CS229M/STATS214: Machine Learning Theory

Lecturer: Tengyu Ma — Lecture # 12
Scribe: Rohan Taori and Jonathan Lee — Feb 24nd, 2021

12.1 Review and overview

In the last lecture, we finished our discussion of matrix completion and introduced the neural tangent kernel (NTK), which we defined via Taylor expansion as follows:

$$f_\theta(x) = f_{\theta^0}(x) + \langle \underbrace{\nabla_\theta f_{\theta^0}(x)}_{\text{features}}, \theta - \theta^0 \rangle + \text{higher order terms}, \tag{12.1}$$

where θ^0 is an appropriate random parameter initialization. Ignoring the higher order terms, we can think of f_θ as an affine transformation of θ, thus allowing us to treat this as a linear model. There are two main questions left unaddressed:

1. Does there exist a small neighborhood around θ^0, denoted $B(\theta^0)$, such that there is a global minimizer in $B(\theta^0)$?
2. Does gradient descent on the loss with respect to $f_\theta(x)$ stay in the neighborhood $B(\theta^0)$ and thus converge to the global minimizer?

The answer to both questions is "yes". In this lecture, we will explain the answer to the first question at an intuitive level, but skip discussion on the second since it requires more technical machinery (and is also easier to believe).

12.2 The two-layer network case

We demonstrate the NTK approach for the two-layer network setup.

12.2.1 Setup

For $i \in [m]$, let $a_i \in \mathbb{R}$ be scalars and let $w_i \in \mathbb{R}^d$ be vectors. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be the ReLU activation function defined as $\sigma(t) = \max\{t, 0\}$. Suppose we have the following two-layer network:

$$\hat{y} = f_\theta(x) = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} a_i \sigma(w_i^T x), \tag{12.2}$$

for some input x.

Our weight matrix is $W = \begin{bmatrix} w_1^T \\ \vdots \\ w_m^T \end{bmatrix} \in R^{m \times d}$. We initialize W randomly using $W_{ij} \overset{i.i.d.}{\sim} \mathcal{N}(0, 1)$ for all i and j. We initialize $a_i \in \{\pm 1\}$ and assume that the a_i's are fixed after initialization, i.e. not updated during training. (We fix a_i for simplicity: the results still hold when we are allowed

to optimize a_i in training.) We also assume that x has norm on the order of 1, and that the true label y is on the order of 1.

In our analysis, we will assume we have sufficiently large $m = \text{poly}(n, d)$. In other words, the width of the network m is sufficiently large such that $\text{poly}(n, d)$ factors are not important in the analysis. For simplicity, we write $O_{d,n}(1)$ to hide polynomial dependencies on d and n. Thus $O_{d,n}(m^c) = m^c \cdot \text{poly}(n, d)$.

Why do we need the scaling factor $1/\sqrt{m}$ in Equation 12.2? It is included to prevent the model outputs from blowing up when we increase the number of neurons m. Note that $\sigma(w_i^T x) \approx O_{d,n}(1)$ since $w_i^T \approx O_{d,n}(1)$ and $x \approx O_{d,n}(1)$. Since $a_i \in \{\pm 1\}$ for all i, this implies that $\sum_{i=1}^m a_i \sigma(w_i^T x) \approx O_{d,n}(\sqrt{m})$. Thus, the scaling factor is needed to obtain $\hat{y} = f_\theta(x) = O_{d,n}(1)$.

Next, we introduce some notation that will be helpful for our analysis. Let $\Delta\theta = \theta - \theta^0$. Suppose we have n examples $x^{(1)}, ..., x^{(n)}$ and labels $y^{(1)}, ..., y^{(n)}$. Let $\overrightarrow{y} = \begin{bmatrix} y^{(1)} & \dots & y^{(n)} \end{bmatrix}^T$. Let

$$\overrightarrow{y'} = \begin{bmatrix} y^{(1)} - f_{\theta^0}(x^{(1)}) \\ \vdots \\ y^{(n)} - f_{\theta^0}(x^{(n)}) \end{bmatrix} \tag{12.3}$$

be the transformed labels where we subsume the affine term in the label, allowing us to treat this as a purely linear model without loss of generality. Note that $\theta = \text{vec}(W) \in R^{dm}$ is the vectorized version of the weights. Let $\varphi^{(i)} = \nabla_\theta f_{\theta^0}(x^{(i)}) \in R^{dm}$ be the feature associated with the ith example, and let φ denote the collection of the features across the examples:

$$\varphi = \begin{bmatrix} \varphi^{(1)^T} \\ \vdots \\ \varphi^{(n)^T} \end{bmatrix} \in R^{n \times dm}. \tag{12.4}$$

