Lecture Notes 9: Per-Coordinate Learning rates, AdaGrad and Adam

Instructor: Ashok Cutkosky

1 Per-Coordinate Learning Rates

Let's consider a simple linear regression problem:

$$\hat{y} = \langle w, x \rangle$$

Suppose for example that we are trying to predict the identity of a speaker from two options given what they said. We'll say that positive $\hat{y}$ indicates that the speaker is person A and negative $\hat{y}$ indicates that the speaker is person B. Suppose that x is be a bag-of-words feature vector. This means that x is a vector with one coordinate for every possible english word, and the value of x at coordinate i is the number of times word i appears in whatever sentence or paragraph x is representing.

In this setting, the ith coordinate of w represents a kind of "score" for how much we believe the ith word is associated with either speaker. Large positive values indicate that the ith word is more likely to be said by speaker A, and large negative values indicate it is more likely to be said by speaker B, while near-zero values indicatres that this is in some sense a "neutral" word.

Now, we may try to learn w using the logistic loss:

$$\ell(w, (x, y)) = \log(1 + \exp(-y\langle w, x \rangle))$$

This loss is 2-smooth, and the gradient is:

$$\nabla \ell(w, (x, y)) = \frac{-yx \exp(-y\langle w, x \rangle)}{1 + \exp(-y\langle w, x \rangle)}$$

From this, if the feature vectors x satisfy $\|x\| \leq r$, we might expect the variance to be bounded by r^2, so we should look for a learning rate of the form

$$\eta_t \propto \frac{r}{\sqrt{t}}$$

However, we might be able to do better. Specifically, language has a long tail of rare words. That is, there are so many different words that any given paragraph is likely to have a few words that do not appear again for a very long time. As a result, in our data many of the coordinates of x will only rarely be non-zero. However, some of these coordinates may be very discriminative. For example, a biologist might use the word "polymerase" occasionally, while a physicist uses the word "quark" occasionally, and it is extremely rare for either person to use the other person's word.

Thus, when we see an example x that encodes a paragraph using the word "polymerase", we would ideally take a large step of gradient descent in order to quickly learn the fact that "polymerase" is a biologist-indicative word.

We will accomplish this via *per-coordinate learning rates*. That is, instead of the standard gradient descent update:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla \ell(\mathbf{w}_t, z_t)$$

we will instead update each coordinate using it's own specific learning rate:

$$\mathbf{w}_{t+1}[i] = \mathbf{w}_t[i] - \eta_t[i] \nabla \ell(\mathbf{w}_t, z_t)[i]$$

where we use $v[i]$ to indicate the ith coordinate of a vector v. Thus, η_t is no longer a scalar but in fact a *vector* of learning rates. Then, we will try to set η_t in such a way that $\eta_t[i]$ is larger for coordinates for which the ith word is rarer. This way we can quickly adapt to the uncommon words like polymerase by taking a large step when they appear.

However, there is still a problem: how can we know ahead of time which coordinates are going to be rare? The solution is to just count them. We could set:

$$\eta_t[i] \propto \frac{1}{\sqrt{n_t[i]}}$$

where $n_t[i]$ is the number of times the ith word has appeared after t iterations.

However, we will be even fancier. Taking inspiration from our adaptive methods, we set:

$$\begin{aligned} \mathbf{g}_t &= \nabla \ell(\mathbf{w}_t, z_t) \\ \eta_t[i] &= \frac{c}{\sqrt{\epsilon^2 + \sum_{\tau=1}^t \mathbf{g}_\tau[i]^2}} \end{aligned}$$

Note that

$$\mathbf{g}_t[i] = \left[\frac{-y_t \exp(-y_t \langle \mathbf{w}_t, \mathbf{x}_t \rangle)}{1 + \exp(-y \langle \mathbf{w}_t, \mathbf{x}_t \rangle)}\right] \mathbf{x}_t[i]$$

so that roughly speaking, $\sum_{\tau=1}^t \mathbf{g}_\tau[i]^2$ is a count of the number of times $\mathbf{x}_t[i]$ is non-zero. However, it is a little better than that - this sum is really more like an estimate of the *per-coordinate variance*:

$$\sigma^2[i] = \mathbb{E}[(\nabla \ell(\mathbf{w}_t, z_t)[i] - \nabla \mathcal{L}(\mathbf{w}_t)[i])^2$$

