CMPUT 654: Mathematical Foundations of Modern AI Systems Fall 2026

Lecture 12: Accelerated and adaptive first-order methods

October 13, 2026

Lecturer: Csaba Szepesvári *Note prepared by: Csaba Szepesvári*

Reading guide. Lecture 11 showed that a β-smooth objective decreases under gradient descent when the stepsize satisfies $\eta \le 1/\beta$. Under a μ-PL inequality, the same update reduces the objective gap geometrically, and the number of iterations needed for a fixed relative reduction is proportional to β/μ. Two questions remain. First, an implementation may not know β. Second, even the best safe scalar step can be slow when different directions have very different curvature.

The two questions concern different limitations of gradient descent. An unknown β prevents the algorithm from choosing a certified scalar stepsize. A large ratio β/μ means that every safe scalar stepsize is slow in some directions. Backtracking addresses the first limitation by testing proposed finite steps. Acceleration and preconditioning address the second limitation by changing how information from different iterations or different parameter directions is used. Keeping stepsize adaptation separate from conditioning improvement is important: a method can choose a safe stepsize automatically and still converge slowly on an ill-conditioned problem.

The first part of this lecture develops backtracking line search, which tests the finite-step inequality needed by the proof and therefore avoids advance knowledge of β. The second part explains the condition-number barrier for gradient descent and proves the main acceleration result: for smooth, strongly convex objectives, the number of gradient evaluations needed to reduce the objective gap by a fixed relative amount improves from order κ to order $\sqrt{\kappa}$. The third part separates scalar stepsize adaptation from adaptation of the update geometry. It first studies preconditioning for a fixed smooth objective and then changes the information available to the update: AdaGrad and Adam adapt to the subgradients or stochastic gradients observed during the run. AdamW adds a separate choice about weight decay.

The principal results for a first reading are the backtracking guarantee, the linear acceleration theorem and the corresponding lower-bound comparison, the preconditioning calculation, and the distinction between scalar online adaptation, diagonal AdaGrad, and Adam. Material marked with * may be skipped on a first reading. The prerequisites are the smoothness, convexity, strong-convexity, and PL results from Lecture 11, together with eigenvalues of symmetric matrices.

12.1 Review: Gradient descent with smooth objectives

This section recalls why the usual gradient-descent stepsize depends on the smoothness constant. The argument also separates the condition used by the convergence proof—a one-step decrease inequality—from one sufficient certificate for that condition—a known global smoothness constant.

Let $L : \mathbb{R}^d \to \mathbb{R}$ be differentiable and bounded below, and put

$$L^\star = \inf_{\theta \in \mathbb{R}^d} L(\theta).$$

The function is β-smooth when

$$\|\nabla L(u) - \nabla L(v)\|_2 \le \beta \|u - v\|_2 \qquad \text{for every } u, v \in \mathbb{R}^d. \tag{12.1}$$

Smoothness converts first-order information at u into the quadratic upper bound

$$L(v) \le L(u) + \langle \nabla L(u), v - u \rangle + \frac{\beta}{2}\|v - u\|_2^2. \tag{12.2}$$

Applying Eq. (12.2) with $u = \theta$ and $v = \theta^+ = \theta - \eta \nabla L(\theta)$ gives

$$L(\theta^+) \le L(\theta) - \eta \left(1 - \frac{\beta\eta}{2}\right) \|\nabla L(\theta)\|_2^2. \tag{12.3}$$

The usual stepsize restriction now enters at one precise point. If $\eta \le 1/\beta$, then

$$L(\theta^+) \le L(\theta) - \frac{\eta}{2}\|\nabla L(\theta)\|_2^2. \tag{12.4}$$

The role of β is therefore quite specific. Knowledge of β guarantees Eq. (12.4) at every parameter, for every step with $\eta \le 1/\beta$. The convergence proof, however, uses the decrease inequality itself; the proof does not otherwise use the numerical value of β. Moreover, the argument gives only a sufficient restriction on the stepsize. A larger step may still satisfy the same decrease inequality at the current parameter. Section 12.2 uses the distinction between the decrease inequality and the numerical value of β to construct a stepsize rule: propose a step, test the required decrease, and shorten the step only when the test fails.

