Neural Tangent Kernel Derivation

By SY Chou, 2021, Jul

$$f(x, \theta) = \sigma(wx + b) \quad w, b \in \theta$$

$$L(f, x, y, \theta) = (f(x, \theta) - y)^2$$

$$\theta_{t+1} = \theta_t + \eta \nabla_w L(f, x, y, \theta)$$

Denote θ_t as dynamic $\theta(t)$

$$\frac{d\theta(t)}{dt} = \nabla_\theta L(f, x, y, \theta)$$

$$= 2(f(x, \theta) - y)\nabla_\theta f(x, \theta)$$

$$= 2(f(x, \theta) - y)\nabla_\theta(f(x, \theta_0) + \nabla_\theta f(x, \theta_0)^\top(\theta - \theta_0))$$

$$= 2(f(x, \theta) - y)\nabla_\theta(f(x, \theta_0) + \nabla_\theta f(x, \theta_0)^\top\theta - \nabla_\theta f(x, \theta_0)^\top\theta_0)$$

$$= 2(f(x, \theta) - y)\nabla_\theta f(x, \theta_0)$$

Abstract

To understand what is the neural tangent kernel(NTK), there are 3 important result that need to remember.

- When the width of the neural network goes to infinity, the network will be equivalent to a Gaussian process.
- When the width of the neural network goes to infinity, the weight of the network will remain almost unchanged during training. That is, the neural network will become a linear model.

- Since the network will become a linear model, the loss surface of the MSE of the infinite-width network will be a convex. As a result, we can optimize the infinite-width network just like optimizing a linear regression and solve it by ODE.

Combine these 3 points, NTK is a kernel that can kernelize the neural network architecture and it provides a closed-form solution of the kernel for anytime. Thus, We can compute the NTK at the end of training without actual training and compute the posterior and the prediction of the testing data with Bayesian inference.

Infinite-Width Neural Network As Gaussian Process

Define A Neural Network

First, given a training dataset $\mathcal{D}$ including $N = |\mathcal{D}|$ data points. We denote the data point as $d_i = (x, y) \in \mathcal{D}\ \forall i$ where the feature vector $x \in \mathbb{R}^{k \times 1}$ and the label $y \in \mathbb{R}$. The set of the feature vectors is $\mathcal{X} = \{x : (x, y) \in \mathcal{D}\}$ and similarly, the set of the label $\mathcal{Y} = \{y : (x, y) \in \mathcal{D}\}$.

We represent the element of the neural network $f(x, \theta),\ x \in \mathcal{X}$ as following

$$h^1 = W^1 x + b^1$$

$$h^{l+1} = W^{l+1} a^l + b^{l+1} = W^{l+1} \phi(h^l) + b^{l+1}$$

$$\hat{y} = f(x, \theta) = h^L$$

where ϕ is the activation function, $h^{l+1} \in \mathbb{R}^{n_{l+1} \times 1}$ and $a^{l+1} \in \mathbb{R}^{n_{l+1} \times 1}$ are the pre-activation and the post-activation respectively. The parameter θ includes $W^l, b^l \in \theta\ \forall l$. $W^{l+1} \in \mathbb{R}^{n_{l+1} \times n_l}$ and $b^{l+1} \in \mathbb{R}^{n_{l+1} \times 1}$ are the weight and the bias respectively. They follow the LeCun and normal distribution respectively. Also, we denote the n_{l+1} as the network width of $l+1$-th layer and $n_0 = k$ is the size of the feature vector. Finally, we denote $\hat{y} \in \mathbb{R}$ is the prediction of the network based on input x. For convenience, we also denote $\hat{\mathcal{Y}} = f(\mathcal{X}, \theta) \in \mathbb{R}^N$.

$$W^{l+1}_{i,j} = \frac{\sigma_w}{\sqrt{n_{l+1}}} w^{l+1}_{i,j}, \quad b^{l+1}_i = \sigma_b \beta^{l+1}_i$$

$$w^{l+1}_{i,j}, \beta^{l+1}_i \overset{i.i.d}{\sim} \mathcal{N}(0, 1)$$

$$\forall l, i, j \quad 1 \leq l \leq L, \quad 1 \leq i \leq n_{l+1}, \quad 1 \leq j \leq n_l$$

where $W_{i,j}^{l+1} \in \mathbb{R}$ is the entry of the matrix of $l+1$-th layer at the i-th row and j-th column. $b_i^l \in \mathbb{R}$ is the i-th entry of the bias vector.

Combine them.

$$h_i^{l+1} = W_i^{l+1} a^l + b_i^{l+1}$$

$$= \sum_{j=1}^{n_{l+1}} W_{i,j}^{l+1} a_j^l + b_i^{l+1}$$

$$= \frac{\sigma_w}{\sqrt{n_{l+1}}} \sum_{j=1}^{n_{l+1}} w_{i,j}^{l+1} \phi(h_j^l) + \sigma_b \beta_i^{l+1}$$

where $W_i^{l+1} \in \mathbb{R}^{1 \times n_l}$ is the i-th row of the matrix W^{l+1}. The post-activation vector is $a^l \in \mathbb{R}^{n_l \times 1}$ and the j-th entry of the vector is $a_j^l \in \mathbb{R}$.

Let θ^{l+1} denote as the all parameters of the layer $l+1$.

$$\theta^{l+1} = vec(W^{l+1}, b^{l+1}) \in \mathbb{R}^{n_{l+1} \times (n_l + 1)}$$

$$\theta = vec(\cup_{l=1}^{L} \theta^l) \in \mathbb{R}^{S \times 1}$$

where $S = \sum_{l=1}^{L} n_l (n_{l-1} + 1)$ is the number of all parameters.

