Chapter 10

Accelerated Gradient Descent

Reading

1. The blog-post titled "Why momentum really works?" at https://distill.pub/2017/momentum

In the previous chapter we saw two results that characterize how many iterations gradient descent requires to reach within an ϵ-neighborhood of the global optimum for convex functions. If the function $\ell(w)$ is convex, GD with a step-size at most $1/L$ requires $\mathcal{O}(1/\epsilon)$ iterations. If the function $\ell(w)$ is strongly convex, then the step-size can be as large as $2/(m+L)$ and GD requires $\mathcal{O}(\kappa \log(1/\epsilon))$ iterations where

$$\kappa = \frac{L}{m}$$

is the condition number of the Hessian $\nabla^2 \ell(w)$. A large value of κ means that there are some directions where the function is highly curved and others where it is relatively flat. The main point to remember is that we often do not know a good value for m, L. Since the step-size of GD depends upon the curvature of the function; if the step-size is too large then GD overshoots on the highly curved directions and if the step-size is too small then GD makes slow progress along a direction.

We will study two algorithms in this chapter which accelerate the progress of gradient descent.

10.1 Polyak's Heavy Ball method

The most natural place to begin is to imagine gradient descent as a kinematic equation. Let w^t be the iterate of GD at time t, let us associate to it an auxiliary

variable called the "velocity" v^t

$$v^t := \frac{w^{t+1} - w^t}{\eta}. \tag{10.1}$$

Gradient descent can then be written as

$$v^t = -\nabla \ell(w^t), \tag{10.2}$$

which allows us to think of the term $-\nabla \ell(w^t)$ is the force that acts on a particle to update its position from w^t to w^{t+1}. This particle has no inertia, so the force applied directly affects its position. If the magnitude of this gradient is small in a certain direction, the velocity is also small in that direction.

We now give our particle some inertia. Instead of the force directly affecting the position we will write down Newton's second law of motion ($F = ma$) for a particle with unit mass $m = 1$ as

$$\begin{aligned} -\nabla \ell(w^t) &=: \frac{v^{t+1} - v^t}{\eta} \\ &= \frac{1}{\eta}\left(w^{t+1} - 2w^t + w^{t-1}\right) \\ \Rightarrow w^{t+1} &= w^t - \eta \nabla \ell(w^t) + \left(w^t - w^{t-1}\right). \end{aligned} \tag{10.3}$$

Notice the third term on the right-hand side above, it is the gap between the current weights w^t and the previous weights w^{t-1}, if we have

$$\left\langle w^t - w^{t-1}, \nabla \ell(w^t) \right\rangle < 0,$$

i.e., the change from the previous time-step is along the descent direction, then the weights w^{t+1} get an extra boost. If instead, the change from the previous direction is not along the gradient descent direction, then the third term reduces the magnitude of the gradient. The third term is effectively the inertia of gradient updates. This method is therefore called Polyak's Heavy Ball method.

> We give ourselves some more control over how inertia enters the update equation using a hyper-parameter ρ
>
> $$w^{t+1} = w^t - \eta \nabla \ell(w^t) + \rho\left(w^t - w^{t-1}\right). \tag{10.4}$$
>
> If $\rho = 0$, we do not use any inertia and Polyak's method boils down to gradient descent. Typically, we choose $\rho \in (0, 1)$. This inertia is called *momentum* in the optimization literature and ρ is called the momentum coefficient.

Polyak's method is simple yet very powerful. In the previous chapter, we showed a lower-bound of Nesterov which indicates that first-order optimization algorithm (that only depends on the gradient of the objective) cannot be faster than $\mathcal{O}(1/\sqrt{\epsilon})$. It turns out that Polyak's method converges at this rate, i.e., if we want

$$\|w^t - w^*\| \leq \epsilon$$

we need to run Polyak's Heavy Ball method for $\mathcal{O}(1/\sqrt{\epsilon})$ iterations for convex functions. If the function is strongly convex, the number of iterations comes down to

$$\mathcal{O}(\sqrt{\kappa}\log(1/\epsilon)).$$

Both of these are improvements upon their convergence rates for gradient descent. These improvements are also quite a lot, we need quadratically fewer iterations than gradient descent in Polyak's method and the only incremental cost of doing so is that we have to maintain a copy of the weights w^{t+1} while implementing the updates in (10.4).

