Problem 3: Adam vs. SGD with Momentum

Setup

Adam (scalar parameter w) at step t:

$$\begin{aligned} g_t &= \nabla \ell_t(w_t), \\ m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t, \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \\ \hat{m}_t &= \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}, \\ w_{t+1} &= w_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\varepsilon}. \end{aligned}$$

Momentum SGD:

$$u_t = \mu u_{t-1} + g_t, \qquad w_{t+1} = w_t - \alpha u_t.$$

Assume $m_0 = 0$, $v_0 = 0$.

Part 1 (3.1): First two Adam updates (6 pts)

We assume $m_0 = 0$, $v_0 = 0$, and keep everything in g_1, g_2, α, β_1, β_2, ε.

First update (after we see g_1)

By definition, the first moment is $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$. For $t = 1$ with $m_0 = 0$ we get $m_1 = (1-\beta_1) g_1$. The second moment is $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$; for $t = 1$ with $v_0 = 0$ we get $v_1 = (1-\beta_2) g_1^2$.

$$\boxed{m_1 = (1-\beta_1) g_1}, \qquad \boxed{v_1 = (1-\beta_2) g_1^2}.$$

Because we started at zero, the raw m_1 and v_1 are biased downward. The algorithm therefore divides them by $1-\beta_1^t$ and $1-\beta_2^t$ respectively. For $t = 1$ that is $1-\beta_1$ and $1-\beta_2$, which cancel the same factors in m_1 and v_1, so the bias-corrected moments are just the gradient and its square:

$$\boxed{\hat{m}_1 = g_1}, \qquad \boxed{\hat{v}_1 = g_1^2}.$$

The update rule $w_{t+1} = w_t - \alpha \hat{m}_t / (\sqrt{\hat{v}_t} + \varepsilon)$ then gives, after substituting $\hat{m}_1$ and $\hat{v}_1$:

$$\boxed{w_2 = w_1 - \alpha \frac{g_1}{\sqrt{g_1^2}+\varepsilon}.}$$

Second update (after we see g_2)

We now have m_1 and v_1. The recurrence for the first moment blends the previous m_1 with the new gradient g_2: $m_2 = \beta_1 m_1 + (1-\beta_1)g_2$. Substituting $m_1 = (1-\beta_1)g_1$ and factoring out $(1-\beta_1)$ gives a weighted combination of g_1 and g_2. The same logic for the second moment uses $v_2 = \beta_2 v_1 + (1-\beta_2)g_2^2$ and $v_1 = (1-\beta_2)g_1^2$:

$$\boxed{m_2 = (1-\beta_1)(\beta_1 g_1 + g_2)}, \qquad \boxed{v_2 = (1-\beta_2)(\beta_2 g_1^2 + g_2^2)}.$$

Again we bias-correct by dividing by $1-\beta_1^t$ and $1-\beta_2^t$; for $t=2$ that is $1-\beta_1^2$ and $1-\beta_2^2$. No cancellation this time, so we keep the fractions:

$$\boxed{\hat{m}_2 = \frac{(1-\beta_1)(\beta_1 g_1 + g_2)}{1-\beta_1^2}}, \qquad \boxed{\hat{v}_2 = \frac{(1-\beta_2)(\beta_2 g_1^2 + g_2^2)}{1-\beta_2^2}}.$$

Applying the same update rule with these corrected moments yields:

$$\boxed{w_3 = w_2 - \alpha \frac{\hat{m}_2}{\sqrt{\hat{v}_2} + \varepsilon}.}$$

Part 2: Compare to momentum SGD (5 pts)

1. **Effective step size and gradient magnitude:**

   We want to see how the *magnitude* of the parameter update depends on the scale of the gradients. So we look at the update formulas and ask: if gradients were uniformly larger or smaller, how would the step size change?

   In momentum SGD the update is

   $$\Delta w_t = -\alpha u_t, \qquad u_t = \mu u_{t-1} + g_t.$$

   So the step magnitude is

   $$|\Delta w_t| = \alpha |u_t|.$$

   Since u_t is an EMA of the g_τ, we have $u_t \propto$ (scale of gradients) (modulo the decay μ). So doubling all gradient magnitudes roughly doubles $|u_t|$ and hence doubles the step: **effective step size scales directly with gradient magnitude**; bigger gradients $\Rightarrow$ bigger steps.