We also denote the empirical loss of all training dataset as $\mathcal{L}(\mathcal{X}, \mathcal{Y})$, and $l(\hat{y}, y)$ as loss function.

$$l(\hat{y}, y) = l(f(x, \theta), y)$$

$$\mathcal{L}_{f(\cdot,)}(\mathcal{X}, \mathcal{Y}) = \sum_{x \in \mathcal{X},\ y \in \mathcal{Y}} l(f(x, \theta), y) \in \mathbb{R}$$

The Mean Of The Layers

For i-th entry of the l-th layer, with data point x, we can derive the expectation of the entry.

$$E[h_i^l(x)] = E[\frac{\sigma_w}{\sqrt{n_{l-1}}} \sum_{j=1}^{n_{l-1}} w_{i,j}^l a_j^{l-1} + \sigma_b \beta_i^l]$$

$$= \frac{\sigma_w}{\sqrt{n_{l-1}}} \sum_{j=1}^{n_{l-1}} E[w_{i,j}^l] a_j^{l-1} + \sigma_b E[\beta_i^l] = 0$$

Thus, for any entry on the any layer, the expectation is $E[h_i^l(x)] = 0, \forall i, l$.

The Covariance Of The First Layer

Consider two different case, derive the covariance of the same entry and the different entries respectively.

Case 1: Consider the same entries.

$$k^1(x, x') = Cov[h_i^1(x), h_i^1(x')]$$

$$= E[(h_i^1(x) - E[h_i^1(x)])(h_i^1(x') - E[h_i^1(x')])]$$

$$= E[h_i^1(x) h_i^1(x')] - E[h_i^1(x)] E[h_i^1(x')]$$

$$= E[(\frac{\sigma_w}{\sqrt{n_0}} \sum_{j=1}^{n_0} w_{i,j}^1 x_j + \sigma_b \beta_i^1)(\frac{\sigma_w}{\sqrt{n_0}} \sum_{j=1}^{n_0} w_{i,j}^1 x'_j + \sigma_b \beta_i^1)]$$

$$= E[\frac{\sigma_w^2}{n_0} (\sum_{j=1}^{n_0} w_{i,j}^1 x_j)(\sum_{k=1}^{n_0} w_{i,k}^1 x'_k)) + (\sigma_b \beta_i^1)(\sigma_b \beta_i^1) + \frac{\sigma_w}{\sqrt{n_0}} (\sigma_b \beta_i^1 \sum_{j=1}^{n_0} w_{i,j}^1 x_j) + \frac{\sigma_w}{\sqrt{n_0}} (\sigma_b \beta_i^1 \sum_{j=1}^{n_0} w_{i,j}^1 x'_j)]$$

$$= \frac{\sigma_w^2}{n_0} (E[\sum_{j=k}^{n_0} w_{i,j}^1 x_j w_{i,k}^1 x'_k] + E[\sum_{j \neq k}^{n_0} w_{i,j}^1 x_j w_{i,k}^1 x'_k]) + \sigma_b^2 E[(\beta_i^1)^2] + \frac{\sigma_w \sigma_b}{\sqrt{n_0}} (E[\beta_i^1] E[\sum_{j=1}^{n_0} w_{i,j}^1 x_j]) + \frac{\sigma_w \sigma_b}{\sqrt{n_0}} (E[\beta_i^1] E[\sum_{j=1}^{n_0}$$

Since $E[\beta_i^1] = E[w_{i,j}^l] = 0$, thus we can derive the following formula.

$$= \frac{\sigma_w^2}{n_0} (\sum_{j=k}^{n_0} E[w_{i,j}^1 w_{i,k}^1] x_j x'_k + \sum_{j \neq k}^{n_0} E[w_{i,j}^1 w_{i,k}^1] x_j x'_k) + \sigma_b^2 (Var[\beta_i^1] + E[\beta_i^1]^2)$$

Since $w_{i,j}^1, w_{i,k}^1 \overset{i.i.d}{\sim} \mathcal{N}(0,1)$, thus, $E[w_{i,j}^1 w_{i,k}^1] = E[w_{i,j}^1] E[w_{i,k}^1] = 0$. Also, $Var[\beta_i^1] = 1$.

$$= \frac{\sigma_w^2}{n_0}(\sum_{j=1}^{n_0}(Var[w_{i,j}^1] + E[w_{i,j}^1]^2)x_j x_k') + \sigma_b^2$$

$$= \frac{\sigma_w^2}{n_0}(\sum_{j=1}^{n_0} x_j x_k') + \sigma_b^2$$

$$= \frac{\sigma_w^2}{n_0} x^\top x' + \sigma_b^2$$

Case 2: Consider the different entries.

$$k_{i\neq j}^1(x,x') = Cov[h_i^1(x), h_j^1(x')],\ i \neq j$$

$$= E[(\frac{\sigma_w}{\sqrt{n_0}}\sum_{k=1}^{n_0} w_{i,k}^1 x_k + \sigma_b \beta_i^1)(\frac{\sigma_w}{\sqrt{n_0}}\sum_{k'=1}^{n_0} w_{j,k'}^1 x_{k'}' + \sigma_b \beta_j^1)]$$

$$= \frac{\sigma_w^2}{n_0}(E[\sum_{k=1}^{n_0}\sum_{k'=1}^{n_0} w_{i,k}^1 x_k w_{j,k'}^1 x_{k'}']) + \sigma_b^2 E[\beta_i^1 \beta_j^1] + \frac{\sigma_w \sigma_b}{\sqrt{n_0}}(E[\beta_i^1]E[\sum_{k=1}^{n_0} w_{i,k}^1 x_k]) + \frac{\sigma_w \sigma_b}{\sqrt{n_0}}(E[\beta_j^1]E[\sum_{k'=1}^{n_0} w_{j,k'}^1 x_{k'}'])$$

$$= \frac{\sigma_w^2}{n_0}(\sum_{k=1}^{n_0}\sum_{k'=1}^{n_0} E[w_{i,k}^1]E[w_{j,k'}^1] x_k x_{k'}') = 0$$

The Covariance Of The Deeper Layers

Till now, we've derived the covariance of the first layer, now we can dive into the case of deeper layers. Similarly, we discuss the 2 cases.

Case 1: Consider the different entries.

