Momentum, RMSProp and Adam: optimizer updates by hand

Ten problems on optimizer update rules: gradient descent on a quadratic and the 2/λ_max learning-rate limit, momentum unrolled as a weighted sum of past gradients and its 1/(1 − β) step, the EMA form, Nesterov, AdaGrad and RMSProp, Adam's bias correction and its first step, two Adam steps by hand, and L2 penalty versus AdamW's decoupled weight decay, with worked solutions and the mistakes that change the step size without anyone noticing.

BEFORE YOU START

An optimizer turns a sequence of gradients into a sequence of parameter updates, and every popular one is a short recurrence that can be run by hand. Running it by hand is how you find out what the step size really is. These ten problems start with plain gradient descent on a quadratic, where the learning-rate limit comes from the Hessian's largest eigenvalue, then unroll momentum into a weighted sum of past gradients, compare its two common forms and Nesterov's variant, follow AdaGrad and RMSProp under a constant gradient, derive Adam's bias correction, take two Adam steps with numbers, and separate an L2 penalty from AdamW's weight decay. The five mistakes at the end each change the effective step size by a constant factor: a stability bound from the wrong eigenvalue, a momentum step taken to be the plain step, a learning rate carried between momentum conventions, an Adam step without bias correction, and an L2 penalty read as weight decay.

- The parameters are $\theta \in \mathbb{R}^d$ and the loss is $L(\theta)$. θ_0 is the starting point and θ_t the parameters after step $t = 1, 2, \ldots$. The gradient used at step t is $g_t = \nabla L(\theta_{t-1})$, taken at the parameters before the step. $\eta > 0$ is the learning rate.
- Gradient descent: $\theta_t = \theta_{t-1} - \eta g_t$.
- Momentum (the "classic" or summed form): $v_0 = 0$, $v_t = \beta v_{t-1} + g_t$, $\theta_t = \theta_{t-1} - \eta v_t$, with $0 \le \beta < 1$. This is PyTorch's `torch.optim.SGD` with `momentum=`β and the default `dampening=0`. The EMA form is $m_0 = 0$, $m_t = \beta m_{t-1} + (1-\beta) g_t$, $\theta_t = \theta_{t-1} - \eta' m_t$.
- Nesterov momentum, in the form PyTorch uses with `nesterov=True`: v_t as above, and $\theta_t = \theta_{t-1} - \eta\,(g_t + \beta v_t)$.
- AdaGrad: $G_0 = 0$, $G_t = G_{t-1} + g_t^2$, $\theta_t = \theta_{t-1} - \eta\, g_t / (\sqrt{G_t} + \epsilon)$. RMSProp: $s_0 = 0$, $s_t = \rho s_{t-1} + (1-\rho) g_t^2$, $\theta_t = \theta_{t-1} - \eta\, g_t / (\sqrt{s_t} + \epsilon)$, with $0 < \rho < 1$.

- Adam: $m_0 = v_0 = 0$, $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$, $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$, the bias-corrected estimates $\hat{m}_t = m_t / (1-\beta_1^t)$ and $\hat{v}_t = v_t / (1-\beta_2^t)$, and $\theta_t = \theta_{t-1} - \eta\, \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$. PyTorch's defaults are $\eta = 10^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$. Here β_1^t is β_1 to the power t; v_t in Adam is its second-moment estimate, not momentum's v_t, and each problem says which one it uses.
- In the adaptive methods, g_t^2, $\sqrt{\cdot}$, $|\cdot|$ and the division act entry by entry, and $\epsilon > 0$ is a small constant that keeps the denominator away from 0.
- A quadratic loss is $L(\theta) = \frac{1}{2}\theta^\top A \theta - b^\top \theta$ with A symmetric positive definite, so $\nabla L(\theta) = A\theta - b$ and the Hessian is A. Its eigenvalues are $0 < \lambda_{\min} \le \cdots \le \lambda_{\max}$, and $\theta^* = A^{-1} b$ is the minimizer.

