18.408 Lecture 3 — Ambar Mehta

Optimization

① Recap

1) $R(f(x;w)) = \sum_{i=1}^{N} \ell(f(x_i;w), y_i)$

N ——————→ one term for each example you want to label

2) in practice stochastic gradient descent / mini-batch gradient descent, K random examples
(K should be not too large b/c randomness helps escape spurious local minima)

↓

reduce variance of unbiased estimator

3) Nesterov acc $y_{t+1} = x_t - \eta \nabla f(x_t)$

$x_{t+1} = y_{t+1} + \mu(\underbrace{y_{t+1} - y_t}_{\text{momentum}})$

iterations to reduce distance to opt. by const. factor

$GD \sim \frac{\beta}{\alpha}$

$AGD \sim \sqrt{\beta/\alpha}$

β = smoothness, α = strong convexity

4) gradient flow: continuous time limit of GD

$\dot{x}(t) = -\nabla f(x(t))$

5) Moving away from convex optimization

Issue #1: can't find global minimum (NP-hard)

Issue #2: minimum we find depends on opt. algo and init. — Langevin Dynamics

Issue #3: generalizability

Neural Tangent Kernel

① Overview

fundamental result about fitting overparameterized deep nets

idea: for a wide enough deep net GD finds a 0-error solution

caveats:

(1) depth ≥ 2, right nonlinearities (~ universal approx)

(2) how wide is wide enough

(3) right scaling parameters for random init

Main Ideas:

(1) if we take linear approx to a deep network, if width > # examples, we can fit perfectly

(2) as we increase width, we need to move less in parameter space

⇒ region doesn't change too much

① Setup

Setup

$$f(w) \triangleq \begin{bmatrix} f(x_1; w) \\ \vdots \\ f(x_N; w) \end{bmatrix} \qquad y \triangleq \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

$$R(2f(w)) \triangleq \frac{1}{2}\left\| 2f(w) - y \right\|^2$$

2N width

Gradient Flows

$$\dot{w}(t) = -\nabla_w R(2f(w(t))) = -2 J_t^T \nabla R(2f(w(t))) = -2J_t^T (2f(w(t)) - y)$$ squared loss

$$J_t \triangleq \begin{bmatrix} \nabla f(x_1; w(t))^T \\ \vdots \\ \nabla f(x_N; w(t))^T \end{bmatrix}$$

Goal

Main Theorem (Informal)

(1) if λ is big enough $R(2f(w(t))) \le R(2f(w(0)))\, e^{-c t \lambda^2}$

(2) $\|w(t) - w(0)\| \le \dfrac{\sqrt{R(2f(w(0)))}}{\lambda}$

proof strategy: introduce auxiliary flow $u(t)$ using linear approximation to f and show that $u(t)$ and $w(t)$ remain close

② Look at linear in gradient flow

$$f_0(u) \triangleq f(w(0)) + J_0 (u - w(0))$$ linear approx (Taylor series) around initialization

gradient flow $\dot{u}(t) = -\nabla_u R(2 f_0(u(t))) = -2 J_0^T \nabla R(2 f_0(u(t))) = -2 J_0^T (2 f_0(u(t)) - y)$ (1)

jacobian doesn't depend on time now!

Lemma 1: if $J_0 J_0^T$ is full rank, then $R(2 f_0(u(t)))$ goes to zero

Proof: look at change in predictions

$$\frac{d}{dt} 2 f_0(u(t)) = \frac{d}{dt}\left[2\left(f(w(0)) + J_0 (u(t) - w(0)) \right] = 2 J_0 \dot{u}(t) \right. \quad (2)$$

(1), (2) $\Rightarrow$ $\frac{d}{dt} 2 f_0(u(t)) = -\lambda^2 J_0 J_0^T (2 f_0(u(t)) - y)$

$$\frac{d}{dt}\underbrace{\left(2 f_0(u(t)) - y \right)}_{\triangleq r(t)} = -\lambda^2 J_0 J_0^T (2 f_0(u(t)) - y)$$

$\dot{r}(t) = -\lambda^2 J_0 J_0^T r(t)$ simple ODE; solution is matrix exponential

$r(t) = e^{-\lambda^2 J_0 J_0^T t} r(0)$ converges by full rank assumption to zero