In much the same way that our previous adaptive learning rate $\eta_t \propto \frac{1}{\sqrt{\sum_{\tau=1}^t \|\mathbf{g}_t\|^2}}$ is an estimate of $\frac{1}{\sigma\sqrt{t}}$, this new learning rate is approximately:

$$\eta_t[i] \approx \frac{c}{\sigma[i]\sqrt{t}}$$

That is, the vector of learning rates is measuring how much we should "trust" each individual coordinate of the gradient. This is the famous AdaGrad algorithm [1] (which was essentially simultaneously described by [2]). As this algorithm was first proposed in the convex setting, we will analyze it there first.

Also, in order to make the algebra look a little cleaner, we will introduce the following notation:

Definition 1. *Given vectors $x_1, \dots, x_t$, we use the notation $x_{a:b}$ to indicate the sum $\sum_{t=a}^b x_t$. Similarly, we use $x_{a:b}^2[i]$ to indicate the sum $\sum_{t=a}^b x_t^2[i]$ and $\|x\|_{a:b}^2$ for the sum $\sum_{t=a}^b \|x_t\|^2$.*

Using this notation, our learning rates can be re-written as:

$$\eta_t[i] = \frac{c}{\sqrt{\epsilon^2 + g_{1:t}^2[i]}}$$

To start with a good baseline, remember that while doing the previous homework, you proved the following result:

Theorem 2 (Adaptive Gradient Descent)**.** *Suppose that $\mathcal{L}$ is a convex function satisfying $\|\mathbf{w}_\star\|_2 \leq D_2$. Then with $c = \sqrt{2}D_2$, the algorithm:*

$$\begin{aligned} \mathbf{g}_t &= \nabla \ell(\mathbf{w}_t, z_t) \\ \mathbf{w}_{t+1} &= \Pi_{\|\mathbf{w}\|_2 \leq D_2} \mathbf{w}_t - \frac{c\mathbf{g}_t}{\sqrt{\|\mathbf{g}\|_{1:t}^2}} \end{aligned}$$

satisfies:

$$\mathbb{E}\left[\sum_{t=1}^T \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)\right] \leq 2\sqrt{2}D_2 \, \mathbb{E}\left[\sqrt{\sum_{t=1}^T \|\mathbf{g}_t\|^2}\right]$$

Note that in this theorem, we have emphasized the fact that we are measuring the norm of $\mathbf{w}_\star$ using the standard Euclidean 2-norm $\|\mathbf{w}_\star\|_2 = \sqrt{\sum_{i=1}^d \mathbf{w}_\star[i]^2}$.

In contrast, we will be able to prove the following result about AdaGrad:

Theorem 3 (AdaGrad Convergence)**.** *Suppose that $\mathcal{L}$ is a convex function satisfying $\|\mathbf{w}_\star\|_\infty \leq D_\infty$. Then with $c = \sqrt{2}D_\infty$, the algorithm:*

$$\mathbf{g}_t = \nabla \ell(\mathbf{w}_t, z_t)$$
$$\mathbf{w}_{t+1}[i] = \Pi_{|\mathbf{w}[i]| \leq D_\infty} \mathbf{w}_t[i] - \frac{c\mathbf{g}_t[i]}{\sqrt{\mathbf{g}_{1:t}^2[i]}}$$

satisfies:

$$\mathbb{E}\left[\sum_{t=1}^T \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)\right] \leq 2\sqrt{2}D_\infty \, \mathbb{E}\left[\sum_{i=1}^d \sqrt{\sum_{t=1}^T \mathbf{g}_t[i]^2}\right]$$

This algorithm is called AdaGrad *(or sometimes* diagonal AdaGrad*).*

Before proving this theorem, let's take a minute to discuss some of the implications.

