18.408 Project

Shawn Im

May 2021

1 Introduction

One of the big problems of deep learning is understanding why deep neural networks work well, and one surprising result is that having overparameterized models can reach zero error and still generalize well, and also that as width increases, training causes increasingly smaller changes to the parameters which leads to the idea of a limiting behavior as the width goes to infinity. This idea of studying the infinite width limit of a neural network is the main idea behind Neural Tangent Kernels.

Neural Tangent Kernels are kernels that describe the behavior of a infinitely wide neural network over training with gradient descent, and this kernel is fixed in the infinite width limit so the same kernel will describe a neural network at any point of training. Some of the nice properties of the NTK are that it only depends on the depth, the nonlinearity, and initialization variance of the neural network, so in a way, it provides a simple view of neural networks and generalizes their behavior, opening up ways to understand properties of how neural networks learn.

However, one of the limiting factors of the NTK was that it considers neural networks in the infinite width limit and being able to compute or understand something with infinite width can be difficult, and understanding how it connects to finite networks which are the ones that are used in real application. So how can we connect the NTK to finite width networks and what do NTKs tell us about neural networks?

There are multiple ways this question can be answered. One direct way is actually showing that a sufficiently wide neural network has the same generalization properties as the NTK by showing equivalence. This is the way that Arora et al. [1] approaches the question, and the paper also introduces a more intuitive formula to the CNTK, the convolutional NTK, that allows for easier understanding and better computation. Another way of approaching this question is by using the eigendecomposition of the NTKs of neural networks to understand properties about the functions that they learn, and this idea is explored in Yang and Salman's [2] paper. In this report, we will be reviewing the results of these two papers and what kind of questions looking at them together opens up.

2 Finite Width Deep Nets

Let us formally introduce NTKs before going more into the results found on them. First, let us define a neural network using notation from the Arora et al. [1] paper as

$$g^{(0)}(x) = x$$

$$f^{(h)}(x) = W^{(h)} g^{(h-1)}(x)$$

$$g^{(h)}(x) = \sqrt{\frac{c_\sigma}{d_h}}\sigma(f^{(h)}(x))$$

for h from 1 to L where L is the number of hidden layers, d_h is the width of the hth hidden layer, $W^{(h)}$ is a $d_h \times h_{h-1}$ real matrix and σ is the nonlinear activation function, and $c_\sigma = (E_{z\sim\mathcal{N}(0,1)}[\sigma(z)^2])^{-1}$, and

$$f(\theta, x) = f^{(L+1)}(x)$$

is the output of the neural network where θ is the collection of all parameters.

Now, we will define a few more things that will allow us to define an NTK. First we will define the covariance of the Gaussian process that the pre-activations tend to in the infinite width limit:

$$\Sigma^{(0)}(x, x') = x^\top x'$$

$$\Lambda^{(h)}(x, x') = \begin{bmatrix} \Sigma^{(h-1)}(x, x) & \Sigma^{(h-1)}(x, x') \\ \Sigma^{(h-1)}(x', x) & \Sigma^{(h-1)}(x', x') \end{bmatrix}$$

$$\Sigma^{(h)}(x, x') = c_\sigma E_{(u,v)\sim\mathcal{N}(0,\Lambda^{(h)})}[\sigma(u)\sigma(v)]$$

$$\dot{\Sigma}^{(h)}(x, x') = c_\sigma E_{(u,v)\sim\mathcal{N}(0,\Lambda^{(h)})}[\dot{\sigma}(u)\dot{\sigma}(v)]$$

$$\dot{\Sigma}^{(L+1)}(x, x') = 1$$

$\Sigma^{(h)}$ is the covariance of the hth hidden layer, and $\dot{\Sigma}^{(h)}$ is the derivative covairance of the hth hidden layer, and these allow us to define the NTK which is

$$\Theta^{(L)}(x, x') = \sum_{h=1}^{L+1}(\Sigma^{(h-1)}(x, x') \prod_{h'=h}^{L+1} \dot{\Sigma}^{(h')}(x, x'))$$

