Lecture 6: Minibatching and Decreasing Step Sizes

CS4787 — Principles of Large-Scale Machine Learning Systems

Where we left off: we looked at how stochastic gradient descent performs without assuming convexity.

Gradient descent for strongly convex objectives. This was without assuming strong convexity. But how does SGD perform on strongly convex problems? As before, we start from this sort of expression

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

and apply the Polyak–Lojasiewicz condition,

$$\left\|\nabla f(x)\right\|^2 \ge 2\mu\left(f(x) - f^*\right);$$

this gives us

$$\mathbf{E}\left[f(w_{t+1})\right] \le \mathbf{E}\left[f(w_t)\right] - \mu\alpha\mathbf{E}\left[f(w_t) - f^*\right] + \frac{\alpha^2\sigma^2 L}{2}.$$

Subtracting f^* from both sides, we get

$$\mathbf{E}\left[f(w_{t+1}) - f^*\right] \le (1-\mu\alpha)\mathbf{E}\left[f(w_t) - f^*\right] + \frac{\alpha^2\sigma^2 L}{2}.$$

Now subtracting the fixed point from both sides gives us

$$\begin{aligned}\mathbf{E}\left[f(w_{t+1}) - f^*\right] - \frac{\alpha^2\sigma^2 L}{2\mu\alpha} &\le (1-\mu\alpha)\mathbf{E}\left[f(w_t) - f^*\right] + \frac{\alpha^2\sigma^2 L}{2} - \frac{\alpha^2\sigma^2 L}{2\mu\alpha} \\ &= (1-\mu\alpha)\left(\mathbf{E}\left[f(w_t) - f^*\right] - \frac{\alpha^2\sigma^2 L}{2\mu\alpha}\right).\end{aligned}$$

Now applying this recursively,

$$\mathbf{E}\left[f(w_T) - f^*\right] - \frac{\alpha^2\sigma^2 L}{4\mu\alpha} \le (1-\mu\alpha)^K\left(f(w_0) - f^* - \frac{\alpha^2\sigma^2 L}{2\mu\alpha}\right),$$

and so since $(1-\mu\alpha) \le \exp(-\mu\alpha)$,

$$\mathbf{E}\left[f(w_T) - f^*\right] \le \exp(-\mu\alpha K)\cdot\left(f(w_0) - f^*\right) + \frac{\alpha\sigma^2 L}{2\mu}.$$

What can we learn from this expression?

Previously, with gradient descent, if we wanted to get a solution of a desired level of accuracy (either small gradient or small objective gap) we could just keep running until we observed a gradient small enough to satisfy our desires. Now though, this won't necessarily happen.

One way to achieve a desired level of error is to choose an α and T as a function of the error level. For example, for non-convex SGD, if for some $\epsilon > 0$ we want to guarantee that we will get

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

it suffices to pick α and T such that

$$\frac{2\left(f(w_0) - f^*\right)}{\alpha T} = \frac{\alpha \sigma^2 L}{2} = \frac{\epsilon}{2}.$$

This happens when

$$\alpha = \frac{\epsilon}{\sigma^2 L} \qquad \text{and} \qquad T = \frac{4\sigma^2 L\left(f(w_0) - f^*\right)}{\epsilon^2}.$$

This can be compared with our results from gradient descent (Lecture 2) where we could get the same guarantee with

$$\alpha = \frac{1}{L} \qquad \text{and} \qquad T \leq \frac{2L(f(w_0) - f^*)}{\epsilon}.$$

Similarly, for strongly convex SGD, if we want to guarantee that

$$\mathbf{E}\left[f(w_T) - f^*\right] \leq \epsilon,$$

it suffices to pick α and T such that

$$\exp(-\mu\alpha T) \cdot (f(w_0) - f^*) = \frac{\alpha\sigma^2 L}{2\mu} = \frac{\epsilon}{2}.$$

This happens when (letting $\kappa = L/\mu$ as usual)

$$\alpha = \frac{\epsilon}{\sigma^2 \kappa} \qquad \text{and} \qquad T = \frac{\sigma^2 \kappa}{\mu \epsilon} \log\left(\frac{2\left(f(w_0) - f^*\right)}{\epsilon}\right).$$

In comparison, gradient descent (Lecture 3) had

$$T \geq \kappa \cdot \log\left(\frac{f(w_0) - f^*}{\epsilon}\right).$$

What can we conclude from this? Here's one thing that we can get: the asymptotic runtime used by these algorithms. For each of non-convex GD/SGD and strongly convex GD/SGD, write a big-$\mathcal{O}$ expression for the total amount of compute that would be done by the algorithm to achieve error ϵ. Give your result in terms of ϵ, κ (for strongly-convex), n, and σ^2, treating all other expressions (such as $f(w_0) - f^*$) as constant.

When might one algorithm be better than the other?

Minibatching. One way to make all these rates smaller is by decreasing the value of σ^2. A simple way to do this is by using *minibatching*. With minibatching, we use a sample of the gradient examples of size larger than 1. That is, our update rule looks likee

$$w_{t+1} = w_t - \alpha_t \sum_{b=1}^{B} \nabla f_{\tilde{i}_{t,b}}(w_t).$$

If the batch size is B, this results in an estimator with variance B times smaller.

How does this trade off work for faster convergence?

Diminishing Step Size Rules. We will see how we can get an “optimal” step size from the analysis of convex SGD, starting with the expression (from the Lecture 4 notes)

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