12.2.2 Analysis of neighborhood size

Recall that in the previous lecture, we defined

$$g_\theta(x) = f_{\theta^0}(x) + \langle \nabla_\theta f_{\theta^0}, \theta - \theta^0 \rangle, \tag{12.5}$$

which is the linear approximation of $f_\theta(x)$ at θ^0. If we wish to fit $g_\theta(x)$ to y with the ℓ_2-loss, we may consider minimizing the following objective function over θ:

$$\sum_{i=1}^n \left(y^{(i)} - f_{\theta^0}(x^{(i)}) - \langle \nabla_\theta f_{\theta^0}(x^{(i)}), \theta - \theta^0 \rangle \right)^2 = \sum_{i=1}^n \left(y^{(i)} - f_{\theta^0}(x^{(i)}) - \Delta\theta^T \varphi^{(i)} \right)^2 \tag{12.6}$$

$$= \| \overrightarrow{y'} - \varphi \Delta\theta \|_2^2. \tag{12.7}$$

This is equivalent to the following optimization problem:

$$\min_{\Delta\theta} \quad \| \overrightarrow{y'} - \varphi \Delta\theta \|_2^2. \tag{12.8}$$

Since $n \ll dm$, so we have an undetermined linear system. Since our goal is to show that the relevant neighborhood around θ^0 is small, we should choose the minimum norm solution which can

be found directly by the pseudoinverse: $\hat{\theta} = \varphi^{\dagger}\overrightarrow{y}$, where $\varphi^{\dagger}$ is the pseudoinverse of φ given by $\varphi^{\dagger} = \varphi^T(\varphi\varphi^T)^{-1}$.

It remains to show that the norm of $\hat{\theta}$ is small. Before we can do that, we will prove some useful claims:

Lemma 12.1 (φ is a well-conditioned matrix)**.** *When θ^0 is random, φ is well-conditioned in the sense that*

$$\sigma_{\min}(\varphi) \approx \frac{1}{\sqrt{n}}\|\varphi\|_F \quad \textit{and} \quad \sigma_{\max}(\varphi) \approx \frac{1}{\sqrt{n}}\|\varphi\|_F. \tag{12.9}$$

($\sigma_{\min}(\varphi)$ and $\sigma_{\max}(\varphi)$ denote the smallest and largest singular values of φ respectively.) Specifically, $\sigma_{min}(\varphi) \gtrsim \Omega\left(\frac{1}{\sqrt{n}}\|\varphi\|_F\right)$, and vice-versa for $\sigma_{\max}(\varphi)$.

We omit the proof as it uses tools from random matrix theory that are not required for this course. (The high-level idea is to use the fact that $\varphi\varphi^T \approx c \cdot I$ for some constant scalar c.)

Remark 12.2. Since $\varphi \in R^{n\times dm}$, φ has at most n singular values $\sigma_1 \geq \ldots \geq \sigma_n \geq 0$. Since $\|\varphi\|_F = \sqrt{\sigma_1^2 + ... + \sigma_n^2}$, the fact that $\sigma_n \approx \frac{1}{\sqrt{n}}\|\varphi\|_F$ means that all the singular values are not very different from each other.

Lemma 12.3 (Frobenius norm of φ is order 1)**.** $\|\varphi\|_F \asymp \Theta_{d,n}(1)$*, which implies that*

$$\|\varphi\|_{op},\ \|\varphi^{\dagger}\|_{op} \asymp \Theta_{d,n}(1). \tag{12.10}$$

Proof. Applying the definition of $\varphi^{(i)}$, we have

$$\varphi^{(i)} = \text{vec}\left(\frac{\partial f_\theta(x^{(i)})}{\partial W}\right) = \frac{1}{\sqrt{m}}(a \odot \sigma'(wx^{(i)})) \cdot \left(x^{(i)}\right)^T. \tag{12.11}$$

Thus, its norm can be written as

$$\|\varphi^{(i)}\|_2 = \frac{1}{\sqrt{m}}\|a \odot \sigma'(wx^{(i)})\|_2 \cdot \|x^{(i)}\|_2 \tag{12.12}$$

$$\approx \Theta_{d,n}\left(\frac{1}{\sqrt{m}} \cdot \sqrt{m} \cdot 1\right) \tag{12.13}$$

$$\approx \Theta_{d,n}(1). \tag{12.14}$$

The first equality is because $\|ab^T\|_2 = \|a\|_2 \cdot \|b\|_2$ for any vectors a and b, and (12.13) is because $\|x^{(i)}\|_2$ is on order of 1, $a \odot \sigma'(wx^{(i)})$ is a vector of length m with each entry being on the order of 1 (each a_i is either 1 or -1, and σ' is either 0 or 1). Summing up over the $\varphi^{(i)}$'s,

$$\|\varphi\|_F = \sqrt{\sum_{i=1}^{n}\|\varphi^{(i)}\|_2^2} \approx \Theta_{d,n}(1). \tag{12.15}$$

Putting this together with Lemma 12.1, all the singular values of φ are $\Theta_{d,n}(1)$. By extension, $\|\varphi^{\dagger}\|_{\text{op}} \asymp \Theta_{d,n}(1)$ as well, since if a matrix A has singular values $\sigma_1, \ldots, \sigma_n$, $A^{\dagger}$ has singular values $1/\sigma_1, \ldots, 1/\sigma_n$. □

