Lecture Notes 5: Short tips on practice: Tuning and Learning Rate Schedules

Instructor: Ashok Cutkosky

1 Tuning Learning Rates

So far, we have discussed setting the learning rate η using what is sometimes called "oracle tuning". That is, the tuning is in some way magical: we do not actually know the various parameters Δ, H, G, σ or T that are required to set the learning rate, so in reality we probably cannot use these theoretically motivated learning rates. In practice, one usually simply guesses the learning rate. That is, you try out a few learning rates and see how well the resulting outputs behave on some test set. Then, you just hope that you tried a learning rate that is good.

- When tuning learning rates (or regularization constants or usually any real-valued parameter), it is practically effective to tune on a *logarithmic scale*. That is, you might test the learning rates $10, 1.0, 0.1, 0.01, 0.001$.
- It is a good idea to make sure that the "best" learning rate is not on the extreme edge of the range you tuned over.
- If you are tuning more than just the learning rate (e.g. also tuning a regularization constant), you might want to tune via *random search*. That is, choose some values $x_1, x_2, \dots$ at random from $[-3, 1]$ and test learning rate of 10^{x_i}. This is helpful for avoiding issue in which the learning rate and the regularization parameter have correlated effects (e.g, increasing regularization may be similar to decreasing learning rates).

2 Learning Rate Schedules

We have discussed setting the learning rate for SGD as $\propto \frac{1}{\sqrt{t}}$ for optimal theoretical bounds. However, in practice there are a wide range of options used. We'll just discuss a few prominent ones.

Many of these schedules are set based on the number of *epochs* that have elapsed, where an epoch is one whole pass over the training set. However, this is not always the case, as some datasets are so large that the total number of epochs may be very small (GPT3 was trained for less than a single epoch). In these cases, one might consider setting the schedule as a function of the number of iterations instead.

- **Exponential Decay**: In this schedule, after a certain number of epochs, we decrease the learning rate by a constant factor. For example, when training models on the ImageNet dataset for object recognition, it is common to employ SGD with a constant learning rate, and decrease the learning rate by a factor of 10 every 30 epochs for 90 total epochs.

  Intuitively, the idea is that after 30 epochs the optimization has converged roughly to the limit of how well it can converge using the given learning rate. By decreasing the learning rate, it is possible for the optimizer to explore higher-resolution features of the loss surface. Typically, one observes a rapid decrease in loss in the first iterations after decreasing the learning rate.

- **Cosine Annealing**. This schedule was introduced by [1] and uses the cosine function to set the learning rate:

$$\eta_t = \eta_{\min} + \frac{1}{2}\left(\eta_{\max} - \eta_{\min}\right)\left(1 + \cos\left(\frac{t}{T}\pi\right)\right)$$

where $\eta_{\min}$ and $\eta_{\max}$ are some chosen minimum and maximum learning rates. It is also possible to combine cosine annealing with exponential decay. In this setup, one would divide the training into N phases. Let $T_1, \ldots, T_N$ be the lengths of the phases. Then in phase i, we re-index the step-counter t so that phase i starts with $t = 1$, and use the learning rate:

$$\eta_t = \eta_{\min,i} + \frac{1}{2}\left(\eta_{\max,i} - \eta_{\min}\right)\left(1 + \cos\left(\frac{t}{T_i}\pi\right)\right)$$

where $\eta_{\min,i}$ and $\eta_{\max,i}$ are potentially decaying learning rate ranges for each phase. This approach means that the learning rate will actually *increase* at the start of each phase, leading to a kind of oscillatory behavior in the loss. Thus, one should take care to only return the parameters $\hat{w}$ that occur at the end of the last completed phase.

The cosine annealing schedule is quite effective empirically, but unfortunately there is essentially no good mathematical justification for it. While it is in some sense inspired by a notion of "restarting" that is useful in convex optimization, there is no formal connection, and indeed the use of the cosine function is essentially unheard of in convex optimization. It's also not clear if the cosine function itself is particularly important here.