For i-th entry of the l-th layer, with data point x and x', we can derive the covariance $k_{i\neq j}^l(x,x')$ of the different entries.

$$k_{i\neq j}^l(x,x') = Cov[h_i^l(x), h_j^l(x')],\ i \neq j$$

$$= E[(\frac{\sigma_w}{\sqrt{n_{l-1}}}\sum_{k=1}^{n_{l-1}} w_{i,k}^l \phi(h_k^{l-1}(x)) + \sigma_b \beta_i^l)(\frac{\sigma_w}{\sqrt{n_{l-1}}}\sum_{k'=1}^{n_{l-1}} w_{j,k'}^l \phi(h_{k'}^{l-1}(x')) + \sigma_b \beta_j^l)]$$

$$= \frac{\sigma_w^2}{n_{l-1}}(E[\sum_{k=1}^{n_{l-1}}\sum_{k'=1}^{n_{l-1}} w_{i,k}^l \phi(h_k^{l-1}(x)) w_{j,k'}^l \phi(h_{k'}^{l-1}(x'))]) + \sigma_b^2 E[\beta_i^l \beta_j^l] + \frac{\sigma_w \sigma_b}{\sqrt{n_{l-1}}}(E[\beta_i^l] E[\sum_{k=1}^{n_{l-1}} w_{i,k}^l \phi(h_k^{l-1}(x))]) + \frac{\sigma_w \sigma_b}{\sqrt{n_{l-1}}}($$

$$= \frac{\sigma_w^2}{n_{l-1}}(\sum_{k=1}^{n_{l-1}}\sum_{k'=1}^{n_{l-1}} E[w_{i,k}^l] E[w_{j,k'}^l] \phi(h_k^{l-1}(x)) \phi(h_{k'}^{l-1}(x'))]) = 0$$

We've got the covariance of different entries $k_{i\neq j}^l(x, x')$ of sample x and x' is 0. Thus, we can conclude that different entries are independent.

Case 2: Consider the same entries.

$$k^l(x, x') = Cov[h_i^l(x), h_i^l(x')]$$

$$= E[(h_i^l(x) - E[h_i^l(x)])(h_i^l(x') - E[h_i^l(x')])]$$

$$= E[h_i^l(x) h_i^l(x')] - E[h_i^l(x)] E[h_i^l(x')]$$

$$= E[(\frac{\sigma_w}{\sqrt{n_{l-1}}} \sum_{j=1}^{n_{l-1}} w_{i,j}^1 \phi(h_j^{l-1}(x)) + \sigma_b \beta_i^1)(\frac{\sigma_w}{\sqrt{n_{l-1}}} \sum_{j=1}^{n_{l-1}} w_{i,j}^1 \phi(h_j^{l-1}(x')) + \sigma_b \beta_i^1)]$$

$$= \frac{\sigma_w^2}{n_{l-1}}(E[\sum_{j=k}^{n_{l-1}} w_{i,j}^l \phi(h_j^{l-1}(x)) w_{i,k}^l \phi(h_j^{l-1}(x'))] + E[\sum_{j\neq k}^{n_l} w_{i,j}^l \phi(h_j^{l-1}(x)) w_{i,k}^l \phi(h_j^{l-1}(x'))]) + \sigma_b^2 E[(\beta_i^l)^2] + \frac{\sigma_w \sigma_b}{\sqrt{n_{l-1}}}(E[\beta$$

$$= \frac{\sigma_w^2}{n_{l-1}}(\sum_{j=1}^{n_{l-1}} \phi(h_j^{l-1}(x)) \phi(h_j^{l-1}(x'))) + \sigma_b^2$$

Recall **central limit theorem(CLT)**, support we have random variables $X_1, X_2, ..., X_n$ are i.i.d, the expectation and the variance of the random variables are $E[X_i] = \mu$ and $Var[X_i] = \sigma^2$. Denote the sum of the random variables is $\bar{X} = \frac{1}{n}\sum_{i=1}^n X_i$. The normalized $\bar{X}$ is $\zeta = \frac{\bar{X}-\mu}{\frac{\sigma}{\sqrt{n}}}$.

$$\lim_{n\to\infty} P_\zeta(\zeta \leq z) = P_Z(Z \leq z)$$

where random variable $Z \sim \mathcal{N}(0, 1)$ and $z \in \mathbb{R}$ is a scalar. The functions P_ζ and P_Z are C.D.F of the random variables ζ and Z respectively.

@ Need Proof: Connection between CLT and infinite width

When $n_{l-1} \to \infty$, apply central limit theorem(CLT)

$$k^l(x,x') = \lim_{n_{l-1}\to\infty} \frac{\sigma_w^2}{n_{l-1}} (\sum_{j=1}^{n_{l-1}} \phi(h_j^{l-1}(x))\phi(h_j^{l-1}(x'))) + \sigma_b^2$$

$$= \sigma_w^2 E_{h_j^{l-1}(x), h_j^{l-1}(x') \sim \mathcal{N}(0, K_{x,x'}^{l-1})} [\phi(h_j^{l-1}(x))\phi(h_j^{l-1}(x'))] + \sigma_b^2$$

$$= \sigma_w^2 E_{u,v \sim \mathcal{N}(0, K_{xx'}^{l-1})} [\phi(u)\phi(v)] + \sigma_b^2$$

$$K_{xx'}^{l-1} = \begin{bmatrix} k^{l-1}(x,x) & k^{l-1}(x,x') \\ k^{l-1}(x',x) & k^{l-1}(x',x') \end{bmatrix}$$

$$k^1(x,x') = \frac{\sigma_w^2}{n_0} x^\top x' + \sigma_b^2$$

Neural Network Gaussian Process(NNGP)

Define an operator Γ, for a positive semi-definite matrix $\Sigma \in \mathbb{R}^{2\times 2}$

$$\Gamma(\Sigma) = \sigma_w^2 E_{u,v \sim \mathcal{N}(0,\Sigma)} [\phi(u)\phi(v)] + \sigma_b^2$$

With Γ, the **NNGP** kernel $\Sigma_{\mathcal{X}}^L$.

$$\Sigma_{\mathcal{X}}^L = \begin{bmatrix} k^L(x_1,x_1) & k^L(x_1,x_2) & ...\ k^L(x_1,x_N) \\ k^L(x_2,x_1) & k^L(x_2,x_2) & ...\ k^L(x_2,x_N) \\ . & . & . \\ . & . & . \\ . & . & . \\ k^L(x_N,x_1) & k^L(x_N,x_2) & ...\ k^L(x_N,x_N) \end{bmatrix}$$

