

The Neural Tangent Kernel

In the previous lecture, we derived the following heuristic that tells us how much each layer contributes to the function value:

$$\delta f_\theta(x) \approx -\eta_0 \ell'(f_\theta(x), y) \sum_{i=1}^L \underbrace{\|\nabla_{\theta_i} f_\theta(x)\|^2}_{\text{contribution of group } i}$$

For a two layer neural network $f_\theta(x) = a^\top \sigma(Wx)$ under standard parameterization, we derived:

$$\|\nabla_a f_\theta(x)\|^2 = \Theta(m) \quad \|\nabla_W f_\theta(x)\|_F^2 = \Theta(d).$$

The largest stable learning rate is set by the larger of the two, so $\eta_0 = \Theta(1/m)$, and at this learning rate the first layer contributes only $\Theta(d/m)$ to the change in the predictions. Therefore, as $m \rightarrow \infty$, the first layer makes no contribution to the output at all, reducing the model to a random features model. We then used this observation to define its infinite width limit.

The last layer needs a learning rate of $O(1/m)$ to train stably, but this freezes the first layer. The fix is obvious: **give each layer its own learning rate**. With per-layer learning rates η_a, η_W :

$$\delta f_\theta(x) \approx -\ell'(f, y) \left[\underbrace{\eta_a \|\nabla_a f_\theta(x)\|^2}_{\Theta(\eta_a m)} + \underbrace{\eta_W \|\nabla_W f_\theta(x)\|_F^2}_{\Theta(\eta_W d)} \right].$$

With d fixed, both layers contribute $\Theta(1)$ if and only if $\eta_a = \Theta(1/m)$ and $\eta_W = \Theta(1)$. In other words, SP used the right learning rate η_a , but η_W was far too small when m is large.

Larger depth

Let's extend our heuristic to an MLP with L layers and hidden widths $m_1, \dots, m_{L-1}$. Write $m_0 = d$ and $m_L = 1$, so that $W_\ell \in \mathbb{R}^{m_\ell \times m_{\ell-1}}$:

$$h_1 = W_1 x \quad h_\ell = W_\ell \sigma(h_{\ell-1}) \quad (2 \leq \ell \leq L) \quad f_\theta(x) = h_L, \\ (W_\ell)_{ij} \sim N(0, 1/m_{\ell-1}).$$

The output layer $W_L \in \mathbb{R}^{1 \times m_{L-1}}$ plays the role of the readout $a^\top$ in the two layer network. Recall that fan-in initialization keeps the coordinates of h_ℓ and $\sigma(h_\ell)$ at order one.

Backward pass. Write $g_\ell := \nabla_{h_\ell} f_\theta(x) \in \mathbb{R}^{m_\ell}$ for the backpropagated signal. The chain rule gives

$$g_L = 1 \quad g_{\ell-1} = \sigma'(h_{\ell-1}) \odot (W_\ell)^\top g_\ell, \\ \nabla_{W_1} f_\theta(x) = g_1 x^\top \quad \nabla_{W_\ell} f_\theta(x) = g_\ell \sigma(h_{\ell-1})^\top.$$

Incorrect calculation. The output layer gives $g_{L-1} = \sigma'(h_{L-1}) \odot W_L^\top$, whose coordinates have size $m_{L-1}^{-1/2}$, so $\|g_{L-1}\| = \Theta(1)$. **Conditional on g_ℓ , each coordinate of $(W_\ell)^\top g_\ell$ is $N(0, \|g_\ell\|^2/m_{\ell-1})$.** Summing squares over $m_{\ell-1}$ coordinates gives order one. Multiplying by $\sigma'(h_{\ell-1})$ preserves the size of each coordinate, so inductively g_ℓ has coordinate scale $m_\ell^{-1/2}$ and norm $\Theta(1)$.

These are informal course notes and have not undergone peer review. AI tools assisted with simplifying proofs, editing, and typesetting. I take responsibility for the content, and corrections are welcome.