For finite width networks, the function

$$\left\langle \frac{\partial f(\theta, x)}{\partial \theta}, \frac{\partial f(\theta, x')}{\partial \theta} \right\rangle$$

serves as an approximation to $\Theta^{(L)}(x, x')$. We would expect and know that this approximation tends to the NTK when the width goes to infinity, but what about how close it is when the width is large but finite? It turns out that with a sufficiently large finite width, the approximation can get arbitrarily close to the NTK at initialization. This is the first result of the Arora et al. paper. The theorem is as follows [1]:

For any $\epsilon > 0$ and $\delta \in (0, 1)$, let $sigma(x)$ is the ReLU function, and $\min_{h\in[L]} d_h \geq \Omega(\frac{L^6}{\epsilon^4}\log(L/\delta))$. Then for any inputs $x, x' \in R^{d_0}$ such that $\|x\| \leq 1$, $\|x'\| \leq 1$, with probability at least $1 - \delta$

$$\left| \left\langle \frac{\partial f(\theta, x)}{\partial \theta}, \frac{\partial f(\theta, x')}{\partial \theta} \right\rangle - \Theta^{(L)}(x, x') \right| \leq (L + 1)\epsilon$$

Although the theorem does provide a non-asymptotic result, one of the problems that could occur with using a finite width deep net to approximate the NTK is that in order to be off by about ϵ, the width has to be at least in the order of $1/\epsilon^4$, so it would be difficult to use a finite width deep net as an approximation with a small error. However, the theorem does provide a much stronger results than previous results which were asymptotic, and having such a theorem may be useful to getting new or stronger results about the NTK or how finite width neural networks behave.

The paper continues to strengthen previous results by giving another non-asymptotic result that shows that after training, a neural net again with sufficiently large width and with all layers

having the same width with scaled output is equivalent to kernel regression with the NTK. In order to write the theorem, we first need to define $\{(x_i, y_i)\}_{i=1}^n$ as the training data and

$$f_{nn}(\theta, x) = \kappa f(\theta, x)$$
$$f_{nn}(x_{te}) = \lim_{t\to\infty} f_{nn}(\theta(t), x_{te})$$

where $\kappa > 0$ is small, and x_{te} is a test point and

$$f_{ntk}(x_{te}) = (\ker_{ntk}(x_{te}, X))^\top (H^*)^{-1} y$$

where H^* is an $n \times n$ matrix where $H*_{i,j} = \Theta^{(L)}(x_i, x_j)$ and $\ker_{ntk}(x_{te}, X)$ is the NTK evaluated on x_{te} and each of the training points. Lastly, let λ_0 be the minimum eigenvalue of H^*. Then we have the following theorem [1]:

Suppose that $\sigma(x)$ is the ReLU function, $1/\kappa = (1/\epsilon, \log(n/\delta))$ and $d_1 = d_2 = \cdots = d_L = m$ with $m \geq (1/\kappa, L, 1/\lambda_0, n, \log(1/\delta))$. Then, for any $x_{te} \in R^d$ with $\|x_{te}\| = 1$, with probability at least $1 - \delta$ over random initialization,

$$|f_{nn}(x_{te}) - f_{ntk}(x_{te})| \leq \epsilon$$

Now, we not only have that a sufficiently wide neural network with finite width converges to the NTK at initialization but also when trained, is equivalent to kernel regression with the NTK. With this, we now have that these wide neural networks have the same generalization properties as the NTK and understanding one would mean understanding the other.