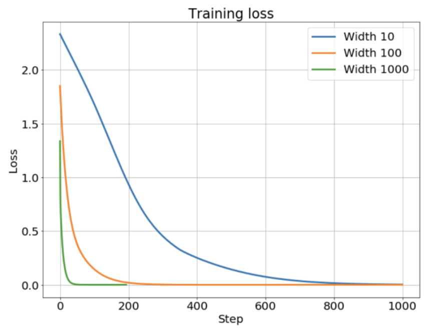
$$k^1(x,x') = \frac{\sigma_w^2}{n_0} x^\top x' + \sigma_b^2$$

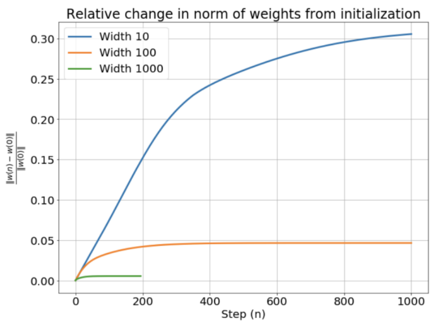
$$k^L(x_1,x_2) = \Gamma(K_{x_1x_2}^{L-1}) = \sigma_w^2 E_{u,v \sim \mathcal{N}(0,K^{L-1})} [\phi(u)\phi(v)] + \sigma_b^2$$

$$K_{x_1x_2}^l = \begin{bmatrix} k^l(x_1,x_1) & k^l(x_1,x_2) \\ k^l(x_2,x_1) & k^l(x_2,x_2) \end{bmatrix}$$

Infinite-Width Neural Network As A Linear Model

Linear Model





As we've shown in the previous post, the parameters of the neural network change more slightly while the width of the network gets larger. In the other words, the neural network **remains almost unchanged during training.** As a result, the parameters $\theta^{(T)} \in \mathbb{R}$ of neural network after training T steps will be very close to the initial parameters $\theta^{(0)}$.

$$\lim_{n_l \to \infty} f(x, \theta^{(t)}) = f(x, \theta^{(0)}), \quad 1 \leq l \leq L, \quad \forall t$$

where $\bar{f}(x, \cdot)$ is the approximation function of neural network $f(x, \cdot)$.

Taylor Expansion

Since the parameters of the infinite-width neural network only change slightly, thus, we can expand the neural network with Taylor expansion.

Taylor expansion

$$g(x) = \sum_{n=0}^{\infty} \frac{g^{(n)}(a)}{n!}(x-a)^n$$

First-order Taylor expansion

$$g(x) \approx \ g(a) + \frac{dg(a)}{dx}(x-a)$$

We denote the parameters of the neural network at training step t as $\theta^{(t)}$. Then, expand the neural network $f(x, \theta^{(t)})$ at training step t with data point x.

$$\hat{y}^{(t)} = f(x, \theta^{(t)}) \approx f_{lin}(x, \theta^{(t)}) = f(x, \theta^{(0)}) + \nabla_\theta f(x, \theta^{(0)})(\theta^{(t)} - \theta^{(0)}), \ \forall t$$

where $\hat{y}^{(t)}$ is the prediction of the data point x from the network at training step t. The divergence $\nabla_\theta f(x, \theta^{(0)}) \in \mathbb{R}^{N \times S}$ is a **Jacobian matrix**.

As for the whole dataset $\mathcal{X}$, the predictions $\hat{\mathcal{Y}}^{(t)}$

$$\hat{\mathcal{Y}}^{(t)} = f(\mathcal{X}, \theta^{(t)}) \approx f_{lin}(x, \theta^{(t)}) = f(\mathcal{X}, \theta^{(0)}) + \nabla_\theta f(\mathcal{X}, \theta^{(0)})(\theta^{(t)} - \theta^{(0)}), \ \forall t$$

$$f_{lin}(\mathcal{X}, \theta^{(t)}) - f(\mathcal{X}, \theta^{(0)}) = \nabla_\theta f(\mathcal{X}, \theta^{(0)})(\theta^{(t)} - \theta^{(0)})$$

Because $f_{lin}(\mathcal{X}, \theta^{(0)}) = f(\mathcal{X}, \theta^{(0)}) + \nabla_\theta f(\mathcal{X}, \theta^{(0)})(\theta^{(0)} - \theta^{(0)}) = f(\mathcal{X}, \theta^{(0)})$, replace $f(\mathcal{X}, \theta^{(0)})$ with $f_{lin}(\mathcal{X}, \theta^{(0)})$

$$f_{lin}(\mathcal{X}, \theta^{(t)}) - f_{lin}(\mathcal{X}, \theta^{(0)}) = \nabla_\theta f_{lin}(\mathcal{X}, \theta^{(0)})(\theta^{(t)} - \theta^{(0)})$$

where $\nabla_\theta f_{lin}(x, \theta^{(0)}) \in \mathbb{R}^{N \times S}$ is also a **Jacobian matrix**.

@ Need Hessian Proof

With Lipschitz continuity, we can guarantee that $\theta^{(t+\Delta t)} - \theta^{(0)} \geq \theta^{(t+\Delta t)} - \theta^{(t)}$ and $f(\mathcal{X}, \theta^{(t+\Delta t)}) - f(\mathcal{X}, \theta^{(0)}) \geq f(\mathcal{X}, \theta^{(t+\Delta t)}) - f(\mathcal{X}, \theta^{(t)})$, and we can also write down the formula.

$$\lim_{\Delta t \to 0} \frac{f_{lin}(\mathcal{X}, \theta^{(t+\Delta t)}) - f_{lin}(\mathcal{X}, \theta^{(t)})}{\Delta t} = \nabla_\theta f_{lin}(\mathcal{X}, \theta^{(0)}) \lim_{\Delta t \to 0} \frac{(\theta^{(t+\Delta t)} - \theta^{(t)})}{\Delta t}$$

$$\frac{\partial f_{lin}(\mathcal{X}, \theta^{(t)})}{\partial t} = \nabla_\theta f_{lin}(\mathcal{X}, \theta^{(0)}) \frac{\partial \theta^{(t)}}{\partial t}$$

