Lecture 16. (Sketch.)

- Today will be some optimization loose ends; this lecture can be useful for project ideas.

Final comments on "handling approximate gradients".

Here was our main result:

Theorem. Suppose closed convex S and convex f given, and $((w_i, g_i))_{i=1}^t$ from subgradient descent with $\mathbb{E}(g_i|w_{i-1}) \in \partial f(w_{i-1})$ and $\eta := D/(G\sqrt{t})$ with $G \geq \max_i \max\{\|g_i\|, \|\mathbb{E}(g_i|w_{i-1})\|\}$. For any $z \in S$,

$$\mathbb{E}\left(f(\hat{w}_t) - f(z)\right) \leq \mathbb{E}\left(\frac{1}{t}\sum_{i\leq t}\left(f(w_i) - f(z)\right)\right) \leq \frac{DG}{\sqrt{t}},$$

and with probability at least $1-\delta$ over the stochastic gradients,

$$f(\hat{w}_t) - f(z) \leq \frac{1}{t}\sum_{i\leq t}\left(f(w_i) - f(z)\right) \leq \frac{DG\left(1+\sqrt{8\ln(1/\delta)}\right)}{\sqrt{t}}.$$

Remarks.

- We have freedom in how we *use* the bound, namely what the random distribution is.
  - We can make $f = \widehat{\mathcal{R}}$ and obtain a guarantee over a finite training set $S := ((x_i, y_i))_{i=1}^n$; e.g., with a convex ℓ and linear predictor, if in every round we sample (x_j, y_j) uniformly at random from S and use the stochastic gradient $-\ell'(-\langle w_{i-1}, x_j y_j\rangle)x_j y_j$, the theorem gives a bound on $\widehat{\mathcal{R}}$. This is a *randomized algorithm* and the probability distribution is over the behavior of this algorithm.
  - If S (in the preceding point) was drawn iid from some distribution and we use each example at most once, then we have a guarantee on $\mathcal{R}$. (Using examples more than once breaks the martingale in the proof! In practice, people do multiple passes, so the preceding guarantee should be used.)
  - People are not consistent about calling one or the other "stochastic gradient descent"! So when you see a theorem of this type, you need to check whether the expectation is over a randomized algorithm or over a distribution providing examples!

Remarks (continued).

- Some people call $\widehat{\mathcal{R}}$ (as above) the "finite sum" setting. **(Project idea)** Some recent algorithms for this setting are SVRG, SDCA, ...
- What does that δ probability mass throw out in the above examples? For instance, the situation that we draw the same example in every round.
- In practice, *random permutation* is generally used: the algorithm picks an ordering of the training set, performs sgd with this ordering, then picks another ordering, performs sgd again, etc. **(Project idea)** There is ongoing work on this topic, but still it is considered open.

Summary of optimization bounds we've covered.

- Lipschitz / approximate gradient setting.
  - Good news: approximate/noisy gradients. Bad news: needs convexity, compactness/projections, rate is $1/sqrtt$, averaged rather than last iterate.
- Smooth setting.
  - Good news: $1/\sqrt{t}$ rate for the gradient norms (though not last iterate), $1/t$ rate for (convex) function value with last iterate. Bad news: smoothness assumption needs work for deep learning.
- Gradient flow.
  - Good news: similar rates and guarantees to smoothness without assuming it, math is easier and people are using it with deep learning. Bad news: maybe gradient descents discrete steps help with bad local optima?
- Smooth and strongly convex: didn't discuss much (other than proving $\exp(-\lambda t/\beta)$ rate) since doesn't seem relevant for deep learning (I think...) and proofs are similar.

Neural network landscape results.

One active line of work studying deep learning focuses on the shape of the landscape, most results focusing on cases where all local optima are global.

- Matrix completion: solve (under RIP)

$$\min_{X \in d \times r} \sum_{(i,j) \in S} (M_{i,j} - XX^\top)^2.$$

  Recently it was shown that all local optima are global, and so gradient descent from random initialization suffices (Ge, Lee, and Ma 2016).
- For linear networks optimized with the squared loss, local optima are global, but there are bad saddle points (Kawaguchi 2016).
- There are also a few works on residual networks *(but I haven't looked closely)*.

