CS 2243 Fall 2024
10/09

Scribes: <names>
Based on notes by: Xiaodong Yang

Lecture 11: Linearized Networks

1 Introduction

Gradient Descent. Given i.i.d. samples $\{(x_i, y_i) : 1 \le i \le n\}$ drawn from some distribution on $\mathbb{R}^d \times \mathbb{R}$, we set up a student network $f_\theta(x)$ and initialize the parameters θ randomly. Then we run gradient descent on the empirical loss $\hat{L}(\theta) = \frac{1}{n}\sum_{i=1}^n (f_\theta(x_i) - y_i)^2$,

$$\theta \leftarrow \theta - \eta \cdot \nabla \hat{L}(\theta),$$

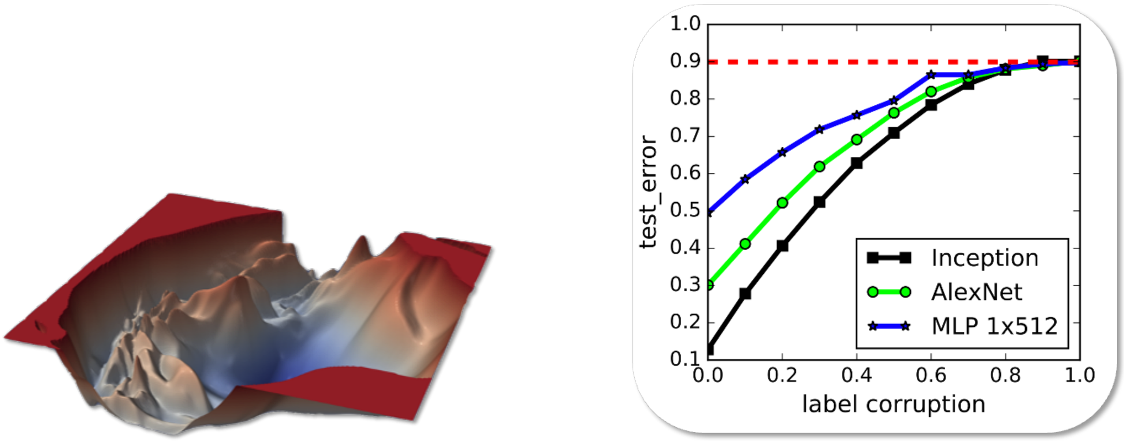
until convergence. Variants of this gradient method include: full-batch, mini-batch, online, adaptive step size, etc. Here we focus on one hidden layer student networks (which already accounts for overwhelming majority of the literature): for $\theta = \{\theta_i\}_{i=1}^N$ where $\theta_i = (a_i, w_i)$, define

$$f_\theta(x) = \gamma \sum_{i=1}^N a_i \cdot \sigma(\langle w_i, x \rangle),$$

where $\sigma(\cdot)$ is a fixed 1D activation, γ is a fixed scaling parameter. We extract γ from the $(a_1, \ldots, a_N)$ mainly to see how scaling shapes the training dynamics. In practice, N is taken to be very large, often much larger than d (overparametrization). Two mysteries emerge from practice.

1. **Optimization.** Empirical loss is a (highly) non-convex function. Yet we are able to efficiently converge to zero empirical loss (i.e. perfectly fit all the training data). See Figure 1a for an illustration of the non-convex landscape.

2. **Generalization.** Highly overparametrized models trained to achieve zero empirical loss also achieve small test loss, i.e. they do not overfit! Figure 1b (taken from [ZBH+16]) displays the generalization ability of deep networks under label corruption.



(a) Empirical Loss Optimization Landscape

(b) Generalization under Label Corruption

Figure 1: Mysteries of GD Training

Thereafter, we have to exploit overparametrization in the analysis. As empirical evidence suggests “the more parameters the better”, we come up with the following hypothesis:

As number N of parameters tending to infinity, training dynamics converge to a limiting process with the desired optimization / generalization properties. The only role of overparametrization is to bring us closer and closer to this limiting process.