Note the differences here: now we are measuring $\mathbf{w}_\star$ using the infinity norm rather than the 2-norm: $\|\mathbf{w}_\star\|_\infty = \max_i |\mathbf{w}_\star[i]|$. The infinity norm is always smaller than the 2-norm, so at first this seems like a purely good thing. However, we may pay for this improvement by having

$$\sum_{i=1}^d \sqrt{\sum_{t=1}^T \mathbf{g}_t[i]^2} \tag{1}$$

in the bound rather than

$$\sqrt{\sum_{t=1}^T \|\mathbf{g}_t\|^2} \tag{2}$$

It is possible for (1) to be bigger than (2). But how much bigger? Let use Cauchy-Schwarz:

$$\begin{aligned}\sum_{i=1}^d \sqrt{\sum_{t=1}^T \mathbf{g}_t[i]^2} &\leq \sqrt{d}\sqrt{\sum_{i=1}^d \sum_{t=1}^T \mathbf{g}_t[i]^2} \\ &= \sqrt{d}\sqrt{\sum_{t=1}^T \sum_{i=1}^d \mathbf{g}_t[i]^2} \\ &= \sqrt{d\sum_{t=1}^T \|\mathbf{g}_t\|^2}\end{aligned}$$

So that (1) is at most $\sqrt{d}$ times bigger than (2).

On the other hand, consider the case that $\mathbf{w}_\star = (\pm w, \pm w, \ldots, \pm w)$ for some constant w. In this case we have:

$$\|\mathbf{w}_\star\|_2 = w\sqrt{d} = \|\mathbf{w}_\star\|_\infty \sqrt{d}$$

Thus, when $\mathbf{w}_\star = (\pm w, \pm w, \ldots, \pm w)$ we would have (assuming the c constants are set to their best values):

$$D_\infty \sum_{i=1}^d \sqrt{\sum_{t=1}^T \mathbf{g}_t[i]^2} \leq D_2 \sqrt{\sum_{t=1}^T \|\mathbf{g}_t\|^2}$$

Of course, this is a strong restriction on $\mathbf{w}_\star$, but it does still allow for 2^d possible different values so some non-trivial optimization must happen to learn $\mathbf{w}_\star$.

Finally, let us consider a setting in which some of the coordinate $\mathbf{g}_t[i]$ are very rare, similar to our original motivating example. This setting is adapted from [1]. Suppose that $\mathbf{g}_t$ is always a 1-hot vector (i.e. has one non-zero coordinate) such that the ith coordinate $\mathbf{g}_t[i]$ is ± 1 with probability proportional to $1/i^2$ and 0 otherwise. Then we have:

$$\begin{aligned}\mathbb{E}[D_\infty \sum_{i=1}^d \sqrt{\mathbf{g}_{1:t}^2[i]}] &\leq D_\infty \sum_{i=1}^d \sqrt{T/i^2} \\ &= D_\infty \sqrt{T} \sum_{i=1}^d \frac{1}{i} = O(D_\infty \log(d) \sqrt{T})\end{aligned}$$

In contrast:

$$\mathbb{E}[D_2 \sqrt{\|\mathbf{g}\|_{1:T}^2}] = D_2 \sqrt{T}$$

Thus, so long as D_∞ is significantly less than D_2, the bound will be significantly better using AdaGrad.

In general, it is not obvious apriori which bound is better: the algorithm you use should depend on some empirical observations/specific knowledge about the data.

Now, let's try proving Theorem 3.

Proof of Theorem 3. Remember that our previous proofs for gradient descent on convex functions made a lot of use of the inequality:

$$\mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)] \leq \mathbb{E}[\langle \nabla \mathcal{L}(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}_\star \rangle] = \mathbb{E}[\langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{w}_\star \rangle]$$

In fact, as we say in our analysis of acceleration (Notes 8, Theorem 5), this was the only part of the analysis that actually used the fact that $\mathbf{g}_t$ was at all related to $\mathcal{L}$: the rest of the analysis was purely algebraic manipulation and held for any $\mathbf{g}_t$ and and $\mathbf{w}_\star$. The key idea behind the analysis of AdaGrad is to break the analysis of $\langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{w}_\star \rangle$ using the following simple but surprisingly useful observation:

$$\langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{w}_\star \rangle = \sum_{i=1}^d g_t[i](\mathbf{w}_t[i] - \mathbf{w}_\star[i])$$