12.2 Adapting to an unknown smoothness constant

This section develops a stepsize rule for the common situation in which a useful value of β is not known. Sometimes the available analysis gives only a conservative bound $B \ge \beta$. The stepsize $1/B$ is safe, but a large ratio B/β makes gradient descent take smaller steps, and possibly more iterations, than the sharper bound would require. The first part of Exercise 1 asks you to quantify the additional iteration cost on a quadratic objective. In other problems, no finite upper bound on β may be available to the practitioner.

The preceding decrease calculation suggests searching for a safe upper estimate of β. At a parameter θ, let $g = \nabla L(\theta)$ and let $M > 0$ be a tentative upper bound on β. The estimate M defines the trial point

$$\theta^+(M) = \theta - \frac{1}{M} g. \tag{12.5}$$

If M is indeed an upper bound, so that $M \ge \beta$, then Eq. (12.3) with $\eta = 1/M$ gives

$$L(\theta^+(M)) \le L(\theta) - \frac{1}{M}\left(1 - \frac{\beta}{2M}\right)\|g\|_2^2 \le L(\theta) - \frac{1}{2M}\|g\|_2^2.$$

Thus $M \ge \beta$ is a universally safe range: every estimate in that range guarantees a decrease of at least $\|g\|_2^2/(2M)$. The factor $1/2$ is the largest constant certified uniformly over all $\beta \le M$, because the worst case $\beta = M$ makes the preceding coefficient exactly $1/2$.

Although the value of β is unknown, the promised decrease can be tested using function values. For a tentative estimate M, accept the trial point when

$$L(\theta^+(M)) \le L(\theta) - \frac{1}{2M}\|g\|_2^2. \tag{12.6}$$

Every quantity in Eq. (12.6) is available: the gradient g determines $\theta^+(M)$, and one objective evaluation gives $L(\theta^+(M))$. If the test fails, increase the estimate geometrically by replacing M with

ρM for a fixed $\rho > 1$, form the new trial point, and test again. Repeating these operations is called *backtracking line search*. The search variable is the curvature estimate M; after M is accepted, the gradient step uses the corresponding stepsize $1/M$.

The search must terminate because every $M \geq \beta$ passes Eq. (12.6). Starting from M_{start}, after k failures the estimate is $\rho^k M_{\text{start}}$, so a safe estimate is reached after only logarithmically many trials. If $M_{\text{start}} < \beta$, the first estimate that reaches the safe range is smaller than $\rho\beta$. Geometric increases therefore find an estimate within a factor ρ of the threshold β; an additive increase would depend on the numerical scale of β.

The name *backtracking* describes the movement of the trial point. Each increase of M shortens the step and moves $\theta^+(M)$ back toward θ along the negative-gradient direction.

The acceptance test is local and can succeed before M reaches β. The constant β controls the quadratic upper bound for every pair of parameters, whereas Eq. (12.6) concerns only θ and the proposed successor $\theta^+(M)$. If the test accepts a value $M < \beta$, the algorithm takes a step larger than $1/\beta$ without requiring the same curvature bound to hold elsewhere.

Backtracking gradient descent. The complete method uses an initial estimate $M_{\text{init}} > 0$ and a multiplier $\rho > 1$. At iteration t:

(1) compute $g_t = \nabla L(\theta_t)$;

(2) set $M = M_{\text{init}}$ when $t = 0$, and set $M = M_{t-1}$ when $t \geq 1$;

(3) form $\theta^+(M) = \theta_t - g_t/M$ and test whether

$$L(\theta^+(M)) \leq L(\theta_t) - \frac{1}{2M}\|g_t\|_2^2.$$

If the test fails, replace M by ρM and repeat this item;

(4) set $M_t = M$ and $\theta_{t+1} = \theta^+(M_t)$.

Rejected trials do not require additional gradient evaluations. The current parameter has not changed, so the method reuses g_t and pays only for the objective values at the new trial points.

At every iteration $t \geq 1$, the algorithm begins with M_{t-1}. Consequently, (M_t) is nondecreasing, and every rejected trial causes a permanent geometric increase in the estimate. The monotone rule gives the simple one-time bound on rejected trials proved below. The limitation is that the method may miss a smaller locally safe estimate, and hence a larger safe stepsize, at a later iteration.