Builds on: Regression gradients: linear, logistic and softmax

Problems

1. Gradient descent on the quadratic $L(\theta) = \frac{1}{2}\theta^\top A \theta - b^\top \theta$. Let $e_t = \theta_t - \theta^*$ be the error after step t. Show that $e_t = (I - \eta A)\, e_{t-1}$ and hence write e_t in terms of e_0. difficulty ·

2. Show that gradient descent on the quadratic converges to θ^* from every starting point exactly when $0 < \eta < 2/\lambda_{\max}$. For $A = \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}$, find that range, and find the η that makes the worst-case error factor per step as small as possible. difficulty ··

3. Momentum, classic form, with any gradients $g_1, g_2, \ldots$ treated as given. Show that $v_t = \sum_{s=1}^{t} \beta^{t-s} g_s$ and that $\theta_t = \theta_0 - \eta \sum_{s=1}^{t} \dfrac{1-\beta^{t-s+1}}{1-\beta}\, g_s$. difficulty ···

4. Classic momentum with a constant gradient, $g_t = g$ for every t. Find v_t and the size of the step as $t \to \infty$. How large is it for $\beta = 0.9$ compared with plain gradient descent at the same η? difficulty ·

5. The EMA form of momentum is $m_t = \beta m_{t-1} + (1-\beta) g_t$, $\theta_t = \theta_{t-1} - \eta' m_t$, with $m_0 = 0$. Show that $m_t = (1-\beta) v_t$ for the classic v_t fed the same gradients, and find the η for which the classic form produces exactly the same parameters as the EMA form with learning rate η'. difficulty ··

6. One parameter, $L(\theta) = \theta^2$, $\theta_0 = 1$, $\eta = 0.1$, $\beta = 0.9$. Compute θ_1 and θ_2 with classic momentum and with Nesterov momentum in the form $\theta_t = \theta_{t-1} - \eta\,(g_t + \beta v_t)$. difficulty ··

7. One parameter with a constant gradient $g \neq 0$, and $\epsilon = 0$. Find the step size $|\theta_t - \theta_{t-1}|$ of AdaGrad and of RMSProp as functions of t. What happens to each as $t \to \infty$, and what is RMSProp's first step for $\rho = 0.99$? difficulty ··

8. Adam. Show that $m_t = (1-\beta_1)\sum_{s=1}^{t} \beta_1^{t-s} g_s$, whose weights sum to $1 - \beta_1^t$, so that a constant gradient g gives $\hat{m}_t = g$ and $\hat{v}_t = g^2$ exactly. Then show that, whatever the gradients, Adam's first step is $\theta_1 - \theta_0 = -\eta\, g_1/(|g_1| + \epsilon)$, entry by entry. difficulty ··

9. One parameter, $L(\theta) = \theta^2$, $\theta_0 = 1$, Adam with $\eta = 0.1$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 0$. Compute θ_1 and θ_2, rounding to four decimal places at the end. difficulty ···

10. Two parameters, $\theta_0 = (1, 1)$, first gradient of the data loss $g_1 = (0, 4)$, $\eta = 10^{-3}$, $\lambda = 10^{-2}$, $\epsilon = 10^{-8}$. Compute θ_1 for (a) Adam with an L2 penalty, which replaces g_t by $g_t + \lambda\theta_{t-1}$ before the moment updates (PyTorch's `Adam` with `weight_decay=`λ), and (b) AdamW, which sets $\theta_t = (1 - \eta\lambda)\,\theta_{t-1} - \eta\,\hat{m}_t/(\sqrt{\hat{v}_t} + \epsilon)$ with the moments built from g_t alone (PyTorch's `AdamW`). Write the decay part of the step in each. difficulty ···

Answers

1. $e_t = (I - \eta A)^t \, e_0$
2. Converges from every start iff $0 < \eta < 2/\lambda_{\max}$; for this A, $0 < \eta < \frac{1}{2}$, and $\eta = \frac{1}{3}$ gives the smallest worst-case factor, $\frac{1}{3}$ per step
3. $v_t = \sum_{s=1}^{t} \beta^{t-s} g_s$ and $\theta_t = \theta_0 - \eta \sum_{s=1}^{t} \frac{1-\beta^{t-s+1}}{1-\beta} \, g_s$
4. $v_t = \frac{1-\beta^t}{1-\beta} \, g$, and the step tends to $\frac{\eta}{1-\beta} \, g$: ten times the plain step ηg when $\beta = 0.9$
5. $m_t = (1-\beta) v_t$; the classic form with $\eta = (1-\beta)\eta'$ reproduces the EMA form exactly
6. Classic: $\theta_1 = 0.8$, $\theta_2 = 0.46$; Nesterov: $\theta_1 = 0.62$, $\theta_2 = 0.2224$
7. AdaGrad: $|\Delta\theta_t| = \eta/\sqrt{t} \to 0$; RMSProp: $|\Delta\theta_t| = \eta/\sqrt{1-\rho^t} \to \eta$, with a first step of 10η for $\rho = 0.99$
8. For a constant gradient, $\hat{m}_t = g$ and $\hat{v}_t = g^2$; for any gradients, $\theta_1 - \theta_0 = -\eta \, g_1/(|g_1| + \epsilon)$
9. $\theta_1 = 0.9$ and $\theta_2 \approx 0.8004$
10. (a) Adam + L2: $\theta_1 \approx (0.999,\ 0.999)$, with the decay divided by $\sqrt{\hat{v}_t} + \epsilon$; (b) AdamW: $\theta_1 = (0.99999,\ 0.99899)$, decay part $\eta\lambda\theta$