Key definition: matrix $K^\infty = J_0 J_0^T$ is called **neural tangent kernel**

$K(x_i, x_j) = \nabla f(x_i; w(0))^T \cdot \nabla f(x_j; w(0))$ kernel function because can be used on new data

Note It is known that under right scaling $f(\cdot;w(\alpha))$ converges to a Gaussian Process property of architecture and random initialization; given by gradients

Note gradient flow for w is something we have seen, in disguise, for least squares

ie: $x_{t+1} = x_t - \eta A^T r_t$, $r_t \triangleq A x_t - b$

rewrite as recurrence for residual r_t $\quad r_{t+1} = r_t - \eta A A^T r_t$

looks like what we had with $A \leftrightarrow J_0$

w is solving a least squares problem using the linear approx to the deep net at initialization

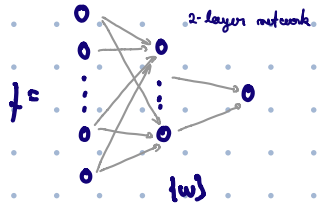
(IV) Interlude: why λ?

λ = abstraction for why increasing λ (or increasing width)

as $\lambda \to \infty$ we are zooming in at $w(0)$ so the linear approx is more accurate

as we increase width we need to move less to affect output

thought experiment $\hat{f} = 2f$



$\Rightarrow$ double the network to $\hat{f}$

$\{w_1, w_2\}$ = both are duplicates of w

random matrix theory to prove this for depth-2

(V) Analyze real gradient flow $w(t)$

(1) evolution in prediction space: $\frac{d}{dt} \lambda f(w(t)) = \lambda J_t \dot{w}(t) = -\lambda^2 \underbrace{J_t J_t^T}_{\text{2 function of time!}} \nabla R(\lambda f(w(t)))$

(2) notation $z(t) \triangleq \lambda f(w(t)) \quad Q(t) \triangleq \lambda J_t J_t^T$

Lemma 2: If: $\dot{z}(t) = -Q(t) \nabla R(z(t))$

$\lambda \triangleq \inf_{t \in [0,z]} \lambda_{\min}(Q) > 0$

Then: for $t \in [0, z]$, $R(z(t)) \le R(z(0))\, e^{-ct\lambda}$

Proof: 1) Gronwall's Inequality

$[a,b]$ interval $\dot{u}(t) \le p(t) u(t)$, v differentiable on $[a,b]$

then u is upper-bounded by sol. to

$$\dot{v}(t) = p(t) v(t)$$

assuming same boundary conditions

$\Rightarrow u(t) \le u(0)\, e^{\int_0^t p(s) ds}$

2) look at

$$\frac{d}{dt} \frac{1}{2} \|z(t) - y\|^2 = \langle \dot{z}, z - y \rangle = \langle -Q(t) \cdot \nabla R(z(t)), z - y \rangle$$

$= \langle -\varphi(t)\cdot(z-y),\ z-y\rangle \le -2\lambda \frac{\|z(t)-y\|^2}{2}$

So $\frac{d}{dt}R(z(t)) \le -2\lambda R(z(t))$. Now we can use Gronwall's Ineq.

Corollary: $\|z(t)-y\| \le \|z(0)-y\|\, e^{-t\lambda}$

Next Steps: where we don't go too far away from init

Notation:

$v(t) \triangleq w(t)$

$g(v(t)) \triangleq \nabla f(w(t))$

$S(t) \triangleq \partial J_\phi$

Lemma 3

$\dot v(t) = -S(t)^T \nabla R(g(v(t)))$

$\varphi(t) = S(t)S(t)^T \quad , t\in[0,2]$

$\lambda I \le \varphi(t) \le \lambda_{max} I$

Then: $\|v(t) - v(0)\| \le \frac{\sqrt{\lambda_{max}}}{\lambda}\|g(v(0)) - y\|$

$\le \frac{\sqrt{2\lambda_{max} R(g(v(0)))}}{\lambda}$

Proof: Triangle Inequality + Corollary

Discussion: We need $z^2\sigma_{min} I \le \varphi(t) \le z^2\sigma_{max} I$ (3)

Epilogue: What random initialization puts you in NTK regime?

$x \to A_i x_i + b_i$; Let $A_i = \frac{1}{\sqrt{m_i}} W_i$