Example 12.2.1 (The safe range is exact on a scalar quadratic)**.** The smoothness argument proves that every $M \geq \beta$ passes the acceptance test. The scalar quadratic shows that this sufficient condition is also necessary in the worst case. Let $L(\theta) = \beta\theta^2/2$. Then

$$\theta^+(M) = \left(1 - \frac{\beta}{M}\right)\theta.$$

For $\theta \neq 0$, substituting into Eq. (12.6) and dividing by $\beta\theta^2/2$ gives the exact equivalence

$$\left(1 - \frac{\beta}{M}\right)^2 \leq 1 - \frac{\beta}{M} \quad \Longleftrightarrow \quad M \geq \beta. \tag{12.7}$$

Thus the threshold β is not an artifact of the proof: on this objective, every smaller estimate fails. More general objectives can still accept $M < \beta$ when the curvature encountered by the current step is smaller than the global worst case. Exercise 1 asks you to analyze both the exact quadratic threshold and the possible gain from local curvature.

Proposition 12.2.2 (Termination and a bound on M_t)**.** *Suppose that L is β-smooth. Run the backtracking gradient descent defined above with $M_{\text{init}} > 0$ and $\rho > 1$. Every search terminates after finitely many trials, and, for every $t \geq 0$,*

$$M_t \leq \max\{M_{\text{init}}, \rho\beta\}. \tag{12.8}$$

Proof. The calculation preceding Eq. (12.6) shows that every $M \geq \beta$ passes the test, so each search terminates. We prove Eq. (12.8) by induction on t. At $t = 0$, the algorithm first tests M_{init}. If the first trial passes, then $M_0 = M_{\text{init}}$. Otherwise, the algorithm multiplies the trial value by ρ until M_0 passes. The immediately preceding value M_0/ρ was rejected and must therefore satisfy $M_0/\rho < \beta$. Hence either $M_0 = M_{\text{init}}$ or $M_0 < \rho\beta$, which proves Eq. (12.8) at $t = 0$.

Now suppose the bound holds at $t - 1$. If iteration t accepts its first trial, then $M_t = M_{t-1}$ and the induction hypothesis applies. If iteration t rejects at least one trial, then its last rejected value is $M_t/\rho < \beta$, so $M_t < \rho\beta$. The bound follows in both cases. □

Proposition 12.2.2 also bounds the extra objective evaluations used by the search. Across all iterations, the number of rejected trial points is at most

$$N_{\text{reject}} \leq \max\left\{0, \left\lceil \log_\rho \frac{\beta}{M_{\text{init}}} \right\rceil\right\}. \tag{12.9}$$

Every rejection multiplies the current estimate by ρ, the sequence (M_t) is nondecreasing, and every estimate at least β passes the test. Hence T iterations require at most $T + N_{\text{reject}}$ objective evaluations at trial points, in addition to the T gradient evaluations.

12.2.1 The deterministic price is a constant

Proposition 12.2.2 and Eq. (12.9) bound the extra objective evaluations used to choose the steps. The next theorem bounds the number of gradient iterations: under the PL inequality, the gradient-iteration bound needed to reach a prescribed objective gap is at most a factor ρ larger than the corresponding known-β bound.

Theorem 12.2.3 (Backtracking under PL)**.** *Suppose that L is β-smooth and satisfies*

$$\frac{1}{2}\|\nabla L(\theta)\|_2^2 \geq \mu(L(\theta) - L^\star) \qquad \text{for every } \theta \in \mathbb{R}^d. \tag{12.10}$$

Run the backtracking gradient descent defined above with multiplier $\rho > 1$ and initial estimate $M_{\text{init}} > 0$. Put $\overline{M} = \max\{M_{\text{init}}, \rho\beta\}$. Then

$$L(\theta_t) - L^\star \leq \left(1 - \frac{\mu}{\overline{M}}\right)^t \big(L(\theta_0) - L^\star\big). \tag{12.11}$$