In the form of dynamic

$$\dot{f}_{lin}(\mathcal{X}, \theta^{(t)}) = \nabla_\theta f_{lin}(\mathcal{X}, \theta^{(0)}) \dot{\theta}(t)$$

Combine With Gradient Descent

Denote the loss of linearized neural network $f_{lin}(\cdot, \cdot)$ on the training dataset $(\mathcal{X}, \mathcal{Y})$ at the training step t as $\mathcal{L}_{lin}^{(t)}(\mathcal{X}, \mathcal{Y}) = \sum_{x \in \mathcal{X}, \ y \in \mathcal{Y}} l(f_{lin}(x, \theta^{(t)}), y)$. The gradient descent can be represented as the following formula.

$$\theta^{(t+1)} = \theta^{(t)} + \eta \nabla_\theta \mathcal{L}_{lin}^{(t)}(\mathcal{X}, \mathcal{Y}) = \theta^{(t)} + \eta \nabla_\theta f_{lin}(\mathcal{X}, \theta^{(t)})^\top \nabla_{f_{lin}(\mathcal{X}, \theta^{(t)})} \mathcal{L}_{lin}^{(t)}(\mathcal{X}, \mathcal{Y})$$

$$\theta^{(t+1)} - \theta^{(t)} = \eta \nabla_\theta f_{lin}(\mathcal{X}, \theta^{(t)})^\top \nabla_{f_{lin}(\mathcal{X},\theta^{(t)})} \mathcal{L}_{lin}^{(t)}(\mathcal{X}, \mathcal{Y})$$

where $\nabla_{f_{lin}(\mathcal{X},\theta^{(t)})} \mathcal{L}_{lin}^{(t)}(\mathcal{X}, \mathcal{Y}) \in \mathbb{R}^{N \times 1}$

In the form of gradient flow dynamic

$$\dot{\theta}(t) = \eta \nabla_\theta f_{lin}(\mathcal{X}, \theta^{(t)})^\top \nabla_{f_{lin}(\mathcal{X},\theta^{(t)})} \mathcal{L}_{lin}^{(t)}(\mathcal{X}, \mathcal{Y})$$

Replace $\dot{\theta}(t)$ with $\eta \nabla_\theta f_{lin}(\mathcal{X}, \theta^{(t)})^\top \nabla_{f_{lin}(\mathcal{X},\theta^{(t)})} \mathcal{L}_{lin}^{(t)}(\mathcal{X}, \mathcal{Y})$

$$\dot{f}_{lin}(\mathcal{X}, \theta^{(t)}) = \eta \nabla_\theta f_{lin}(\mathcal{X}, \theta^{(t)}) \nabla_\theta f_{lin}(\mathcal{X}, \theta^{(t)})^\top \nabla_{f_{lin}(\mathcal{X},\theta^{(t)})} \mathcal{L}_{lin}^{(t)}(\mathcal{X}, \mathcal{Y})$$

Then, we can expand $\nabla_\theta f_{lin}(\mathcal{X}, \theta^{(t)})$ with Taylor expansion.

$$\nabla_\theta f_{lin}(\mathcal{X}, \theta^{(t)})$$

$$= \nabla_\theta (f(\mathcal{X}, \theta^{(0)}) + \nabla_\theta f(\mathcal{X}, \theta^{(0)})(\theta^{(t)} - \theta^{(0)}))$$

$$= \nabla_\theta (f(\mathcal{X}, \theta^{(0)}) + \nabla_\theta f(\mathcal{X}, \theta^{(0)})\theta^{(t)} - \nabla_\theta f(\mathcal{X}, \theta^{(0)})\theta^{(0)})$$

$$= \nabla_\theta f(\mathcal{X}, \theta^{(0)})$$

Finally, we can replace $\nabla_\theta f_{lin}(\mathcal{X}, \theta^{(t)})$ with $\nabla_\theta f(\mathcal{X}, \theta^{(0)})$, since they are identical due to the linearity.

$$\dot{f}_{lin}(\mathcal{X}, \theta^{(t)}) = \eta \nabla_\theta f(\mathcal{X}, \theta^{(0)}) \nabla_\theta f(\mathcal{X}, \theta^{(0)}) \nabla^\top_{f_{lin}(\mathcal{X},\theta^{(t)})} \mathcal{L}_{lin}^{(t)}(\mathcal{X}, \mathcal{Y})$$

Let $T_{\mathcal{X}\mathcal{X}}^{(0)} = \nabla_\theta f(\mathcal{X}, \theta^{(0)}) \nabla_\theta f(\mathcal{X}, \theta^{(0)})^\top$

$$\dot{f}_{lin}(\mathcal{X}, \theta^{(t)}) = \eta T_{\mathcal{X}\mathcal{X}}^{(0)} \nabla_{f_{lin}(\mathcal{X},\theta^{(t)})} \mathcal{L}_{lin}^{(t)}(\mathcal{X}, \mathcal{Y})$$

where $T_{\mathcal{X}\mathcal{X}}^{(0)} \in \mathbb{R}^{|\mathcal{D}| \times |\mathcal{D}|}$ is the **Neural Tangent Kernel(NTK)**

Gradient Flow Of MSE As A Linear Regression

Mean Square Error(MSE)

In the previous section, we've derive the relation between the prediction of the neural network and the gradient descent at time step t. In this section, we'll dive into the point of view of gradient flow. Now, we've known that the linear approximation of the neural network regression.

$$\dot{f}_{lin}(\mathcal{X}, \theta^{(t)}) = \eta T^{(0)}_{\mathcal{X}\mathcal{X}} \nabla_{f_{lin}(\mathcal{X}, \theta^{(t)})} \mathcal{L}^{(t)}_{lin}(\mathcal{X}, \mathcal{Y})$$

Usually, in regression tasks, we use **Mean Square Error(MSE)** as error function.

