Generalizability of Neural Networks with a Fourier Feature Embedding

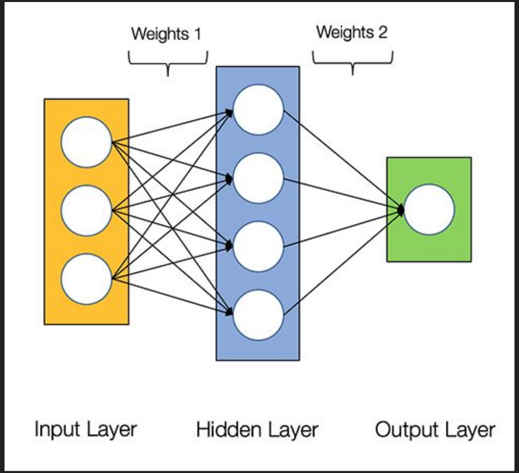
Nithin Raghavan
Richard Hu
Alberto Checcone

Background

- Why do we measure the generalizability of a neural network?
    - We want to measure the training accuracy and testing accuracy over time.
    - 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.
    - This can also measure the richness of the class of functions that a neural network can learn.
        - 3D shape regression, 2D image regression, CT, MRI, etc.
        - Occupancy networks

Background

- Why measure Neural Network generalizability?
  - Relates training accuracy to test accuracy
  - Less complex = more generalizable



Background

- How do we measure the generalizability of a neural network?
  - Empirical Rademacher complexity directly gives an upper bound on generalization error
  - Rademacher complexity can give us an easily verifiable measure that can differentiate between true labels and random labels.
  - Using the neural tangent kernel, we can obtain closed-form bounds for the Rademacher complexity over training and testing set for neural networks.

Background

- How is generalizability measured?
    - Rademacher complexity upper bounds generalization error.
    - Allows for NN optimization.

What is a neural tangent kernel?

- The Neural Tangent Kernel (NTK) is a kernel function that describes the evolution of artificial neural networks during training using gradient descent.
- The NTK tools additionally lend themselves useful when using a 1-Lipschitz loss function.

$$\Theta\left(x, y; \theta\right) = \sum_{p=1}^{P} \partial_{\theta_p} f\left(x; \theta\right) \partial_{\theta_p} f\left(y; \theta\right).$$

Neural Networks and the Neural Tangent Kernel

- 2-layer Neural Networks reduce to kernelized ridge regression

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

What is a neural tangent kernel?

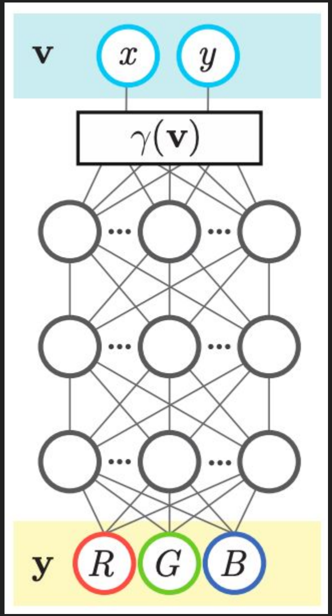
- A kernel function $k(x, y)$ is a positive-definite function that measures the similarity between vectors x and y.
    - Dot product kernels are of the form $k(x, y) = \Phi(x) \cdot \Phi(y)$, where Φ is a lifting into some different feature space.
- A Gram matrix H is an n by n Hermitian matrix such that $H_{i,j} = k(x_i, x_j)$, where k is a kernel
    - n is the number of sample points.
- Overview:
    - Generalize bound on NN
    - NN at infinite epochs → kernel RR with no ridge (where kernel = NTK)
    - Nithin paper => can use four yay feet to modify NTK
    - Note that generalizability → small values of projections of labels onto NTK eigenvectors
    - So we can create bound 2 with good B

Kernels

- The kernel function $k(x, y) := \Phi(x) \cdot \Phi(y)$ for some Φ
- The (Positive Definite) Gram matrix is $H_{i,j} = k(x_i, x_j)$

Coordinate-Based MLP

- MLPs with a ReLU activation function taking in points from $\mathbb{R}^d$
- Input is generally very low dimensional
- Want to learn labels