Worked solutions

Problem 1

> Gradient descent on the quadratic $L(\theta) = \frac{1}{2}\theta^\top A\theta - b^\top\theta$. Let $e_t = \theta_t - \theta^*$ be the error after step t. Show that $e_t = (I - \eta A)\, e_{t-1}$ and hence write e_t in terms of e_0.

$\nabla L(\theta^*) = A\theta^* - b = 0$, so $b = A\theta^*$.	$\theta^* = A^{-1}b$ is where the gradient vanishes; writing b this way lets the gradient be expressed through the error.
$g_t = A\theta_{t-1} - b = A\theta_{t-1} - A\theta^* = Ae_{t-1}$.	The gradient of the quadratic is $A\theta - b$; substitute step 1 and factor out A.
$e_t = \theta_{t-1} - \eta g_t - \theta^* = e_{t-1} - \eta A e_{t-1} = (I - \eta A)\, e_{t-1}$.	Subtract θ^* from both sides of the update and use step 2.
$e_t = (I - \eta A)^t\, e_0$	Step 3 applied t times. On a quadratic, gradient descent is a fixed linear map applied to the error over and over, so whether it converges depends only on the eigenvalues of $I - \eta A$ (Problem 2).

Problem 2

> Show that gradient descent on the quadratic converges to θ^* from every starting point exactly when $0 < \eta < 2/\lambda_{\max}$. For $A = \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}$, find that range, and find the η that makes the worst-case error factor per step as small as possible.

$A = Q\Lambda Q^\top$ with Q orthogonal and $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_d)$, so $I - \eta A = Q(I - \eta\Lambda)Q^\top$.	A symmetric matrix has an orthonormal basis of eigenvectors, and $I = QQ^\top$, so I and A are diagonal in the same basis.
With $c_t = Q^\top e_t$, $c_{t,i} = (1 - \eta\lambda_i)^t\, c_{0,i}$.	Problem 1 in the eigenvector basis: each coordinate is multiplied by its own factor $1 - \eta\lambda_i$ at every step, independently of the others.

$e_t \to 0$ for every e_0 iff $\lvert 1-\eta\lambda_i\rvert < 1$ for every i, iff $0 < \eta\lambda_i < 2$ for every i.	$\lVert e_t\rVert = \lVert c_t\rVert$ because Q is orthogonal. A start along eigenvector i has only $c_{0,i} \neq 0$, and $r^t \to 0$ exactly when $\lvert r\rvert < 1$.
Every $\lambda_i > 0$, so the condition is $0 < \eta < 2/\lambda_{\max}$.	The largest eigenvalue gives the tightest upper bound; the others are then satisfied automatically.
$\det(A-\lambda I) = (3-\lambda)^2 - 1 = 0$ gives $\lambda = 2$ and $\lambda = 4$.	$3-\lambda = \pm 1$.
The worst factor is $\max(\lvert 1-2\eta\rvert, \lvert 1-4\eta\rvert)$; it is smallest where $1-2\eta = -(1-4\eta)$, at $\eta = \frac{1}{3}$, giving $\frac{1}{3}$.	As η grows, $\lvert 1-2\eta\rvert$ falls while $\lvert 1-4\eta\rvert$ rises once $\eta > \frac{1}{4}$; the maximum of the two is least where they cross. In general the crossing is at $\eta = 2/(\lambda_{\min}+\lambda_{\max})$, with factor $(\lambda_{\max}-\lambda_{\min})/(\lambda_{\max}+\lambda_{\min})$.
Converges from every start iff $0 < \eta < 2/\lambda_{\max}$; for this A, $0 < \eta < \frac{1}{2}$, and $\eta = \frac{1}{3}$ gives the smallest worst-case factor, $\frac{1}{3}$ per step	The steep direction ($\lambda_{\max}$) caps the learning rate and the flat direction ($\lambda_{\min}$) then sets the speed; the ratio $\lambda_{\max}/\lambda_{\min}$, the condition number, is what momentum and the adaptive methods below try to work around.

Problem 3

> Momentum, classic form, with any gradients $g_1, g_2, \ldots$ treated as given. Show that $v_t = \sum_{s=1}^{t} \beta^{t-s} g_s$ and that $\theta_t = \theta_0 - \eta \sum_{s=1}^{t} \dfrac{1-\beta^{t-s+1}}{1-\beta}\, g_s$.