$$l(f_{lin}(x, \theta), y) = \frac{1}{2} ||f_{lin}(x, \theta) - y||_2^2$$

Plugin the MSE into the neural network regression

$$\dot{f}_{lin}(\mathcal{X}, \theta^{(t)}) = \eta T^{(0)}_{\mathcal{X}\mathcal{X}} \nabla_{f_{lin}(\mathcal{X}, \theta^{(t)})} \mathcal{L}^{(t)}_{lin}(\mathcal{X}, \mathcal{Y})$$

$$= \eta T^{(0)}_{\mathcal{X}\mathcal{X}} \nabla_{f_{lin}(\mathcal{X}, \theta^{(t)})} \sum_{x \in \mathcal{X},\ y \in \mathcal{Y}} \frac{1}{2} ||f_{lin}(x, \theta^{(t)}) - y||_2^2$$

$$= \eta T^{(0)}_{\mathcal{X}\mathcal{X}} \sum_{x \in \mathcal{X},\ y \in \mathcal{Y}} ||f_{lin}(x, \theta^{(t)}) - y||_2$$

$$= \eta T^{(0)}_{\mathcal{X}\mathcal{X}} (f_{lin}(\mathcal{X}, \theta^{(t)}) - \mathcal{Y})$$

The Dynamic

In the previous post, we've shown that we can continue the discrete step and take the discrete trajectory as a continuous trajectory as if the step is small enough. For instance, Malthus' population growth model says that "the population growth rate is proportional to the total population".

$$\begin{cases} \frac{dP(t)}{dt} = \lambda P(t),\ \lambda > 0 \\ P(t_0) = P_0 \end{cases}$$

where the dynamic $P(t)$ is the population at time t and the P_0 is the initial population. Usually, we also denote $\frac{dP(t)}{dt} = \dot{P}(t)$.

Now, we can solve the dynamic $P(t)$.

$$\int_{t_0}^{t_1} \frac{dP(t)}{dt}\frac{1}{P(t)}dt = \int_{t_0}^{t_1} \lambda dt$$

$$\int_{t_0}^{t_1} \frac{d\ ln(P(t))}{dt}dt = \lambda(t_1 - t_0)$$

$$ln(P(t_1)) - ln(P(t_0)) = \lambda(t_1 - t_0)$$

$$ln(\frac{P(t_1)}{P(t_0)}) = \lambda(t_1 - t_0)$$

$$P(t_1)) = P(t_0)e^{\lambda(t_1 - t_0)}$$

The solution is

$$P(t) = P_0 e^{\lambda(t - t_0)}$$

Prediction Dynamic

$$\dot{f}_{lin}(\mathcal{X}, \theta^{(t)}) = \eta T^{(0)}_{\mathcal{X}\mathcal{X}}(f_{lin}(\mathcal{X}, \theta^{(t)}) - \mathcal{Y})$$

@ Need Proof

Thus, we can write down the ODE.

$$\dot{f}_{lin}(\mathcal{X}, \theta^{(t)}) = \eta T^{(0)}_{\mathcal{X}\mathcal{X}} f_{lin}(\mathcal{X}, \theta^{(t)})$$

It's trivial that since the term's derivation is itself, thus the solution of the term must contain natural exponential e. Usually, we guess the solution in the form of $Ae^{\lambda t}$

$$f_{lin}(\mathcal{X}, \theta^{(t)}) = Ae^{\eta T^{(0)}_{\mathcal{X}\mathcal{X}} t}$$

Consider the training time step $t = 0$.

$$f_{lin}(\mathcal{X}, \theta^{(0)}) = Ae^{\eta T^{(0)}_{\mathcal{X}\mathcal{X}} 0} = A$$

Plug into the $Ae^{\eta T^{(0)}_{\mathcal{X}\mathcal{X}} t}$

$$f_{lin}(\mathcal{X}, \theta^{(t)}) = f_{lin}(\mathcal{X}, \theta^{(0)}) e^{\eta T^{(0)}_{\mathcal{X}\mathcal{X}} t}$$

Weight Dynamic

$$\theta^{(t+1)} - \theta^{(t)} = \eta \nabla_\theta f(\mathcal{X}, \theta^{(t)}) \nabla_{f(\mathcal{X}, \theta^{(t)})} \mathcal{L}^{(t)}(\mathcal{X}, \mathcal{Y})$$

We can write down the ODE of the dynamic $\dot{\theta}(t)$

$$\lim_{\Delta t \to 0} \frac{\theta^{(t+\Delta t)} - \theta^{(t)}}{\Delta t} = \frac{\partial \theta^{(t)}}{\partial t} = \dot{\theta}(t) = \eta \nabla_\theta f(\mathcal{X}, \theta^{(t)}) \nabla_{f(\mathcal{X}, \theta^{(t)})} \mathcal{L}^{(t)}(\mathcal{X}, \mathcal{Y})$$

Now, since the ODE $\dot{\theta}(t)$ is linear, it has cloesd form solution

$$\dot{\theta}(t) = \eta \nabla_\theta f(\mathcal{X}, \theta^{(t)}) \nabla_{f(\mathcal{X}, \theta^{(t)})} \sum_{x \in \mathcal{X},\ y \in \mathcal{Y}} \frac{1}{2} ||f(x, \theta^{(t)}) - y||_2^2$$

$$= \eta \nabla_\theta f(\mathcal{X}, \theta^{(t)}) \sum_{x \in \mathcal{X},\ y \in \mathcal{Y}} ||f(x, \theta^{(t)}) - y||_2$$

$$= \eta \nabla_\theta f(\mathcal{X}, \theta^{(t)}) (\hat{\mathcal{Y}}^{(t)} - \mathcal{Y})$$

Prediction of Trained NN

Gradient Kernel

Recall the NTK $T^{(0)}_{\mathcal{X}\mathcal{X}} = \nabla_\theta f(\mathcal{X}, \theta^{(0)}) \nabla_\theta f(\mathcal{X}, \theta^{(0)})^\top \in \mathbb{R}^{N \times N}$. Now we only consider single entry of output, and thus, $f(x, \theta) \in \mathbb{R}$. $\nabla_\theta f(\mathcal{X}, \theta) \in \mathbb{R}^{N \times S}$ is a Jacobian matrix.

$$T^{(0)}_{\mathcal{X}\mathcal{X}} = \nabla_\theta f(\mathcal{X}, \theta^{(0)}) \nabla_\theta f(\mathcal{X}, \theta^{(0)})^\top$$

The matrix multiplication can be written as the following matrix.