Table 1: Signal propagation under fan-in initialization (hidden layers $\ell < L$)

	Coordinates	L_2 norm
Forward h_ℓ	$\Theta(1)$	$\Theta(\sqrt{m_\ell})$
Backward g_ℓ	$\Theta(m_\ell^{-1/2})$	$\Theta(1)$

This calculation gives the right answer, for the wrong reason. The line in red fails because g_ℓ depends on W_ℓ through the forward activations. This is known as the gradient independence assumption, which gives the correct infinite-width limit here under NTK parameterization (Yang, 2020).

Instead, we will compute a single coordinate $(g_\ell)_j$ by tracking how a perturbation of the j th coordinate of h_ℓ changes the output. If we replace h_ℓ by $h_\ell + \varepsilon e_j$, the perturbation starts with RMS norm $\varepsilon/\sqrt{m_\ell}$. Each linear layer asymptotically maintains the RMS norm (this is the purpose of fan-in initialization), and each activation changes it by a constant factor, which means the output changes by order $\varepsilon/\sqrt{m_\ell}$. Therefore, the j th entry of g_ℓ must have order $1/\sqrt{m_\ell}$, so $\|g_\ell\| = \Theta(1)$.

This demonstrates that for $2 \leq \ell \leq L$, including the output layer where $g_L = 1$, the gradient has squared Frobenius norm

$$\|\nabla_{W_\ell} f\|_F^2 = \|g_\ell \sigma(h_{\ell-1})^\top\|_F^2 = \underbrace{\|g_\ell\|^2}_{\Theta(1)} \underbrace{\|\sigma(h_{\ell-1})\|^2}_{\Theta(m_{\ell-1})} = \Theta(m_{\ell-1}).$$

The same calculation gives $\|\nabla_{W_1} f\|_F^2 = \|g_1\|^2 \|x\|^2 = \Theta(m_0)$ for the first layer. Each layer should therefore use a learning rate of $\Theta(1/m_{\ell-1})$.

Table 1 is a useful reference for fan-in initialization, but will change once we move to μ P.

NTK Parameterization

We saw that one way to derive the NTK parameterization is to use fan-in initialization and learning rate $\eta_W = \eta_0/\text{fan in}$. However, the same dynamics are often written using *forward multipliers*:

Proposition 1 (learning rates as forward multipliers). *Let W be a parameter with initialization $W^{(0)}$, and let $c > 0$ be a learning rate multiplier. Initialize a new parameter $\widehat{W}^{(0)} := c^{-1/2} W^{(0)}$, and use $\sqrt{c} \widehat{W}$ in place of W in the forward pass: $\widehat{L}(\widehat{W}) := L(\sqrt{c} \widehat{W})$. Then the updates*

$$\begin{array}{ll} W \leftarrow W - c\eta_0 \nabla_W L(W) & \widehat{W} \leftarrow \widehat{W} - \eta_0 \nabla_{\widehat{W}} \widehat{L}(\widehat{W}) \\ \text{LR multipliers} & \text{Forward multipliers} \end{array}$$

satisfy $W^{(t)} = \sqrt{c} \widehat{W}^{(t)}$ for all $t \geq 0$.

Proof. Induction on t and the chain rule. □

Taking $c = 1/\text{fan in}$ gives two equivalent ways to write the NTK parameterization:

Definition 1 (NTK parameterization). For each layer (Jacot et al., 2018):

	Initialization	Multiplier	Learning rate
LR multipliers	$W_{ij} \sim N(0, 1/\text{fan in})$	1	$\eta_0/\text{fan in}$
Forward multipliers	$W_{ij} \sim N(0, 1)$	$1/\sqrt{\text{fan in}}$	η_0

For now, we will use **LR multipliers**. For our two layer network, this means $f_\theta(x) = a^\top \sigma(Wx)$, with $a_j \sim N(0, 1/m)$, $W_{ij} \sim N(0, 1/d)$, and learning rates $\eta_a = \eta_0/m$, $\eta_W = \eta_0/d$.