An alternative way to write Polyak's updates We can rewrite the updates in (10.4) using a dummy variable u^t as

$$\begin{aligned} u^t &= (1+\rho)w^t - \rho\, w^{t-1} \\ w^{t+1} &= u^t - \eta\,\nabla\,\ell(w^t). \end{aligned} \tag{10.5}$$

This is how these updates are implemented in PyTorch. This is convenient: effectively, the code needs to maintain only the difference $u^t = (1+\rho)w^t - \rho w^{t-1}$ in a buffer u^t and subtract the gradient $\nabla\,\ell(w^t)$ from this update to result in the new updates. GD can be implemented with a simple change by setting $u^t := w^t$. *The dummy variable is initialized to* $u^0 = w^0$.

A yet another way to write Polyak's updates We can also rewrite the updates in (10.5) as

$$\begin{aligned} u^{t+1} &= \rho\, u^t - \nabla\,\ell(w^t) \\ w^{t+1} &= w^t + \eta\, u^{t+1}. \end{aligned} \tag{10.6}$$

This set of updates brings out idea of momentum more clearly. The variable u^t in this case is exactly the velocity v^t that we have seen above except that it is updated slightly different than our expression ($F = ma$) in the first equation. The first term

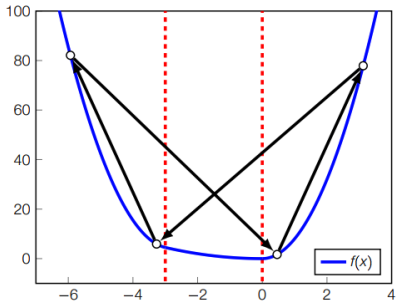
$$u^{t+1} = \rho\, u^t - \nabla\,\ell(w^t)$$

reduces the velocity u^t by a factor ρ before adding the gradient to it.

⚠ Draw Polyak's updates for a two-dimensional function.

10.1.1 Polyak's method can fail to converge

The caveat with relying on the inertia of the particle to make progress is that near the global minimum, when the iterates overshoot the global minimum, the inertia is often very different from the gradient. Polyak's method can become unstable and can result in oscillations under such conditions, e.g.,



However it is a very simple method to accelerate gradient descent and works great in practice.

10.2 Nesterov's method

Nesterov's method is an advanced version of Polyak's method and removes the oscillations towards the end. Let us understand these oscillations better. We saw that incorporating a notion of inertia in Polyak's method gave us accelerated convergence; this is intuitive, if the velocity is along the descent direction the particle descends faster. The same inertia hurts towards the end because the velocity can be very different than the gradient when the particle overshoots the minimum.

A simple way to fix this is to incorporate damping (friction) into Newton's law of motion; you can read about the simple harmonic oscillator at https://en.wikipedia.org/wiki/Harmonic_oscillator. We are going to write

$$ma = F - cv.$$

where m is the mass, c is the coefficient of damping, a and v are acceleration and velocity respectively and F is the force as usual. The effective force decreases with the velocity. Doing so does not slow down the weight updates much when the gradient magnitude is large at the beginning of training. But when the gradient magnitude is small (when the particle is in the neighborhood of the global minimum), this friction prevents excessive overshooting.

With that background, let us see how Nesterov's updates for gradient descent look.

We will write a similar set up of updates as that of (10.6). Nesterov's

updates correspond to

$$\begin{aligned} u^{t+1} &= \rho\, u^t - \nabla\, \ell(w^t + \eta\, \rho\, u^t) \\ w^{t+1} &= w^t + \eta\, u^{t+1}. \end{aligned} \tag{10.7}$$

The only difference between (10.7) and (10.6) is the term in blue; effectively the gradient is computed as if the weights w^t moved using the velocity u^t; and then this new velocity u^{t+1} is used to obtain the new updates w^{t+1}. Nesterov's method solves the problem of instability in Polyak's method by essentially computing the gradient (the blue term) as given by the current velocity. You can see how this *slows down* the updates if the velocity is well-aligned with the gradient; we are reducing the benefit of inertia/momentum. However, doing so prevents overshooting as the iterates reach the neighborhood of the global minimum.