$$= \begin{bmatrix} \dot{k}^L(x_1, x_1) & \dot{k}^L(x_1, x_2) & ... & \dot{k}^L(x_1, x_N) \\ \dot{k}^L(x_2, x_1) & \dot{k}^L(x_2, x_2) & ... & \dot{k}^L(x_2, x_N) \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ \dot{k}^L(x_N, x_1) & \dot{k}^L(x_N, x_2) & ... & \dot{k}^L(x_N, x_N) \end{bmatrix}$$

where the function$\dot{k}^L(x, x') = \nabla_\theta f(x, \theta^{(0)})^\top \nabla_\theta f(x', \theta^{(0)})$ is the kernel function of NTK. For the gradient $\nabla_\theta f(x, \theta) \in \mathbb{R}^{S \times 1}$ of a single output network, define the kernel function.

$$\dot{k}^L(x,x') = \nabla_\theta f(x,\theta^{(0)})^\top \nabla_\theta f(x',\theta^{(0)})$$

$$= \nabla_{\theta^{\leq L}} h^L(x)^\top \nabla_{\theta^{\leq L}} h^L(x')$$

Divide the network along the depth of the layers $\nabla_{\theta^L} h^L(x)^\top = [\nabla_{\theta^L} h_1^L(x),\ \nabla_{\theta^{\leq L-1}} h^L(x)]$,

$$= \nabla_{\theta^L} h_1^L(x)^\top \nabla_{\theta^L} h_1^L(x') + \nabla_{\theta^{\leq L-1}} h^L(x)^\top \nabla_{\theta^{\leq L-1}} h^L(x')$$

Since $\nabla_{\theta^L} h^L(x)^\top = [\nabla_{\theta_1^L} h_1^L(x),\ \nabla_{\theta_2^L} h_2^L(x), \dots, \nabla_{\theta_{n_L}^L} h_{n_L}^L(x)] \in \mathbb{R}^{1\times n_L}$, the inner product of the vector can be written as a form of summation. Note that $\theta_i^{\leq l}$ represents the i-th entry of the l-th layer.

$$= \nabla_{\theta_1^L} h_1^L(x)^\top \nabla_{\theta_1^L} h_1^L(x') + (\nabla_{\theta^{\leq L-1}} h^{L-1}(x) \nabla_{h^{L-1}(x)} h^L(x))^\top (\nabla_{\theta^{\leq L-1}} h^{L-1}(x') \nabla_{h^{L-1}(x')} h^L(x'))$$

$$= \nabla_{\theta_1^L} h_1^L(x)^\top \nabla_{\theta_1^L} h_1^L(x') + \nabla_{h^{L-1}(x)} h^L(x)^\top \nabla_{\theta^{\leq L-1}} h^{L-1}(x)^\top \nabla_{\theta^{\leq L-1}} h^{L-1}(x') \nabla_{h^{L-1}(x')} h^L(x')$$

$$= \nabla_{\theta_1^L} h_1^L(x)^\top \nabla_{\theta_1^L} h_1^L(x') + \nabla_{h^{L-1}(x)} h^L(x)^\top \dot{k}^{L-1}(x,x') \nabla_{h^{L-1}(x')} h^L(x')$$

$$= k^L(x,x') + \dot{k}^{L-1}(x,x') \dot{\Gamma}(K_{xx'}^{L-1})$$

$$K_{xx'}^{l-1} = \begin{bmatrix} k^{l-1}(x,x) & k^{l-1}(x,x') \\ k^{l-1}(x',x) & k^{l-1}(x',x') \end{bmatrix}$$

In the following section, we'll dive into the detail of the recurrent structure of the NTK.

The First Term

In general, derive the inner product of gradient at i-th entry.

$$\nabla_{\theta_i^l} h_i^l(x)^\top \nabla_{\theta_i^l} h_i^l(x')$$

$$= \nabla_{\theta_i^l} \left(\frac{\sigma_w}{\sqrt{n_{l-1}}} \sum_{j=1}^{n_{l-1}} w_{i,j}^l \phi(h_j^{l-1}(x)) + \sigma_b \beta_i^l\right)^\top \nabla_{\theta_i^l} \left(\frac{\sigma_w}{\sqrt{n_{l-1}}} \sum_{j=1}^{n_{l-1}} w_{i,j}^l \phi(h_j^{l-1}(x')) + \sigma_b \beta_i^l\right)$$

$$= [\frac{\sigma_w}{\sqrt{n_{l-1}}} \sum_{j=1}^{n_{l-1}} \phi(h_j^{l-1}(x)),\ \sigma_b][\frac{\sigma_w}{\sqrt{n_{l-1}}} \sum_{j=1}^{n_{l-1}} \phi(h_j^{l-1}(x')),\ \sigma_b]^\top$$

Since they are identical entries(i-th), we can combine them into single term.

$$= \frac{\sigma_w^2}{n_{1-1}} (\sum_{j=1}^{n_{l-1}} \phi(h_j^{l-1}(x))\phi(h_j^{l-1}(x')) + \sum_{j\neq j'}^{n_{l-1}} \phi(h_j^{l-1}(x))\phi(h_{j'}^{l-1}(x'))) + \sigma_b^2$$

By central limit theorem(CLT), if $n_l \to \infty$, it will converge.

$$= \sigma_w^2 (E_{h_j^{l-1}(x), h_j^{l-1}(x') \sim \mathcal{N}(0, K_{xx'}^{l-1})}[\phi(h_j^{l-1}(x))\phi(h_j^{l-1}(x'))] + E_{h_j^{l-1}(x), h_{j'}^{l-1}(x') \sim \mathcal{N}(0, K_{xx'}^{l-1}), j\neq j'}[\phi(h_j^{l-1}(x))\phi(h_{j'}^{l-1}(x'))]$$

$$= \sigma_w^2 E_{u,v \sim \mathcal{N}(0, K_{xx'}^{l-1})}[\phi(u)\phi(v)] + E_{u',v' \sim \mathcal{N}(0, K_{xx'}^{l-1})}[\phi(u')\phi(v')] + \sigma_b^2$$

Because different entries are independent, thus $E_{u',v' \sim \mathcal{N}(0, K_{xx'}^{l-1})}[\phi(u')\phi(v')] = 0$.