The Neural Tangent Kernel

The NTK parameterization was designed to ensure that every layer contributes $O(1)$ to the model's output. However, surprisingly, this does **not** imply that the intermediate weights are moving significantly or that the network is learning new features!

Consider an intermediate layer $W_\ell \in \mathbb{R}^{m_\ell \times m_{\ell-1}}$ and one step of gradient descent on an example (x, y) . Holding the layer's input $\sigma(h_{\ell-1})$ fixed, its weight update changes the preactivations by

$$\delta h_\ell = (\delta W_\ell) \sigma(h_{\ell-1}) = - \underbrace{\eta_0 \ell'(f_\theta(x), y)}_{O(1)} \underbrace{\frac{\|\sigma(h_{\ell-1})\|^2}{m_{\ell-1}}}_{\Theta(1)} g_\ell.$$

Because $\|g_\ell\|_\infty = \tilde{O}(m_\ell^{-1/2})$, $\|\delta h_\ell\|_\infty = \tilde{O}(m_\ell^{-1/2})$. **Thus, training this layer changes every activation by a vanishing amount.** This is the main source of confusion with NTK: if the activations h_ℓ barely change, how can the function change and the loss go down?

The key is that the effect of δh_ℓ on the function output is measured by g_ℓ through the inner product $\langle g_\ell, \delta h_\ell \rangle$. Both vectors have order-one norm, with typical coordinates of order $1/\sqrt{m_\ell}$, and the formula for δh_ℓ shows that they are perfectly aligned up to a sign. Each neuron therefore changes by order $1/\sqrt{m_\ell}$, and propagating through g_ℓ , this changes the function value by order $1/m_\ell$. Because there are m_ℓ neurons and their contributions have the same sign, these add up to a $\Theta(1)$ effect. The output remains sensitive to these small, coordinated changes in the intermediate activations.

In fact, the activations change as much as they can without blowing up the loss. If each activation changed by $O(1)$ in the gradient direction, the effect on the output $\delta f \approx \langle g_\ell, \delta h_\ell \rangle$ would scale like $\sqrt{m_\ell}$, so the function output would diverge. A proper feature learning limit therefore requires scaling down g_ℓ so that we can scale up δh_ℓ . This is the main insight behind μP (Yang and Hu, 2021).

The next observation is that because the activations don't change, **the model gradients remain fixed at their initialization values throughout training**, in the infinite-width limit. To see why, return to our two layer network. For a fixed input x , write $h_j = w_j \cdot x$. The gradients are

$$\frac{\partial f_\theta(x)}{\partial a_j} = \sigma(h_j) \quad \nabla_{w_j} f_\theta(x) = a_j \sigma'(h_j) x.$$

We already know that h_j barely changes, so both $\sigma(h_j)$ and $\sigma'(h_j)$ barely change, assuming a smooth activation with bounded first and second derivatives. The only other factor is a_j . Its learning rate

is η_0/m , so each update changes it by $O(1/m)$, compared with its initial scale $1/\sqrt{m}$. Over a fixed number of training steps, this is a vanishing fraction of its initial scale. Thus, every factor in the gradient stays close to its initial value, and so does the gradient itself (Chizat et al., 2019).

With the gradient frozen, the network is effectively linearized around initialization:

$$f_\theta(x) \approx f_\theta^{\text{lin}}(x) := \underbrace{f_{\theta(0)}(x)}_{\text{init}} + \underbrace{\langle \nabla_\theta f_{\theta(0)}(x), \theta - \theta^{(0)} \rangle}_{\text{tangent features}}.$$

The following theorem makes this precise.

