Non-Convex Optimization

CS6787 Lecture 7 — Fall 2017

First some words about grading

- I sent out a bunch of grades on the course management system
- Everyone should have all their grades in
 - Not including paper review #6
- If you submitted something and it's not on CMS, send me an email

Also some reminders about the reviews

- Paper reviews should be at least one page in length
 - You can format it however you want, but please don't do things that are obviously intended to pad the length (like making the font size larger than 12pt, or making the margins huge).
- Also, be sure to do at least:
 - 1. Summarize the paper
 - 2. Discuss the paper's strengths and weaknesses
 - 3. Discuss the paper's impact.

Non-Convex Optimization

CS6787 Lecture 7 — Fall 2017

Review — We've covered many methods

- Stochastic gradient descent
- Mini-batching
- Momentum
- Variance reduction
- Nice convergence proofs that give us a rate
- But only for convex problems!

Non-Convex Problems

• Anything that's **not convex**



What makes non-convex optimization hard?

- Potentially many local minima
- Saddle points
- Very flat regions
- Widely varying curvature



https://commons.wikimedia.org/wiki/File:Saddle_point.svg

But is it actually that hard?

- Yes, non-convex optimization is at least NP-hard
 - Can encode most problems as non-convex optimization problems
- Example: subset sum problem
 - Given a set of integers, is there a non-empty subset whose sum is zero?
 - Known to be NP-complete.
- How do we encode this as an optimization problem?

Subset sum as non-convex optimization

- Let a_1, a_2, \ldots, a_n be the input integers
- Let $\mathbf{x_1}, \mathbf{x_2}, \dots, \mathbf{x_n}$ be 1 if $\mathbf{a_i}$ is in the subset, and 0 otherwise
- Objective: minimize $(a^T x)^2 + \sum_{i=1}^n x_i^2 (1 - x_i)^2$
- What is the optimum if subset sum returns true? What if it's false?

So non-convex optimization is pretty hard

- There can't be a general algorithm to solve it efficiently in all cases
- Downsides: theoretical guarantees are weak or nonexistent
 - Depending on the application
 - There's usually no theoretical recipe for setting hyperparameters
- Upside: an endless array of **problems to try to solve better**
 - And gain theoretical insight about
 - And improve the performance of implementations

Examples of non-convex problems

- Matrix completion, principle component analysis
- Low-rank models and tensor decomposition
- Maximum likelihood estimation with hidden variables
 - Usually non-convex
- The big one: deep neural networks

Why are neural networks non-convex?

- They're often made of convex parts!
 - This by itself would be convex.



- Composition of convex functions is not convex
 - So deep neural networks also aren't convex

Why do neural nets need to be non-convex?

- Neural networks are universal function approximators
 - With enough neurons, they can learn to approximate any function arbitrarily well
- To do this, they need to be able to approximate non-convex functions
 - Convex functions can't approximate non-convex ones well.
- Neural nets also have many symmetric configurations
 - For example, exchanging intermediate neurons
 - This symmetry means they can't be convex. Why?

How to solve non-convex problems?

- Can use many of the same techniques as before
 - Stochastic gradient descent
 - Mini-batching
 - SVRG
 - Momentum
- There are also specialized methods for solving non-convex problems
 - Alternating minimization methods
 - Branch-and-bound methods
 - These generally aren't very popular for machine learning problems

Varieties of theoretical convergence results

- Convergence to a stationary point
- Convergence to a local minimum
- Local convergence to the global minimum
- Global convergence to the global minimum

Non-convex Stochastic Gradient Descent

Stochastic Gradient Descent

• The update rule is the same for non-convex functions

$$w_{t+1} = w_t - \alpha_t \nabla \tilde{f}_t(w_t)$$

- Same intuition of moving in a direction that lowers objective
- Doesn't necessarily go towards optimum
 - Even in expectation



Non-convex SGD: A Systems Perspective

- It's exactly the same as the convex case!
- The hardware doesn't care whether our gradients are from a convex function or not
- This means that all our intuition about computational efficiency from the convex case directly applies to the non-convex case
- But does our intuition about statistical efficiency also apply?

When can we say SGD converges?

- First, we need to decide what type of convergence we want to show
 - Here I'll just show convergence to a stationary point, the weakest type
- Assumptions:
 - Second-differentiable objective

$$-LI \preceq \nabla^2 f(x) \preceq LI$$

- Lipschitz-continuous gradients
- Noise has bounded variance

$$\mathbf{E}\left[\left\|\nabla \tilde{f}_t(x) - f(x)\right\|^2\right] \le \sigma^2$$

• But no convexity assumption!

