CSE 543 / 599I - Deep Learning

Lecture 8/9: Global convergence of Gradient Descent

April 26, 2022

Lecturer: Simon Du *Scribe: Prashant Rangarajan*

One would expect that locating the global minimum when training a neural network via gradient descent would be a very difficult task. However, surprisingly, in practice we find that the optimization error does go to zero while training, indicating that these models do generalize quite well and reach the global minima.[1] Approaches such as landscape analysis that analyze the geometry of the objective function fail to sufficiently demonstrate why gradient descent attains the global minimum.

We now try to prove this phenomenon using an approach that directly analyzes the trajectory of gradient descent.

1 Main Result

Theorem 1.1. *(Du et al. '18 [2], Allen-Zhu et al. '18 [3], Zou et al '19 [4]) If the width of each layer is $poly(n)$ where n is the number of data. Using random initialization with a particular scaling, gradient descent finds an approximate global minimum in polynomial time.*

In the over parameterized regime, we can show that randomly initialized gradient descent provably optimizes deep neural networks.
If gradient descent approximates the global minimum w^* as follows: $|f(w) - f(w^*)| \leq \epsilon$, then the convergence time would of the form $poly(\log(\frac{1}{\epsilon}), n, L)$.

2 Gradient Flow

To analyze the trajectory of gradient descent, we use the previously studied concept of gradient flow. This can be generalized to the case of gradient step with finite step size by introducing an additional discretization step to the process.
Let $u_i(t) = f(\theta, x_i)$ denote the prediction function for input x_i at time t, and let the loss be of the form

$$\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(\theta, x_i), y_i) = \frac{1}{n}\sum_{i=1}^{n} \ell(u_i(t), y_i)$$

Then,

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta} = \frac{1}{n}\sum_{i=1}^{n} \ell'(u_i(t), y_i) \cdot \frac{\partial u_i(t)}{\partial \theta}$$

Also, by the definition of gradient flow, we know that:

$$\frac{d\theta(t)}{dt} = -\frac{\partial \mathcal{L}(\theta)}{\partial \theta}$$

Now, in the typical optimization problem where the loss function is strongly convex, by classical theory we know that there exists a unique global minimum θ^* s.t. $\theta(t) \to \theta^*$. However, in the case of over-parameterized neural networks, where we have $m = poly(n)$, many combinations of parameters can give a training loss of zero i.e. there is no unique global minimizer.
Instead, if we look at the prediction function $u_i(t)$, we do have a unique global minimum where $u_i(t) \to y_i$. So in this approach we prefer to study the trajectory of $u_i(t)$.

3 Dynamics of the prediction function

As we know, the predictor is given by $u_i(t) = f(\theta, x_i)$. In vectorized notation let $\boldsymbol{u}(t) = (u_1(t), \dots u_n(t))$ and $\boldsymbol{y} = (y_1, \dots y_n)$. Then, by Chain Rule we can say:

$$\begin{aligned}
\frac{du_i(t)}{dt} &= \left\langle \frac{\partial u_i(t)}{\partial \theta}, \frac{\theta(t)}{dt} \right\rangle \\
&= \left\langle \frac{\partial u_i(t)}{\partial \theta}, -\frac{1}{n} \sum_{j=1}^{n} \ell'(u_j(t), y_j) \cdot \frac{\partial u_j(t)}{\partial \theta(t)} \right\rangle \\
&= [\ell'(u_1(t), y_1), \dots, \ell'(u_n(t), y_n)] \cdot \left[\left\langle \frac{\partial u_i(t)}{\partial \theta(t)}, \frac{\partial u_1(t)}{\partial \theta(t)} \right\rangle, \dots, \left\langle \frac{\partial u_i(t)}{\partial \theta(t)}, \frac{\partial u_n(t)}{\partial \theta(t)} \right\rangle \right]
\end{aligned}$$

We have rewritten $\frac{du_i(t)}{dt}$ as the inner product of two vectors. The second vector itself consists of inner products as elements and could be interpreted as a type of kernel.
Introducing some notation, let $\boldsymbol{H}(t) \in \mathbb{R}^{n \times n}$ be a Gram matrix such that

$$(\boldsymbol{H}(t))_{ij} = \left\langle \frac{\partial u_i(t)}{\partial \theta(t)}, \frac{\partial u_j(t)}{\partial \theta(t)} \right\rangle$$