Theorem 1 (NTK linearization, informal, Jacot et al., 2018; Lee et al., 2019; Yang and Littwin, 2021). *Let f_θ be a two layer neural network in the NTK parameterization, and assume σ' , σ'' , and ℓ'' are bounded. Train the network and its linearization using SGD on a fixed dataset, with the same initialization, learning rates, and sequence of samples. Let f_t and f_t^{lin} denote the two models after t steps. Then for any fixed T and input x , with high probability,*

$$\max_{0 \leq t \leq T} |f_t(x) - f_t^{\text{lin}}(x)| \leq \frac{C_T}{\sqrt{m}}.$$

Here C_T can depend on T and the data, but not the width m .

We can now study SGD on this linear model. Recall that SGD on a linear model has an equivalent description in terms of a kernel. We can derive the same description with learning rate multipliers. Let c_i be the learning rate multiplier for parameter group θ_i , so its learning rate is $\eta_0 c_i$. If we take a step of SGD on a sample (x, y) and then evaluate the result on an example x' , we get

$$\delta f_\theta^{\text{lin}}(x') = \langle \nabla_\theta f_\theta^{\text{lin}}(x'), \delta \theta \rangle = -\eta_0 \ell'(f_\theta^{\text{lin}}(x), y) \sum_i c_i \langle \nabla_{\theta_i} f_{\theta(0)}(x'), \nabla_{\theta_i} f_{\theta(0)}(x) \rangle.$$

Here we used that the linear model's gradients are fixed at initialization. We can therefore define the **neural tangent kernel** as a sum of contributions from the parameter groups:

$$K(x, x') := \sum_i K_i(x, x'), \quad K_i(x, x') := c_i \langle \nabla_{\theta_i} f_{\theta(0)}(x), \nabla_{\theta_i} f_{\theta(0)}(x') \rangle.$$

Then the prediction update is simply

$$\delta f_\theta^{\text{lin}}(x') = -\eta_0 \ell'(f_\theta^{\text{lin}}(x), y) K(x, x').$$

The kernel tells us how much training on x changes the prediction at x' . This can also be interpreted through the normalized feature map:

$$\psi(x) := (\sqrt{c_i} \nabla_{\theta_i} f_{\theta(0)}(x))_i, \quad K(x, x') = \langle \psi(x), \psi(x') \rangle.$$

With forward multipliers, this is just the gradient at initialization, without the extra c_i factors, which is why this convention is often preferred.

The Infinite-Width Neural Tangent Kernel

Two-layer networks. Write $K^{(0)}$ for the kernel at initialization and $f_0 := f_{\theta(0)}$. The NTK of the two layer network at initialization splits into the two per-layer kernels:

$$K^{(0)}(x, x') = \underbrace{\frac{1}{m} \sum_{j=1}^m \sigma(w_j \cdot x) \sigma(w_j \cdot x')}_{K^{\text{RF}}(x, x')} + \underbrace{\frac{x \cdot x'}{d} \sum_{j=1}^m a_j^2 \sigma'(w_j \cdot x) \sigma'(w_j \cdot x')}_{\text{first layer}}.$$

The second sum is also an average: write $a_j^2 = (ma_j^2)/m$, where the i.i.d. variables ma_j^2 have mean one. The law of large numbers therefore gives

$$K^{(0)}(x, x') \longrightarrow K^\infty(x, x') := \mathbb{E}_w[\sigma(w \cdot x) \sigma(w \cdot x')] + \frac{x \cdot x'}{d} \mathbb{E}_w[\sigma'(w \cdot x) \sigma'(w \cdot x')].$$

Here $w \sim N(0, I_d/d)$. These are the two expectations from the previous lecture, and $x \cdot x'/d = \|x\|_{\text{RMS}} \|x'\|_{\text{RMS}} \cos \theta$. For ReLU, the first is the ReLU arc-cosine kernel, and since $\sigma'(h) = \mathbf{1}_{h \geq 0}$ the second is the step kernel. With θ the angle between x and x' ,

$$K^\infty(x, x') = \frac{\|x\|_{\text{RMS}} \|x'\|_{\text{RMS}}}{2\pi} [\sin \theta + (\pi - \theta) \cos \theta] + \frac{x \cdot x'}{d} \frac{\pi - \theta}{2\pi}.$$

Note. There are now two different kernels at initialization, playing two different roles. Conditional on the first layer, the distribution of the *initial outputs* is $f_0 \mid W \sim N(0, K^{\text{RF}})$. The dynamics are governed by the NTK $K^\infty = K^{\text{RF}} + K_1$, where K_1 is the NTK of the first layer.