For provable guarantees, it suffices to (i) bound error in approximating limiting process in terms of N; (ii) prove limiting process converges quickly to small train/test error. In this and next lectures, we will see two regimes in which people have tried implementing this recipe:

1. **Lazy Training/ linearized neural networks/ kernel limit:** Network is well approximated by its first order Taylor expansion around the parameters at initialization. But this regime is bottlenecked at kernel methods. Related works include [JGH18, DZPS18, DLL+19, AZLS19, ZCZG20, COB19].
2. **Non-lazy training / feature learning regime / mean field limit:** Training dynamics are highly nonlinear. Learned features (i.e. first layer weights) are very different from those at initialization. Related works include [MMN18, AZL22].

In this lecture, we would focus on the first regime.

1.1 Linearized Networks

Suppose parameters at initialization are given by $\theta_i^0 = (a_i^0, w_i^0)$. Take $\theta = \theta^0 + \Delta$ to deviate from the initialization a little,

$$\theta_i = \left(a_i^0 + [\Delta a_i], w_i^0 + [\Delta w_i]\right).$$

Then the Taylor expansion of f_θ around θ^0 is given by

$$\begin{aligned} f_\theta &= f_{\theta^0} + \langle \Delta, \nabla_\theta f_\theta \rangle + \cdots \\ &= f_{\theta^0} + \underbrace{\gamma \sum_{i=1}^{N} [\Delta a_i] \cdot \sigma(\langle w_i^0, \cdot \rangle)}_{\text{random features model}} + \underbrace{\gamma \sum_{i=1}^{N} a_i^0 \langle [\Delta w_i], \cdot \rangle \cdot \sigma'(\langle w_i^0, \cdot \rangle)}_{\text{neural tangent model}} + \cdots \end{aligned}$$

Therefore, we formally define random feature maps by

$$\phi_{\mathrm{RF}}(x) = \gamma \cdot \left(\sigma(\langle w_1^0, x \rangle); \ldots ; \sigma(\langle w_N^0, x \rangle)\right);$$

and neural tangent feature maps by

$$\phi_{\mathrm{NT}}(x) = \frac{\gamma}{\sqrt{d}} \cdot \left(a_1^0 \sigma'(\langle w_1^0, x \rangle) \cdot x^\top; \ldots ; a_N^0 \sigma'(\langle w_N^0, x \rangle) \cdot x^\top\right).$$

These are random functions whose parameters only depend on the weights of the network at initialization. Given this notion, the previous expansion is reinterpreted into

$$f_{\theta+\Delta}(x) = f_{\theta^0}(x) + \langle \phi_{\mathrm{RF}}(x), [\Delta a] \rangle + \sqrt{d} \cdot \langle \phi_{\mathrm{NT}}(x), [\Delta w] \rangle.$$

1.2 Interlude: Kernel Methods

Let us first understand the generalization properties of "non-deep learning" a.k.a. kernel ridge regression. Given feature map Φ and dataset $\{(x_i, y_i) : 1 \le i \le n\}$, our goal is to run regularized linear regression on the pairs $\{(\vec{\Phi}_i, y_i) : 1 \le i \le n\}$, where $\vec{\Phi}_i \overset{\text{def}}{=} \Phi(x_i)$:

$$\min_b \left\| y - \vec{\Phi} b \right\|^2 + \lambda \|b\|^2.$$

This admits an explicit solution:

$$b^*(\lambda) \overset{\text{def}}{=} \vec{\Phi}^\top \left(\lambda \mathrm{Id}_n + \vec{\Phi}\vec{\Phi}^\top\right)^{-1} y$$

Given an input x, the prediction for its label is given by $\langle b^*(\lambda), \Phi(x)\rangle$. So called kernel trick is the fact that prediction can be made even if feature map is only given implicitly. Define the kernel function

$$K(x_1, x_2) \overset{\text{def}}{=} \langle \Phi(x_1), \Phi(x_2)\rangle .$$

Then for

$$K(x, \cdot) \overset{\text{def}}{=} (K(x, x_1), \ldots, K(x, x_n))$$

and the empirical kernel matrix

$$K_n \overset{\text{def}}{=} \left(K(x_i, x_j)\right)_{i,j=1}^n ,$$

the prediction is represented by

$$\langle b^*(\lambda), \Phi(x)\rangle = K(x, \cdot)^\top \left(\lambda \mathrm{Id}_n + K_n\right)^{-1} y.$$