3 Neural Tangent Kernel Eigenvalues

One of the ways that people have studied the generalization properties of neural networks is by studying the eigenvalues of the corresponding NTK. A paper by Yang et al. [2] studies how these eigenvalues can tell us about any simplicity bias, a bias towards learning simple functions, in a neural network. This paper focuses on functions over the boolean cube and shows that the simplicity bias does not always exist and that any bias depends on the depth of a network, the nonlinearity, and the sampling variance which are the properties that determine the NTK. [2]

We can study these eigenvalues of the CK or NTK and how they affect the functions that a model learns by taking the eigendecomposition of a kernel

$$K = \sum_{i\geq 1} \lambda_i u_i \otimes u_i$$

where λ_i are the eigenvalues in decreasing order and u_i are the corresponding eigenfunctions. Then, each sample from the Gaussian process from the kernel can be written as

$$\sum_{i\geq 1} \sqrt{\lambda_i}\omega_i u_i$$

where $\omega_i \sim \mathcal{N}(0, 1)$, and in the case of the CK or NTK, the neural network (at initialization or after training respectively) is a sample from this Gaussian process in the infinite width limit. Then, if one of the eigenvalues is much larger than the others, the sample will typically just be the corresponding eigenfunction slightly perturbed, and in the case of ReLU activation networks, this is the case leading to learning simple functions rather than a wider variety. [2]

The first theorem of the paper states that for both the CK and NTK with the data distribution being uniform over the d-dimensional boolean cube, uniform over the sphere $\sqrt{d}S^{d-1}$, or the standard Gaussian, for any $d, k > 0$, let $\kappa_{d,k}$ be the kth largest unique eigenvalue of K, K being the NTK or CK,over the data distribution. Then, for any $k > 0$, for sufficiently large d which depends on k, $\kappa_{d,k}$ corresponds to an eigenspace of dimension $d^k/k! + o(d^k)$. [2]

Since the dimension of the subspace spanned by the top k unique eigenvalues grows so quickly, it is only necessary to look at the first few $\kappa_{d,k}$ as the number of samples in most problems is usually much smaller than the dimension of the subspace spanned by the first few unique eigenvalues. For example, in the paper, they mention that if $d = 576$ which is the number of pixels in an MNIST image, and $k = 5$, the number of dimensions spanned is about 5.2×10^{11}. [2]

Then, over the boolean cube, the eigenfunctions of the kernels are the parity functions of all subsets of the d dimensions, and the eigenvalues actually only depend on the cardinality of the the set of dimensions in the parity function. [2] This makes the simplicity of functions learned easy to understand as if μ_k is the eigenvalue for functions over k dimensions, then if μ_k is larger for bigger k, then there is more complexity and if μ_k is larger for smaller k, then there is more simplicity. [2] For two layer ReLU networks with $\sigma_w = \sigma_b = 2$, μ_0 was the largest and accounted for 80% of the variance so it would tend to learn almost constant functions and for the erf activation and $\sigma_b = 0$, μ_1 was largest so it would tend to learn almost linear functions. [2] Although the same conclusions cannot be made when looking at the eigenvalues when the data distribution is more complex as typically is in real datasets, the same idea of looking at the top unique eigenvalues and their eigenfunctions can be used to understand generalization properties.

One measure of how the eigenvalues affect the learned function is through degree k fractional variance which is

$$\frac{\binom{d}{k}\mu_k}{\sum_{i=0}^{d}\binom{d}{i}\mu_i}$$

where the degree k fractional variance is the fraction of the variance of the sampled function f from K being the NTK or CK. [2] Then, degree k fractional variance is roughly a measure of how well a degree k polynomial could be learned. Through experiments of maximizing degree k fractional variance for different k by changing the depths of the network, it was found that infinite depth is not optimal and that past a certain point, degree k fractional variance will start decreasing. It was also found in the experiments that the point at which that occurs increases with k, and this idea of an optimal depth was also found to appear in real datasets.

Using degree k fractional variance as a measure of complexity, it is also found that the NTK tends to learn more complex functions than the CK as for larger k, the NTK has higher degree k fractional variance.

The eigendecomposition of the kernels also provides a way of finding the maximum learning rate needed to converge which is simply the inverse of the largest eigenvalue and this was found to be accurate for real datasets as well.