Also, we define $\boldsymbol{\ell}'(\boldsymbol{u}(t), \boldsymbol{y}) \in \mathbb{R}^n$ s.t. $[\boldsymbol{\ell}'(\boldsymbol{u}(t), \boldsymbol{y})]_i = \ell'(u_i(t), y_i)$.
Then, the dynamics of the overall prediction is:

$$\frac{d\boldsymbol{u}(t)}{dt} = -\frac{1}{n} \boldsymbol{H}(t) \boldsymbol{\ell}'(\boldsymbol{u}(t), \boldsymbol{y})$$

For example, if we consider a simple quadratic loss $\boldsymbol{\ell}(\boldsymbol{u}(t), \boldsymbol{y}) = \frac{1}{2} \|\boldsymbol{u}(t) - \boldsymbol{y}\|^2$, then we get:

$$\frac{d\boldsymbol{u}(t)}{dt} = -\frac{1}{n} \boldsymbol{H}(t)(\boldsymbol{u}(t) - \boldsymbol{y})$$

Note that $\boldsymbol{H}(t)$ does not depend on the loss function, but only on the predictor function.

Theorem 3.1. *If $\exists \lambda_0 > 0 \; \forall t$ s.t. $\lambda_{min}(\boldsymbol{H}(t)) \geq \lambda_0$, then we get global convergence for gradient flow.*

$$\begin{aligned}\frac{d\left(\frac{1}{2}\|\boldsymbol{u}(t)-\boldsymbol{y}\|^2\right)}{dt} &= -\frac{1}{n}(\boldsymbol{u}(t)-\boldsymbol{y})^\top \boldsymbol{H}(t)(\boldsymbol{u}(t)-\boldsymbol{y}) \\ &\leq -\frac{\lambda_0}{n}\|\boldsymbol{u}(t)-\boldsymbol{y}\|^2\end{aligned}$$

The derivative of the loss being bounded by a negative factor of the loss clearly indicates that there the dynamics will converge. We can more formally show this as follows:

Consider the term $\frac{d}{dt}\left(\exp\left(\frac{\lambda_0 t}{n}\right)\frac{1}{2}\|\boldsymbol{u}(t)-\boldsymbol{y}\|^2\right)$. Expanding and using the above inequality, we get:

$$\begin{aligned}\frac{d}{dt}\left(\exp\left(\frac{\lambda_0 t}{n}\right)\frac{1}{2}\|\boldsymbol{u}(t)-\boldsymbol{y}\|^2\right) &= \frac{\lambda_0}{2n}\exp\left(\frac{\lambda_0 t}{n}\right)\|\boldsymbol{u}(t)-\boldsymbol{y}\|^2 \\ &\quad + \frac{d\left(\frac{1}{2}\|\boldsymbol{u}(t)-\boldsymbol{y}\|^2\right)}{dt}\exp\left(\frac{\lambda_0 t}{n}\right) \\ &\leq \exp\left(\frac{\lambda_0 t}{n}\right)\|\boldsymbol{u}(t)-\boldsymbol{y}\|^2\left(\frac{\lambda_0}{2n}-\frac{\lambda_0}{n}\right) < 0\end{aligned}$$

Clearly $\left(\exp(\frac{\lambda_0 t}{n})\frac{1}{2}\|\boldsymbol{u}(t)-\boldsymbol{y}\|^2\right)$ is decreasing with t. At $t=0$, let $\frac{1}{2}\|\boldsymbol{u}(0)-\boldsymbol{y}\|^2 = C$. Then $\forall t$,

$$\left(\exp\left(\frac{\lambda_0 t}{n}\right)\frac{1}{2}\|\boldsymbol{u}(t)-\boldsymbol{y}\|^2\right) \leq C$$

In other words

$$\left(\frac{1}{2}\|\boldsymbol{u}(t)-\boldsymbol{y}\|^2\right) \leq C\exp\left(\frac{-\lambda_0 t}{n}\right)$$

As $t \to \infty$, the loss converges linearly to 0 with $\boldsymbol{u}(t) \to \boldsymbol{y}$.