Cubic regularization and friends.

Recall how Nesterov-Polyak cubic regularization selects its next iterate:

$$\arg\min_{w'} \left(f(w) + \left\langle \nabla f(w), w' - w \right\rangle + \frac{1}{2} \left\langle \nabla^2 f(w)^{-1}(w' - w), w' - w \right\rangle + \frac{L}{6} \|w' - w\|^3 \right),$$

where $\|\nabla^2 f(x) - \nabla^2 f(y)\| \le L\|x - y\|$; after t iterations, some iterate w satisfies

$$\|\nabla f(w)\| \le \frac{\mathcal{O}(1)}{t^{2/3}}, \qquad \nabla^2 f(w) \succeq -\frac{\mathcal{O}(1)}{t^{1/3}}.$$

This is better than what smooth gradient descent gave, but we have to solve that (cubic polynomial!) minimization problem.

(Project idea.) Many papers either improve this method (Carmon and Duchi 2018), or give alternative algorithms with similar guarantees (Jin et al. 2017). (Crawl the citation graph from there to find many more.)

Implicit regularization.

Another approach to deep learning is to show that gradient descent not only minimizes empirical risk, but also finds low complexity solutions.

(Project idea) For instance, crawl the citation graph from here: (Bartlett, Foster, and Telgarsky 2017), (Soudry, Hoffer, and Srebro 2017), (Ji and Telgarsky 2018).

I am involved in this line of research so perhaps treat my comments with skepticism.

Momentum and acceleration.

- Consider gradient descent *with momentum*: x_0 arbitrary, and thereafter

$$y_{i+1} := x_i - \eta_i \nabla f(x_i), \qquad x_{i+1} := y_{i+1} + \gamma_i (y_{i+1} - y_i)$$

- This seems to help in deep learning, but no one knows why.
- If set $\eta_i = 1/\beta$ and $\gamma_i = i/(i+3)$ **(constants matter)**, $f(x_i) - \inf_{x \in X} f(x) \leq \mathcal{O}(1/t^2)$ ("Nesterov's accelerated method"). This rate is tight amongst algorithms outputting iterates in the span of gradients, under some assumptions people treat as standard.
- **(Project idea)** Accelerated methods (in both convex and non-convex cases) are an active area of research.

Online learning and online-to-batch.

- We briefly discussed online learning, but didn't discuss how to convert an online guarantee into a guarantee in expectation. There are also versions of this in high probability.
- An algorithm is called "no regret" if

$$\sum_i \ell_i(w_{i-1}) - \min_{w \in \mathcal{W}} \ell_i(w) = o(t).$$

  Suppose $((x_i, y_i))_{i=1}^t$ are drawn iid and define $\ell_i(w) = \ell(\langle w, -x_i y_i \rangle)$; we want a guarantee about $\mathbb{E}\ell(w) = \mathbb{E}\ell(\langle w, -XY \rangle)$.
- For random classifier $\tilde{w}$ and average classifier $\hat{w}$ with convex ℓ:

$$\mathbb{E} \sum_i \ell(w_{i-1}) = \mathbb{E} \sum_i \ell_i(w_{i-1}),$$

$$\mathbb{E}\ell(\tilde{w}) = \frac{1}{t} \mathbb{E} \sum_i \ell_i(w_{i-1}),$$

$$\mathbb{E}\ell(\hat{w}) \leq \frac{1}{t} \mathbb{E} \sum_i \ell_i(w_{i-1}).$$

Proof. First guarantee:

$$\begin{aligned}
\mathbb{E} \sum_i \ell(w_{i-1}) &= \sum_i \mathbb{E}\ell(w_{i-1}) \\
&= \sum_i \mathbb{E}\mathbb{E}(\ell(w_{i-1}) | w_{i-1}) \\
&= \sum_i \mathbb{E}\mathbb{E}(\ell(\langle w_{i-1}, -x_i y_i \rangle) | w_{i-1}) \\
&= \mathbb{E} \sum_i \ell_i(w_{i-1}).
\end{aligned}$$