$v_1 = \beta v_0 + g_1 = g_1$.	$v_0 = 0$; this matches the claimed sum for $t = 1$, which has the single term $\beta^0 g_1$.
If $v_{t-1} = \sum_{s=1}^{t-1} \beta^{t-1-s} g_s$, then $v_t = \sum_{s=1}^{t-1} \beta^{t-s} g_s + g_t = \sum_{s=1}^{t} \beta^{t-s} g_s$.	Multiplying by β raises every exponent by one, and the new gradient enters with weight $\beta^0 = 1$. Induction on t completes the first claim.
$\theta_t = \theta_0 - \eta \sum_{k=1}^{t} v_k = \theta_0 - \eta \sum_{k=1}^{t} \sum_{s=1}^{k} \beta^{k-s} g_s$.	Each step subtracts ηv_k, so after t steps the parameters have moved by the sum of all of them.

$\sum_{k=1}^{t}\sum_{s=1}^{k}\beta^{k-s}g_s = \sum_{s=1}^{t} g_s \sum_{k=s}^{t}\beta^{k-s}.$	Swap the order of summation: the pairs with $1 \le s \le k \le t$ can be listed by s first, with k running from s to t.
$\sum_{k=s}^{t}\beta^{k-s} = \sum_{j=0}^{t-s}\beta^{j} = \dfrac{1-\beta^{t-s+1}}{1-\beta}.$	A geometric series with $t-s+1$ terms; $\beta < 1$ so the denominator is not zero.
$v_t = \sum_{s=1}^{t}\beta^{t-s}g_s$ **and** $\theta_t = \theta_0 - \eta\sum_{s=1}^{t}\frac{1-\beta^{t-s+1}}{1-\beta}\,g_s$	Step 2 and steps 3 to 5. An old gradient's total effect on θ approaches $\eta/(1-\beta)$ times the gradient: every gradient is eventually applied $1/(1-\beta)$ times over, spread across the following steps.

Problem 4

> Classic momentum with a constant gradient, $g_t = g$ for every t. Find v_t and the size of the step as $t \to \infty$. How large is it for $\beta = 0.9$ compared with plain gradient descent at the same η?

$v_t = \sum_{s=1}^{t}\beta^{t-s}g = g\sum_{j=0}^{t-1}\beta^{j}.$	Problem 3 with every $g_s = g$; substitute $j = t-s$.
$v_t = \dfrac{1-\beta^{t}}{1-\beta}\,g.$	A geometric series with t terms.
$\beta^{t} \to 0$, so $\eta v_t \to \dfrac{\eta}{1-\beta}\,g.$	$0 \le \beta < 1$.
$v_t = \frac{1-\beta^{t}}{1-\beta}\,g$, **and the step tends to** $\frac{\eta}{1-\beta}\,g$: **ten times the plain step** ηg **when** $\beta = 0.9$	On a long stretch where the gradient barely changes, momentum moves $1/(1-\beta)$ times as far per step as gradient descent with the same learning rate. Where the gradient flips sign from step to step, the terms of step 1 alternate and partly cancel instead, which is what damps oscillation across a narrow valley.

Problem 5

> The EMA form of momentum is $m_t = \beta m_{t-1} + (1-\beta) g_t$, $\theta_t = \theta_{t-1} - \eta' m_t$, with $m_0 = 0$. Show that $m_t = (1-\beta) v_t$ for the classic v_t fed the same gradients, and find the η for which the classic form produces exactly the same parameters as the EMA form with learning rate η'.

$m_0 = 0 = (1-\beta) v_0$.	Both start at zero.
If $m_{t-1} = (1-\beta) v_{t-1}$, then $m_t = (1-\beta)\beta v_{t-1} + (1-\beta) g_t = (1-\beta)(\beta v_{t-1} + g_t) = (1-\beta) v_t$.	Substitute and factor out $1-\beta$; the bracket is the classic recurrence. Induction gives the claim for every t, as long as both are fed the same g_t.
$\eta' m_t = \eta'(1-\beta) v_t$, so the two updates agree when $\eta = (1-\beta)\eta'$.	With the same starting point and the same steps, the two runs pass through the same parameters, so their gradients g_t are the same too and step 2 keeps applying.
$m_t = (1-\beta) v_t$; **the classic form with $\eta = (1-\beta)\eta'$ reproduces the EMA form exactly**	With $\beta = 0.9$, an EMA-form learning rate of 0.1 is a classic-form learning rate of 0.01. Problem 4's factor $1/(1-\beta)$ lives in the classic form; the EMA form has already divided it out, since its weights $(1-\beta)\beta^{t-s}$ sum to $1 - \beta^t \le 1$.