Deep networks. We can now generalize the NTK to networks of arbitrary fixed depth. We use the depth L network from above, with Gaussian fan-in initialization and learning rates $\eta_{W_\ell} = \eta_0/m_{\ell-1}$.

Proposition 2 (deep NTK, [Jacot et al., 2018](#); [Yang, 2020](#)). Assume σ is continuously differentiable, with σ and σ' of polynomial growth. As all hidden widths grow with fixed positive ratios, the NTK at initialization converges in probability to $K^\infty = \mathcal{K}_L$, where

$$\mathcal{K}_1 = H_1, \quad \mathcal{K}_\ell = H_\ell + \dot{H}_\ell \mathcal{K}_{\ell-1}.$$

The forward covariances are computed recursively by

$$H_1(x, x') = \frac{\langle x, x' \rangle}{d}, \quad H_\ell(x, x') = \mathbb{E}[\sigma(U) \sigma(V)], \quad \dot{H}_\ell(x, x') = \mathbb{E}[\sigma'(U) \sigma'(V)],$$

for $2 \leq \ell \leq L$, where

$$(U, V) \sim N\left(0, \begin{pmatrix} H_{\ell-1}(x, x) & H_{\ell-1}(x, x') \\ H_{\ell-1}(x', x) & H_{\ell-1}(x', x') \end{pmatrix}\right).$$

Proof. Homework 1. □

Here $H_\ell(x, x')$ is the limit of $\frac{1}{m_\ell} \langle h_\ell(x), h_\ell(x') \rangle$ for $\ell < L$. For depth 2, this reduces to $K^\infty = H_2 + H_1 \dot{H}_2$, exactly the two-layer formula above.

Note. For equal hidden widths, deep SP at learning rate η_0/m scales the first layer's NTK learning rate down by d/m . Its infinite-width kernel therefore uses the same recursion with $\mathcal{K}_1 = 0$: the first layer contributes no tangent features.

Global convergence

Most coupling results in this course, including Theorem 1, only hold for a fixed number of steps T . However, we can extend this to a global convergence result by arguing that the linearized model converges quickly enough to keep the NTK coupling valid throughout training. This is one of the few general settings in which we can prove that neural networks converge to global minimizers of the training loss (Du et al., 2019; Allen-Zhu et al., 2019; Oymak and Soltanolkotabi, 2019a).

For the convergence argument, it is simpler to use a common learning rate and ordinary Euclidean distance in parameter space. We therefore switch to **forward multipliers**:

$$f_\theta(x) = \frac{1}{\sqrt{m}} a^\top \sigma\left(\frac{1}{\sqrt{d}} Wx\right), \quad a_j, W_{ij} \sim N(0, 1), \quad \eta_a = \eta_W = \eta_0.$$

The training dynamics and the NTK are unchanged. The multipliers are now part of the forward pass, so the kernel is simply $K_\theta(x, x') = \langle \nabla_\theta f_\theta(x), \nabla_\theta f_\theta(x') \rangle$.

For this proof, we will use the MSE loss and directly study gradient flow:

$$L(\theta) = \frac{1}{2n} \|f_\theta(X) - y\|^2, \quad \frac{d\theta}{dt} = -\nabla_\theta L(\theta).$$

Here we have absorbed the base learning rate η_0 into time. We will use the shorthand $f_t(x) := f_{\theta(t)}(x)$, $r(t) := f_{\theta(t)}(X) - y$, and $K(t) := K_{\theta(t)}(X, X)$, omitting time arguments in differential equations.