An alternative way to write Nesterov's updates We can rewrite the updates in (10.7) to look like those in (10.5), to get

$$\begin{aligned} u^t &= (1+\rho) w^t - \rho\, w^{t-1} \\ w^{t+1} &= u^t - \eta \nabla\, \ell(u^t). \end{aligned} \tag{10.8}$$

Again the blue term is the only difference between Polyak's method and Nesterov's method. The term u^t is conceptually a forecasted version of the weights w^t because notice that

$$u^t = w^t + \rho(w^t - w^{t-1}).$$

The new weights w^{t+1} are now obtained by thinking of u^t as the actual weights.

10.2.1 Yet another way to write Nesterov's updates

We now tie back Nesterov's updates to our introductory narrative on friction. We will set the damping coefficient (ρ) in (10.8) to a special value

$$\rho = \frac{t-1}{t+2};$$

effectively as $t \to \infty$ the friction becomes larger and larger. This simplifies our updates to

$$\begin{aligned} u^t &= w^t + \frac{t-1}{t+2}\left(w^t - w^{t-1}\right) \\ w^{t+1} &= u^t - \eta\, \nabla\, \ell(u^t). \end{aligned}$$

which upon collapsing together give

$$w^{t+1} - w^t = \frac{t-1}{t+2}\left(w^t - w^{t-1}\right) - \eta\, \nabla\, \ell(u^t). \tag{10.9}$$

We now choose a different way of interpreting these updates. We will imagine that the sequence

$$\left\{w^0, w^1, \ldots, w^t, w^{t+1}, \ldots\right\}$$

is the discretization of a smooth curve

$$\{W(\tau) : \tau \in [0, \infty)\}.$$

How is this curve $W(\tau)$ related to the original sequence? We are going to study the updates under the setting

$$\tau := \sqrt{\eta}\, t. \tag{10.10}$$

Essentially the time of the discrete-time updates (10.9) increments in intervals of 1, but the time of the curve $W(\tau)$ is real-number. Each increment in discrete-time corresponds to $\sqrt{\eta}$ increment of the time τ for the curve $W(\tau)$. This gives

$$\begin{aligned} W(\tau) &= w^t \\ W(\tau + \sqrt{\eta}) &= w^{t+1}. \end{aligned}$$

We now do a Taylor expansion for the continuous curve $X(\tau)$ to get

$$\begin{aligned} w^{t+1} - w^t &= W(\tau + \sqrt{\eta}) - W(\tau) \\ &= \dot{W}(\tau)\sqrt{\eta} + \frac{1}{2}\ddot{W}(\tau)\eta + \mathcal{O}(\sqrt{\eta}). \end{aligned} \tag{10.11}$$

Here

$$\dot{W}(\tau) = \frac{\mathrm{d}}{\mathrm{d}\tau} W(\tau), \quad \ddot{W}(\tau) = \frac{\mathrm{d}^2}{\mathrm{d}\tau^2} W(\tau)$$

are the first and second derivative of the curve with respect to time and $\mathcal{O}(\sqrt{\eta})$ denotes higher-order terms. Similarly

$$\begin{aligned} w^t - w^{t-1} &= W(\tau) - W(\tau - \sqrt{\eta}) \\ &= \dot{W}(\tau)\sqrt{\eta} - \frac{1}{2}\ddot{W}(\tau)\eta + \mathcal{O}(\sqrt{\eta}). \end{aligned}$$

We now do a Taylor expansion of the loss term $\nabla \ell(u^t)$ to get

$$\begin{aligned} \nabla \ell(u^t) &= \nabla \ell \left(w^t + \frac{t-1}{t+2} \left(w^t - w^{t-1} \right) \right) \\ &= \nabla \ell(w^t) + \text{higher order terms} \\ &= \nabla \ell(W(\tau)) + \mathcal{O}(\sqrt{\eta}). \end{aligned} \tag{10.12}$$