Proof. Acceptance and Eq. (12.10) give

$$\begin{aligned} L(\theta_{t+1}) - L^\star &\leq L(\theta_t) - L^\star - \frac{1}{2M_t}\|\nabla L(\theta_t)\|_2^2 \\ &\leq \left(1 - \frac{\mu}{M_t}\right)\big(L(\theta_t) - L^\star\big). \end{aligned}$$

By Proposition 12.2.2, $M_t \leq \overline{M}$. Replacing μ/M_t by the smaller number $\mu/\overline{M}$ and iterating proves the claim. □

With the fixed stepsize $1/\beta$, the objective gap satisfies

$$L(\theta_{t+1}) - L^\star \le \left(1 - \frac{\mu}{\beta}\right)\big(L(\theta_t) - L^\star\big).$$

If $M_{\text{init}} \le \beta$, backtracking instead guarantees the one-step objective-gap multiplier $1 - \mu/(\rho\beta)$ and adds the trial objective evaluations counted above. The number of gradient iterations remains

$$O\left(\frac{\beta}{\mu}\log\frac{L(\theta_0) - L^\star}{\epsilon}\right).$$

Thus adaptation to an unknown deterministic smoothness constant changes the iteration bound only by the constant factor ρ. Backtracking may do better than the worst-case factor-ρ comparison when the values M_t that pass the local test are smaller than the global constant β.

The conclusion relies on access to exact objective values at the trial points. A noisy minibatch loss does not generally certify the same inequality for the full empirical objective or the population objective. Adaptation in the stochastic setting therefore requires a different argument.

12.3 The condition number and gradient descent

This section identifies the limitation that remains after backtracking has found a safe scalar step. One scalar must accommodate every parameter direction, so a step that is stable in a high-curvature direction can make little progress in a low-curvature direction. A strongly convex quadratic makes this condition-number obstruction exact and prepares the acceleration analysis in the next section.

Consider

$$L(\theta) = \frac{1}{2}\theta^\top H\theta - b^\top\theta, \qquad 0 < \mu I \preceq H \preceq \beta I. \tag{12.12}$$

The quadratic objective has the unique minimizer $\theta^\star = H^{-1}b$. If $e_t = \theta_t - \theta^\star$, gradient descent gives

$$e_{t+1} = (I - \eta H)e_t. \tag{12.13}$$

After diagonalizing $H = Q\,\mathrm{diag}(\lambda_1, \dots, \lambda_d)Q^\top$, every eigencomponent evolves as

$$\langle q_j, e_t\rangle = (1 - \eta\lambda_j)^t\langle q_j, e_0\rangle. \tag{12.14}$$

For this quadratic, the sharp choices of the smoothness constant and the strong-convexity or PL constant are $\beta = \lambda_{\max}(H)$ and $\mu = \lambda_{\min}(H)$. With these choices, the condition number defined in Lecture 11 equals the spectral ratio

$$\kappa = \frac{\beta}{\mu} = \frac{\lambda_{\max}(H)}{\lambda_{\min}(H)}. \tag{12.15}$$

If β and μ are instead loose bounds in $\mu I \preceq H \preceq \beta I$, then β/μ is an upper bound on the spectral ratio. A step small enough to be stable in the largest-curvature direction changes a smallest-curvature component by only a fraction of order $1/\kappa$ per iteration.

The two-dimensional diagonal case makes the obstruction explicit. For

$$L(\theta) = \frac{1}{2}\big(\mu\theta_1^2 + \beta\theta_2^2\big),$$

the step $\eta = 1/\beta$ eliminates the second coordinate in one iteration, whereas the first coordinate obeys

$$\theta_{t+1,1} = \left(1 - \frac{1}{\kappa}\right)\theta_{t,1}.$$

The same scalar stepsize must accommodate both directions. Backtracking can discover that $1/\beta$ is safe, but discovering the safe value does not change the recursion $\theta_{t+1,1} = (1 - 1/\kappa)\theta_{t,1}$ in the flatter direction.

