Neural Tangent Kernel

Huihui Xu
MPhil
RBM
HKUST(GZ)

***Abstract*—Neural Tangent Kernel (NTK) is one of the latest and important devices in the theoretical analysis of deep neural networks. This report will illustrate the basic concepts for NTK and its applications to analyse wide neural networks.**

***Keywords*—neural tangent kernel, wide neural networks, deep learning theory**

I. Introduction

Multilayer feed-forward neural networks trained with gradient descent can often generalize well on test samples. And this is true even when the number of model parameters exceeds the number of training samples. Moreover, the high-dimensional non-convex loss surface renders the characterization of optimization dynamics intractable. Recently, neural tangent kernel (NTK) [1] is an important concept for understanding neural network training via gradient descent. At its core, it explains how updating the model parameters on one data sample affects the predictions for other samples.

To understand why the effect of one gradient descent is so similar for different initializations of network parameters, several pioneering theoretical work starts with infinite width networks. We will look into detailed proof using NTK of how it guarantees that infinite width networks can converge to a global minimum when trained to minimize an empirical loss.

II. Preliminary

Suppose a supervised training dataset is $\{X \in R^{N\times d}, Y \in R^{N\times c}\} = \{(x_i, y_i)\}_{i=1}^N$, where $x_i \in R^d$ and $y_i \in R^c$ is the training sample and its supervision signal. Suppose m is any positive integer, a model is denoted as $f : (R^d)^{\otimes m} \times \Theta \to (R^c)^{\otimes m}$, where Θ is the parameter space. The loss function is denoted as $L : (R^c)^{\otimes m} \times (R^c)^{\otimes m} \to R$. The notations here are different to convention, in the sense that, f can take either one vector in R^d or a $m \times d$ matrix as input and output the corresponding model predictions, i.e., one vector in R^c or a $m \times c$ matrix.

Kernel Function A scalar-valued kernel function $K : (R^d)^{\otimes m_1} \times (R^d)^{\otimes m_2} \to R^{m_1\times m_2}$ measures the similarity between any two samples. If the input of the kernel function is all the training samples X, then $K(X, X) \in R^{N\times N}$ is called the Gram matrix. The simplest kernel function is inner product $K(x, x') = x^T x'$. For linear regression model optimized with Tikhonov regularization, we can have a dual representation:

$$\begin{aligned} f(x; w^*) &= w^{*T} x \\ &= Y^T X(X^T X + \lambda I_d)^{-1} x \\ &= Y^T (XX^T + \lambda I_N)^{-1} Xx \\ &= Y^T (K(X, X) + \lambda I_N)^{-1} K(X, x). \end{aligned} \tag{1}$$

Therefore, when predicting a test sample, in addition to Y, the information extracted from the training data is the Gram matrix. We can view the Gram matrix as an information bottleneck that must transmit enough information about the training data for the model to be able to perform its task.

Infinite-width network at initialization is a Gaussian process Suppose a network of L layers is parametrized as: for $l = 0, ..., L-1$, $x^0 = x, n_0 = d, n_L = c$, $z^{l+1} = \frac{1}{\sqrt{n_l}} W^l x^l + \beta b^l, x^l = \sigma(z^l)$, where $W^l \in R^{n_l \times n_{l-1}}$, $b^l \in R^{n_l}$, and n_l are the weight matrix, bias vector, and layer width of the l^{th} layer, respectively. x_l and z_l are called the activations and pre-activations of the l^{th} layer. β is a hyperparameter to tune the influence of the bias on the training. Our model is denoted as $f(x; \theta) = z^L$. Generally, before training, the weights and biases of a neural network are i.i.d. sample from Gaussian distribution, in that $W_{ij}^l \sim \mathcal{N}(0, \frac{1}{n_l})$ and $b_i^l \sim \mathcal{N}(0, 1)$.

Proposition 1 [1], [2], [5] For a network of depth L at initialization, with a Lipschitz nonlinearity σ, and in the limit as $n_1, ..., n_{L-1} \to \infty$, the output functions $f(\cdot; \theta)_k : R^d \to R$, for $k = 1, ..., n_L$, tend (in law) to i.i.d. centered Gaussian processes of covariance Σ^L, where Σ^L is defined recursively by:

$$\begin{aligned} \Sigma^1(x, x') &= \frac{1}{n_0} x^T x' + \beta^2 \\ \Sigma^{L+1}(x, x') &= E_{f\sim N(0,\Sigma^L)}[\sigma(f(x))\sigma(f(x'))] + \beta^2 \end{aligned} \tag{2}$$

taking the expectation w.r.t. a centered Gaussian process f of covariance Σ^L.

