Lecture 9: Diminishing step sizes and accelerating SGD with averaging.

CS4787 — Principles of Large-Scale Machine Learning Systems

Recap: Over the last week, we looked at a couple of ways to accelerate learning by using techniques that help decrease the dependence on the condition number of the problem κ. One of these methods was AdaGrad, which we saw had the side effect of decreasing the step size over time. Doing this will cause the algorithm to converge asymptotically towards having low gradient magnitude, unlike plain constant-step-size SGD which converges to a noise ball.

Recall that from the Lecture 4 notes, we had that (for strongly convex SGD)

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

Last week, we looked at the effect of the κ term on this and other algorithms; today, we'll look at the effect of the α and σ^2 parts of the noise ball, and discuss some ways to address the effect of these to accelerate SGD.

Increasing minibatch sizes. One approach to target the variance term is to use a minibatch size that becomes larger over time. Here, we'll look at how this can effect convergence in the non-convex optimization setting (just to vary our setting from our diminishing step size case, not because either method is particularly suited to either setting).

If we use a constant learning rate and a minibatch of size B_t at time t, then (from the analysis we did in Lecture 5), we would get

$$\mathbf{E}\left[f(w_{t+1})\right] \le \mathbf{E}\left[f(w_t)\right] - \frac{\alpha}{2} \cdot \mathbf{E}\left[\|\nabla f(w_t)\|^2\right] + \underbrace{\frac{\alpha^2 L}{2} \cdot \mathbf{E}\left[\left\|\frac{1}{B_t}\sum_{b=1}^{B_t} \nabla f_{i_{b,t}}(w_t) - \nabla f(w_t)\right\|^2\right]}_{\text{second-order variance/error term}}.$$

If (as we usually assume) the variance of an individual example gradient is bounded by σ^2 as

$$\frac{1}{n}\sum_{i=1}^{n} \|\nabla f_i(w_t) - \nabla f(w_t)\|^2 \le \sigma^2,$$

then by the analysis we've already done for minibatch SGD we know that

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

which implies that

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

Summing this up over T iterations of SGD, and telescoping the sum as usual, we get

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

which implies that

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

This means that any increasing batch size scheme is going to converge, and if the sum of the reciprocals of the batch sizes converges, then SGD with this scheme will converge at a rate of $1/T$.

We can do a similar analysis for convex problems...I won't discuss this in class but there's a great analysis in Chapter 5 of "Optimization Methods for Large-Scale Machine Learning" if you are curious.

Polyak averaging. Intuition: SGD is converging to a "noise ball" where the iterates are randomly jumping around some space surrounding the optimum. We can think about these iterates as random samples that approximate the optimum.

What can we do when we have a bunch of random samples that approximate something to improve the precision of our estimate?

Technique: run regular SGD and just average the iterates. That is,

$$\begin{aligned}w_{t+1} &= w_t - \alpha_t \nabla f_{i_t}(w_t) \\ \bar{w}_{t+1} &= \frac{t}{t+1}\cdot \bar{w}_t + \frac{1}{t+1}\cdot w_{t+1};\end{aligned}$$

eventually we output the average $\bar{w}_T$ at the end of execution. This is equivalent to writing

$$\bar{w}_T = \frac{1}{T}\sum_{t=1}^{T} w_t.$$

The main idea here is that we're averaging out a bunch of iterations w_t that are all in the noise ball. When you average out a bunch of noisy things, often you are able to reduce the noise.