4 Gradient Flow for Kernel Regression

In order to prove Theorem 3.1, we need to apply different analysis techniques for different functions to get the corresponding λ_0. Let us consider the example of a Kernel Function. A kernel can be thought of as an infinite dimensional linear regression. The predictor function in this case would be $u_i(t) = \phi(x_i)^\top \theta(t)$ where $\phi(x_i)$ is the corresponding feature map of the kernel function. Then, the Gram Matrix $\boldsymbol{H}(t)$ is actually equivalent to the Kernel matrix:

$$H_{i,j}(t) = \left\langle \frac{\partial u_i(t)}{\partial \theta(t)}, \frac{\partial u_j(t)}{\partial \theta(t)} \right\rangle = \langle \phi(x_i), \phi(x_j) \rangle = K(x_i, x_j)$$

In this case, since the kernel function doesn't depend on t, neither does $\boldsymbol{H}$.
If the kernel is full rank, then we get $\lambda_{\min}(\boldsymbol{H}(t)) \geq \lambda_0$. Also, in general, if the kernel is universal, the result holds and hence we get global convergence.

5 Gradient Flow for Neural Networks

While in the case of kernels we have a gram matrix that is independent of t. But the analysis won't be as straightforward in the case of a neural network.

We use the example of a simple 2-layer neural network. Let the predictor be

$$f(\theta, x) = \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r \sigma(w_r^\top x)$$

where $x \in \mathbb{R}^d, w_r \in \mathbb{R}^d, a_r \in \mathbb{R}, \sigma(\cdot) : ReLU$. Note that we need to scale by a factor of $1/\sqrt{m}$ in order to prove the required bounds.
For simplicity, let us assume the following:
Initialization: $a_r \sim Unif\{1, -1\}, w_r \sim \mathcal{N}(0, I)$.
Training: Only the first layer of weights w_i are trained. The problem still continues to be non-convex despite this assumption. So, the optimization problem is

$$\min_{w_1,\ldots,w_m} \frac{1}{n} \sum_{i=1}^{n} f(x_i, \boldsymbol{a}, \boldsymbol{w}) - y_i)^2$$

Now, as we know,

$$\frac{d\boldsymbol{u}(t)}{dt} = -\frac{1}{n}\boldsymbol{H}(t)(\boldsymbol{u}(t) - \boldsymbol{y})$$

Key Idea: We show that $\forall t$, $\boldsymbol{H}(t) \approx \boldsymbol{H}^*$, where $\boldsymbol{H}^*$ is a constant matrix (called the *neural tangent kernel*) of the form

$$H^*_{i,j} = \lim_{m\to\infty} \mathbb{E}_{w\sim \boldsymbol{D}_{init}} \left\langle \frac{\partial f(x_i, a, w)}{\partial w(t)}, \frac{\partial f(x_j, a, w)}{\partial w(t)} \right\rangle$$

This implies that there isn't much change in the Gram matrix with t. This would mean that

$$\frac{d\boldsymbol{u}(t)}{dt} \approx -\frac{1}{n}\boldsymbol{H}^*(\boldsymbol{u}(t) - \boldsymbol{y})$$

If $\boldsymbol{H}^*$ is a full rank matrix, then it follows from the prior discussion that gradient descent will globally converge.

Let $\boldsymbol{u}(t) = f(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{w})$. Then, $\boldsymbol{H}(t)$ can be simplified as follows:

$$\begin{aligned}
H_{i,j}(t) &= \left\langle \frac{\partial u_i(t)}{\partial w(t)}, \frac{\partial u_j(t)}{\partial w(t)} \right\rangle \\
&= \sum_{r=1}^{m} \left\langle \frac{\partial u_i(t)}{\partial w_r(t)}, \frac{\partial u_j(t)}{\partial w_r(t)} \right\rangle \\
\frac{\partial u_i(t)}{\partial w_r(t)} &= \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r x_i \mathbb{1}\{w_r(t)^\top x_i \geq 0\} \\
\therefore\ H_{i,j}(t) &= \sum_{r=1}^{m} \frac{1}{m} \left\langle a_r x_i \mathbb{1}\{w_r(t)^\top x_i \geq 0\}, a_r x_j \mathbb{1}\{w_r(t)^\top x_j \geq 0\} \right\rangle \\
&= \frac{1}{m} x_i^\top x_j \sum_{r=1}^{m} \mathbb{1}\{w_r(t)^\top x_i \geq 0, w_r(t)^\top x_j \geq 0\}
\end{aligned}$$

Proposition 5.1. *$\boldsymbol{H}(t) \approx \boldsymbol{H}^*$ in the case of the 2 layer neural network.*

Proof. In order to prove this, we claim the following:

1. $\boldsymbol{H}(0) \approx \boldsymbol{H}^*$ (Initialization)
2. $\boldsymbol{H}(t) \approx \boldsymbol{H}(0)\ \forall t \geq 0$ (Training)