Substitute (10.11) and (10.12) in (10.9) to get

$$\begin{aligned} \dot{W}(\tau) + \frac{1}{2}\ddot{W}(\tau)\sqrt{\eta} + \mathcal{O}(\sqrt{\eta}) = \left(1 - \frac{3\sqrt{\eta}}{\tau}\right) \left(\dot{W}(\tau) - \frac{1}{2}\ddot{W}(\tau)\sqrt{\eta} + \mathcal{O}(\sqrt{\eta}) \right) \\ - \sqrt{\eta}\ \nabla \ell(W(\tau)) + \mathcal{O}(\sqrt{\eta}). \end{aligned}$$

This equation is true for all values of η, so we can compare the coefficients of $\sqrt{\eta}$ on both sides to get

$$\ddot{W} + \frac{3}{\tau}\dot{W} + \nabla \ell(X) = 0. \tag{10.13}$$

This equation looks very similar to Newton's law with friction $ma + cv = F$. Again, the term $\nabla \ell(W)$ is acting as the force, the second derivative $\ddot{W}$ is the acceleration and the friction term $\frac{3}{t}\dot{W}$ increases with velocity. We have shown

that for a particularly chosen value of the momentum coefficient, Nesterov's updates result in an ordinary differential equation that looks much like that simple harmonic oscillator that most of you have seen before in high-school. This approach gives an alternative, and very simple, way of understanding Nesterov's updates which is nice because the updates in (10.7) and (10.8) were quite non-intuitive and created by Nesterov through a sheer *tour de force*.

Remark 1. Derive a similar ordinary differential equation for Polyak's updates using the same setting of friction $(t-1)/(t+2)$ as that in (10.9). You will notice that if viewed in continuous-time Polyak's updates are exactly the same as Nesterov's updates. This suggests that that the continuous-time construct is a more abstract point-of-view and eliminates the subtle differences between the updates between the two algorithms.

Such continuous-time constructs are however very useful to understand what these updates actually do, e.g., we know that Nesterov's (and Polyak's) updates correspond to having friction in Newton's law which is not apparent by looking at the equations in (10.8). It is also very easy to obtain the convergence rate of the continuous-time version; it is an ordinary differential equation and we can use a lot of tools from dynamical systems, in particular Lyapunov functions. It will amuse you to know that obtaining the convergence rate for Nesterov's updates using the continuous-time version (10.13) takes about half a page.

10.2.2 How to pick the momentum parameter?

Nesterov's updates converge at a rate that is similar to that of Polyak's updates. For convex functions, we need

$$\mathcal{O}(1/\sqrt{\epsilon})$$

iterations to get within the ϵ-neighborhood of the global minimum if we set

$$\rho = (t-1)/(t+2)$$

in (10.6). If we are minimizing a strongly-convex function we can pick the momentum coefficient to depend on m, L: we can set

$$\rho = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} \tag{10.14}$$

and $\eta < 2/(m+L)$. Doing so entails that we need only

$$\mathcal{O}(\sqrt{\kappa}\log(1/\epsilon))$$

weight updates to reach within an ϵ-neighborhood of the global minimum. The expression in (10.14) gives some insight in how momentum accelerates things. If $\kappa \approx 1$, i.e., the Hessian of the objective is well-conditioned without a big diversity in the curvature in different directions, we do not really need friction $\rho \approx 0$ to avoid overshooting close to the minimum; progress in all directions is balanced. On the other hand, if $\kappa \gg 1$, the objective is badly conditioned and the friction coefficient $\rho \approx 1$ should be large to avoid overshooting near the global minimum.

How to pick ρ in practice? If we know what m, L are for a given problem (you will see an example of this in HW 4), it is straightforward to pick the momentum coefficient and get accelerated convergence of gradient descent. If we do not know m, L, we pick some constant value of ρ. For instance, $\rho = 0.9$ is popularly used in most deep learning libraries. Typically, the momentum coefficient is not increased with the number of weight updates using $(t-1)/(t+2)$. You will some heuristics for modifying the momentum coefficient in this week's recitation.