To gain intuition about Polyak averaging, let's look at a simple one-dimensional quadratic...

$$f(w) = \frac{1}{2}w^2$$

with example gradients

$$\nabla f_i(w) = w + u_i$$

where u_i considered as a random variable over all $i \in \{1, \ldots, n\}$ has mean 0 and variance σ^2, i.e.

$$\frac{1}{n}\sum_{i=1}^{n} u_i = 0 \qquad \text{and} \qquad \frac{1}{n}\sum_{i=1}^{n} u_i^2 = \sigma^2.$$

Suppose that we run SGD on this with a constant learning rate α. Our update step will be

$$w_{t+1} = w_t - \alpha w_t - \alpha u_{i_t} = (1-\alpha) w_t - \alpha u_{i_t}.$$

Applying this recursively, we get

$$w_T = (1-\alpha)^T w_0 - \underbrace{\alpha \sum_{t=0}^{T-1} (1-\alpha)^{T-1-t} u_{i_t}}_{\text{noise term}}.$$

To tell how basic SGD compares with averaged SGD, we want to compare the expected loss f evaluated at both these points. For regular SGD, it is

$$\begin{aligned}
\mathbf{E}\left[f(w_T)\right] &= \frac{1}{2}\mathbf{E}\left[w_T^2\right] \\
&= \frac{1}{2}\mathbf{E}\left[\left((1-\alpha)^T w_0 - \alpha \sum_{t=0}^{T-1}(1-\alpha)^{T-1-t} u_{i_t}\right)^2\right] \\
&= \frac{1}{2}\mathbf{E}\left[\left((1-\alpha)^T w_0 - \alpha \sum_{t=0}^{T-1}(1-\alpha)^{T-1-t} u_{i_t}\right)\left((1-\alpha)^T w_0 - \alpha \sum_{s=0}^{T-1}(1-\alpha)^{T-1-s} u_{i_s}\right)\right] \\
&= \frac{1}{2}\Bigg((1-\alpha)^{2T} w_0^2 - 2\alpha(1-\alpha)^T w_0 \sum_{t=0}^{T-1}(1-\alpha)^{T-1-t} \cdot \mathbf{E}\left[u_{i_t}\right] \\
&\qquad + \alpha^2 \sum_{t=0}^{T-1}\sum_{s=0}^{T-1}(1-\alpha)^{T-1-t} \cdot (1-\alpha)^{T-1-s} \cdot \mathbf{E}\left[u_{i_t} u_{i_s}\right]\Bigg).
\end{aligned}$$

Now, since the u_{i_t} are zero mean and independent, it follows that

$$\mathbf{E}\left[\mathbf{E}\left[u_{i_t}\right]\right] = 0.$$

For the same reason, it will also be the case that if $s \neq t$,

$$\mathbf{E}\left[u_{i_t} u_{i_s}\right] = 0.$$

On the other hand, if $s = t$, then

$$\mathbf{E}\left[u_{i_t} u_{i_s}\right] = Exvu_{i_t}^2 = \sigma^2.$$

As a result, all the cross terms in the above sum will cancel out to zero, and we can write the expected loss as

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

Next, if we let $k = T-1-t$, then summing from $t=0$ to $T-1$ is equivalent to summing from $k=0$ to $T-1$, so

$$\begin{aligned}
\mathbf{E}\left[f(w_T)\right] &= \frac{1}{2}\left((1-\alpha)^{2T} w_0^2 + \alpha^2 \sum_{k=0}^{T-1}(1-\alpha)^{2k}\sigma^2\right) \\
&= \frac{1}{2}\left((1-\alpha)^{2T} w_0^2 + \alpha^2\sigma^2 \frac{1-(1-\alpha)^{2T}}{1-(1-\alpha)^2}\right) \\
&= \frac{1}{2}\left((1-\alpha)^{2T} w_0^2 + \alpha\sigma^2 \frac{1-(1-\alpha)^{2T}}{2-\alpha}\right) \\
&= (1-\alpha)^{2T}\left(f(w_0) - \frac{\alpha\sigma^2}{2(2-\alpha)}\right) + \frac{\alpha\sigma^2}{2(2-\alpha)}
\end{aligned}$$

where in the derivation above, we used the fact that for any x,

$$\sum_{n=0}^{N-1} x^n = \frac{1 - x^N}{1 - x}.$$

Note that this expression is exact! We didn't use any approximations or inequalities here, but if we want a somewhat simpler expression we can upper bound this with

$$\mathbf{E}\left[f(w_T)\right] = (1-\alpha)^{2T} \cdot f(w_0) + \frac{\alpha\sigma^2}{2(2-\alpha)}$$

