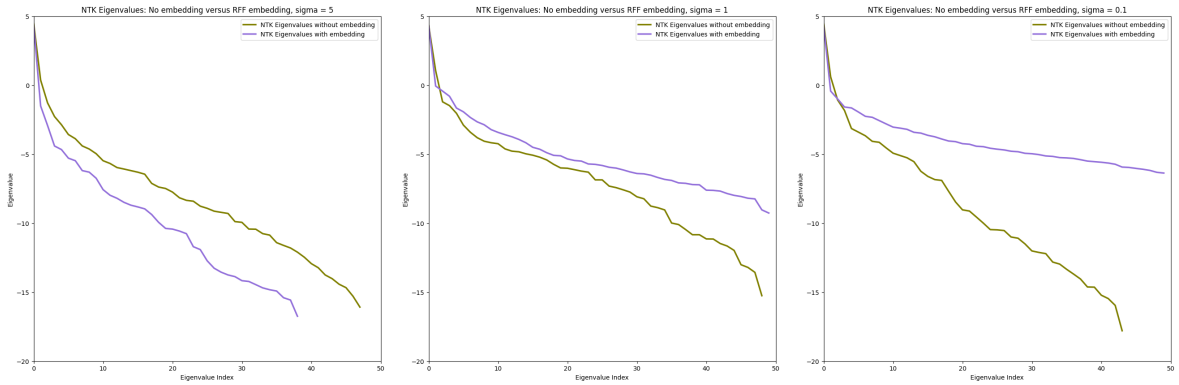

Figure 4: Eigenvalues of four networks, plotted on a logarithmic scale. When an RFF embedding is used and $\sigma \to 0$, the eigenvalues of f_θ are compressed to a smaller range.

This final set of experiments suggests that the analytical result for two-layer networks where $m = 1, d = 1$ extends to networks of arbitrary depth. As $\sigma \to 0$, the eigenvalues of $\hat{\mathbf{K}}$ become compressed to a smaller range, suggesting that all components of our target function f are learned equally quickly.

Conclusion

We demonstrated that spectral bias can be ascribed to the eigendecomposition of the neural tangent kernel in both infinite- and finite-width settings. Specifically, we showed that low-frequency components in a target function correspond to larger NTK eigenvalues, and are thus learned quickly, while high-frequency components correspond to smaller NTK eigenvalues and are learned slowly.

We discussed how passing training samples through an RFF embedding overcomes spectral bias in low-dimensional settings. Further, we showed how networks' outputs can be explicitly tuned by changing k and its parameters. Finally, we tied the efficacy of RFF embeddings to their compressive effect on the NTK spectrum through computation and direct analysis.

Further research is needed to extend the theoretical links between spectral bias, the NTK and RFF embeddings. Many results tying spectral bias to the NTK make theoretically necessary assumptions that don't generalize to arbitrary networks – for instance, that $\mathcal{X}$ is distributed on $\mathbb{S}^d$ or that f_θ is a two-layer network. Additionally, we lack analytical results regarding the effect of RFF embeddings on the NTK spectrum for $m, d > 1$ and for non-RBF kernels.

References

[1] Ronen Basri, David W. Jacobs, Yoni Kasten, and Shira Kritchman. The convergence rate of neural networks for learned functions of different frequencies. *CoRR*, abs/1906.00425, 2019.

[2] Yuan Cao, Zhiying Fang, Yue Wu, Ding-Xuan Zhou, and Quanquan Gu. Towards understanding the spectral bias of deep learning. *CoRR*, abs/1912.01198, 2019.

[3] Zekun Hao, Arun Mallya, Serge Belongie, and Ming-Yu Liu. Implicit neural representations with levels-of-experts. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 2564–2576. Curran Associates, Inc., 2022.

[4] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *CoRR*, abs/1806.07572, 2018.

[5] Jaehoon Lee, Lechao Xiao, Samuel S Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent *. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(12):124002, December 2020.

[6] 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. *CoRR*, abs/2003.08934, 2020.

[7] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.

[8] Sameera Ramasinghe and Simon Lucey. Beyond periodicity: Towards a unifying framework for activations in coordinate-mlps. *CoRR*, abs/2111.15135, 2021.

[9] 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. *CoRR*, abs/2006.10739, 2020.

[10] Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of fourier feature networks: From regression to solving multi-scale pdes with physics-informed neural networks. *CoRR*, abs/2012.10047, 2020.