III. Neural Network Kernel

Gradient Descent in Θ can be characterized as:

$$\begin{aligned} \dot\theta_t &= -\eta \nabla_\theta L(f(X; \theta), Y) \\ &= -\eta \nabla_2 f(X; \theta_t)\ \nabla_1 L(f(X; \theta_t); Y) \end{aligned} \tag{3}$$

$$\begin{aligned} \dot f(X; \theta_t) &= \nabla_2 f(X; \theta_t)^T \dot\theta_t \\ &= -\eta K_f(X, X; \theta_t) \nabla_1 L(f(X; \theta_t); Y) \end{aligned} \tag{4}$$

$$\dot f(x; \theta_t) = -\eta K_f(x, X; \theta_t) \nabla_1 L(f(X; \theta_t); Y),\ \forall x \in R^d \tag{5}$$

where

$$K_f(X, X; \theta_t) = \nabla_2 f(X; \theta_t)^T \nabla_2 f(X; \theta_t) \in (R^{c\times c})^{\otimes N\times N} \tag{6}$$

is a *multi-dimensional Gram matrix* for the current parametrization, induced by the *f-specific parametrized multi-dimensional kernel function*:

$$\begin{aligned} K_f : (R^d)^{\otimes m_1} \times (R^d)^{\otimes m_2} \times \Theta &\to (R^{c\times c})^{\otimes m1\times m2} \\ K_f(x, x; \theta) &= \nabla_2 f(x; \theta)^T \nabla_2 f(x; \theta) \end{aligned} \tag{7}$$

We call this parametrized multi-dimensional kernel function K_f **the NTK of the model** f, which takes any inputs and any parametrization as input. We call the matrix $K_f(X, X; \theta)$ **the Gram matrix of the model** f **parametrized by** θ **w.r.t the training inputs** X**.**

Updating θ according to (3) will cause the loss value to change:

$$\begin{aligned}\dot{L}(f(X;\theta_t);Y) &= \nabla_1 L(f(X;\theta_t);Y)^T \dot{f}(X;\theta_t) \\ &= -\eta \underbrace{\nabla_1 L(f(X;\theta_t);Y)^T}_{(1\times c)^{\otimes N}} \underbrace{K_f(X,X;\theta_t)}_{(c\times c)^{\otimes N\times N}} \underbrace{\nabla_1 L(f(X;\theta_t);Y)}_{(c\times 1)^{\otimes N}}\end{aligned} \tag{8}$$

Generally, fixing the second input and m, the loss function $L : (R^c)^{\otimes m} \times (R^c)^{\otimes m} \to R$ is a convex function in its first input. For example, we usually use a l2-norm loss for regress tasks. And we can prove using definition that $L(Y, \hat{Y}) = \frac{1}{N}\sum_{i=1}^{N} ||y_i - \hat{y}_i||_2^2$ is convex in $Y \in (R^c)^{\otimes N}$. And we usually use the log softmax compositied by negative cross-entropy loss for classification tasks. And assuming all the entries of Y are positive, by definition, we can prove that $L(Y, \hat{Y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{c} y_i^j \text{logsoftmax}(y_i^j)$ is convex in $Y \in (R^c)^{\otimes N}$.

Therefore, we can analyse the optimization dynamics from Θ to $(R^c)^{\otimes N}$, under the fact that $L(\cdot, Y)$ is a convex function in the model predictions $f(X;\theta) \in (R^c)^{\otimes N}$.

(8) shows that the derivative of the loss is a quadratic form in $(R^c)^{\otimes N}$, whose matrix is just the Gram matrix, given by (6), of the model f parametrized by θ_t w.r.t the training inputs X

Therefore, we have following conclusions:

- if at some t, we can know that $K_f(X, X; \theta_t)$ is a positive definite second-order tensor in $(R^{c\times c})^{\otimes N\times N}$, then the training loss $L(X; \theta_{t+1})$ will decrease, at least for sufficiently small η, by gradient descent in Θ.
- if we further know that for any θ, $K_f(X, X; \theta)$ is positive definite, then the loss will decrease during the whole training process; Moreover, since $L(\cdot; Y)$ is convex in $(R^{c\times c})^{\otimes N\times N}$, the optimization process will converge to a stationary point, which is global optimal.