Now let's do the same analysis for the Polyak-averaged updates. If we use averaging, the output of the algorithm becomes

$$\begin{aligned}\bar{w}_T &= \frac{1}{T}\sum_{k=0}^{T-1}\left((1-\alpha)^k w_0 - \alpha\sum_{t=0}^{k-1}(1-\alpha)^{k-1-t}u_{i_t}\right)\\ &= \frac{1}{T}\left(\sum_{k=0}^{T-1}(1-\alpha)^k w_0 - \alpha\sum_{k=0}^{T-1}\sum_{t=0}^{k-1}(1-\alpha)^{k-1-t}u_{i_t}\right).\end{aligned}$$

The pair of sums in the second term here results in the value $(1-\alpha)^{k-1-t}u_{i_t}$ being summed up for all pairs (k,t) that satisfy

$$0 \le k < T \qquad \text{and} \qquad 0 \le t < k.$$

This condition is equivalent to $0 \le t < k < T$, and so we could also write it as

$$0 \le t < T-1 \qquad \text{and} \qquad t < k < T.$$

That is, these inequalities correspond to the same ordered pairs of (k,t). As a result, we can re-order our summations above into

$$\bar{w}_T = \frac{1}{T}\left(\sum_{k=0}^{T-1}(1-\alpha)^k w_0 - \alpha\sum_{t=0}^{T-2}\sum_{k=t+1}^{T-1}(1-\alpha)^{k-1-t}u_{i_t}\right),$$

since the resulting expression sums up the exact same terms. Next, if we substitute $l = k-1-t$ in the inner sum, then

$$\begin{aligned}\bar{w}_T &= \frac{1}{T}\left(\sum_{k=0}^{T-1}(1-\alpha)^k w_0 - \alpha\sum_{t=0}^{T-2}\sum_{l=0}^{T-t-2}(1-\alpha)^{l}u_{i_t}\right)\\ &= \frac{1}{T}\left(\frac{1-(1-\alpha)^T}{1-(1-\alpha)}\cdot w_0 - \alpha\sum_{t=0}^{T-2}\frac{1-(1-\alpha)^{T-t-1}}{1-(1-\alpha)}\cdot u_{i_t}\right)\\ &= \frac{1}{T}\left(\frac{1-(1-\alpha)^T}{\alpha}\cdot w_0 - \sum_{t=0}^{T-2}\left(1-(1-\alpha)^{T-t-1}\right)\cdot u_{i_t}\right).\end{aligned}$$

If we compute the expected value of the loss function f evaluated at this point, then as before, all the cross terms will cancel, and we'll just be left with

$$\begin{aligned}\mathbf{E}\left[f(\bar{w}_T)\right] &= \frac{1}{2}\mathbf{E}\left[\bar{w}_T^2\right]\\ &= \frac{1}{2}\mathbf{E}\left[\left(\frac{1}{T}\left(\frac{1-(1-\alpha)^T}{\alpha}\cdot w_0 - \sum_{t=0}^{T-2}\cdot\left(1-(1-\alpha)^{T-t-1}\right)\cdot u_{i_t}\right)\right)^2\right]\\ &= \frac{1}{2T^2}\left(\left(\frac{1-(1-\alpha)^T}{\alpha}\cdot w_0\right)^2 + \sum_{t=0}^{T-2}\left(1-(1-\alpha)^{T-t-1}\right)^2\cdot\mathbf{E}\left[u_{i_t}^2\right]\right)\\ &= \frac{1}{2T^2}\left(\left(\frac{1-(1-\alpha)^T}{\alpha}\cdot w_0\right)^2 + \sum_{t=0}^{T-2}\left(1-(1-\alpha)^{T-t-1}\right)^2\cdot\sigma^2\right).\end{aligned}$$