In conclusion, we only need to know $K(\cdot, \cdot)$, i.e. how to compute inner products in feature space! Going back to neural networks, for finite N, we have the following two kernel functions: random features

$$K_N^{\mathrm{RF}}(x_1, x_2) = \gamma^2 \sum_{i=1}^N \sigma(\langle w_i^0, x_1\rangle) \cdot \sigma(\langle w_i^0, x_2\rangle),$$

and neural tangent features:

$$K_N^{\mathrm{NT}}(x_1, x_2) = \frac{\gamma^2}{d} \sum_{i=1}^N \langle x_1, x_2\rangle \cdot \sigma'(\langle w_i^0, x_1\rangle) \cdot \sigma'(\langle w_i^0, x_2\rangle).$$

We want to take $N \longrightarrow \infty$, so take the scaling $\gamma = \frac{1}{\sqrt{N}}$. As $N \longrightarrow \infty$, these two kernels respectively have limit

$$\begin{aligned}
K_N^{\mathrm{RF}}(x_1, x_2) &\longrightarrow \mathbb{E}_w \left[\sigma(\langle w, x_1\rangle)\sigma(\langle w, x_2\rangle)\right] = K^{\mathrm{RF}}(x_1, x_2),\\
K_N^{\mathrm{NT}}(x_1, x_2) &\longrightarrow \frac{1}{d}\mathbb{E}_w \left[\langle x_1, x_2\rangle \cdot \sigma'(\langle w, x_1\rangle) \cdot \sigma'(\langle w, x_2\rangle)\right] = K^{\mathrm{NT}}(x_1, x_2).
\end{aligned}$$

If, e.g., $w \sim \mathcal{N}(0, \mathrm{Id}/d)$, these two kernels are both rotationally invariant: $K(x, x')$ only depends on $\angle(x, x')$.

Example: One example of which we can prove a generalization bound is: $x_i \sim \sqrt{d} \cdot \mathbb{S}^{d-1}$ and $y_i = f(x_i)$ for some arbitrary f such that $\|f\|_{L^2} = 1$. And we have the following theorems.

Theorem 1 (informal version of [GMMM21]). *For any rotationally invariant kernel with non-vanishing Hermite coefficients at all degrees, if $d^{l+\delta} \ll n \ll d^{l+1-\delta}$ for integer l and small constant δ, then for any sufficiently small λ>0, kernel ridge regression achieves test loss*

$$\|\mathrm{P}_{>l} f^*\|^2 + o_d(1),$$

where $\mathrm{P}_{>l}$ is the projection to orthogonal complement of subspace of degree-$\leq l$ polynomials.

See Figure 2 for an illustration of the polynomial projection. We can also use off-the-shelf matrix concentration inequalities to quantify convergence of empirical kernel to limiting kernel.

Theorem 2 (informal version of [MZ22]). *For the neural tangent kernel, with high probability over x_i's and w_i's, there holds*

$$\left\| K^{-1/2} K_N K^{-1/2} - \mathrm{Id}_n \right\|_{op} \leq \widetilde{O}\left(\sqrt{\frac{n+d}{Nd}}\right).$$

As long as the number Nd of parameters is much larger than number n of samples, the limiting object is accurate. Similar picture holds for random features. And this proof is based on applying matrix Bernstein inequality.

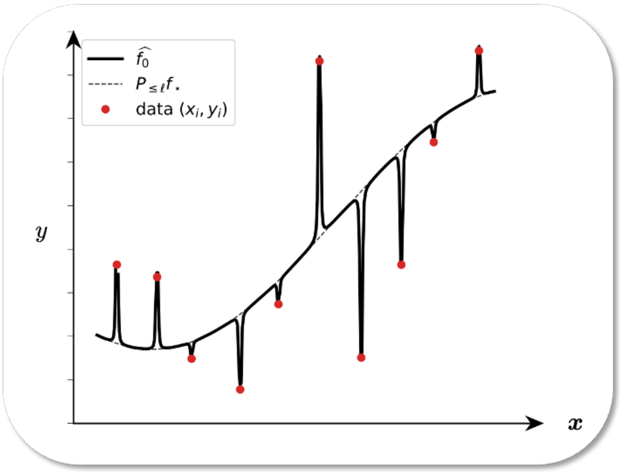


Figure 2: Polynomial Projection

1.3 Back to Gradient Descent

Previous section mainly discusses the performance of a linearized network

$$f_\theta^{\text{lin}}(x) = f_{\theta^0}(x) + \gamma \sum_{i=1}^{N} (a_i - a_i^0) \cdot \sigma(\langle w_i^0, x\rangle) + \gamma \sum_{i=1}^{N} a_i^0 \langle w_i - w_i^0, x\rangle \cdot \sigma'(\langle w_i^0, x\rangle).$$