Problem 6

> One parameter, $L(\theta) = \theta^2$, $\theta_0 = 1$, $\eta = 0.1$, $\beta = 0.9$. Compute θ_1 and θ_2 with classic momentum and with Nesterov momentum in the form $\theta_t = \theta_{t-1} - \eta\,(g_t + \beta v_t)$.

$g_t = 2\theta_{t-1}$.	$\frac{d}{d\theta}\theta^2 = 2\theta$, evaluated before the step.
Classic, step 1: $g_1 = 2$, $v_1 = 2$, $\theta_1 = 1 - 0.1 \cdot 2 = 0.8$.	$v_1 = \beta \cdot 0 + g_1$.
Classic, step 2: $g_2 = 1.6$, $v_2 = 0.9 \cdot 2 + 1.6 = 3.4$, $\theta_2 = 0.8 - 0.34 = 0.46$.	$v_2 = \beta v_1 + g_2$, then $\theta_2 = \theta_1 - \eta v_2$.

Nesterov, step 1: $g_1 = 2$, $v_1 = 2$, $g_1 + \beta v_1 = 3.8$, $\theta_1 = 1 - 0.38 = 0.62$.	The buffer is updated exactly as in the classic form; only the direction used for the step changes.
Nesterov, step 2: $g_2 = 1.24$, $v_2 = 1.8 + 1.24 = 3.04$, $g_2 + \beta v_2 = 1.24 + 2.736 = 3.976$, $\theta_2 = 0.62 - 0.3976 = 0.2224$.	$g_2 = 2 \cdot 0.62$, then the same two lines as step 4.
Classic: $\theta_1 = 0.8$, $\theta_2 = 0.46$; **Nesterov:** $\theta_1 = 0.62$, $\theta_2 = 0.2224$	Nesterov's direction is $g_t + \beta v_t = (1 + \beta) g_t + \beta^2 v_{t-1}$: the newest gradient counts $1 + \beta$ times and the older history is damped by one more factor of β, so the step leans toward where momentum is about to carry the parameters. Under a constant gradient both forms tend to the same step, $\eta g/(1 - \beta)$, because $g + \beta g/(1 - \beta) = g/(1 - \beta)$; they differ when the gradient changes.

Problem 7

> One parameter with a constant gradient $g \neq 0$, and $\epsilon = 0$. Find the step size $|\theta_t - \theta_{t-1}|$ of AdaGrad and of RMSProp as functions of t. What happens to each as $t \to \infty$, and what is RMSProp's first step for $\rho = 0.99$?

AdaGrad: $G_t = \sum_{s=1}^{t} g^2 = t\, g^2$, so $\sqrt{G_t} = \sqrt{t}\, \lvert g \rvert$.	G adds the squared gradient at every step and never forgets.
AdaGrad step: $\eta \, \lvert g \rvert / (\sqrt{t}\, \lvert g \rvert) = \eta / \sqrt{t}$.	The gradient's size cancels; only the count of steps is left.
RMSProp: $s_t = (1 - \rho) \sum_{s=1}^{t} \rho^{t-s} g^2 = (1 - \rho^t)\, g^2$.	Problem 3's unrolling with ρ for β and the extra factor $1 - \rho$, then the geometric sum $\sum_{j=0}^{t-1} \rho^j = (1 - \rho^t)/(1 - \rho)$.
RMSProp step: $\eta \, \lvert g \rvert / (\sqrt{1 - \rho^t}\, \lvert g \rvert) = \eta / \sqrt{1 - \rho^t}$.	$\sqrt{s_t} = \sqrt{1 - \rho^t}\, \lvert g \rvert$.

At $t = 1$ with $\rho = 0.99$: $\eta/\sqrt{0.01} = 10\eta$; as $t \to \infty$ the step falls to η.	$\rho^t \to 0$, and $1 - \rho^t$ is smallest at $t = 1$.
AdaGrad: $\lvert\Delta\theta_t\rvert = \eta/\sqrt{t} \to 0$; **RMSProp:** $\lvert\Delta\theta_t\rvert = \eta/\sqrt{1-\rho^t} \to \eta$, **with a first step of** 10η **for** $\rho = 0.99$	AdaGrad's sum grows without limit, so its steps keep shrinking even when the gradient does not; RMSProp's moving average forgets, so its step settles at η. Neither depends on $\lvert g\rvert$: multiplying the loss by a constant leaves both updates unchanged when $\epsilon = 0$. RMSProp's oversized early steps come from $s_0 = 0$ biasing s_t toward zero, which is the bias Adam corrects (Problem 8).