Proposition 12.3.1 (Best constant step on a quadratic)**.** *For Eq. (12.12), define*

$$q(\eta) = \max_{\lambda \in [\mu, \beta]} |1 - \eta\lambda|. \tag{12.16}$$

For every H satisfying $\mu I \preceq H \preceq \beta I$, gradient descent satisfies

$$\|e_{t+1}\|_2 \le q(\eta)\|e_t\|_2. \tag{12.17}$$

The constant step

$$\eta_{\text{opt}} = \frac{2}{\beta + \mu} \tag{12.18}$$

minimizes $q(\eta)$ over $\eta > 0$:

$$\min_{\eta > 0} q(\eta) = q(\eta_{\text{opt}}) = \frac{\kappa - 1}{\kappa + 1}. \tag{12.19}$$

Consequently,

$$\|e_t\|_2 \le \left(\frac{\kappa - 1}{\kappa + 1}\right)^t \|e_0\|_2. \tag{12.20}$$

If $e_0 \neq 0$ and $He_0 = \lambda e_0$ for some $\lambda \in \{\mu, \beta\}$, then equality holds in Eq. (12.20). Hence, when $\kappa > 1$, for every fixed $c \in (0, 1)$ and every such pair (H, e_0),

$$\min\{t \ge 0 : \|e_t\|_2 \le c\|e_0\|_2\} = \left\lceil \frac{\log(1/c)}{\log((\kappa + 1)/(\kappa - 1))} \right\rceil = \Theta(\kappa) \qquad (\kappa \to \infty). \tag{12.21}$$

Proof. Diagonalizing H in Eq. (12.13) gives

$$\|e_{t+1}\|_2 \le \max_j |1 - \eta\lambda_j| \, \|e_t\|_2 \le q(\eta)\|e_t\|_2,$$

which proves Eq. (12.17). For fixed $\eta > 0$, the function $\lambda \mapsto |1 - \eta\lambda|$ is convex, so its maximum on $[\mu, \beta]$ occurs at $\lambda = \mu$ or $\lambda = \beta$. Expanding the two absolute values gives four affine functions. The inequalities $1 - \eta\beta \le 1 - \eta\mu$ and $\eta\mu - 1 \le \eta\beta - 1$ remove the two dominated functions, so $q(\eta) = \max\{1 - \eta\mu, \eta\beta - 1\}$. The first function decreases with η, the second increases with η, and they are equal exactly at $\eta = \eta_{\text{opt}} = 2/(\beta + \mu)$. Consequently,

$$q(\eta) = \begin{cases} 1 - \eta\mu, & 0 < \eta \le \eta_{\text{opt}}, \\ \eta\beta - 1, & \eta \ge \eta_{\text{opt}}. \end{cases}$$

The maximum therefore decreases up to η_{opt} and increases after η_{opt}, so its minimum occurs at η_{opt}. Substitution gives Eq. (12.19). Iterating Eq. (12.17) proves Eq. (12.20). If $He_0 = \lambda e_0$ for $\lambda \in \{\mu, \beta\}$, then $e_t = (1 - \eta_{\text{opt}}\lambda)^t e_0$. For $\lambda = \mu$, the multiplier is $(\kappa - 1)/(\kappa + 1)$; for $\lambda = \beta$, the multiplier is $-(\kappa - 1)/(\kappa + 1)$. Hence $\|e_t\|_2 = ((\kappa - 1)/(\kappa + 1))^t \|e_0\|_2$ in both cases. The sign alternates when $\lambda = \beta$, but the norm is unchanged. Thus equality holds in Eq. (12.20). Solving $q(\eta_{\text{opt}})^t \le c$ gives the exact iteration count in Eq. (12.21). The $\Theta(\kappa)$ relation follows from $\log((\kappa + 1)/(\kappa - 1)) = 2/\kappa + O(1/\kappa^2)$. □

The error after t steps has the form

$$e_t = p_t(H)e_0, \qquad p_t(\lambda) = (1 - \eta\lambda)^t.$$

A method that combines past iterates can generate a degree-t polynomial with a smaller maximum magnitude on $[\mu, \beta]$. Acceleration chooses such a polynomial without computing an eigendecomposition.

The polynomial formulation also explains why acceleration concerns more than a larger stepsize. Repeated gradient descent applies the same linear factor to every eigenvalue at every iteration. A method with memory can choose the coefficients of a degree-t polynomial to reduce $\max_{\lambda\in[\mu,\beta]} |p_t(\lambda)|$. The iterates implement the recurrence for the chosen polynomial using gradients and a small amount of stored state.