Theorem 2 (global convergence). *Assume the inputs have bounded RMS norm and σ' and σ'' are bounded. Let $\lambda := \lambda_{\min}(K^\infty(X, X)) > 0$ and $L_0 := L(\theta(0))$. If $m = \tilde{\Omega}(n^2/\lambda^2)$, then with high probability over initialization, gradient flow satisfies, for every $t \geq 0$,*

$$L(\theta(t)) \leq e^{-\lambda t/n} L_0.$$

Proof. First, we need to show that the kernel at initialization is well conditioned. For a fixed pair of inputs x, x' , write $h_j(x) = w_j \cdot x/\sqrt{d}$. With all variables at initialization,

$$K(0; x, x') = \frac{1}{m} \sum_{j=1}^m \left[\sigma(h_j(x)) \sigma(h_j(x')) + a_j^2 \frac{\langle x, x' \rangle}{d} \sigma'(h_j(x)) \sigma'(h_j(x')) \right].$$

This is an average of m i.i.d. neuron contributions, with expectation $K^\infty(x, x')$. Since σ' is bounded and the initialization is Gaussian, these contributions have sub-exponential tails. Standard scalar concentration therefore gives fluctuations of order $1/\sqrt{m}$, up to logarithmic factors. A union bound over the n^2 pairs of training inputs gives, with high probability,

$$\max_{i,k} |K(0; x_i, x_k) - K^\infty(x_i, x_k)| = \tilde{O}(m^{-1/2}).$$

Converting this entrywise bound to an operator norm bound,

$$\|K(0) - K^\infty(X, X)\|_{\text{op}} \leq \|K(0) - K^\infty(X, X)\|_F \leq n \max_{i,k} |K_{ik}(0) - K_{ik}^\infty| = \tilde{O}(n/\sqrt{m}).$$

Thus, if $m = \tilde{\Omega}(n^2/\lambda^2)$, then $\lambda_{\min}(K(0)) \geq 3\lambda/4$.

Next, we need to control how much the kernel can change during training. To do this, we will fix a radius R to be chosen later and bound how much the gradients change in the ball:

$$B_R := \{\theta : \|\theta - \theta(0)\| \leq R\}.$$

Fix an input x and write $h = Wx/\sqrt{d}$. For any $\theta \in B_R$, boundedness of the input RMS norm gives

$$\|h - h(0)\| = \|(W - W(0))x/\sqrt{d}\| \leq \|W - W(0)\|_F \|x\|_{\text{RMS}} \lesssim R.$$

The gradient with respect to a is $\nabla_a f_\theta(x) = \sigma(h)/\sqrt{m}$. Since σ' is bounded,

$$\|\nabla_a f_\theta(x) - \nabla_a f_{\theta(0)}(x)\| = \frac{1}{\sqrt{m}} \|\sigma(h) - \sigma(h(0))\| \lesssim \frac{1}{\sqrt{m}} \|h - h(0)\| \lesssim \frac{R}{\sqrt{m}}.$$

The gradient with respect to W is

$$\nabla_W f_\theta(x) = \frac{1}{\sqrt{m}} (a \odot \sigma'(h)) \frac{x^\top}{\sqrt{d}}.$$

Here both a and h change, so we write

$$a \odot \sigma'(h) - a(0) \odot \sigma'(h(0)) = (a - a(0)) \odot \sigma'(h) + a(0) \odot (\sigma'(h) - \sigma'(h(0))).$$

Using boundedness of σ' for the first term and σ'' for the second gives

$$\|\nabla_W f_\theta(x) - \nabla_W f_{\theta(0)}(x)\|_F \lesssim \frac{1}{\sqrt{m}} (\|a - a(0)\| + \|a(0)\|_\infty \|h - h(0)\|).$$