Now we can leverage the previous two lemmas to produce a bound on the ℓ_2-norm of the solution of the optimization problem (12.8), $\hat{\theta}$. Recall that $\hat{\theta} = \varphi^\dagger \overrightarrow{y'}$. Upper bounding the norm yields

$$\begin{aligned}
\|\hat{\theta}\|_2 &\leq \|\varphi^\dagger\|_{\text{op}} \cdot \|\overrightarrow{y'}\|_2 && (12.16)\\
&\leq O_{d,n}(1) \cdot \|\overrightarrow{y'}\|_2 \qquad \text{(by Lemma 12.3)} && (12.17)\\
&\leq O_{d,n}(1), && (12.18)
\end{aligned}$$

where the last inequality is because $\overrightarrow{y'}$ is of dimension n and each entry is on the order of 1 (the original labels are on the order of 1 and the shifts $f_{\theta^0}(x^{(i)})$ are also on the order of 1). Although $\|\hat{\theta}\|_2$ may not appear to be small, since it may still be poly(d, n), we can view it as comparatively small relative to the size of θ^0 since

$$\begin{aligned}
\|\theta^0\|_2 &= \|W^0\|_F^2 && (12.19)\\
&\asymp \sqrt{dm} \qquad (\because W^0 \text{ has } dm \text{ entries of order 1}) && (12.20)\\
&= \Theta_{d,n}(\sqrt{m}). && (12.21)
\end{aligned}$$

Thus, the neighborhood size is much smaller than the norm of the initialization in terms of m. Further justification of the ℓ_2-norm as a reasonable metric for defining neighborhood size may require deeper inspection of the higher order terms and their behavior within the neighborhood. The intuition is that one only needs to move a little to reach the solution. Relative to the norm of the initialization, the neighborhood size is shrinking.

Remark 12.4. While we do not cover this in detail, the main takeaway on the optimization front is that the problem of fitting $g_\theta(x)$ is a standard strongly convex optimization problem, which enjoys the geometric rate of convergence.

12.3 Limitations of NTK

The NTK approach has its limitations.

- Empirically, optimizing $g_\theta(x)$ as described in the theory does not work as well as state-of-the-art (or even standard) deep learning methods. For example, using the NTK approach (i.e., taking the Taylor expansion and optimizing $g_\theta(x)$) with a ResNet generally does not perform as well as ResNet with best-tuned hyperparameters.

- The NTK approach requires a specific initialization scheme and learning rate which may not coincide with what is commonly used in practice.

- The analysis above was for gradient descent, while stochastic gradient descent is used in practice, introducing noise in the procedure. This means that NTK with stochastic gradient descent requires a small learning rate to stay in the initialization neighborhood. Deviating from the requirements can lead to leaving the initialization neighborhood.

One possible explanation for the gap between theory and practice is because NTK effectively requires a fixed kernel, so there is no incentive to select the right features. Furthermore, the minimum ℓ_2-norm solution is typically dense. This is similar to the difference between sparse

and dense combinations of features observed in the ℓ_1-SVM/two-layer network versus the standard kernel method SVM (or ℓ_2-SVM) analyzed previously.

To make these ideas more concrete, consider the following example [WLLM20].

Example 12.5. Let $x \in \mathbb{R}^d$ and $y \in \{-1, 1\}$. Assume that each component of x satisfies $x_i \in \{-1, 1\}$. Define the output $y = x_1 x_2$, that is, y is only a function of the first two components of x.

This output function can be described exactly by a neural network consisting of a sparse combination of the features (4 neurons to be exact):

$$\hat{y} = \frac{1}{2}\left[\phi_{\text{relu}}(x_1 + x_2) + \phi_{\text{relu}}(-x_1 - x_2) - \phi_{\text{relu}}(x_1 - x_2) - \phi_{\text{relu}}(x_2 - x_1)\right] \tag{12.22}$$

$$= \frac{1}{2}\left(|x_1 + x_2| - |x_1 - x_2|\right) \tag{12.23}$$

$$= x_1 x_2. \tag{12.24}$$

(12.23) follows from the fact that $\phi_{\text{relu}}(t) + \phi_{\text{relu}}(-t) = |t|$ for all t, while (12.24) follows from evaluating the 4 possible values of (x_1, x_2). Thus, we can solve this problem exactly with a very sparse combination of features.

However, if we were to use the NTK approach (kernel method), the network's output will always involve $\sigma(w^\top x)$ where w is random so it includes all components of x (i.e. a dense combination of features), and cannot isolate just the relevant features x_1 and x_2. This is illustrated in the following informal theorem (which will not be proved in this lecture).

Theorem 12.6. *The kernel method with NTK requires $n = \Omega(d^2)$ samples to learn Example 12.5 well. In contrast, the neural network regularized by $\sum_{j=1}^m |u_j| \|w_j\|_2$ only requires $n = O(d)$ samples.*

Bibliography

[WLLM20] Colin Wei, Jason D. Lee, Qiang Liu, and Tengyu Ma, *Regularization matters: Generalization and optimization of neural nets v.s. their induced kernel*, 2020.