Problem 8

> Adam. Show that $m_t = (1-\beta_1)\sum_{s=1}^{t} \beta_1^{t-s} g_s$, whose weights sum to $1 - \beta_1^t$, so that a constant gradient g gives $\hat{m}_t = g$ and $\hat{v}_t = g^2$ exactly. Then show that, whatever the gradients, Adam's first step is $\theta_1 - \theta_0 = -\eta\, g_1/(\lvert g_1\rvert + \epsilon)$, entry by entry.

$m_t = (1-\beta_1)\sum_{s=1}^{t} \beta_1^{t-s} g_s$.	Problem 5, step 2 shows the EMA recurrence is $(1-\beta_1)$ times the classic one, and Problem 3 unrolls the classic one.
$(1-\beta_1)\sum_{s=1}^{t} \beta_1^{t-s} = (1-\beta_1)\,\dfrac{1-\beta_1^t}{1-\beta_1} = 1 - \beta_1^t$.	Geometric series with t terms. The weights fall short of 1 by β_1^t, the weight that $m_0 = 0$ would have carried.
With $g_s = g$: $m_t = (1-\beta_1^t)\,g$, so $\hat{m}_t = m_t/(1-\beta_1^t) = g$.	Step 2 times g. Dividing by the sum of the weights turns a sum weighted toward zero into a true weighted average.
The same steps for v_t, with β_2 and g^2: $v_t = (1-\beta_2^t)\,g^2$ and $\hat{v}_t = g^2$.	v_t is the same recurrence applied to g_t^2.

At $t = 1$ for any g_1: $m_1 = (1-\beta_1)g_1$ and $v_1 = (1-\beta_2)g_1^2$, so $\hat{m}_1 = g_1$ and $\hat{v}_1 = g_1^2$, $\sqrt{\hat{v}_1} = \lvert g_1 \rvert$.	One gradient is a constant sequence of length one, so step 3 applies; $\sqrt{x^2} = \lvert x \rvert$ entry by entry.
For a constant gradient, $\hat{m}_t = g$ and $\hat{v}_t = g^2$; for any gradients, $\theta_1 - \theta_0 = -\eta\, g_1/(\lvert g_1 \rvert + \epsilon)$	Each entry of the first step is $-\eta$ times the sign of its gradient (slightly less when $\lvert g_1 \rvert$ is comparable to ϵ), whatever the gradient's size. Adam's learning rate is therefore close to the actual distance each parameter moves early in training, which is why η does not need retuning when the loss is rescaled.

Problem 9

> One parameter, $L(\theta) = \theta^2$, $\theta_0 = 1$, Adam with $\eta = 0.1$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 0$. Compute θ_1 and θ_2, rounding to four decimal places at the end.

$g_1 = 2$, $m_1 = 0.1 \cdot 2 = 0.2$, $v_1 = 0.001 \cdot 4 = 0.004$.	$g = 2\theta$ at $\theta_0 = 1$; both moments start at 0.
$\hat{m}_1 = 0.2/0.1 = 2$, $\hat{v}_1 = 0.004/0.001 = 4$, step $= 0.1 \cdot 2/\sqrt{4} = 0.1$, so $\theta_1 = 0.9$.	$1-\beta_1^1 = 0.1$ and $1-\beta_2^1 = 0.001$; the first step is η times the sign, as Problem 8 predicts.
$g_2 = 1.8$, $m_2 = 0.9 \cdot 0.2 + 0.1 \cdot 1.8 = 0.36$, $v_2 = 0.999 \cdot 0.004 + 0.001 \cdot 3.24 = 0.007236$.	$g = 2 \cdot 0.9$ and $1.8^2 = 3.24$; then the two recurrences.
$\hat{m}_2 = 0.36/0.19 \approx 1.89474$, $\hat{v}_2 = 0.007236/0.001999 \approx 3.61981$.	$1-0.9^2 = 0.19$ and $1-0.999^2 = 0.001999$.
Step $= 0.1 \cdot 1.89474/\sqrt{3.61981} \approx 0.1 \cdot 1.89474/1.90258 \approx 0.09959$.	$\sqrt{3.61981} \approx 1.90258$.

$\theta_1 = 0.9$ **and** $\theta_2 \approx 0.8004$	The second step is 0.0996, again almost exactly η, although the gradient fell by 10%: early on $\hat{m}_t/\sqrt{\hat{v}_t}$ is a ratio of two averages of the same few gradients and stays near 1. Gradient descent with $\eta = 0.1$ would have moved 0.2 and then 0.16.