- **Scaling learning rate when increasing batch size**. This rule is fairly simple: if you scale the batch-size by a factor B, but keep the same number of total gradient computations (i.e. *epochs*), then you should scale the learning rate also by a factor B.

  One of the main points of increasing the batch size is to train "faster" in the sense of spending less time by parallelizing the computation of the batch gradients. This is only useful if we keep the total computation constant, so if we keep the number of passes over the training set constant (each pass is called one "epoch"), then increasing the batch size from 1 to B is accompanied by a corresponsing *decrease* in the total number of SGD update from T to T/B. Thus, looking back at the previous lecture notes, we see that if the optimal learning rate with batch size 1 and T iterations is $O(1/\sqrt{T})$, then with batch size B and T/B iterations, the optimal learning rate is $O(\sqrt{B}/\sqrt{T/B}) = O(B/\sqrt{T})$ because the variance of the gradients goes down by a factor of B.

- **Warm-up**. See [2]. This schedule is empirically motivated in the large batch-size regime. In this idea, when training with a very large batch size, one should actually start with a very small learning rate and slowly increase the learning rate during the earlier iterations. Once a certain small number of epochs has elapsed, then the learning rate is allowed to remain constant or decrease following some other more typical schedule. For example, it is common to apply *linear warm-up:*

$$\eta_t = \begin{cases} \frac{t\eta_0}{T_{\text{warm-up}}} & t \leq T_{\text{warm-up}} \\ \text{some other schedule (e.g. } \propto 1/\sqrt{t}) & t > T_{\text{warm-up}} \end{cases}$$

  Here η_0 is some base or "target" learning rate, that we are slowly increasing η_t until this rate is reached. Afterwards, we follow some arbitrary other schedule, typically such that the first learning rate of the other schedule is η_0.

2.1 Some more technical intuition for warm-up

Let's try to gain some intuition behind why this might be a good idea. The essential claim is that this procedure roughly causes the iterates produced by SGD using a large batch of size B to be close to the iterates produced via SGD using a batch size of 1 but a B-times smaller learning rate. In the following, we'll flesh out this argument more formally. Note that we will *not* show a strong theorem here, instead we'll just try to provide a little intuition about what might be going on. So far as I am aware, a truly complete formal understanding of warm-up is not yet known.

We have already discussed how increasing the batch size should be generally accompanied by an increase in the learning rate, so that it seems reasonable that these two learning rate/batch size schemes should have similar guarantees about convergence to critical points.

Intuitively, however, there is an alternative way to justify scaling the learning rate when the batch size is increased. Instead of trying to reason about the function value $\mathcal{L}(\mathbf{w}_t)$, we could try to understand the actual iterates $\mathbf{w}_t$ themselves. Specifically, consider two scenarios, a batch-size of 1 and a learning rate of η, and a batch size of B and a learning rate $\eta^{(B)}$. Let $\mathbf{w}_t^{(1)}$ be the tth iterate with batch size 1 and $\mathbf{w}_t^{(B)}$ be the tth iterate with batch size B. Suppose both batch sizes start at the same initial point $\mathbf{w}_1$. Then after B steps with batch-size 1, we have:

$$\mathbf{w}_{B+1}^{(1)} = \mathbf{w}_1 - \eta \sum_{t=1}^{B} \nabla \ell(\mathbf{w}_t^{(1)}, z_t)$$

while after 1 step of batch-size B,

$$\mathbf{w}_2^{(B)} = \mathbf{w}_1 - \eta^{(B)} \frac{1}{B} \sum_{t=1}^{B} \nabla \ell(\mathbf{w}_1, z_t)$$

Now, if it happened to be true that $\nabla \ell(\mathbf{w}_1, z_t) \approx \nabla \ell(\mathbf{w}_t^{(1)}, z_t)$, and $\eta^{(B)} = B\eta$, then we would have $\mathbf{w}_{B+1}^{(1)} \approx \mathbf{w}_2^{(B)}$, so that after B gradient evaluations, the two procedures have reached nearly the same point.