This means that we can write:

$$\begin{aligned}\mathbb{E}\left[\sum_{t=1}^T \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)\right] &\leq \sum_{t=1}^T \mathbb{E}[\langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{w}_\star \rangle] \\ &= \sum_{t=1}^T \sum_{i=1}^d \mathbb{E}[\mathbf{g}_t[i](\mathbf{w}_t[i] - \mathbf{w}_\star[i])] \\ &= \sum_{i=1}^d \mathbb{E}\left[\sum_{t=1}^T \mathbf{g}_t[i](\mathbf{w}_t[i] - \mathbf{w}_\star[i])\right]\end{aligned}$$

So now, it suffices to just bound each sum $\sum_{t=1}^T \mathbf{g}_t[i](\mathbf{w}_t[i] - \mathbf{w}_\star[i])$ *individually*. These are just d 1-d versions of the same adaptive analysis you did on the homework. We'll do it again here for completeness. First, using the fact that $|\mathbf{w}_\star[i]| \leq D_\infty$:

$$\begin{aligned}(\mathbf{w}_{t+1}[i] - \mathbf{w}_\star[i])^2 &\leq (\mathbf{w}_t[i] - \eta_t[i]\mathbf{g}_t[i] - \mathbf{w}_\star[i])^2 \\ &= (\mathbf{w}_t[i] - \mathbf{w}_\star[i])^2 - 2\eta_t[i]\mathbf{g}_t[i](\mathbf{w}_t[i] - \mathbf{w}_\star[i]) + \eta_t[i]^2 \mathbf{g}_t[i]^2 \\ \mathbf{g}_t[i](\mathbf{w}_t[i] - \mathbf{w}_\star[i]) &\leq \frac{(\mathbf{w}_t[i] - \mathbf{w}_\star[i])^2 - (\mathbf{w}_{t+1}[i] - \mathbf{w}_\star[i])^2}{2\eta_t[i]} + \frac{\eta_t[i]\mathbf{g}_t[i]^2}{2}\end{aligned}$$

summing over t and reordering:

$$\begin{aligned}\sum_{t=1}^{T} \mathbf{g}_t[i](\mathbf{w}_t[i] - \mathbf{w}_\star[i]) &\leq \frac{(\mathbf{w}_1[i] - \mathbf{w}_\star[i])^2}{2\eta_1[i]} - \frac{(\mathbf{w}_{T+1}[i] - \mathbf{w}_\star[i])^2}{2\eta_T[i]} \\ &\quad + \sum_{t=2}^{T} (\mathbf{w}_t[i] - \mathbf{w}_\star[i])^2 \left(\frac{1}{2\eta_t[i]} - \frac{1}{2\eta_{t-1}[i]}\right) + \sum_{t=1}^{T} \frac{\eta_t[i]\mathbf{g}_t[i]^2}{2} \\ &\leq \frac{D_\infty^2}{2\eta_1[i]} + \sum_{t=2}^{T} D_\infty^2 \left(\frac{1}{2\eta_t[i]} - \frac{1}{2\eta_{t-1}[i]}\right) + \sum_{t=1}^{T} \frac{\eta_t[i]\mathbf{g}_t[i]^2}{2} \\ &\leq \frac{D_\infty^2}{2\eta_T[i]} + \sum_{t=1}^{T} \frac{\eta_t[i]\mathbf{g}_t[i]^2}{2}\end{aligned}$$

Now, we observe that from our sum-to-integration lemma (Notes 6 Lemma 2), we have

$$\sum_{t=1}^{T} \eta_t[i]\mathbf{g}_t[i]^2 = \sum_{t=1}^{T} \frac{\mathbf{g}_t[i]^2}{\sqrt{\sum_{\tau=1}^{t} \mathbf{g}_\tau[i]^2}} \leq \int_0^{\sum_{t=1}^{T} \mathbf{g}_t[i]^2} \frac{dx}{\sqrt{x}} = \sqrt{2\sum_{t=1}^{T} \mathbf{g}_t[i]^2}$$