Both differences on the right are $\lesssim R$, and Gaussian initialization gives $\|a(0)\|_\infty = \tilde{O}(1)$ with high probability. Thus the change in $\nabla_W f$ is $\tilde{O}(R/\sqrt{m})$ as well. Combining the two gradient blocks by the triangle inequality, $\|\nabla_\theta f_\theta(x) - \nabla_\theta f_{\theta(0)}(x)\| = \tilde{O}(R/\sqrt{m})$. Now, let $J_\theta \in \mathbb{R}^{n \times p}$ have rows $\nabla_\theta f_\theta(x_i)^\top$, and write $J(0) := J_{\theta(0)}$. Applying this bound to each of the n inputs gives

$$\|J_\theta - J(0)\|_F^2 = \sum_{i=1}^n \|\nabla_\theta f_\theta(x_i) - \nabla_\theta f_{\theta(0)}(x_i)\|^2 \leq \tilde{O}\left(\frac{nR^2}{m}\right).$$

Therefore, we can choose $R = \tilde{\Theta}(\sqrt{m\lambda/n})$ so that, throughout B_R ,

$$\|J_\theta - J(0)\|_{\text{op}} \leq \|J_\theta - J(0)\|_F \leq \sqrt{3\lambda/4} - \sqrt{\lambda/2}.$$

This keeps the smallest eigenvalue of $K_\theta = J_\theta J_\theta^\top$ at least $\lambda/2$, since

$$\sigma_{\min}(J_\theta) \geq \sqrt{3\lambda/4} - \|J_\theta - J(0)\|_{\text{op}} \geq \sqrt{\lambda/2}, \quad K_\theta \succeq \frac{\lambda}{2} I.$$

It remains to show that training never leaves this ball.

We now use a first-exit argument (Oymak and Soltanolkotabi, 2019b). Let τ be the first time the parameters reach the boundary of B_R . For $t < \tau$, $K(t) \succeq \frac{\lambda}{2} I$, so

$$\frac{dL}{dt} = -\|\nabla L\|^2 = -\frac{1}{n^2} r^\top K r \leq -\frac{\lambda}{n} L,$$

so $L(\theta(t)) \leq e^{-\lambda t/n} L_0$ for $t < \tau$. Now bound how far the parameters can move before τ . Let s denote distance traveled along the parameter trajectory, so $ds/dt = \|\dot{\theta}\| = \|\nabla L\|$. While $L > 0$,

$$\frac{dL}{ds} = \frac{dL/dt}{ds/dt} = -\|\nabla L\| \leq -\sqrt{\frac{\lambda}{n}} \sqrt{L}.$$

If we now look at $\sqrt{L}$ instead, the right-hand side becomes a constant:

$$\frac{d\sqrt{L}}{ds} = \frac{1}{2\sqrt{L}} \frac{dL}{ds} \leq -\frac{1}{2} \sqrt{\frac{\lambda}{n}}.$$

Thus $\sqrt{L}$ decreases by at least a constant amount per unit distance traveled. Since it starts at the finite value $\sqrt{L_0}$ and cannot become negative, the path before τ has finite length:

$$\|\theta(t) - \theta(0)\| \leq s(t) \leq 2\sqrt{\frac{nL_0}{\lambda}}.$$

Because $L_0 = \tilde{O}(1)$, we can take the implicit constant in $m = \tilde{\Omega}(n^2/\lambda^2)$ sufficiently large that this bound on $\|\theta(t) - \theta(0)\|$ is less than R . Therefore, τ cannot be finite. If it were, continuity would place the parameters at time τ strictly inside the ball, a contradiction. $\square$

Note. Theorem 2 does not apply to ReLU because σ' is not Lipschitz. The ball argument still works, but controlling the change in J requires bounding the number of neurons whose preactivations change sign, rather than using Lipschitz continuity of σ' . See [Du et al. \(2019\)](#).

Memorizing random data

The convergence theorem requires a lower bound on the smallest eigenvalue of the infinite-width NTK. For random data, this holds even with far more samples than input dimensions.