Let's try to characterize when this happens more formally. We already should suspect that the learning rate should never exceed $\frac{1}{H}$ when $\mathcal{L}$ is an H-smooth function. Let's assume a slightly stronger statement: $\eta^{(B)} \leq \frac{1}{2H}$ so that $\eta^{(1)} \leq \frac{1}{2BH}$. Further, define $r_t^{(1)} = \nabla \ell(\mathbf{w}_t^{(1)}, z_t) - \nabla \mathcal{L}(\mathbf{w}_t^{(1)})$. Notice that $\mathbb{E}[r_t^{(1)}] = 0$. Let's also suppose $\mathbb{E}[\|r_t^{(1)}\|^2] \leq \sigma^2$ for some σ. Similarly, define $r_t^{(B)} = \nabla \ell(\mathbf{w}_1, z_t) - \nabla \mathcal{L}(\mathbf{w}_1)$, and assume $\mathbb{E}[\|r_t^{(B)}\|^2] \leq \sigma^2$. Now we can write:

$$\begin{aligned}
\mathbf{w}_{B+1}^{(1)} - \mathbf{w}_1 &= -\eta \sum_{t=1}^{B} \nabla \ell(\mathbf{w}_t^{(1)}, z_t) \\
&= -\eta \sum_{t=1}^{B} (\nabla \ell(\mathbf{w}_t^{(1)}, z_t) - \nabla \mathcal{L}(\mathbf{w}_t^{(1)})) + (\nabla \mathcal{L}(\mathbf{w}_t^{(1)}) - \nabla \mathcal{L}(\mathbf{w}_1)) + \nabla \mathcal{L}(\mathbf{w}_1) \\
&= -\eta B \nabla \mathcal{L}(\mathbf{w}_1) - \eta \sum_{t=1}^{B} r_t^{(1)} - \eta \sum_{t=1}^{B} (\nabla \mathcal{L}(\mathbf{w}_t^{(1)}) - \nabla \mathcal{L}(\mathbf{w}_1))
\end{aligned}$$

Also, we have:

$$\begin{aligned}
\mathbf{w}_2^{(B)} - \mathbf{w}_1 &= -\eta^{(B)} \frac{1}{B} \sum_{t=1}^{B} \nabla \ell(\mathbf{w}_1, z_t) \\
&= -\eta \sum_{t=1}^{B} \nabla \mathcal{L}(\mathbf{w}_1) + r_t^{(B)} \\
&= e - \eta B \nabla \mathcal{L}(\mathbf{w}_1) - \eta \sum_{t=1}^{B} r_t^{(B)}
\end{aligned}$$

Thus, we subtract to find the distance between $\mathbf{w}_{B+1}^{(1)}$ and $\mathbf{w}_2^{(B)}$:

$$\mathbf{w}_{B+1}^{(1)} - \mathbf{w}_2^{(B)} = -\eta \sum_{t=1}^{B} (r_t^{(1)} - r_t^{(B)}) - \eta \sum_{t=1}^{B} (\nabla \mathcal{L}(\mathbf{w}_t^{(1)}) - \nabla \mathcal{L}(\mathbf{w}_1))$$

Now, let us define $p_t = r_t^{(1)} - r_t^{(B)}$. Notice that $\mathbb{E}[p_t] = 0$ and $\mathbb{E}[\|p_t\|^2] \leq \mathbb{E}[2\|r_t^{(1)}\|^2 + 2\|r_t^{(B)}\|^2] \leq 4\sigma^2$. Then, we have shown:

$$\begin{aligned}
\mathbf{w}_{B+1}^{(1)} - \mathbf{w}_2^{(B)} &= -\eta \left[\sum_{t=1}^{B} p_t + \sum_{t=1}^{B} (\nabla \mathcal{L}(\mathbf{w}_t^{(1)}) - \nabla \mathcal{L}(\mathbf{w}_1)) \right] \\
\|\mathbf{w}_{B+1}^{(1)} - \mathbf{w}_2^{(B)}\| &\leq \eta \left[\left\| \sum_{t=1}^{B} p_t \right\| + \sum_{t=1}^{B} \|\nabla \mathcal{L}(\mathbf{w}_t^{(1)}) - \nabla \mathcal{L}(\mathbf{w}_1)\| \right] \\
&\leq \eta \left[\left\| \sum_{t=1}^{B} p_t \right\| + H \sum_{t=1}^{B} \|\mathbf{w}_t^{(1)} - \mathbf{w}_1\| \right]
\end{aligned}$$

Now, observe that we have p_t are mean zero, variance at most $4\sigma^2$ and also p_t and $p_{t'}$ are uncorrelated for any $t \neq t'$. Therefore:

$$\mathbb{E}\left[\left\| \sum_{t=1}^{B} p_t \right\| \right] \leq \sqrt{\mathbb{E}\left[\left\| \sum_{t=1}^{B} p_t \right\|^2 \right]} \leq 2\sigma\sqrt{B}$$

Thus:

$$\mathbb{E}[\|\mathbf{w}_{B+1}^{(1)} - \mathbf{w}_2^{(B)}\|] \leq 2\sigma\eta\sqrt{B} + \eta H \, \mathbb{E}\left[\sum_{t=1}^{B} \|\mathbf{w}_t^{(1)} - \mathbf{w}_1\| \right]$$

Now, let's turn to bounding $\mathbb{E}\left[\sum_{t=1}^{B} \|\mathbf{w}_t^{(1)} - \mathbf{w}_1\|\right]$. By previous arguments, we have:

$$\begin{aligned}
\mathbf{w}_t^{(1)} - \mathbf{w}_1 &= -\eta(t-1)\nabla \mathcal{L}(\mathbf{w}_1) - \eta \sum_{t'=1}^{t-1} r_{t'}^{(1)} - \eta \sum_{t'=1}^{t-1} (\nabla \mathcal{L}(\mathbf{w}_{t'}^{(1)}) - \nabla \mathcal{L}(\mathbf{w}_1)) \\
\|\mathbf{w}_t^{(1)} - \mathbf{w}_1\| &\leq \eta(t-1)\|\nabla \mathcal{L}(\mathbf{w}_1)\| + \eta \left\| \sum_{t'=1}^{t-1} r_{t'}^{(1)} \right\| + \eta \sum_{t'=1}^{t-1} H \|\mathbf{w}_{t'}^{(1)} - \mathbf{w}_1\|
\end{aligned}$$

summing over t:

$$\sum_{t=1}^{B} \|\mathbf{w}_t^{(1)} - \mathbf{w}_1\| \leq \frac{\eta B(B-1)}{2} \|\nabla \mathcal{L}(\mathbf{w}_1)\| + \eta \sum_{t=1}^{B} \left\| \sum_{t'=1}^{t-1} r_{t'}^{(1)} \right\| + \eta H B \left(\sum_{t=1}^{B} \|\mathbf{w}_t^{(1)} - \mathbf{w}_1\| \right)$$

Using $\eta \leq \frac{1}{2BH}$:

$$\begin{aligned}
&\leq \frac{\eta B(B-1)}{2} \|\nabla \mathcal{L}(\mathbf{w}_1)\| + \eta \sum_{t=1}^{B} \left\| \sum_{t'=1}^{t-1} r_{t'}^{(1)} \right\| + \frac{1}{2} \sum_{t=1}^{B} \|\mathbf{w}_t^{(1)} - \mathbf{w}_1\| \\
\sum_{t=1}^{B} \|\mathbf{w}_t^{(1)} - \mathbf{w}_1\| &\leq \eta B(B-1)\|\nabla \mathcal{L}(\mathbf{w}_1)\| + 2\eta \sum_{t=1}^{B} \left\| \sum_{t'=1}^{t-1} r_{t'}^{(1)} \right\| \\
\mathbb{E}\left[\sum_{t=1}^{B} \|\mathbf{w}_t^{(1)} - \mathbf{w}_1\| \right] &\leq \eta B^2 \|\nabla \mathcal{L}(\mathbf{w}_1)\| + 4\eta\sigma B^{3/2}
\end{aligned}$$