Thus, plugging in the definition of c and η_t now shows:

$$\sum_{t=1}^{T} \mathbf{g}_t[i](\mathbf{w}_t[i] - \mathbf{w}_\star[i]) \leq 2\sqrt{2} D_\infty \sqrt{\sum_{t=1}^{T} \mathbf{g}_t[i]^2}$$

now using the fact that

$$\begin{aligned}\mathbb{E}\left[\sum_{t=1}^{T} \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)\right] &\leq \sum_{i=1}^{d} \mathbb{E}\left[\sum_{t=1}^{T} \mathbf{g}_t[i](\mathbf{w}_t[i] - \mathbf{w}_\star[i])\right] \\ &\leq 2\sqrt{2} D_\infty \sum_{i=1}^{d} \sqrt{\sum_{t=1}^{T} \mathbf{g}_t[i]^2}\end{aligned}$$

as desired. □

2 Adam

At this point, we have seen 3 (maybe 4) distinct components of stochastic optimization algorithms based on gradient descent:

1. **Adaptive learning rates**: these are lsearning rate schedules that depend on the observed statistics of the gradients. These make the optimizer robust to various unknown parameters, like the variance and sometimes the smoothness constant.

1.5. Related to adaptive learning rates, we had the notion of *exponentially weighted moving averages* for collecting the statistics of the gradients. Intuitively, this modifies the adaptive learning rates to "track" the relevant statistics of the losses over time.

2. **Momentum:** we saw two forms of momentum, both of which can be motivated as integrating some physical equations of motion. The first form of momentum simply does an Euler integration, while the second evaluates the gradient at an kind of "auxiliary point" in order to improve the integration. We saw that for deterministic smooth convex losses, appropriate settings for momentum result in *accelerated* convergence rates.

Algorithm 1 Adam

Input: Initial Point $\mathbf{w}_1$, learning rates η_t, momentum parameters β_1, β_2 (defaults typically 0.9 and 0.999).
Set $\mathbf{v}_0 = 0$.
Set $A_0[i] = \epsilon^2$
for $t = 1 \dots T$ **do**
  Set $\mathbf{g}_t = \nabla \ell(\mathbf{w}_t, z_t)$
  Set $\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} - (1-\beta_1)\mathbf{g}_t$.
  Set $\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1-\beta_1^t}$.
  Set $\mathbf{v}_t[i] = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\mathbf{g}_t[i]^2$.
  Set $\hat{\mathbf{v}}_t[i] = \frac{\mathbf{v}_t[i]}{1-\beta_2^t}$
  Set $\mathbf{w}_{t+1}[i] = \mathbf{w}_t[i] - \eta_t \frac{\hat{\mathbf{m}}_t[i]}{\sqrt{\hat{\mathbf{v}}_t[i]}}$.
end for

3. **Per-coordinate updates:** we have just seen that by essentially running a independent parallel copy of an adaptive algorithm on each coordinate, we can adapt to varying scales of the different coordinates and sometimes achieve improved guarantees.

The *Adam* algorithm [3] simple combines all of these ideas together into one big pot:

Note that in this algorithm we use $\mathbf{m}$ to refer to the momentum and $\mathbf{v}$ to refer to the denominator. Don't be confused by this! Previously, we used $\mathbf{v}$ (for velocity) to refer to the momentum. We are presenting Adam in this way now to be consistent with the literature for this algorithm. the v in $\mathbf{v}$ is supposed to stand for "variance".

In addition to incorporating the above ideas, Adam adds one more wrinkle: dividing by $1-\beta^t$. This is meant to *de-bias* some of the estimates. To see why, let us expand out the expression for $\mathbf{v}_t[i]$. Since we focus on just one coordinate, we'll drop the $[i]$'s for simpler equations:

$$\begin{aligned}\mathbf{v}_t &= \beta_2 \mathbf{v}_{t-1} + (1-\beta_2) g_t^2 \\ &= \beta_2^2 \mathbf{v}_{t-2} + \beta_2(1-\beta_2) g_{t-1}^2 + (1-\beta_2) g_t^2\end{aligned}$$

expanding for t steps:

$$= \beta_2^t \mathbf{v}_0 + \beta_2^{t-1}(1-\beta_2) g_1^2 + \cdots + \beta_2(1-\beta_2) g_{t-1}^2 + (1-\beta_2) g_t^2 \qquad = \beta_2^t \epsilon^2 + (1-\beta_2) \sum_{\tau=0}^{t-1} \beta_2^\tau g_{t-\tau}^2$$