   In Adam the update is

   $$\Delta w_t = -\alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}.$$

   Here $\hat{m}_t$ is a bias-corrected first moment (EMA of g_t) and $\hat{v}_t$ is a bias-corrected second moment (EMA of g_t^2), so $\sqrt{\hat{v}_t}$ approximates the *root-mean-square* of recent gradients. Thus the step has the form $\alpha \cdot (\text{direction from } \hat{m}_t)/(\text{RMS scale} + \varepsilon)$: the numerator and the scale in the denominator both grow with gradient magnitude, so they partially cancel. The *effective step magnitude* is

   $$\alpha \frac{|\hat{m}_t|}{\sqrt{\hat{v}_t} + \varepsilon}.$$

   If we scale all g_t by c, then $\hat{m}_t \propto c$ and $\hat{v}_t \propto c^2$, so $|\hat{m}_t|/\sqrt{\hat{v}_t}$ is scale-invariant (see (b)); hence the step size is **much less sensitive** to raw gradient magnitude. Moreover each parameter has its own $\hat{v}_t$, so adaptation is **per-parameter**: a coordinate with consistently large $|g|$ gets a larger $\hat{v}$ and thus a smaller effective step; one with small or rare gradients gets a smaller $\hat{v}$ and a relatively larger step. So Adam reduces dependence on gradient magnitude via **RMS normalization**.

2. **Scaling gradients by a constant c:**

   We ask: if every gradient g_t is replaced by cg_t, how do the two algorithms' updates change? This tells us whether the optimizer is sensitive to the global scale of the loss/gradients (e.g. loss scaling or batch size).

   In momentum SGD, $u_t = \mu u_{t-1} + g_t$ is linear in g_t, so under $g_t \mapsto cg_t$ we get

   $$u_t \mapsto cu_t.$$

   The step is $\Delta w_t = -\alpha u_t$, so the step scales by c: **momentum SGD is not scale-invariant**; changing gradient scale changes step size proportionally.

   In Adam, $\hat{m}_t$ is linear in the g_τ (bias-corrected EMA of g_t), so $\hat{m}_t \mapsto c\hat{m}_t$. The second moment v_t is an EMA of g_t^2, so $g_t^2 \mapsto c^2 g_t^2$ implies $\hat{v}_t \mapsto c^2\hat{v}_t$, hence

   $$\sqrt{\hat{v}_t} \mapsto |c|\sqrt{\hat{v}_t}.$$

   The ratio in the update is $\hat{m}_t/(\sqrt{\hat{v}_t} + \varepsilon)$. So under scaling: numerator $\mapsto c\hat{m}_t$, denominator $\mapsto |c|\sqrt{\hat{v}_t} + \varepsilon$. For $c > 0$,

   $$\frac{\hat{m}_t}{\sqrt{\hat{v}_t}} \mapsto \frac{c\hat{m}_t}{c\sqrt{\hat{v}_t}} = \frac{\hat{m}_t}{\sqrt{\hat{v}_t}} :$$

   magnitude of the ratio is invariant. For $c < 0$, the direction of $\hat{m}_t$ flips (we move the opposite way), which is correct. So when $\sqrt{\hat{v}_t}$ dominates the denominator, Adam is **approximately scale-invariant**. The caveat: when $\varepsilon > 0$ and $\hat{v}_t$ is small (e.g. early steps or rarely updated parameters), $\sqrt{\hat{v}_t} + \varepsilon \approx \varepsilon$, so the denominator does not scale with c; then the step $\propto c\hat{m}_t$ and scale-invariance is **slightly broken**.

3. **Role of ε:**

   The update divides by $\sqrt{\hat{v}_t}$; we need to avoid division by zero and uncontrolled updates when $\hat{v}_t$ is very small. So we ask what happens when $\hat{v}_t \to 0$ and how ε fixes it.