Two main theorems proposed by [1] are listed below:

Theorem 1 For a feed-forward network of depth L at initialization, with a Lipschitz nonlinearity σ, and in the limits as the layers width $n_1, \ ..., n_{L-1} \to \infty$, its NTK K_f^L converges in probability to a deterministic limiting kernel:

$$K_f^L \to K_\infty^L \otimes Id_{n_L}. \tag{9}$$

The scalar kernel $K_\infty^L : R^d \times R^d \to R$ is defined recursively by

$$\begin{aligned}K_\infty^L(x, x') &= \Sigma^1(x, x') \\ K_\infty^{L+1}(x, x') &= K_\infty^L(x, x')\dot{\Sigma}^{L+1}(x, x') + \Sigma^{L+1}(x, x')\end{aligned}$$

where

$$\dot{\Sigma}^{L+1}(x, x') = E_{f\sim\mathcal{N}(0,\Sigma^L)}[\dot{\sigma}(f(x))\dot{\sigma}(f(x'))]$$

taking the expectation w.r.t. a centered Gaussian process f of covariance Σ^L given by (2).

Interpretation Unlike (7), the the limiting kernel (9) is not dependent to the specific initialized parameters. Therefore, the NTK of a network of infinite-width after being initialized only depends on the depth of the network, the choice of the nonlinearity and the chosen weight/biases variances.

Theorem 2 In stead of assuming doing kernel gradient descent in $(R^c)^{\otimes N}$, we generalize and define the training process as just moving along any direction $d_t \in (R^c)^{\otimes N}$ at any t. Assume that σ is a Lipschitz, twice differentiable nonlinearity function, with bounded second derivative. For any T such that the integral $\int_0^T ||d_t||^2 dt$ stays stochastically bounded, as $n_1, ..., n_{L-1} \to \infty$, we have, uniformly for $t \in [0, T]$,

$$K_f^L(;\theta_t) \to K_\infty^L \otimes Id_{n_L} \tag{10}$$

As a consequence, in this limit, according to (5), the dynamics of $f(\cdot;\theta)$ during training is described by the differential equation:

$$\dot{f}(x) = -\eta K_\infty^L \otimes Id_{n_L}(x, X)\ d_t \tag{11}$$

Interpretation This theorem shows that, as long as the change of $f(X;\theta)$ is bounded, the NTK of a network of infinite width can stay asymptotically close to the limiting kernel (9) during training. As [1] noted, the limiting Gram matrix defined by the limiting kernel, i.e. $K_\infty^L \otimes Id_{n_L}(X, X)$, is a positive definite second-order tensor in $(R^c)^{\otimes N}$ for many X, and σ. [3], [4] further gives some classical theorems about a sufficient condition to render the limiting Gram matrix positive definite.

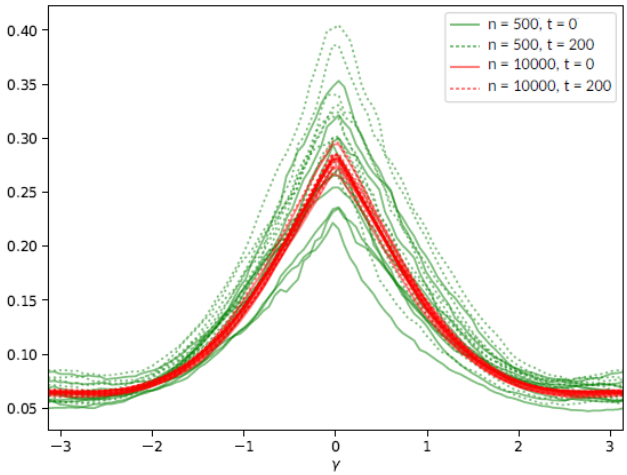

Fig. 1: Convergence of the NTK to a fixed limit for two widths and two times

The authors uses two networks of different widths (500/10000), trained using a artificial dataset with $c = 1, \ d = 2$. The results are shown in Fig.1. For the wider network, the NTK shows less variance and is smoother. The expectation of the NTK is very close for both networks widths. After 200 steps of training, we observe that the NTK tends to "inflate". As expected, this effect is much less apparent for the wider network (n = 10000) where the NTK stays almost fixed, than for the smaller network (n = 500).

IV. Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent

[2] shows that in the infinite width limit, the learning dynamics for doing gradient descent are governed by a linear model obtained from the first-order Taylor expansion of the network around its initial parameters.

