Recall for strongly convex SGD

$$\mathbb{E}[f(w_t) - f^*] \leq \underbrace{e^{-\mu\alpha T}(f(w_0) - f^*)}_{\text{exponential convergent "initialization"}} + \underbrace{\frac{\alpha\sigma^2 K}{2}}_{\text{noise ball}}$$

Previously we examined K term; now we see the impact of α, σ^2 on the noise ball and how to address it.

Diminishing Step Size Schemes

Since α is a parameter we can just set, a natural way to "get at" this term is to just decrease the value of α over time.

This involves a time-varying step size, resulting in an algorithm with update step

$$w_{t+1} \leftarrow w_t - \alpha_t \nabla f_{i_t}(w_t)$$

Problem: Naively, each α_t is a seperate hyperparameter we would need to tune... but choosing so many hyperparameters is prohibitive.

Can we get better intuition for how fast we decrease α_t?

Example: Strongly Convex SGD

For μ-strongly convex, L-Lipschitz gradient, SGD w/ step size $\alpha \leq 1/L$,

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

It follows by the same argument that if instead we use the time varying step size α_t,

$$\mathbb{E}[f(w_{t+1}) - f^*] \leq (1 - \mu\alpha_t)\,\mathbb{E}[f(w_t) - f^*] + \frac{\alpha_t^2 \sigma^2 L}{2}$$

How to choose α_t? Assume we somehow **KNOW** what $\mathbb{E}[f(w_t) - f^*]$ is. What value of α_t should we choose to minimize our bound on the objective gap at the next timestep?

Calculus!

$$-\mu\,\mathbb{E}[f(w_t) - f^*] + \alpha_t \sigma^2 L = 0$$

$$\Rightarrow \qquad \alpha_t = \frac{\mu\,\mathbb{E}[f(w_t) - f^*]}{\sigma^2 L}$$

Assume we know $\mathbb{E}[f(w_t) - f^*] \leq \rho_t$

Setting α_t in this way (assume $\alpha_t \leq 1/L$ still), then

$$\mathbb{E}[f(\omega_{t+1}) - f^*] \leq (1 - \mu\alpha_t)\rho_t + \frac{\alpha_t^2 \sigma^2 L}{2}$$

$$\leq \rho_t - \mu\rho_t \frac{\mu\rho_t}{\sigma^2 L} + \frac{\sigma^2 L}{2} \frac{\mu^2 \rho_t^2}{\sigma^2 L^2}$$

$$\leq \rho_t - \frac{\mu^2 \rho_t^2}{2\sigma^2 L}$$

Set

$$\rho_{t+1} = \rho_t - \frac{\mu^2 \rho_t^2}{2\sigma^2 L}$$

Algebra Trick

$$x^2 \geq x^2 - y^2 = (x+y)(x-y) \Rightarrow \frac{1}{x+y} \geq \frac{x-y}{x^2} = \frac{1}{x} - \frac{y}{x^2}$$

It follows that if we invert the whole expression, then

$$\frac{1}{\rho_{t+1}} = \frac{1}{\rho_t - \frac{\mu^2 \rho_t^2}{2\sigma^2 L}} \geq \frac{1}{\rho_t} + \frac{\mu^2 \rho_t^2}{2\sigma^2 L} \cdot \frac{1}{\rho_t^2}$$

$$= \frac{1}{\rho_t} + \frac{\mu^2}{2\sigma^2 L}$$

By induction, it immediately follows that

$$\frac{1}{\rho_T} \geq \frac{1}{\rho_0} + \frac{\mu^2}{2\sigma^2 L} T$$

$$\Rightarrow \rho_T = \frac{2\sigma^2 L \rho_0}{2\sigma^2 L + \mu^2 T \rho_0}$$

This corresponds to a setting of α_T of

$$\alpha_T = \frac{\mu}{\sigma^2 L} \cdot \frac{2\sigma^2 L \rho_0}{2\sigma^2 L \rho_0 + \mu^2 T \rho_0}$$

$$= \frac{2\mu\rho_0}{2\sigma^2 L + \mu^2 T \rho_0}$$

Observe that this varying learning rate/step size is of the form

$$\alpha_T = \frac{\alpha_0}{1 + \eta T}$$

which is referred to as a $\frac{1}{T}$ step size scheme.

This is a nice way to parametrize a diminishing step size scheme, requiring only two hyperparameters, α_0 and η (as opposed to one for each iteration).

NOTE

Need: $\alpha_0 \leq \frac{1}{L}$

Other options

① For non-convex optimization and non-strongly-convex optimization, often a $1/T$ step size is decreasing too quickly. Here, we adopt a $1/\sqrt{T}$ step size rate that is structured similarly.

② We can also decrease the step sizes quickly in **epochs**, rather than at every iteration

③ For some tasks (e.g. deep learning architectures), it makes sense to **increase** the step size first, and then decrease it. For example, if we expect to move from a region of very high noise early in the optimization to a region of much lower noise later, this could make sense. This approach, especially when repeated is related to annealing and is sometimes called that.

Increasing Minibatch Sizes

Another thing we can do to target the variance term is to use a minibatch size that becomes larger over time.