(We had to be a little careful here and break out a conditional expectation because w_{i-1} depends on $((x_j, y_j))_{j=1}^{i-1}$.) Using this, the randomize classifier satisfies

$$\mathbb{E}\ell(\tilde{w}) = \mathbb{E}\frac{1}{t}\sum_i \ell(w_i) = \frac{1}{t}\mathbb{E}\sum_i \ell_i(w_{i-1}),$$

and the averaged classifier satisfies (in the convex case, by Jensen's inequality)

$$\mathbb{E}\ell(\hat{w}) = \mathbb{E}\ell(\frac{1}{t}\sum_i w_i) \leq \mathbb{E}\frac{1}{t}\sum_i \ell(w_i) = \frac{1}{t}\mathbb{E}\sum_i \ell_i(w_{i-1}).$$

Adapting to problem geometry; proximal gradient and mirror descent.

- Gradient descent iteration can be written as

$$\arg\min_{w'} \left(\left\langle \nabla f(w), w' \right\rangle + \frac{1}{2\eta} \|w - w_i\|^2 \right).$$

- We can generalize this to *proximal gradient* update (to minimize $f + h$):

$$\arg\min_{w'} \left(h(w') + \left\langle \nabla f(w), w' \right\rangle + \frac{1}{2\eta} \|w - w_i\|^2 \right).$$

  How tractable this is depends on h; e.g., $h = 0$ is gradient descent, $h = \iota_C$ (indicator on a convex set) we handled last lecture, but another case is $h(w') = \|w'\|_1$ (see lasso solvers and "iterative shrinkage").

- Define *divergence* $D(w', w) = \|w' - w\|_2^2/2$; can use this generalize gradient descent to **mirror descent**:

$$\arg\min_{w'} \left(\eta \left\langle \nabla f(w), w' \right\rangle + D_g(w', w) \right),$$

where **Bregman divergence** D_g is of the form

$$D_g(w', w) = g(w') - \left(g(w) + \left\langle \nabla g(w), w' - w \right\rangle \right)$$

where g is convex, generally strongly convex. (Gradient descent uses $g(w) = \|w\|_2^2/2$. Another key setting has $g(w) = \sum_i w_i \ln(w_i)$ and $D_g(w', w) = \mathrm{KL}(w', w) = \sum_i w_i' \ln(w_i'/w_i)$.)

- Applying first-order optimality conditions to this minimization:

$$\nabla g(w') = \nabla g(w) - \eta \nabla f(w),$$

which by properties of the **Fenchel conjugate** g^*

$$w' = \nabla g^*(\nabla g(w) - \eta \nabla f(w)).$$

This gives the second standard form of mirror descent.

- Taking these ideas further gives "AdaGrad". AFAICS, AdaGrad and SVRG/SDCA were used as the basis of Adam/AdaDelta, etc. **(Project idea)** study all this...

References

Bartlett, Peter, Dylan Foster, and Matus Telgarsky. 2017. "Spectrally-Normalized Margin Bounds for Neural Networks." *NIPS*.

Carmon, Yair, and John C. Duchi. 2018. "Analysis of Krylov Subspace Solutions of Regularized Nonconvex Quadratic Problems." In *NIPS*.

Ge, Rong, Jason D. Lee, and Tengyu Ma. 2016. "Matrix Completion Has No Spurious Local Minimum." In *NIPS*.

Ji, Ziwei, and Matus Telgarsky. 2018. "Gradient Descent Aligns the Layers of Deep Linear Networks." *arXiv:1810.02032 [cs.LG]*.

Jin, Chi, Rong Ge, Praneeth Netrapalli, Sham M. Kakade, and Michael I. Jordan. 2017. "How to Escape Saddle Points Efficiently." In *ICML*.

Kawaguchi, Kenji. 2016. "Deep Learning Without Poor Local Minima." In *NIPS*.

Soudry, Daniel, Elad Hoffer, and Nathan Srebro. 2017. "The Implicit Bias of Gradient Descent on Separable Data." *arXiv Preprint arXiv:1710.10345*.