Problem 10

> Two parameters, $\theta_0 = (1, 1)$, first gradient of the data loss $g_1 = (0, 4)$, $\eta = 10^{-3}$, $\lambda = 10^{-2}$, $\epsilon = 10^{-8}$. Compute θ_1 for (a) Adam with an L2 penalty, which replaces g_t by $g_t + \lambda\theta_{t-1}$ before the moment updates (PyTorch's `Adam` with `weight_decay=`λ), and (b) AdamW, which sets $\theta_t = (1 - \eta\lambda)\,\theta_{t-1} - \eta\,\hat{m}_t/(\sqrt{\hat{v}_t} + \epsilon)$ with the moments built from g_t alone (PyTorch's `AdamW`). Write the decay part of the step in each.

(a) The gradient Adam sees is $\tilde{g}_1 = g_1 + \lambda\theta_0 = (0.01,\ 4.01)$.	The penalty $\frac{\lambda}{2}\|\theta\|^2$ has gradient $\lambda\theta$, added to the data gradient before anything else happens.
(a) $\theta_1 = \theta_0 - \eta\,\tilde{g}_1/(\|\tilde{g}_1\| + \epsilon) \approx (1 - 0.001,\ 1 - 0.001) = (0.999,\ 0.999)$.	Problem 8: the first step is $-\eta$ times the sign of the gradient it is given; $0.01/(0.01 + 10^{-8})$ and $4.01/(4.01 + 10^{-8})$ are both 1 to six decimal places.
(a) In general $\hat{m}_t$ contains a weighted average of the past $\lambda\theta$ terms, and that share of the step is divided entry by entry by $\sqrt{\hat{v}_t} + \epsilon$, exactly like the data gradient.	m_t is linear in the gradients it is fed, so it splits into a data part and a penalty part; the division by $\sqrt{\hat{v}_t} + \epsilon$ applies to both, with $\hat{v}_t$ built from the squared total gradient.
(b) $\hat{m}_1 = g_1$, $\sqrt{\hat{v}_1} = \|g_1\| = (0, 4)$, so the Adam part is $-\eta\,(0/(0 + 10^{-8}),\ 4/(4 + 10^{-8})) \approx (0,\ -0.001)$.	Problem 8 with the data gradient alone; the first entry's gradient is exactly 0, so its Adam step is 0.
(b) $\theta_1 = (1 - 10^{-5})(1, 1) + (0, -0.001) = (0.99999,\ 0.99899)$.	$\eta\lambda = 10^{-5}$: each weight shrinks by the same fraction before the Adam step.

(a) Adam + L2: $\theta_1 \approx (0.999,\ 0.999)$, **with the decay divided by** $\sqrt{\hat{v}_t} + \epsilon$; **(b) AdamW:** $\theta_1 = (0.99999,\ 0.99899)$, **decay part** $\eta\lambda\theta$	With L2, the weight that has no data gradient moved 10^{-3}, a hundred times the $\eta\lambda\theta = 10^{-5}$ that "weight decay" suggests, while for the weight with a large gradient the penalty barely changed the step. Dividing by $\sqrt{\hat{v}_t}$ makes the decay strongest on weights with small gradient history and weakest on those with large ones. AdamW keeps the decay a fixed fraction $\eta\lambda$ of every weight, which is why its λ (PyTorch's default is 0.01) behaves like weight decay in plain SGD.

Where this goes wrong

1. Learning-rate limit from the smallest eigenvalue

The minimum lies along the flat directions, and it is natural to tune the step to the direction that has furthest to go.

The error along eigenvector i is multiplied by $1 - \eta\lambda_i$ at each step	Right so far: Problem 2, step 2.
“The slow direction is the bottleneck, so the learning rate should be set by $\lambda_{\min}$.”	The reasoning that causes the mistake: the flat direction sets how fast gradient descent can go, but the steep direction sets whether it converges at all.
For A with eigenvalues 2 and 4, any $\eta < 2/\lambda_{\min} = 1$ converges, so take $\eta = 0.9$ ✗	At $\eta = 0.9$ the steep direction's factor is $1 - 0.9 \cdot 4 = -2.6$: the error there flips sign and grows by $2.6\times$ every step. The limit is $2/\lambda_{\max} = \frac{1}{2}$ (Problem 2); the flat direction's slowness has to be fixed by momentum or preconditioning, not by a larger η.