According to (11), when doing kernel gradient descent in $(R^c)^{\otimes N}$, the derivative of the predictions for each input is:

$$\dot{f}(x;\theta_t) = -\eta K_\infty^L \otimes Id_{n_L}(x,X)\nabla_1 L(f(X;\theta_t);Y)$$

To track the evolution of θ in time, let's consider it as a function of time step t. With Taylor expansion, the network learning dynamics can be simplified as:

$$f(\cdot;\theta(t)) \approx f^{lin}(\cdot;\theta(t)) = f(\cdot;\theta(0)) + \nabla_\theta f(\cdot;\theta(0))(\theta(t)-\theta(0)) \tag{12}$$

which f^{lin} is referred to as the **linearized model at** $\theta(0)$.

Assuming that the incremental time step t is extremely small and the parameter is updated by gradient descent:

$$\begin{aligned}
&\theta(t)-\theta(0) = -\eta\nabla_\theta L(f(X;\theta(0));Y)\\
&= -\eta\nabla_2 f(X;\theta(0))\nabla_1 L(f(X;\theta(0));Y)\\
&f^{lin}(X;\theta(t)) - f(X;\theta(0))\\
&= -\eta\nabla_\theta f(X;\theta(0))^T\nabla_\theta f(X;\theta(0))\nabla_1 L(f(X;\theta(0));Y)\\
&\dot{f}(X;\theta(t)) = -\eta K(X,X;\theta(0))\nabla_1 L(f(X;\theta(0));Y)
\end{aligned} \tag{13}$$

Since the Gram matrix after initialization, i.e.,$K(X,X;\theta(0))$, converges to be the limiting kernel (9) for infinite width networks, we eventually get the same learning dynamics, which implies that a neural network with infinite width can be considerably simplified as governed by above linearized model.

In a simple case when the empirical loss is an MSE loss, $\nabla_1 L = f(X;\theta) - Y$, the dynamics of the network becomes a simple linear ODE and it can be solved in a closed form:

$$\begin{aligned}
&\dot{f}(X;\theta) = -\eta K_\infty(X;X)(f(X;\theta)-Y)\\
&\text{let } g(X;\theta) = f(X;\theta) - Y\\
&\dot{g}(X;\theta) = -\eta K_\infty(X,X)g(X;\theta)\\
&\int \dot{g}(X;\theta)dt = -\eta\int K_\infty(X;X)dt\\
&f(X;\theta) - Y = Ce^{-\eta K_\infty(X,X)t}
\end{aligned} \tag{14}$$

When $t=0$, we have $C = f(\theta(0)) - Y$, therefore, the training dynamics has a closed form:

$$f(X;\theta(t)) = (f(X;\theta(0)) - Y)e^{-\eta K_\infty(X,X)t} + Y \tag{15}$$

$$= f(X;\theta(0))e^{-K_\infty(X,X)} + (I - e^{-\eta K_\infty(X,X)t})Y \tag{16}$$

V. Comparisons

The first work [1] introduced the concept of Neural Tangent Kernel. It proves that the NTK of networks of infinite width at initialization only depends on the network depth, activation function and the chosen variances, and will stay asymptotically close the the limiting NTK during training. The corresponding Gram matrix induced by the limiting NTK will be a positive definite second-order tensor for many training data. The second work [2] showed theoretically that the learning dynamics in parameter space of deep nonlinear neural networks are exactly described by a linearized model in the infinite width limit.

VI. Conclusion

The NTK is a analysis tool to analyse the generalization and optimization of deep neural networks. The [1] shows that the limiting kernel converges to be positive semi definite w.r.t to many X during the whole training process. However, these two papers only considered X is not changing. In reality, the Catastrophic Forgetting problem [6] will appear, in that if X is sampled from a different distribution after learning one task, the naive training based on the new X will cause the neural network to forget the knowledge of previous tasks. But NTK is an extreme versatile analysis tool. We can measure the changing dynamics of the predictions of any inputs, during training with any training data. I think using NTK to analyze Catastrophic Forgetting is a very promising work.

References

[1] A. Jacot, F. Gabriel, and C. Hongler, Neural tangent kernel: Convergence and generalization in neural networks," in Advances in Neural Information Processing Systems, vol. 31, 2018..

[2] J. Lee, L. Xiao, S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington, Wide neural networks of any depth evolve as linear models under gradient descent," in Advances in Neural Information Processing Systems, vol.32, 2019.

[3] K. Hornik, M. Stinchcombe, and H. White, Multilayer feedforward networks are universal approximators," Neural networks, vol.2, no.5, pp.359-366, 1989.

[4] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," Neural networks, vol.6, no.6, pp. 861-867, 1993.

[5] Neal, Radford M. Bayesian learning for neural networks. Vol. 118. Springer Science & Business Media, 2012.

[6] Kirkpatrick, James, et al. Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences 114.13 (2017): 3521-3526.