Fourier Feature embedding

- Coordinate-based multi-layer perceptrons
  - Feed-forward neural networks that only take in the coordinates of an input, instead of the entire input itself
- A Fourier Feature embedding is an embedding γ(v) of the input to a coordinate-based multi-layer perceptron of the following form:

$$\gamma(\mathbf{v}) = \left[a_1 \cos(2\pi \mathbf{b}_1^{\mathrm{T}}\mathbf{v}), a_1 \sin(2\pi \mathbf{b}_1^{\mathrm{T}}\mathbf{v}), \ldots, a_m \cos(2\pi \mathbf{b}_m^{\mathrm{T}}\mathbf{v}), a_m \sin(2\pi \mathbf{b}_m^{\mathrm{T}}\mathbf{v})\right]^{\mathrm{T}}$$

Fourier Feature embedding

- Low feature space = MLPs can't approximate well
- Idea: lift features onto surface of hypersphere
- A Fourier Feature embedding γ(v) defined below:
    - $B = [b_1,...,b_m]$ and m are hyperparameters
    - a_i's are constant with $||a|| = 1$

$$\gamma(\mathbf{v}) = \left[a_1 \cos(2\pi \mathbf{b}_1^{\mathrm{T}} \mathbf{v}), a_1 \sin(2\pi \mathbf{b}_1^{\mathrm{T}} \mathbf{v}), \ldots, a_m \cos(2\pi \mathbf{b}_m^{\mathrm{T}} \mathbf{v}), a_m \sin(2\pi \mathbf{b}_m^{\mathrm{T}} \mathbf{v})\right]$$

Why use a Fourier Feature embedding?

- MLPs with ReLU have a “wide” kernel that leads to over-smoothing.
- Necessary to learn high frequency functions
- Fourier feature embedding allows modification of NTK kernel matrix

What is a Fourier Feature embedding?

- Why use a Fourier Feature embedding?
  - Recent advances in NTK theory have shown that MLPs with ReLU have a “wide” kernel that leads to “over-smoothing”.
  - Useful in high-dimensional tasks in order to prevent overfitting In low dimensions
    - Such an embedding is necessary to learn high dimensional features
  - We would like to measure the generalizability of coordinate-based neural networks using the Fourier Feature embedding with a fixed sample size, but varying the number of allowed frequencies.

Basic Rademacher complexity bound

- Observation: norm of a vector passed through Fourier Feature embedding is bounded by the number of frequencies:

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

- First idea: create weak bound on generalizability using only number of frequencies

Basic Rademacher complexity bound

- To conduct this kind of analysis, we consider the class of neural networks whose weights are bounded by some constant (usually the case in practice)

$$\mathcal{H}' \doteq \{f_\Theta : \|w\|_2 \leq B_2', \|u_j\|_2 \leq B_2 \;\; \forall j = [m]\}$$

- Where the neural network itself is a 2-layer ReLU network defined as:

$$f_\Theta = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} w_i \phi(u_i^\top x)$$

Basic Rademacher complexity bound

- After some analysis, we concluded that

$$\mathrm{Rad}_S(\mathcal{H}') \leq 2B_2B_2'\sqrt{\frac{r}{n}}$$

Advancements in NTK theory

- Arora, et al. (2019)
  - Relates generalizability to NTK eigendecomposition

$$\sqrt{\frac{2\mathbf{y}^\top \left(\mathbf{H}^\infty\right)^{-1}\mathbf{y}}{n}}$$

$$y^\top (H^\infty)^{-1} y = y^\top \left(\sum_{i=1}^{n} \frac{1}{\lambda_i} v_i^\top v_i y \right) = \sum_{i=1}^{n} \frac{(v_i^\top y)^2}{\lambda_i}$$