If we use a constant learning rate and a minibatch of size B_t at time t, then we would get

$$\mathbb{E}[f(w_{t+1})] \leq \mathbb{E}[f(w_t)] - \frac{\alpha}{2}\mathbb{E}\left[\|\nabla f(w_t)\|^2\right] + \frac{\alpha^2 L}{2}\mathbb{E}\left[\left\|\frac{1}{B_t}\sum_{b=1}^{B_t}\nabla f_{i_{b,t}}(w_t) - \nabla f(w_t)\right\|^2\right]$$

Assume the variance of an individual example is bounded by σ^2,

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

then by analysis done earlier for minibatch SGD we know that

$$\mathbb{E}[f(w_{t+1})] \leq \mathbb{E}[f(w_t)] - \frac{\alpha}{2}\mathbb{E}[\|\nabla f(w_t)\|^2] + \frac{\alpha^2\sigma^2 L}{2B_t}$$

which implies that

$$\frac{\alpha}{2}\mathbb{E}[\|\nabla f(w_t)\|^2] \leq \mathbb{E}[f(w_0)] - \mathbb{E}[f(w_{t+1})] + \frac{\alpha^2\sigma^2 L}{2B_t}$$

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

$$\frac{\alpha}{2}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla f(w_t)\|^2] \leq \mathbb{E}[f(w_0)] - \mathbb{E}[f(w_T)] + \sum_{t=0}^{T-1}\frac{\alpha^2\sigma^2 L}{2B_t}$$

$$\leq f(w_0) - f^* + \sum_{t=0}^{T-1}\frac{\alpha^2\sigma^2 L}{2B_t}$$

implying that

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

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

Polyak Averaging

Intuition: SGD converges 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.

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

$$w_{t+1} = w_t - \alpha_t \nabla f_{i_t}(w_t)$$

$$\bar{w}_{t+1} = \frac{t}{t+1} \bar{w}_t + \frac{1}{t+1} w_{t+1}$$

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

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

Extra Memory $O(d)$ → keep running average

Extra Compute $O(Td)$

The main idea is that we're averaging out a bunch of iterations w_t that are all in the noise ball. This tends to reduce the noise.

One issue with this is that we are averaging with equal weight iterates from the very start of training, when we have not even 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 usually produce better results in practice than averaging SGD w/o any warmup.

For **Polyak averaging on non-convex** problems, we run into same issues that we ran into w/ 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 to RMSProp. That is, for decay factor $0 < \rho < 1$, we run

$$w_{t+1} \longleftarrow w_t - \alpha_t \nabla f_{i_t}(w_t)$$

$$\bar{w}_{t+1} \longleftarrow \rho \bar{w}_t + (1-\rho) w_{t+1}$$

* This running avg approach often outperforms other averaging methods in non-convex settings

Variance Reduction

If I know that for some $\tilde{w} \in \mathbb{R}$, $\nabla f(\tilde{w}) = \tilde{g}$, then

$$\nabla f_i(w) - \nabla f_i(\tilde{w}) + \nabla f(\tilde{w}) = \nabla h_i(w)$$

$$\mathbb{E}[\nabla h_i(w)] = \frac{1}{n}\sum_{i=1}^{n} \nabla h_i(w) = \frac{1}{n}\sum_{i=1}^{n}\left(\nabla f_i(w) - \nabla f_i(\tilde{w}) + \nabla f(\tilde{w})\right)$$

$$= \nabla f(w) - \nabla f(\tilde{w}) + \nabla f(\tilde{w})$$

$$= \nabla f(w)$$

$$\mathbb{E}[\|\nabla h_i(w) - \nabla f(w)\|^2] = \mathbb{E}[\|\nabla f_i(w) - \nabla f_i(\tilde{w}) + \nabla f(\tilde{w}) - \nabla f(w)\|^2]$$

$$\leq \mathbb{E}[(\|\nabla f_i(w) - \nabla f_i(\tilde{w})\| + \|\nabla f(w) - \nabla f(\tilde{w})\|)^2]$$

$$\leq \mathbb{E}[(2L\|w - \tilde{w}\|)^2]$$

$$\leq 4L^2\|w - \tilde{w}\|^2$$

Complexity

GD: $\left(nK \log \frac{1}{\varepsilon}\right)$ SVRG: $O\left((n+K)\log \frac{1}{\varepsilon}\right)$

i.e the idea is to reduce the variance of the gradient estimators by using an infrequent full-gradient step.

Procedure SVRG

Parameters: update frequency m and learning rate η

Initialize $\tilde{\omega}_0$

Iterate for $s = 1, 2, \ldots$

$\tilde{\omega} = \tilde{\omega}_{s-1}$

$\tilde{\mu} = \frac{1}{n} \sum_{i=1}^{n} \nabla \psi_i(\tilde{\omega})$

$\omega_0 = \tilde{\omega}$

Iterate for $t = 1, 2, \ldots, m$

Randomly pick $i_t \in \{1, \ldots, n\}$ and update weight

$$\omega_t = \omega_{t-1} - \eta \left(\nabla \psi_{i_t}(\omega_{t-1}) - \nabla \psi_{i_t}(\tilde{\omega}) + \tilde{\mu} \right)$$

end

OPTION I: set $\tilde{\omega}_s = \omega_m$

OPTION II: set $\tilde{\omega}_s = \omega_t$ for random $t \in \{0, \ldots, m-1\}$

end