12.4 Reducing the condition-number dependence with acceleration

This section asks whether a first-order method can make faster progress in low-curvature directions without taking an unstable gradient step in high-curvature directions. The backtracking procedure used smoothness, and its linear-rate theorem added the PL inequality without convexity. The direct acceleration argument below narrows the setting. Throughout the derivation, L is β-smooth and μ-strongly convex, and $\theta^\star$ is its unique minimizer. Acceleration retains the preceding displacement and reduces the iteration-count dependence from κ to $\sqrt{\kappa}$.

The design begins by deciding what the method may store and what quantity the analysis should decrease. Gradient descent stores one parameter vector. We permit one additional parameter-sized vector $v_t \in \mathbb{R}^d$ and treat (θ_t, v_t) as the state. Thus the method uses twice the parameter memory of gradient descent, apart from the gradient and temporary vectors. For an undetermined coefficient $\nu > 0$, consider the Lyapunov function

$$V(\theta, v) = L(\theta) - L^\star + \frac{\nu}{2}\|v - \theta^\star\|_2^2. \tag{12.22}$$

The objective gap is the quantity that we want to control. The second term allows the analysis to record progress that is stored in the auxiliary state. Since

$$L(\theta) - L^\star \le V(\theta, v) \qquad \text{for every } \theta, v \in \mathbb{R}^d, \tag{12.23}$$

a geometric bound on V immediately gives the same bound on the objective gap. We therefore seek an update and a number $\tau \in (0, 1]$ such that

$$V(\theta_{t+1}, v_{t+1}) \le (1 - \tau)V(\theta_t, v_t). \tag{12.24}$$

We search within a four-parameter update family. Requiring the intermediate point to equal the common stored vector when $\theta_t = v_t$ suggests an affine combination of θ_t and v_t. The parameter update takes a gradient step from that point, and the auxiliary-state update combines the two state vectors affinely and adds a gradient correction. A general linear two-state update would introduce more coefficients. We use the restricted family below with $b \in (0, 1]$, so the state part of the v_t update is a convex combination. The verification will determine a, b, c, η, and the potential coefficient ν:

$$\begin{aligned} y_t &= (1 - a)\theta_t + av_t, \\ \theta_{t+1} &= y_t - \eta g_t, \qquad g_t = \nabla L(y_t), \\ v_{t+1} &= (1 - b)v_t + by_t + cg_t. \end{aligned} \tag{12.25}$$

We now substitute the update directly into the next potential. Smoothness gives

$$\begin{aligned} V(\theta_{t+1}, v_{t+1}) &= L(\theta_{t+1}) - L^\star + \frac{\nu}{2}\|v_{t+1} - \theta^\star\|_2^2 \\ &\le L(y_t) - L^\star - \left(\eta - \frac{\beta\eta^2}{2}\right)\|g_t\|_2^2 \\ &\quad + \frac{\nu}{2}\,\|(1-b)(v_t - \theta^\star) + b(y_t - \theta^\star) + cg_t\|_2^2\,. \end{aligned} \tag{12.26}$$

The coefficient subtracted from $\|g_t\|_2^2$ is largest at $\eta = 1/\beta$. With that choice, expanding the squared norm gives

$$\begin{aligned} V(\theta_{t+1}, v_{t+1}) \le L(y_t) - L^\star + \frac{\nu}{2}\,\|(1-b)(v_t - \theta^\star) + b(y_t - \theta^\star)\|_2^2 \\ + \nu c\,\langle g_t, (1-b)(v_t - \theta^\star) + b(y_t - \theta^\star)\rangle + \left(\frac{\nu c^2}{2} - \frac{1}{2\beta}\right)\|g_t\|_2^2. \end{aligned} \tag{12.27}$$

The coefficient b in the auxiliary-state update suggests trying $\tau = b$ in the contraction target Eq. (12.24). The state part of the update satisfies

$$\|(1-b)(v_t - \theta^\star) + b(y_t - \theta^\star)\|_2^2 = (1-b)\|v_t - \theta^\star\|_2^2 + b\|y_t - \theta^\star\|_2^2 - b(1-b)\|v_t - y_t\|_2^2. \tag{12.28}$$