Next, I'm going to simplify the above expression by noting that for sufficiently small α (i.e. $0 < \alpha \leq 1$) the term $1 - (1-\alpha)^{T-t-1} \leq 1$, and so

$$\sum_{t=0}^{T-2} \left(1 - (1-\alpha)^{T-t-1}\right)^2 \leq T.$$

Using this, and the fact that $1 - (1-\alpha)^T \leq 1$, we can get the simplified upper bound

$$\begin{aligned}
\mathbf{E}\left[f(\bar{w}_T)\right] &\leq \frac{1}{2T^2}\left(\left(\frac{1}{\alpha} \cdot w_0\right)^2 + T \cdot \sigma^2\right) \\
&= \frac{w_0^2}{2\alpha^2 T^2} + \frac{\sigma^2}{2T} \\
&= \frac{f(w_0)}{\alpha^2 T^2} + \frac{\sigma^2}{2T}.
\end{aligned}$$

Now, let's compare our rates. For SGD and averaged SGD, we have

$$\mathbf{E}\left[f(w_T)\right] \leq (1-\alpha)^{2T} \cdot f(w_0) + \frac{\alpha\sigma^2}{2(2-\alpha)} \qquad \text{and} \qquad \mathbf{E}\left[f(\bar{w}_T)\right] \leq \frac{f(w_0)}{\alpha^2 T^2} + \frac{\sigma^2}{2T}.$$

When we compare, we notice that the second term (the term that depends on σ^2, i.e. the *variance term*) is actually going to zero as T increases. That is, even with a constant learning rate α, we can get loss values that are arbitrarily close to the actual optimal loss $f^* = f(0) = 0$. (Unfortunately, this is not guaranteed to happen for general convex losses, and only happens here because our loss is a quadratic function.) On the other hand, the term that depends on the initialization w_0 (the first term) is no longer decaying exponentially.

One issue with this is that we are averaging with equal weight iterates from the very start of training, when we have not reached the noise ball. In order to address this, we often run averaged SGD by first using **a warm-up period** during which we do not average, and then only starting to average after the warm-up period is over. Long warm-up periods (such as half of the total number of epochs ran) usually produce better results in practice than averaged SGD without any warmup.

For **Polyak averaging on non-convex problems**, we run into the same issue that we ran into with AdaGrad: as we move through a non-convex landscape, we may get very far away from the parameter values we were at during previous iterations. If this happens, averaging together with those parameter values doesn't really make sense, and can hurt the performance of our algorithm. So, instead, a standard approach is to use an *exponential moving average*, just like was done to transform AdaGrad into RMSProp. That is, we pick some decay factor $0 < \rho < 1$ and run

$$\begin{aligned}
w_{t+1} &= w_t - \alpha_t \nabla f_{i_t}(w_t) \\
\bar{w}_{t+1} &= \rho \cdot \bar{w}_t + (1-\rho) \cdot w_{t+1}.
\end{aligned}$$

This running average approach often outperforms other averaging methods in non-convex settings.

Stochastic variance-reduced gradient. Idea: reduce the variance of the gradient estimators by using an infrequent full-gradient step.

```
Procedure SVRG

Parameters update frequency $m$ and learning rate $\eta$
Initialize $\tilde{w}_0$
Iterate: for $s = 1, 2, \ldots$
  $\tilde{w} = \tilde{w}_{s-1}$
  $\tilde{\mu} = \frac{1}{n} \sum_{i=1}^{n} \nabla \psi_i(\tilde{w})$
  $w_0 = \tilde{w}$
  Iterate: for $t = 1, 2, \ldots, m$
    Randomly pick $i_t \in \{1, \ldots, n\}$ and update weight
      $w_t = w_{t-1} - \eta(\nabla \psi_{i_t}(w_{t-1}) - \nabla \psi_{i_t}(\tilde{w}) + \tilde{\mu})$
  end
  option I: set $\tilde{w}_s = w_m$
  option II: set $\tilde{w}_s = w_t$ for randomly chosen $t \in \{0, \ldots, m-1\}$
end
```