Convergence of Non-Convex SGD

• Start with the update rule:

$$w_{t+1} = w_t - \alpha_t \nabla \tilde{f}_t(w_t)$$

• At the next time step, by Taylor's theorem, the objective will be

$$f(w_{t+1}) = f(w_t - \alpha_t \nabla \tilde{f}_t(w_t))$$

= $f(w_t) - \alpha_t \nabla \tilde{f}_t(w_t)^T \nabla f(w_t) + \frac{\alpha_t^2}{2} \nabla \tilde{f}_t(w_t)^T \nabla^2 f(y_t) \nabla \tilde{f}_t(w_t)$

• Taking the expected value

$$\mathbf{E}\left[f(w_{t+1})|w_t\right] \le f(w_t) - \alpha_t \mathbf{E}\left[\nabla \tilde{f}_t(w_t)^T \nabla f(w_t) \middle| w_t\right] + \frac{\alpha_t^2 L}{2} \mathbf{E}\left[\left\|\nabla \tilde{f}_t(w_t)\right\|^2 \middle| w_t\right]$$

• So now we know how the expected value of the objective evolves.

$$\mathbf{E}\left[f(w_{t+1})|w_t\right] \le f(w_t) - \left(\alpha_t - \frac{\alpha_t^2 L}{2}\right) \left\|\nabla f(w_t)\right\|^2 + \frac{\alpha_t^2 \sigma^2 L}{2}$$

• If we set α small enough that 1 - $\alpha L/2 > 1/2$, then

$$\mathbf{E}[f(w_{t+1})|w_t] \le f(w_t) - \frac{\alpha_t}{2} \|\nabla f(w_t)\|^2 + \frac{\alpha_t^2 \sigma^2 L}{2}$$

• Now taking the full expectation,

$$\mathbf{E}\left[f(w_{t+1})\right] \leq \mathbf{E}\left[f(w_t)\right] - \frac{\alpha_t}{2}\mathbf{E}\left[\left\|\nabla f(w_t)\right\|^2\right] + \frac{\alpha_t^2 \sigma^2 L}{2}.$$

 \bullet And summing up over an epoch of length ${\bf T}$

$$\mathbf{E}[f(w_T)] \le f(w_0) - \sum_{t=0}^{T-1} \frac{\alpha_t}{2} \mathbf{E}\left[\|\nabla f(w_t)\|^2 \right] + \sum_{t=0}^{T-1} \frac{\alpha_t^2 \sigma^2 L}{2}.$$

- Now we need to decide how to set the step size $\boldsymbol{\alpha}_t$
 - Let's just set it to be decreasing like we did in the convex setting

$$\alpha_t = \frac{\alpha_0}{t+1}$$

• So our bound on the objective becomes

$$\mathbf{E}\left[f(w_T)\right] \le f(w_0) - \sum_{t=0}^{T-1} \frac{\alpha_0}{2(t+1)} \mathbf{E}\left[\|\nabla f(w_t)\|^2\right] + \sum_{t=0}^{T-1} \frac{\alpha_0^2 \sigma^2 L}{2(t+1)^2}.$$

• Rearranging the terms,

$$\sum_{t=0}^{T-1} \frac{\alpha_0}{2(t+1)} \mathbf{E} \left[\|\nabla f(w_t)\|^2 \right] \le f(w_0) - \mathbf{E} \left[f(w_T) \right] + \sum_{t=0}^{T-1} \frac{\alpha_0^2 \sigma^2 L}{2(t+1)^2}$$

Now, we're kinda stuck

• How do we use the bound on this term to say something useful?

$$\sum_{t=0}^{T-1} \frac{\alpha_0}{2(t+1)} \mathbf{E} \left[\left\| \nabla f(w_t) \right\|^2 \right]$$

- Idea: rather than outputting w_T , instead output some randomly chosen w_i from the history.
 - You might recall this trick from the proof in the SVRG paper.

Let
$$z_T = w_t$$
 with probability $\frac{1}{H_T(t+1)}$, where $H_t = \sum_{t=0}^{T-1} \frac{1}{t+1}$

Using our randomly chosen output

Let $z_T = w_t$ with probability $\frac{1}{H_T(t+1)}$, where $H_t = \sum_{t=0}^{T-1} \frac{1}{t+1}$

• So the expected value of the gradient at this point is

$$\mathbf{E}\left[\left\|\nabla f(z_T)\right\|^2\right] = \sum_{t=0}^{T-1} \mathbf{P}\left(z_T = w_t\right) \cdot \mathbf{E}\left[\left\|\nabla f(w_t)\right\|^2\right]$$
$$= \sum_{t=0}^{T-1} \frac{1}{H_T(t+1)} \mathbf{E}\left[\left\|\nabla f(w_t)\right\|^2\right].$$