The identity assigns weight $1-b$ to the squared distance involving v_t and weight b to the squared distance involving y_t. To use the same split for the objective gap, fix $0 < \nu \le \mu$ and apply strong convexity to its b-weighted part:

$$b\big(L(y_t) - L^\star\big) \le b\langle g_t, y_t - \theta^\star\rangle - \frac{b\nu}{2}\|y_t - \theta^\star\|_2^2. \tag{12.29}$$

Combining Eqs. (12.27) to (12.29) then gives

$$\begin{aligned} V(\theta_{t+1}, v_{t+1}) \le (1-b)\left(L(y_t) - L^\star + \frac{\nu}{2}\|v_t - \theta^\star\|_2^2 - \frac{b\nu}{2}\|v_t - y_t\|_2^2\right) \\ + b\langle g_t, y_t - \theta^\star\rangle + \nu c\,\langle g_t, (1-b)(v_t - \theta^\star) + b(y_t - \theta^\star)\rangle \\ + \left(\frac{\nu c^2}{2} - \frac{1}{2\beta}\right)\|g_t\|_2^2. \end{aligned} \tag{12.30}$$

The first line has the proposed factor $1-b$. The next two lines determine the remaining coefficient choices.

At this point, the design has imposed

$$\eta = \frac{1}{\beta}, \qquad \tau = b, \qquad 0 < \nu \le \mu.$$

The values of a, b, c, and ν remain to be selected within these restrictions. To eliminate $\theta^\star$ from the two inner products in Eq. (12.30) and produce another factor $1-b$, set

$$\nu c = -b \qquad \Longleftrightarrow \qquad c = -\frac{b}{\nu}. \tag{12.31}$$

Indeed, the two inner products then satisfy

$$\begin{aligned} b\langle g_t, y_t - \theta^\star\rangle - b\,\langle g_t, (1-b)(v_t - \theta^\star) + b(y_t - \theta^\star)\rangle \\ = -b(1-b)\langle g_t, v_t - y_t\rangle. \end{aligned} \tag{12.32}$$

Substituting Eqs. (12.31) and (12.32) into Eq. (12.30) gives

$$V(\theta_{t+1}, v_{t+1}) \le (1-b)\left(L(y_t) - L^\star + \frac{\nu}{2}\|v_t - \theta^\star\|_2^2 - \frac{b\nu}{2}\|v_t - y_t\|_2^2 - b\langle g_t, v_t - y_t\rangle\right) + \left(\frac{b^2}{2\nu} - \frac{1}{2\beta}\right)\|g_t\|_2^2. \tag{12.33}$$

The squared-gradient term in Eq. (12.33) is nonpositive when $b^2 \le \nu/\beta$. Under this condition,

$$V(\theta_{t+1}, v_{t+1}) \le (1-b)\left(L(y_t) - L^\star + \frac{\nu}{2}\|v_t - \theta^\star\|_2^2 - \frac{b\nu}{2}\|v_t - y_t\|_2^2 - b\langle g_t, v_t - y_t\rangle\right). \tag{12.34}$$

Add and subtract $\langle g_t, \theta_t - y_t\rangle$ in Eq. (12.34). Convexity gives $L(y_t) + \langle g_t, \theta_t - y_t\rangle \le L(\theta_t)$, and hence

$$V(\theta_{t+1}, v_{t+1}) \le (1-b)\left(V(\theta_t, v_t) - \frac{b\nu}{2}\|v_t - y_t\|_2^2 + \langle g_t, b(y_t - v_t) - (\theta_t - y_t)\rangle\right). \tag{12.35}$$

Choose the intermediate point so that the last inner product vanishes:

$$\theta_t - y_t = b(y_t - v_t). \tag{12.36}$$

For $y_t = (1-a)\theta_t + a v_t$,

$$\begin{aligned} \theta_t - y_t &= a(\theta_t - v_t), \\ b(y_t - v_t) &= b(1-a)(\theta_t - v_t). \end{aligned} \tag{12.37}$$