Now, let us assume $\beta_2^t \epsilon^2 \approx= 0$. This is reasonable: β_2^t already decreases exponentially fast, and ϵ^2 was already very small to begin with. Further, let us assume that each g_t satisfies $\mathbb{E}[g_t^2] = g^2$ for some *fixed* g^2. This is not true - is only somehow approximately true in the sense that the we have $\mathbb{E}[g_t^2] = \nabla \mathcal{L}(\mathbf{w}_t)^2 + \sigma_t^2$, and we might expect both $\nabla \mathcal{L}(\mathbf{w}_t)$ and σ_t to be a slowly varying function of t. However, under this assumption:

$$\begin{aligned}\mathbb{E}[\mathbf{v}_t] &= (1-\beta_2) \sum_{\tau=0}^{t-1} \beta_2^\tau g^2 \\ &= (1-\beta_2) \frac{1-\beta_2^t}{1-\beta_2} g^2 \\ &= (1-\beta_2^t) g^2\end{aligned}$$

Therefore, by dividing by $1-\beta_2^t$, we obtain an "unbiased" estimate of g. Of course, for any reasonably large t, $1-\beta_2^t$ is nearly 1, so it is unclear how and whether this is actually important in theory (or even in practice for that matter, as you will see on the homework).

In the case of the momentum parameter m_t, dividing by $1-\beta_1^t$ has a similar de-biasing effect. This also motivates a second way to think about momentum beyond the physical intuition we discussed earlier.

2.1 Momentum as Averaging

The momentum parameter in Adam can be interpreted as artificially increasing the batch size, To see this, let's follow the same "unraveling" of the recursive formula for m_t as we did for $\mathbf{v}_t$:

$$\begin{aligned} m_t &= \beta_1^t \epsilon^2 + (1-\beta_1)\sum_{\tau=0}^{t-1} \beta_1^\tau g_{t-\tau} \\ &= (1-\beta_1)\sum_{\tau=0}^{t-1} \beta_1^\tau g_{t-\tau} \end{aligned}$$

Now, writing $\frac{1-\beta_1^t}{1-\beta_1} = 1 + \beta_1 + \beta_1^2 + \ldots \beta_t^{t-1}$, we have:

$$\hat{m}_t = \frac{\sum_{\tau=0}^{t-1} \beta_1^\tau g_{t-\tau}}{\sum_{\tau=0}^{t-1} \beta_1^\tau}$$

That is, just like $\mathbf{v}_t$, m_t is a weighted average of the gradient values g_t. Let us write:

$$g_t = \nabla \mathcal{L}(\mathbf{w}_t) + r_t$$

where r_t is some mean-zero random value. Further, let's assume that w_t is varying slowly enough that we can approximate $\mathcal{L}(\mathbf{w}_{t'}) = \mathcal{L}(\mathbf{w}_t)$ for $t \leq t'$. Then we have:

$$\begin{aligned} \hat{m}_t &= \nabla \mathcal{L}(\mathbf{w}_t) + \frac{\sum_{\tau=0}^{t-1} \beta_1^\tau r_{t-\tau}}{\sum_{\tau=0}^{t-1} \beta_1^\tau} \\ \mathbb{E}[\hat{m}_t] &= \nabla \mathcal{L}(\mathbf{w}_t) \end{aligned}$$

What about the variance of $\hat{m}_t$? Supposing each g_t has variance at most σ^2 we have $\mathbb{E}[r_t^2] \leq \sigma^2$ and so:

$$\begin{aligned} \text{Var}(\hat{m}_t) &= \frac{\sum_{\tau=0}^{t-1} \beta_1^{2\tau}\, \mathbb{E}[r_{t-\tau}^2]}{(\sum_{\tau=0}^{t-1} \beta_1^\tau)^2} \\ &= \sigma^2 \frac{(1-\beta_1)(1-\beta_1^{2t-1})}{(1-\beta_1^2)(1-\beta_1^t)^2} \end{aligned}$$

Now, before analyzing this, let's try to gain some quick back-of-the-envelope intuition for what will happen. For large t, we can hopefully ignore the β_1^t and β_t^{2t-1} terms, so that we are left with:

$$\sigma^2 \frac{(1-\beta_1)^2}{1-\beta_1^2} = \sigma^2 \frac{1-\beta_1}{1+\beta_1}$$

Now, by first-order taylor expansion we might expect that this is approximately:

$$\sigma^2(1-\beta_1)$$

So that if $\beta_1 = \frac{N-1}{N}$, we would have variance of about $\frac{\sigma^2}{N}$, which corresponds to what would happen we we had a batch-size of N. Note that this should seem intuitively reasonable from the initial formula:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ &= \frac{N-1}{N} m_{t-1} + \frac{1}{N} g_t \end{aligned}$$

If m_{t-1} happened to be the average of $N-1$ elements, then this formula would set m_t to be the average of those $N-1$ elements and g_t, a total of N elements.