   The Adam step is

   $$\Delta w_t = -\alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}.$$

   Without ε, when $\hat{v}_t = 0$ (e.g. $t = 1$ with a single zero gradient, or a parameter that has never received a non-zero gradient) we would divide by zero. More generally, when $\hat{v}_t \ll 1$, $\sqrt{\hat{v}_t}$ is tiny and $1/\sqrt{\hat{v}_t}$ can be huge, so a single large $\hat{m}_t$ would produce an **exploding update**. Adding ε in the denominator gives

   $$\sqrt{\hat{v}_t} + \varepsilon \geq \varepsilon > 0,$$

   so the denominator is **lower-bounded**: the step magnitude is at most

   $$\frac{\alpha|\hat{m}_t|}{\varepsilon},$$

   which is bounded for bounded $\hat{m}_t$. So ε (1) prevents division by zero, (2) lower-bounds the denominator and prevents exploding updates when $\hat{v}_t$ is tiny (early in training or for rarely updated coordinates), and (3) ensures numerical stability. In practice ε is small (e.g. 10^{-8}) so that when $\hat{v}_t$ is already large, the denominator is hardly affected.

Part 3: Noisy gradients and sparse features (4 pts)

(a) **Learning-rate adaptation across parameters:**

We ask how each algorithm sets the *effective* step size per parameter. In high dimensions, different coordinates can have very different gradient scales; an optimizer that adapts per coordinate may behave better.

In Adam, the effective step for a parameter is

$$\Delta w_t = -\alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}.$$

Each parameter has its *own* m_t, v_t (and thus $\hat{m}_t$, $\hat{v}_t$) from its own gradient history. So $\sqrt{\hat{v}_t}$ is the RMS of that coordinate's recent gradients. A coordinate with consistently large $|g_t|$ has large $\hat{v}_t$, so the denominator is large and the **effective step is smaller**; one with small or rarely non-zero gradients has small $\hat{v}_t$, so the denominator is smaller and the **effective step is relatively larger**. So Adam adapts the learning rate **per parameter** using $\hat{v}_t$. In momentum SGD, the update is $-\alpha u_t$ with a single global α; u_t smooths gradients via

$$u_t = \mu u_{t-1} + g_t,$$

but there is no per-parameter rescaling by a second-moment term. So momentum SGD uses one global α and does **not** adapt step size per parameter based on gradient scale.

(b) **Noisy gradients and sparse features:**

With noisy gradients, we want to avoid overreacting to a few large spurious values. With sparse features, some parameters are updated rarely; we want them to still get meaningful updates when they do get a gradient. We compare how Adam's structure (especially $\hat{v}_t$) addresses these versus momentum.

Noisy gradients: In Adam, the step is normalized by $\sqrt{\hat{v}_t}$, which is an EMA of g_t^2:

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}.$$

So $\hat{v}_t$ reflects the *typical squared magnitude* of recent gradients. A single sporadic large gradient increases $\hat{v}_t$ only slightly (EMA smooths it), and the denominator $\sqrt{\hat{v}_t} + \varepsilon$ is already of the order of the typical scale; so the **RMS normalization** down-weights the impact of that large gradient. In other words, directions with high variance (noise) get a larger denominator and smaller effective steps, reducing the effect of spurious large gradients and stabilizing updates. In momentum SGD, u_t is an EMA of g_t, so one large g_t can still move u_t noticeably; there is no second-moment normalization, so noisy spikes can cause larger, less stable steps.

Sparse features: For a coordinate that is rarely non-zero (e.g. a sparse feature), most g_t are zero. In Adam, v_t (and $\hat{v}_t$) for that coordinate stays **small** because most $g_t^2 = 0$. When a non-zero gradient finally appears, the denominator $\sqrt{\hat{v}_t} + \varepsilon$ is still small, so the **effective step is relatively large**: that parameter gets a meaningful update and can "catch up." In momentum SGD, the same coordinate has small u_t (mostly zeros), so when a gradient appears the step is α times that gradient with no automatic boost from a small denominator; sparse features may therefore learn more slowly. Combined with noise, momentum can be less stable because there is no per-parameter scaling by $\sqrt{\hat{v}_t}$.