Proposition 3 (informal version of Theorems 3.1 and 3.2 in [Montanari and Zhong \(2022\)](#)). *Let $x_1, \dots, x_n$ be i.i.d. uniform on the sphere of radius $\sqrt{d}$, with $n = d^{O(1)}$, and let σ be ReLU. If $m = \tilde{\Omega}(1 + n/d)$, then with high probability as $d \rightarrow \infty$,*

$$\lambda_{\min}(K^\infty) \asymp \lambda_{\min}(K^m) = \Theta(1).$$

Here K^m is the width- m NTK at initialization.

The first conclusion is that $\lambda_{\min}(K^\infty)$ is order one. A bound of this kind lets us apply Theorem 2 at width $m = \tilde{\Omega}(n^2)$. However, the second part shows that the linearized model can memorize at a much smaller width: for $n \geq d$, we only need $m = \tilde{\Omega}(n/d)$, or equivalently $md = \tilde{\Omega}(n)$. The left-hand side is the number of parameters and the right-hand side is the number of samples. Therefore, up to logarithmic factors, the NTK can memorize one sample per parameter, which is near optimal!

While we have focused on regression, the NTK is also remarkably effective at classification and factual recall (e.g. mapping countries to capitals). For example, [Nichani et al. \(2025\)](#) show that the NTK with random token embeddings can memorize n associations $x \mapsto y$ under the decision rule $\hat{y} = \arg \max_y f_y(x)$ using only $\tilde{O}(n)$ parameters, which is within a $\tilde{O}(1)$ factor of the optimal limit.

References

- Z. Allen-Zhu, Y. Li, and Z. Song. A Convergence Theory for Deep Learning via Over-Parameterization. In *ICML*, 2019. URL <https://arxiv.org/abs/1811.03962>.
- L. Chizat, E. Oyallon, and F. Bach. On Lazy Training in Differentiable Programming. In *NeurIPS*, 2019. URL <https://arxiv.org/abs/1812.07956>.
- S. S. Du, X. Zhai, B. Póczos, and A. Singh. Gradient Descent Provably Optimizes Over-parameterized Neural Networks. In *ICLR*, 2019. URL <https://arxiv.org/abs/1810.02054>.
- A. Jacot, F. Gabriel, and C. Hongler. Neural Tangent Kernel: Convergence and Generalization in Neural Networks. In *NeurIPS*, 2018. URL <https://arxiv.org/abs/1806.07572>.
- J. Lee, L. Xiao, S. 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 *NeurIPS*, 2019. URL <https://arxiv.org/abs/1902.06720>.
- A. Montanari and Y. Zhong. The Interpolation Phase Transition in Neural Networks: Memorization and Generalization under Lazy Training. *Annals of Statistics*, 2022. URL <https://arxiv.org/abs/2007.12826>.
- E. Nichani, J. D. Lee, and A. Bietti. Understanding Factual Recall in Transformers via Associative Memories. In *ICLR*, 2025. URL <https://openreview.net/forum?id=hwSmPOAmhk>.
- S. Oymak and M. Soltanolkotabi. Towards Moderate Overparameterization: Global Convergence Guarantees for Training Shallow Neural Networks. arXiv, 2019a. URL <https://arxiv.org/abs/1902.04674>.
- S. Oymak and M. Soltanolkotabi. Overparameterized Nonlinear Learning: Gradient Descent Takes the Shortest Path? In *ICML*, volume 97, pages 4951–4960, 2019b. URL <https://proceedings.mlr.press/v97/oymak19a.html>.
- G. Yang. Tensor Programs II: Neural Tangent Kernel for Any Architecture. arXiv, 2020. URL <https://arxiv.org/abs/2006.14548>.
- G. Yang and E. J. Hu. Tensor Programs IV: Feature Learning in Infinite-Width Neural Networks. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 11727–11737. PMLR, 2021. URL <https://proceedings.mlr.press/v139/yang21c.html>.
- G. Yang and E. Littwin. Tensor Programs IIb: Architectural Universality of Neural Tangent Kernel Training Dynamics. In *ICML*, volume 139, pages 11762–11772, 2021. URL <https://proceedings.mlr.press/v139/yang21f.html>.