Let's try to see this without the taylor approximations. We need to control $\frac{(1-\beta_1)^2(1-\beta_1^{2t-1})}{(1-\beta_1^2)(1-\beta_1^t)^2}$. We'll start by looking at just one part, $\frac{(1-\beta_1)^2}{1-\beta_1^2}$. Let's write $\beta_1 = 1-\alpha$. Then we have:

$$\frac{(1-\beta_1)^2}{1-\beta_1^2} = \frac{\alpha^2}{2\alpha-\alpha^2} = \frac{\alpha}{2-\alpha} \leq \alpha$$

Further, so long as $t \geq \log(1/2)/\log(\beta_1)$, we will have $\beta_1^t \leq 1/2$ so that:

$$\frac{(1-\beta_1)^2(1-\beta_1^{2t-1})}{(1-\beta_1^2)(1-\beta_1^t)^2} \leq 4\alpha = 4(1-\beta)$$

which (up to the factor 4), is the same result we had with the taylor series.

On the surface, this seems really good - we decreased the variance by a large amount! Unfortunately, this decrease in variance is actually accompanied by a corresponding increase in *bias*. The issue is with our assumption that $\mathbf{w}_t$ is constant. In order to deal with the variation of $\mathbf{w}_t$ properly, it turns out that m_t becomes a *biased* estimate of the gradient. This extra bias will almost exactly cancel out the decreased variance so that in the end it is hard to show that we gain anything. Indeed, *no current analysis of Adam or similar algorithms shows any advantage over adagrad in smooth losses*. Nevertheless, in practice these algorithms perform much better, so clearly this is a interesting hole in our understanding of the properties of the loss functions and how the algorithms work.

2.2 Convergence failure of Adam

It turns out that the Adam algorithm actually can fail to converge. To get some idea of what could go wrong, notice that the definition of $\mathbf{v}_t$ is essentially an average of previous gradients. Therefore, if there is some constant variance so that previous gradients always have $|g_t| = 1$ for example, regardless of the true value of $\nabla\mathcal{L}(\mathbf{w}_t)$, then $\hat{\mathbf{v}}_t$ will always be 1. Thus, we can see that it must be important for η_t to decrease or be very small in order to guarantee convergence (unlikely with adagrad). However, even when η_t decreases like $1/\sqrt{t}$, there can still be a problem, as observed by [4], in which an example is provided for which Adam makes consistent progress *in the wrong direction*.

To see why this might be, consider an extreme case in which $\beta_2 = \beta_1 = 0$. This corresponds to the update:

$$\mathbf{w}_{t+1}[i] = \mathbf{w}_t[i] - \eta_t \frac{\mathbf{g}_t[i]}{|\mathbf{g}_t[i]|}$$

That is, we update update using the *sign* of the gradient rather than the actual gradient. In a previous homework problem, you already provided a distribution for $\mathbf{g}_t$ such that the sign of $\mathbf{g}_t$, on average, is different than the sign of $\mathbb{E}[\mathbf{g}_t]$, so that the above update will actually make negative progress in expectation.

On the other hand, it might be that with more realistic values for β_2 and β_1 (which are usually very close to 1), it is possible to show some convergence guarantee. So far as I know, this is still open.

References

[1] J. Duchi, E. Hazan, and Y. Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Conference on Learning Theory (COLT)*. 2010.

[2] H. Brendan McMahan and Matthew Streeter. "Adaptive Bound Optimization for Online Convex Optimization". In: *Proceedings of the 23rd Annual Conference on Learning Theory (COLT)*. 2010.

[3] Diederik Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[4] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. "On the Convergence of Adam and Beyond". In: *6th International Conference on Learning Representations, ICLR 2018*. 2018.