We have to examine training dynamics of gradient methods to understand the difference

$$\underbrace{\frac{1}{n}\sum_{i=1}^{n}(y_i - f_\theta(x_i))^2}_{\text{deep learning}} \quad \text{v.s.} \quad \underbrace{\frac{1}{n}\sum_{i=1}^{n}(y_i - f_\theta^{\text{lin}}(x_i))^2}_{\text{linear regression}}.$$

Theorem 3 (informal version of [DZPS18, DLL⁺19, AZLS19, ZCZG20, COB19]). *When $\gamma \gg 1/N$, the trajectories of gradient flow for training f_θ versus f_θ^{lin} are vanishingly close as $N \to \infty$.*

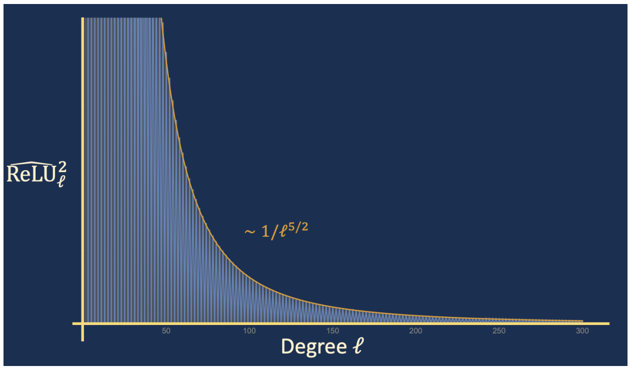


Figure 3: Approximation Error in Kernel Regime for ReLU Labels

We hereby also provide a simple negative example in which the kernel method does not suffice to conclude the empirical success of deep learning. Consider labels generated by $f(x) = \text{ReLU}(\langle w^*, x\rangle)$. It is shown that polynomial regression (and by extension any kernel method, as well as GD in NTK regime) requires $d^{\text{poly}(1/\epsilon)}$ samples to learn to error ϵ. See Figure 3 for an illustration of the implication of Theorem 1 onto this example of ReLU labels. However, as suggested by the following theorem, a single-neuron student network should already suffice to learn this example.

Theorem 4. *With $O(d)$ samples, gradient descent for a one-hidden-layer student network with a single neuron converges exponentially quickly for this problem (for Gaussian inputs).*

This phenomenon is possibly due to that we have to choose $\gamma \propto \frac{1}{N}$ to stay in the kernel regime. And this regime does not perfectly match the true dynamics which practical gradient method is really going through. We will talk about the scaling $\gamma \propto \frac{1}{N^2}$ (a.k.a. the mean field regime) in next lecture.

2 NTK Analysis

The rest of this lecture is devoted to provide details to the NTK analysis. In fact we will prove a generic result that does not even need the assumption that the student network is a one-hidden-layer MLP. This proof is mainly adapted from [COB19] and Telgarsky's Lecture Note. Consider a dataset $\{(x_i, y_i) : 1 \le i \le n\}$ and a student network $f_\theta : \mathbb{R}^d \to \mathbb{R}, \theta \in \mathbb{R}^p$ initialized to some θ_0. Use short hand $f_\theta(x) - y$ to denote

$$(f_\theta(x_1) - y_1, \dots, f_\theta(x_n) - y_n)\,.$$

Define a scaling parameter by $\alpha > 0$ and empirical loss

$$\hat{L}(g) = \frac{1}{2}\|g(x) - y\|_2^2, \quad \hat{L}_0 = \hat{L}(\gamma f_{\theta_0}).$$

Consider the gradient flow

$$\begin{aligned} \mathrm{d}\theta_t &= -\nabla_\theta \hat{L}(\gamma f_{\theta_t})\mathrm{d}t \\ &= -\gamma J_t^\top \nabla_\theta \hat{L}(\gamma f_{\theta_t})\mathrm{d}t, \end{aligned}$$

where the Jacobian J_t is taken as

$$J_t = J_{\theta_t} = \begin{pmatrix} \nabla_\theta f_{\theta_t}(x_1) \\ \nabla_\theta f_{\theta_t}(x_2) \\ \cdots \\ \nabla_\theta f_{\theta_t}(x_n) \end{pmatrix} \in \mathbb{R}^{n \times p}\,.$$