2. Momentum step taken to equal the plain step ηg

Adding momentum to a working SGD setup looks like adding smoothing, not like changing the learning rate.

$v_t = \beta v_{t-1} + g_t$, $\theta_t = \theta_{t-1} - \eta v_t$	Right so far: classic momentum, PyTorch's `SGD` with `momentum=`β.
"Momentum averages the recent gradients, so the step is still about ηg, only smoother."	The analogy that causes the mistake: reading the classic form as if it were the EMA form, whose weights sum to at most 1.
Switching from plain SGD at $\eta = 0.1$ to momentum 0.9 at $\eta = 0.1$ keeps the step size at about $0.1\,g$ ✗	The classic buffer sums the gradients, so on a steady gradient the step tends to $\eta g/(1-\beta) = g$, ten times larger (Problem 4). A learning rate that was near the stability limit of Problem 2 is now well past it; to keep the step the same, multiply η by $1-\beta$.

3. Same learning rate for the EMA and summed momentum forms

Papers and libraries write momentum both ways, and η is called the learning rate in both.

$m_t = (1-\beta)v_t$	Right so far: Problem 5, step 2.
"Both forms are momentum with the same β, so a learning rate from one carries over to the other."	The shortcut that causes the mistake: treating the factor $1-\beta$ in the EMA recurrence as a detail of the bookkeeping rather than a scale on every step.
An EMA-form run with $\eta' = 0.1$, $\beta = 0.9$ is reproduced by the classic form with $\eta = 0.1$ ✗	The classic form needs $\eta = (1-\beta)\eta' = 0.01$ (Problem 5). With $\eta = 0.1$ every step is ten times too large; going the other way, from classic to EMA with the number unchanged, every step is ten times too small and training looks merely slow.

4. First Adam step computed without the bias correction

The bias correction looks like a small fix-up that matters only for a few steps, so a hand-written Adam often leaves it out.

$m_1 = (1-\beta_1)g_1,\ v_1 = (1-\beta_2)g_1^2$	Right so far: the first moment updates from zero.
"The correction only matters for small t; use m_t and v_t directly."	The shortcut that causes the mistake: assuming the two biases are similar and cancel in the ratio $m_t/\sqrt{v_t}$.
$\theta_1 - \theta_0 = -\eta\, m_1/\sqrt{v_1} = -\eta\, \dfrac{0.1\, g_1}{\sqrt{0.001}\,\lvert g_1\rvert} \approx -3.16\,\eta\ \operatorname{sign}(g_1)$ ✗	The biases do not cancel: m_1 is shrunk by $1-\beta_1 = 0.1$, $\sqrt{v_1}$ by $\sqrt{1-\beta_2} \approx 0.0316$. The uncorrected first step is 3.16 times η instead of η (Problem 8), and the factor $(1-\beta_1^t)/\sqrt{1-\beta_2^t}$ by which every uncorrected step is too large rises to about 6.5 near $t = 10$ and is still about 1.26 at $t = 1000$, so the early steps are oversized just when the parameters are furthest from sensible.

5. Adam's L2 penalty taken as a decay of ηλθ

In plain SGD an L2 penalty and weight decay are the same thing: the penalty's gradient $\lambda\theta$ times η shrinks each weight by $\eta\lambda\theta$.

The gradient Adam sees is $g_t + \lambda\theta_{t-1}$	Right so far: Problem 10, step 1, PyTorch's `Adam` with `weight_decay=` λ.
"Adding $\lambda\theta$ to the gradient is weight decay, so each step shrinks the weights by $\eta\lambda\theta$ as in SGD."	The analogy that causes the mistake: SGD multiplies the whole gradient by the same η; Adam divides it entry by entry by $\sqrt{\hat{v}_t} + \epsilon$.

In Problem 10, Adam with `weight_decay=0.01` moves the weight with zero data gradient by $\eta\lambda\theta = 10^{-5}$ ✗	It moves by 10^{-3}, a full Adam step, because the penalty gradient is normalized by its own size (Problem 10, step 2). Weights with small gradients are decayed far more than $\eta\lambda\theta$ and weights with large gradients far less; AdamW applies $\eta\lambda\theta$ separately, outside the normalization.

More at eulerpad.app. This is one of 82 free practice sets with worked solutions at eulerpad.app/practice, alongside guides to the topics behind them.

EulerPad, the app, reads your handwritten math on iPad, iPhone and Android and checks it line by line. It's free to start.

This set is also Chapter 21 of *Deep Learning Math by Hand*, a 250-problem workbook with worked solutions: eulerpad.app/workbooks

eulerpad.app/practice/momentum-rmsprop-and-adam