4 Convolutional NTK

The Arora paper not only provided stronger results on the NTK, but also improved upon its extension to convolutional neural nets by creating a more intuitive formula that includes convolution. The formula for the convolutional NTK (CNTK) will not be written out, but it is defined inductively starting from the first layer to the last and each one involves convolution with the kernel of the previous layer. This new formula provides for a fast way to compute the CNTK and can be used with GPUs, and exactly computing the CNTK or NTK is important as the paper finds that when

using a linearization of the network which only uses the kernel and output at initialization to train, the performance of the model was worse by 5% on the CIFAR dataset. [1] Efficient computing also allows for new or more experimentation that could lead to further results on understanding the CNTK or CNNs.

5 Discussion

The results above provide stronger results and new tools to understand deep learning, and they use the NTK to learn more about finite width neural nets, and with these new ideas come new questions to explore. Here we discuss some of those questions.

One question is what happens if we were to take the results on the equivalence between NTKs and wide networks and used them to show results in the opposite direction. With the theorem that given a ReLU network's depth L, and an ϵ and a δ, that one can find a width such that the network is $(L+1)\epsilon$ close to the NTK as initialization with probability at least $1-\delta$ leads to the possibility of being given a width and choosing an ϵ and δ so that the theorem holds. A similar idea applies to the theorem about the equivalence between a trained network and kernel regression on the NTK, and this could possibly be used to explore the behavior of finite width neural networks and possibly finding some bounds on this behavior. More generally, while this paper shows that sufficiently wide trained networks are equivalent to the the NTK, it could be interesting to explore the networks that are not sufficiently wide and seeing how close they are to the NTK and trying to use this to learn about their generalization properties. One way this idea could be used is to see how far the behavior of the spectra of the weights of the neural networks varies from the NTK based on its width.

A completely different route of studying generalization properties is studied in [2], and one possible extension of this study is by trying to understand apply these ideas to more specific regimes. The simplicity bias of the NTK or CK helps us understand the generalization properties of neural networks, in particular about the functions they learn, and this leads to questions about how this simplicity bias could be studied in fields that apply neural networks such as natural language processing and seeing whether the eigendecomposition of a model's NTK and the simplicity bias could correspond to some linguistic property or simply provide insight to how the models work. With the CNTK, the idea of simplicity bias could also possibly be studied with convolutional neural networks, and this opens up many new questions to explore such as trying to understand the simplicity bias of a CNTK and how it compares to the NTK or CK. This could also extend specifically to images as CNNs are known for performing well on image classification and understanding what simplicity could mean in terms of learning images, and with the results in [1], the computation part should be less of an obstacle.

An idea that appears from looking at the two papers is the role of small eigenvalues. [2] finds that the large eigenvalues are important to analyzing the type of function a neural network learns, and that small eigenvalues are hard to even detect in the functions learned. However, [1] finds a use for the minimum eigenvalue λ_0 as it is involved in the bound on the width for a trained neural network to be equivalent to kernel regression with the NTK. It seems that although the smallest eigenvalues do not really signal the types of functions a neural network learns, they could provide a bound on how spread out the variance from unique eigenvalues are or provide some other generalization bound.

6 Conclusion

NTKs provide a way of understanding a limiting behavior of neural networks, and with results from [1], they provide a way of also understanding not just a limiting behavior but the actual behavior for some neural networks by showing equivalence. Also, analysing the spectrum of the NTK provides insight into the behavior of neural networks by giving a tool of measuring the simplicity and variance of what functions can be learned. [2] There seems to be many approaches to take when considering how to extract information from the NTK, and when we look at two of them together, we get a more complete view and new methods of approach seems to appear and new questions arise and our understanding of deep learning seems to grow.

References

[1] Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. 2019.

[2] Greg Yang and Hadi Salman. A fine-grained spectral perspective on neural networks. 2019.