If both these claims hold, then the result follows (and we can also say that gradient descent converges globally). We validate these claims individually in the following sections. □

5.1 Claim 1: Initialization

We want to show that $\boldsymbol{H}(0) \approx \boldsymbol{H}^*$. To do this, we need the following result:

Theorem 5.2. ***Hoeffding's Inequality:*** *Let $Z_1, \ldots, Z_n$ be i.i.d bounded random variables such that $|Z_i| \leq 1$. If $n = \Omega\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$ where $0 < \delta, \epsilon < 1$, then with probability $(1 - \delta)$, we have*

$$\left| \frac{1}{n} \sum_{i=1}^{n} Z_i - \mathbb{E}[Z_i] \right| \leq \epsilon$$

Now, we know that

$$H_{i,j}(0) = x_i^\top x_j \frac{1}{m} \sum_{r=1}^{m} \mathbb{1}\{w_r(0)^\top x_i \geq 0, w_r(0)^\top x_j \geq 0\}$$

This is an average of m bounded random variables (the indicator variables), so we can directly apply Hoeffding's inequality to say that it converges to $H^*_{i,j}$, where

$$H^*_{i,j} = \mathbb{E}_{w \sim \mathbb{N}(0,I)} x_i^\top x_j \mathbb{1}\{w^\top x_i \geq 0, w^\top x_j \geq 0\}$$

So, when m is large enough, we get $|H_{i,j}(0) - H^*_{i,j}| \leq \epsilon$.
$\therefore \ \|\boldsymbol{H}^* - \boldsymbol{H}(0)\|_F \leq \sum_{i,j} |H_{i,j}(0) - H^*_{i,j}| \leq n^2\epsilon$. We can always pick a small enough ϵ to have an arbitrary approximation. This proves our claim.
Note: This result also highlights the power of over-parameterization of neural networks, as we need a large m (i.e. width of the layer) in order to prove the required concentration bound for result.

5.2 Claim 2: Training

We want to show that $\boldsymbol{H}(t) \approx \boldsymbol{H}(0) \ \forall t$.

For simplicity we assume the following:

1. We train till time t (can extend to $t \to \infty$ case.)
2. $y_i = \mathcal{O}(1)$ i.e y_i is a constant
3. $\|x_i\| = 1$

Note: With these assumptions, the matrix $\boldsymbol{H}^*$ obtained in claim 1 would reduce to: $H^*_{i,j} = x_i^\top x_j \left(\frac{\pi - \arccos x_i^\top x_j}{2\pi}\right)$
Key Idea - Lazy Training: Every weight vector only moves a little bit, and this change scales with $\frac{1}{\sqrt{m}}$. In other words, as the width increases, the movement decreases. This regime of training is called *lazy training*.

Now, we can say:

$$\begin{aligned}\|w_r(t) - w_r(0)\|_2 &= \left\| \int_0^t \frac{dw_r(\tau)}{d\tau} d\tau \right\| \leq \int_0^t \left\| \frac{dw_r(\tau)}{d\tau} \right\| d\tau \\ &= \int_0^t \left\| \frac{1}{\sqrt{m}} \frac{1}{n} \sum_{i=1}^n (u_i(\tau) - y_i) a_r x_i \mathbb{1}\{w_r^\top x_i \geq 0\} \right\| d\tau\end{aligned}$$

Now, if we assume $u_i(\tau)$ to be constant in $[0, t]$, then the integrand would entirely be a constant (say C), and we would get:

$$\|w_r(t) - w_r(0)\|_2 \leq C \int_0^t \frac{1}{\sqrt{m}} d\tau \leq \frac{Ct}{\sqrt{m}}$$

With an increase in m, clearly the weights move less.

ReLU Smoothness: Now, to complete the proof, we need to bound the change in $\boldsymbol{H}(t)$. We show this in the case of the ReLU activation function.
Smoothness is a property whereby a small deviation in the parameters will result in a small deviation in the quantity of interest. While ReLU does not have a smooth derivative, when we have

a Gaussian initialization of weights, we can derive a version of smoothness for the kernel matrix. We know that:

$$H_{i,j}(t) = \frac{1}{m} x_i^\top x_j \sum_{r=1}^{m} \mathbb{1}\{w_r(t)^\top x_i \geq 0, w_r(t)^\top x_j \geq 0\}$$