• Now we can continue to bound this with

$$\mathbf{E}\left[\left\|\nabla f(z_T)\right\|^2\right] = \frac{2}{\alpha_0 H_T} \sum_{t=0}^{T-1} \frac{\alpha_0}{2(t+1)} \mathbf{E}\left[\left\|\nabla f(w_t)\right\|^2\right]$$

 ${\mbox{ \bullet}}$ This means that for some fixed constant ${\mbox{ C}}$

$$\mathbf{E}\left[\left\|\nabla f(z_T)\right\|^2\right] \le \frac{C}{\log T}$$

• And so in the limit

$$\lim_{T \to \infty} \mathbf{E} \left[\left\| \nabla f(z_T) \right\|^2 \right] = 0.$$

Convergence Takeaways

- So even non-convex SGD converges!
 - In the sense of getting to points where the gradient is arbitrarily small
- But this doesn't mean it goes to a local minimum!
 - Doesn't rule out that it goes to a saddle point, or a local maximum.
 - Doesn't rule out that it goes to a region of very flat but nonzero gradients.
- Certainly doesn't mean that it finds the global optimum
- And the theoretical rate here was really slow

Strengthening these theoretical results Convergence to a local minimum

- Under stronger conditions, can prove that SGD converges to a local minimum
 - For example using the strict saddle property (Ge et al 2015)
- Using even stronger properties, can prove that SGD converges to a local minimum with an explicit convergence rate of 1/T
- But, it's unclear whether common classes of non-convex problems, such as neural nets, actually satisfy these stronger conditions.

Strengthening these theoretical results Local convergence to the global minimum

- Another type of result you'll see are local convergence results
- Main idea: if we **start close enough to the global optimum**, we will converge there with high probability
- Results often give **explicit initialization scheme** that is guaranteed to be close
 - But it's often expensive to run
 - And limited to specific problems



Strengthening these theoretical results Global convergence to a global minimum

- The strongest result is convergence no matter where we initialize
 Like in the convex case
- To prove this, we need a global understanding of the objective
 So it can only apply to a limited class of problems
- For many problems, we know empirically that this doesn't happen
 - Deep neural networks are an example of this

Other types of results

- Bounds on generalization error
 - Roughly: we can't say it'll converge, but we can say that it won't overfit
- Ruling out "spurious local minima"
 - Minima that exist in the training loss, but not in the true/test loss.
- Results that use the Hessian to escape from saddle points
 - By using it to find a descent direction, but rarely enough that it doesn't damage the computational efficiency

One Case Where We Can Show Global Convergence: PCA

Recall: Principal Component Analysis

• Setting: find the dominant eigenvalue-eigenvector pair of a positive semidefinite symmetric matrix **A**.

$$u_1 = \arg\max_x \frac{x^T A x}{x^T x}$$

• Many ways to write this problem, e.g.

 $||B||_F$ is Frobenius norm

$$\|B\|_F^2 = \sum_i \sum_j B_{i,j}^2$$

$$\sqrt{\lambda_1}u_1 = \arg\min_x \|xx^T - A\|_F^2$$

Recall: PCA is Non-Convex

- PCA is not convex in any of its formulations
- Why? Think about the solutions to the problem: u and -u
 - Two distinct solutions \rightarrow can't be convex
- But it turns out that we can still show that with appropriately chosen step sizes, gradient descent converges globally!
 - This is one of the easiest non-convex problems, and a good place to start to understand how a method works on non-convex problems.

Gradient Descent for PCA

• Gradient of the objective is

$$f(x) = \frac{1}{4} \|xx^T - A\|_F^2, \, \nabla f(x) = (xx^T - A)x$$

• Gradient descent update step

$$x_{t+1} = x_t - \alpha_t \left(x_t x_t^T x_t - A x_t \right)$$

Gradient Descent for PCA (continued)

• Choose adaptive step size for parameter η :

$$\alpha_t = \frac{\eta}{1 + \eta x_t^T x_t}$$

• Then we get:

$$\begin{aligned} x_{t+1} &= \left(1 - \frac{\eta}{1 + \eta x_t^T x_t} x_t^T x_t\right) x_t + \frac{\eta}{1 + \eta x_t^T x_t} A x_t \\ &= \frac{1}{1 + \eta x_t^T x_t} \left(\left((1 + \eta x_t^T x_t) - \eta x_t^T x_t\right) x_t + \eta A x_t\right) \\ &= \frac{1}{1 + \eta x_t^T x_t} \left(x_t + \eta A x_t\right) \end{aligned}$$

Gradient Descent for PCA (continued)

• So we're left with

$$x_{t+1} = \frac{1}{1 + \eta x_t^T x_t} (I + \eta A) x_t$$

• And applying this inductively gives us

$$x_T = (I + \eta A)^T x_0 \prod_{t=0}^{T-1} \frac{1}{1 + \eta \|x_t\|^2}$$

Convergence in Direction

• It should be clear that the direction of the iterates converges

$$\frac{x_T}{\|x_T\|} = \frac{(I + \eta A)^T x_0}{\|(I + \eta A)^T x_0\|}$$

- Same as power iteration!
 - So we know that it converges at a linear rate.
- But what about the magnitude of the iterates?