where in the last line, we have use $\mathbb{E}\left[\left\|\sum_{t'=1}^{t-1} r_{t'}^{(1)}\right\|\right] \leq 2\sigma\sqrt{t}$ and $\sum_{t=1}^{B} \sqrt{t} \leq 2B^{3/2}$. To see the latter identity, notice that since $\sqrt{x}$ is an increasing function:

$$\int_{t}^{t+1} \sqrt{x} \, dx \geq \sqrt{t}$$

So that

$$\begin{aligned}\sum_{t=1}^{B}\sqrt{t} &\leq \int_{1}^{B+1}\sqrt{x}\,dx = \frac{2}{3}(B+1)^{3/2} - \frac{2}{3} \\ &\leq 2B^{3/2}\end{aligned}$$

Therefore:

$$\begin{aligned}\mathbb{E}[\|\mathbf{w}_{B+1}^{(1)} - \mathbf{w}_{2}^{(B)}\|] &\leq 2\sigma\eta\sqrt{B} + \eta H\,\mathbb{E}\left[\sum_{t=1}^{B}\|\mathbf{w}_{t}^{(1)} - \mathbf{w}_{1}\|\right] \\ &\leq 2\sigma\eta\sqrt{B} + \eta^{2}HB^{2}\|\nabla\mathcal{L}(\mathbf{w}_{1})\| + 4\eta^{2}H\sigma B^{3/2}\end{aligned}$$

Using $\eta \leq \frac{1}{2HB}$:

$$\begin{aligned}&\leq 4\sigma\eta\sqrt{B} + \frac{\eta B}{2}\|\nabla\mathcal{L}(\mathbf{w}_{1})\| \\ &\leq \frac{2\sigma}{H\sqrt{B}} + \frac{\eta B}{2}\|\nabla\mathcal{L}(\mathbf{w}_{1})\|\end{aligned}$$

Now, the first term above is $O(1/\sqrt{B})$, and so is actually small as B increases. However, the second term may be as large as $O(\|\nabla\mathcal{L}(\mathbf{w}_{1})\|)$ and so could actually be quite large, *unless* $\|\nabla\mathcal{L}(\mathbf{w}_{1})\|$ *is somewhat small*.

This provides some alternative view of linearly scaling the learning rate: if the gradient is already somewhat small, then we should expect that B steps of batch-size 1 is approximately *equal* to 1 step of batch size B. If the gradient is *not* small, this will only be true if we set the learning rate much smaller than the $\frac{1}{H}$ limit suggested by simply looking at the function decrease calculations we performed previously.

As a result, in the earlier iterations, when we might intuitively expect $\nabla\mathcal{L}(\mathbf{w})$ to be large, we might try using an artificially small learning rate instead of $\eta^{(B)} = \frac{1}{2H}$, and slowly increase as $\nabla\mathcal{L}(\mathbf{w})$ (hopefully) decreases, until we reach a point when $\nabla\mathcal{L}(\mathbf{w})$ is small enough that we can use our original target learning rate of $\frac{1}{2H}$.

References

[1] Ilya Loshchilov and Frank Hutter. "Sgdr: Stochastic gradient descent with warm restarts". In: *arXiv preprint arXiv:1608.03983* (2016).

[2] Priya Goyal et al. "Accurate, large minibatch sgd: Training imagenet in 1 hour". In: *arXiv preprint arXiv:1706.02677* (2017).