Hence, we can say:

$$\begin{aligned} H_{i,j}(t) - H_{i,j}(0) &= \frac{1}{m} x_i^\top x_j \sum_{r=1}^{m} \mathbb{1}\{w_r(t)^\top x_i \geq 0, w_r(t)^\top x_j \geq 0\} - \mathbb{1}\{w_r(0)^\top x_i \geq 0, w_r(0)^\top x_j \geq 0\} \\ &\leq \frac{1}{m} x_i^\top x_j \sum_{r=1}^{m} \mathbb{1}\{sgn(w_r(t)^\top x_i) \neq sgn(w_r(0)^\top x_i)\} \\ &+ \frac{1}{m} x_i^\top x_j \sum_{r=1}^{m} \mathbb{1}\{sgn(w_r(t)^\top x_j) \neq sgn(w_r(0)^\top x_j)\} \end{aligned}$$

The term $\frac{1}{m}\sum_{r=1}^{m} \mathbb{1}\{sgn(w_r(t)^\top x_i) \neq sgn(w_r(0)^\top x_i)\}$ represents the number of pattern changes in the time interval $[0, t]$. We would like to bound this quantity.

Anti-Concentration Bound: While concentration inequalities bound how much a random variable deviates from a quantity (usually the mean), anti-concentration inequalities give an upper bound on how much a random variable can concentrate around a quantity.
There is some non-trivial mass of the Gaussian distribution away from the mean that we would like to bound. The Gaussian Anti-Concentration bound is as follows:

$$\mathcal{P}_{Z\sim\mathcal{N}(0,I)}(|Z| < R) \leq \frac{2R}{\sqrt{2\pi}}$$

If $w_r(0) \sim \mathcal{N}(0, I)$, then by rotational symmetry, we have $w_r(0)^\top x_i \sim \mathcal{N}(0, 1)$ where $\|x_i\|_2 = 1$.

If $\forall r, \|w_r(t) - w_r(0)\|_2 \leq \Delta w$, then we know that $\Delta w \to 0$ as $m \to \infty$ because $\Delta w = \mathcal{O}(\frac{1}{\sqrt{m}})$ (from the lazy training regime).
Suppose $|w_r(0)^\top x_i| \geq \Delta w$, then we can say $sgn(w_r(t)^\top x_i) = sgn(w_r(0)^\top x_i)$ as $|w_r(t)^\top x_i) - w_r(0)^\top x_i| \leq \|x_i\|_2 \|w_r(t) - w_r(0)\|_2 \leq \Delta w \leq |w_r(0)^\top x_i|$. Thus,

$$\mathcal{P}(|w_r(0)^\top x_i| < \Delta w) \leq \frac{2\Delta w}{\sqrt{2\pi}}$$

represents the bound on the probability that each pattern changes. But we know that $\Delta w \to 0$ as $m \to \infty$, hence $\frac{1}{m}\sum_{r=1}^{m} \mathbb{1}\{sgn(w_r(t)^\top x_i) \neq sgn(w_r(0)^\top x_i)\} \to 0$ as $m \to \infty$.

This means that $H_{i,j}(t) \to H_{i,j}(0)$ as $m \to \infty$. More formally, if $m = \mathcal{O}(\frac{n^6}{\epsilon^2})$, then we get $\|\boldsymbol{H}(t) - \boldsymbol{H}(0)\|_F \leq \epsilon$. Thus means that $\boldsymbol{H}(t) \approx \boldsymbol{H}(0)$ and hence we have proved claim 2.

5.3 Neural Nets as a Kernel Predictor

We showed that the dynamics of gradient descent on a neural network is equivalent to a kernel. We can also show that prediction for a neural network can be equivalent to a kernel predictor.

If $X_{te} \in \mathbb{R}^n$ is a test point, then when $m \to \infty$, even though we train on a neural network, the predictor $f(\theta, X_{te})$ is equivalent to a kernel predictor $K_{te}(\boldsymbol{H}^*)^{-1}\boldsymbol{y}$, where $\boldsymbol{H}^*$ is the NTK, $y \in \mathbb{R}^n$, and $K_{te} \in \mathbb{R}^n$ with $[K_{te}]_i = K(x_i, X_{te})$.

References

[1] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization, 2016.

[2] Simon S. Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks, 2018.

[3] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 242–252. PMLR, 09–15 Jun 2019.

[4] Difan Zou and Quanquan Gu. An improved analysis of training over-parameterized deep neural networks, 2019.