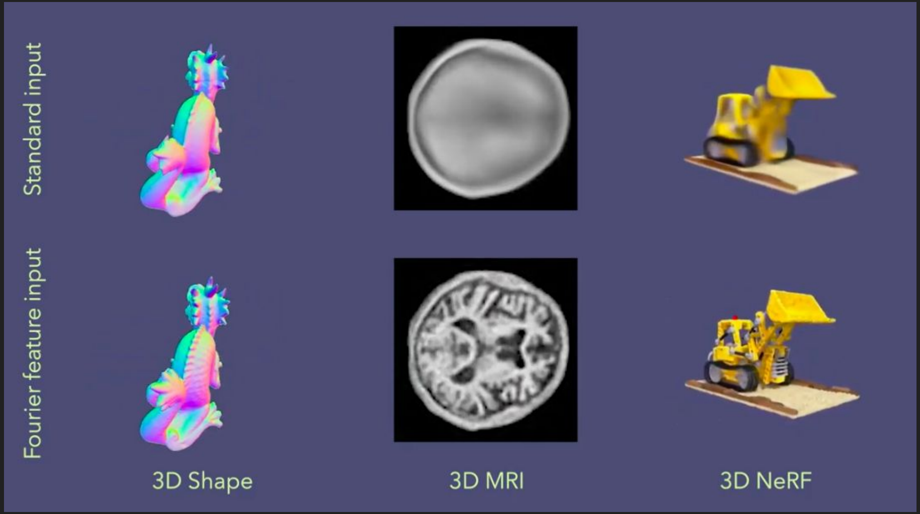
Why are NTKs useful here?

- Arora, et al. (2019) found that the generalizability during training of a neural network follows this (where v_i refers to the eigenvectors of the NTK):

$$\sum_{i=1}^{n} \frac{(v_i^\top y)^2}{\lambda_i}$$

Relating Fourier Feature embedding to the NTK

- NTK matrix modifiable using Fourier Feature embedding
- How do we quantify this?



Why are NTKs useful here?

- This implies that if we modify the NTK of the neural network, we can directly control the eigenvalues of the NTK, and therefore control the generalizability of the network as a whole.
- Two aspects: More frequencies means that the network is able to quickly learn high dimensional information, but also changing the b-values implies that different frequencies in the training set can be learned faster.
- How do we quantify this?

Better Rademacher complexity bound

- Arora, et al. 2019 showed NTK for a two-layer neural network is as follows:

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

- Which bounds the Rademacher complexity:

$$\mathcal{R}_S\left(\mathcal{F}_{R,B}^{\mathbf{W}(0),\mathbf{a}}\right) \leq \frac{B}{\sqrt{2n}}\left(1 + \left(\frac{2\log\frac{2}{\delta}}{m}\right)^{1/4}\right) + \frac{2\sqrt{2}R^2\sqrt{m}}{\sqrt{\pi}\kappa} + R\sqrt{2\log\frac{2}{\delta}},$$

Better Rademacher complexity bound

- We obtain the following derivation:

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

Better Rademacher complexity bound

- Neural Tangent Kernel theory was used here to obtain a testing generalization bound of (where H^∞ is the NTK Gram matrix)

$$\sqrt{\frac{2y^\top (H^\infty)^{-1} y}{n}}$$

- Fourier Feature embedding reduces equation on previous slide to:

$$\sqrt{\frac{2}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j h_{\mathrm{NTK}}(\gamma(x_i)^\top \gamma(x_j))^{-1}}$$

Next steps: Measuring Empirical Generalizability

- Convert inputs to be uniformly sampled over $[0, 1)^d$ forms spherical convolution over input space
  - i.e., 2-D occupancy network, image memorization
- Implies eigenvectors of Gram matrix form DFT matrix

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

Next steps: Rademacher bound optimization

- Use experiments to calculate which b values achieve best bound
- Quantify importance of b values through occupancy networks

Next steps: Rademacher bound of RKHS norm

- Another idea: NNs trained using gradient descent are in the span of the training data in a reproducing kernel Hilbert space (RKHS) induced by the NTK.

Next steps: Rademacher bound of RKHS norm

- Li, et al. (2020) finds the complexity of functions evaluated with an RKHS-norm converging to kernel ridge regression
  - Much harder than the other path
- We can then apply Rademacher analysis on the kernel ridge regression problem using the RKHS norm, and using FF embedding

$$\left\| \hat{f}_l - \hat{f}_{l-1} \right\|_{\mathcal{H}}$$

$$\hat{\mathfrak{R}}_{v^l}\left(\mathcal{L}(\hat{f}_{l-1}, D; f_l, B_l)\right) \leq \frac{DR}{\sqrt{M}}.$$