$$= \sigma_w^2 E_{u,v \sim \mathcal{N}(0, K_{xx'}^{l-1})}[\phi(u)\phi(v)] + \sigma_b^2$$

$$= k^L(x, x') = \Gamma(K_{xx'}^{l-1})$$

$$K_{xx'}^{l-1} = \begin{bmatrix} k^{l-1}(x,x) & k^{l-1}(x,x') \\ k^{l-1}(x',x) & k^{l-1}(x',x') \end{bmatrix}$$

The Second Term

$$\nabla_{h^{l-1}(x)} h^l(x)^\top \dot{k}^{l-1}(x,x') \nabla_{h^{l-1}(x')} h^l(x')$$

$$= \dot{k}^{l-1}(x,x') \nabla_{h^{l-1}(x)} h^l(x)^\top \nabla_{h^{l-1}(x')} h^l(x')$$

$$= \dot{k}^{l-1}(x,x') \nabla_{h^{l-1}(x)} (\frac{\sigma_w}{\sqrt{n_l}} \sum_{j=1}^{n_l} w_{i,j}^l \phi(h_j^{l-1}(x)) + \sigma_b \beta_i^l)^\top \nabla_{h^{l-1}(x')} (\frac{\sigma_w}{\sqrt{n_l}} \sum_{j=1}^{n_l} w_{i,j}^l \phi(h_j^{l-1}(x')) + \sigma_b \beta_i^l)$$

Due to $\nabla_{h^{l-1}(x)} h^l(x)^\top = [\nabla_{h_1^{l-1}(x)} h^l(x), \nabla_{h_2^{l-1}(x)} h^l(x), ..., \nabla_{h_{n_l}^{l-1}(x)} h^l(x)] \in \mathbb{R}^{1\times n_l}$, $\nabla_{h^{l-1}(x)} h^l(x)^\top \nabla_{h^{l-1}(x')} h^l(x')$ is an inner product.

$$= \dot{k}^{l-1}(x,x')\frac{\sigma_w^2}{n_l}(\sum_{j=1}^{n_l} w_{i,j}^l \nabla_{h^{l-1}(x)}\phi(h_j^{l-1}(x)) w_{i,j}^l \nabla_{h^{l-1}(x')}\phi(h_j^{l-1}(x')))$$

$$= \dot{k}^{l-1}(x,x')\frac{\sigma_w^2}{n_l}(\sum_{j=1}^{n_l} w_{i,j}^l \nabla_{h^{l-1}(x)}\phi(h_j^{l-1}(x)) w_{i,j}^l \nabla_{h^{l-1}(x')}\phi(h_j^{l-1}(x')))$$

With central limit theorem(CLT), we can take an expectation over the equation. Since $w_{i,j}^l$ is independent to $h_j^{l-1}(x)$ and $h_j^{l-1}(x')$, we can separate the expectation $E[w_{i,j}^l w_{i,j}^l]$.

$$= \dot{k}^{l-1}(x,x')\sigma_w^2 E_{w_{i,j}^l \sim \mathcal{N}(0,1)}[w_{i,j}^l w_{i,j}^l] E_{h_j^{l-1}(x), h_j^{l-1}(x') \sim \mathcal{N}(0, \dot{K}_{xx'}^{l-1})}[\nabla_{h^{l-1}(x)}\phi(h_j^{l-1}(x)) \nabla_{h^{l-1}(x')}\phi(h_j^{l-1}(x'))]$$

$$= \dot{k}^{l-1}(x,x')\sigma_w^2 E_{h_j^{l-1}(x), h_j^{l-1}(x') \sim \mathcal{N}(0, \dot{K}_{xx'}^{l-1})}[\nabla_{h^{l-1}(x)}\phi(h_j^{l-1}(x)) \nabla_{h^{l-1}(x')}\phi(h_j^{l-1}(x'))]$$

$$= \dot{k}^{l-1}(x,x')\sigma_w^2 E_{u,v \sim \mathcal{N}(0, K_{xx'}^{l-1})}[\nabla_u \phi(u) \nabla_v \phi(v)]$$

$$= \dot{k}^{l-1}(x,x')\dot{\Gamma}(K_{xx'}^{l-1})$$

$$K_{xx'}^{l-1} = \begin{bmatrix} k^{l-1}(x,x) & k^{l-1}(x,x') \\ k^{l-1}(x',x) & k^{l-1}(x',x') \end{bmatrix}$$

The Inner Product Of The Derivative Activation Function

ReLU Activation Function The ReLU activation function is $\phi_{relu}(x) = \max(0,x)$ and the derivative is

$$\frac{d\phi_{relu}(x)}{dx} = H(x)$$

where $H(x)$ is Heaviside function.

$$H(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

Then, we can derive the kernel of the derivative ReLU which we won't cover in this article. For more detail, please refer to the original paper "Kernel Method for Deep Learning" on NIPS'09.

Then, the kernel of the ReLU function

$$\nabla_x \phi_{relu}(x) \nabla_y \phi_{relu}(y) = \frac{1}{2\pi} J_0(\theta_{xy}), \quad J_0(\theta_{xy}) = \pi - \theta_{xy}$$

$$\phi_{relu}(x)\phi_{relu}(y) = \frac{1}{2\pi} J_1(\theta_{xy}), \quad J_1(\theta_{xy}) = \sin\theta_{xy} + (\pi - \theta_{xy})\cos\theta_{xy}$$

where θxy is the angle between x and y.

$$\theta_{xy} = \arccos \frac{x \cdot y}{||x||\;||y||}$$

Erf Activation Function

$$\phi_{erf}(x)\phi_{erf}(y) = \frac{2}{\pi} \sin^{-1} \frac{2x \cdot y}{\sqrt{(1 + 2x \cdot c)(1 + 2y \cdot y)}}$$

$$\nabla_x \phi_{erf}(x) \nabla_y \phi_{erf}(y) = \frac{4}{\pi} \sin^{-1} det(I + 2\Sigma)^{-\frac{1}{2}}$$

Also, the derivation won't cover in this article. For more detail, please refer to the paper "Computing with infinite networks" on NIPS'97

Neural Tangent Kernel After Training