Convergence in Magnitude

• Imagine we've already converged in direction. Then our update becomes

$$x_{t+1} = \frac{1}{1+\eta \|x_t\|^2} (1+\eta \lambda_1) x_t$$

- Why does this converge?
 - If x is large, it will become small in a single step
 - If x is small, it will increase slowly by a factor of about $(1 + \eta \lambda_1)$ until it converges to the optimal value.

Why did this work?

- The PCA objective has no non-optimal local minima
 - This means finding a local optimum is as good as solving the problem

$$f(x) = \frac{1}{4} \|xx^T - A\|_F^2, \, \nabla f(x) = (xx^T - A)x$$

- We took advantage of the algebraic properties
 - And the fact that we already knew about power iteration
 - We used this to choose an adaptive step size seemingly out of nowhere

Stochastic Gradient Descent for PCA

- We can use the same logic to show that a variant of SGD with the same adaptive step sizes works for PCA
 - And we can give an **explicit convergence rate**
- The proof is long and involved
- With more work, can even show that variants with momentum and variance reduction also work
 - Means we can use the same techniques we are used to for this problem too

Can we generalize these results?

- Difficult to generalize!
 - Especially to problems like neural nets that are hard to analyze algebraically
- This PCA objective is one of the simplest non-convex problems
 It's just a degree-4 polynomial
- But these results can **give us intuition** about how our methods apply to the non-convex setting
 - To understand a method, PCA is a good place to start

Deep Learning as Non-Convex Optimization

Or, "what could go wrong with my non-convex learning algorithm?"

Lots of Interesting Problems are Non-Convex

- Including deep neural networks
- Because of this, we almost always **can't prove convergence** or anything like that when we run backpropagation (SGD) on a deep net
- But can we use intuition from PCA and convex optimization to understand what could go wrong when we run non-convex optimization on these complicated problems?

What could go wrong? We could converge to a bad local minimum

- Problem: we converge to a local minimum which is bad for our task
 Often in a very steep potential well
- One way to debug: re-run the system with **different initialization**
 - Hopefully it will converge to some other local minimum which might be better
- Another way to debug: add extra noise to gradient updates
 - Sometimes called "stochastic gradient Langevin dynamics"
 - Intuition: extra noise pushes us out of the steep potential well

What could go wrong? We could converge to a saddle point

- Problem: we converge to a saddle point, which is not locally optimal
- Upside: usually doesn't happen with plain SGD
 - Because noisy gradients push us away from the saddle point
 - But can happen with more sophisticated SGD-like algorithms
- One way to debug: find the Hessian and compute a descent direction

What could go wrong? We get stuck in a region of low gradient magnitude

- Problem: we converge to a region where the gradient's magnitude is small, and then stay there for a very long time
 - Might not affect asymptotic convergence, but very bad for real systems
- One way to debug: use specialized techniques like batchnorm
 - There are many methods for preventing this problem for neural nets
- Another way to debug: design your network so that it doesn't happen
 - Networks using a **RELU activation** tend to avoid this problem

What could go wrong? Due to high curvature, we do huge steps and diverge

- Problem: we go to a region where the gradient's magnitude is very large, and then we make a series of very large steps and diverge
 - Especially bad for real systems using floating point arithmetic
- One way to debug: use adaptive step size
 - Like we did for PCA
 - Adam (which we'll discuss on Wednesday) does this sort of thing
- A simple way to debug: just limit the size of the gradient step
 - But this can lead to the low-gradient-magnitude issue

What could go wrong? I don't know how to set my hyperparameters

- Problem: without theory, how on earth am I supposed to set my hyperparameters?
- We already have discussed the solution: hyperparameter optimization
 All the techniques we discussed apply to the non-convex case.
- To avoid this: just use hyperparameters from folklore

Takeaway

- Non-convex optimization is hard to write theory about
- But it's just as easy to compute SGD on
 - This is why we're seeing a renaissance of empirical computing
- We can use the techniques we have discussed to get speedup here too
- We can apply intuition from the convex case and from simple problems like PCA to learn how these techniques work

Questions?

- Upcoming things
 - Paper review #6 due Today
 - Paper Presentation #7 on Wednesday read paper before class