We will compare it to linearized network / dynamics:

$$\begin{aligned} f_\theta^{\mathrm{lin}}(x) &= f_{\theta_0}(x) + J_0 \cdot (\theta - \theta_0), \\ \mathrm{d}\tilde{\theta}_t &= -\gamma J_0^\top \nabla_\theta \hat{L}(\gamma f_{\tilde{\theta}_t}^{\mathrm{lin}})\mathrm{d}t, \end{aligned}$$

in which Jacobian J_0 does not change over training. We assume the following conditions:

1). J_θ is Lipschitz in θ, i.e.

$$\|J_\theta - J_{\theta'}\|_{op} \le \beta \|\theta - \theta'\|_2.$$

2). initialized Jacobian J_{θ_0} is of full rank (and our bounds will depend on $\sigma_{min}, \sigma_{max}$ of J_0).

As shown by the following lemma, linearized dynamics are very easy to analyze.

Lemma 1. *If $Q(t) \succeq \lambda \mathrm{Id}_n$ for all t, then for (g_t) given by*

$$\mathrm{d}g_t = -Q(t)\nabla \hat{L}(g_t)\mathrm{d}t,$$

we have

$$\hat{L}(g_t) \le \hat{L}(g_0) \cdot \exp(-2\lambda t).$$

Proof. It follows chain rule that

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\hat{L}(g_t) &= \langle -Q(t)\nabla - Q(t)(g_t(x) - y), g_t(x) - y \rangle \\ &\le -\lambda \|g_t(x) - y\|_2^2 \\ &= -2\lambda \cdot \hat{L}(g_t). \end{aligned}$$

Then integrating over it (i.e. using Gronwall's inequality) completes this proof. □

For the linearized dynamics

$$\mathrm{d}\tilde{\theta}_t = -\gamma J_0^\top \nabla_\theta \hat{L}(\gamma f^{\mathrm{lin}}_{\tilde{\theta}_t})\mathrm{d}t,$$

chain rule implies

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\gamma f^{\mathrm{lin}}_{\tilde{\theta}_t}\right) = \gamma \nabla_\theta f^{\mathrm{lin}}_{\tilde{\theta}_t}\Big|_{\theta=\tilde{\theta}_t}\frac{\mathrm{d}\tilde{\theta}_t}{\mathrm{d}t} = -\gamma^2 J_0 J_0^\top \nabla_\theta \hat{L}(\gamma f^{\mathrm{lin}}_{\tilde{\theta}_t}).$$

So we can apply $Q(t) = J_0 J_0^\top$ and $g_t = \gamma f^{\mathrm{lin}}_{\tilde{\theta}_t}$ to find

$$\hat{L}(\gamma f^{\mathrm{lin}}_{\tilde{\theta}_t}) \le \exp\left(-2\gamma^2 t \sigma_{min}(J_0)\right).$$

Unsurprisingly, training loss for linearized network drops exponentially fast. We will complete this NTK analysis in next lecture.

References

[AZL22] Zeyuan Allen-Zhu and Yuanzhi Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. In *The Eleventh International Conference on Learning Representations*, 2022.

[AZLS19] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International conference on machine learning*, pages 242–252. PMLR, 2019.

[COB19] Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in neural information processing systems*, 32, 2019.

[DLL$^+$19] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International conference on machine learning*, pages 1675–1685. PMLR, 2019.

[DZPS18] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2018.

[GMMM21] Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Linearized two-layers neural networks in high dimension. *THE ANNALS*, 49(2):1029–1054, 2021.

[JGH18] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[MMN18] Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018.

[MZ22] Andrea Montanari and Yiqiao Zhong. The interpolation phase transition in neural networks: Memorization and generalization under lazy training. *The Annals of Statistics*, 50(5):2816–2847, 2022.

[ZBH$^+$16] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2016.

[ZCZG20] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. Gradient descent optimizes over-parameterized deep relu networks. *Machine learning*, 109:467–492, 2020.