Thus Eq. (12.36) determines

$$a = b(1-a) \qquad \Longleftrightarrow \qquad a = \frac{b}{1+b}. \tag{12.38}$$

Using Eq. (12.36) in Eq. (12.35) gives

$$V(\theta_{t+1}, v_{t+1}) \le (1-b)\left(V(\theta_t, v_t) - \frac{\nu}{2b}\|\theta_t - y_t\|_2^2\right) \le (1-b)V(\theta_t, v_t). \tag{12.39}$$

The contraction factor is $1-b$, subject to $b^2 \le \nu/\beta$ and $0 < \nu \le \mu$. The largest admissible b is obtained from

$$\nu = \mu, \qquad b = \tau = \sqrt{\frac{\mu}{\beta}} = \frac{1}{\sqrt{\kappa}}. \tag{12.40}$$

This choice proves the target Eq. (12.24). The completed coefficient choices give the coupled updates

$$y_t = \frac{\theta_t + \tau v_t}{1+\tau}, \tag{12.41}$$

$$\theta_{t+1} = y_t - \frac{1}{\beta}\nabla L(y_t), \tag{12.42}$$

$$v_{t+1} = (1-\tau)v_t + \tau\left(y_t - \frac{1}{\mu}\nabla L(y_t)\right). \tag{12.43}$$

The auxiliary state v_t was useful for choosing the coefficients and proving the contraction. Eliminating v_t gives an equivalent implementation that stores the two most recent parameter iterates. Set $\theta_{-1} = \theta_0$ and

$$\gamma = \frac{1-\tau}{1+\tau} = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}. \tag{12.44}$$

The coupled update is equivalent to the two-step recursion

$$y_t = \theta_t + \gamma(\theta_t - \theta_{t-1}), \tag{12.45}$$

$$\theta_{t+1} = y_t - \frac{1}{\beta}\nabla L(y_t). \tag{12.46}$$

This two-step recursion is the usual form of Nesterov's accelerated gradient method. Exercise 2 asks you to verify the equivalence and complete the proof of the accelerated rate.

The point y_t extrapolates the displacement $\theta_t - \theta_{t-1}$, and the second line takes an ordinary gradient step from the extrapolated point. The update evaluates one gradient and stores two parameter vectors per iteration. The derivation illustrates the general strategy: specify the desired progress measure, introduce state that can retain the missing information, and choose the coefficients so that the available inequalities prove the desired one-step change. Eliminating the auxiliary state then gives a simpler implementation.

Theorem 12.4.1 (Accelerated linear convergence)**.** *Suppose that L is β-smooth and μ-strongly convex, let $\kappa = \beta/\mu$, and let $\theta^\star$ be its minimizer. Set γ as in Eq. (12.44) and initialize $\theta_{-1} = \theta_0$. The iterates defined by Eqs. (12.45) and (12.46) satisfy*

$$L(\theta_t) - L^\star \le 2\left(1 - \frac{1}{\sqrt{\kappa}}\right)^t \big(L(\theta_0) - L^\star\big). \tag{12.47}$$

Consequently, for $0 < \epsilon < 2(L(\theta_0) - L^\star)$, $L(\theta_t) - L^\star \le \epsilon$ whenever

$$t \ge \sqrt{\kappa}\log\frac{2(L(\theta_0) - L^\star)}{\epsilon}. \tag{12.48}$$

The proof is completed in Exercise 2. The preceding calculation supplies the potential contraction Eq. (12.39); the exercise eliminates v_t, bounds the initial potential using strong convexity, and derives Eqs. (12.47) and (12.48).

Quadratic interpretation. The quadratic calculation explains how the resulting method acts in different curvature directions. Let q be an eigenvector of H with eigenvalue λ, and write

$$e_t(\lambda) = \langle q, \theta_t - \theta^\star\rangle.$$

For the quadratic objective in Eq. (12.12), Eqs. (12.45) and (12.46) give

$$e_{t+1}(\lambda) = \left(1 - \frac{\lambda}{\beta}\right)\big((1+\gamma)e_t(\lambda) - \gamma e_{t-1}(\lambda)\big). \tag{12.49}$$

Gradient descent uses a first-order recurrence in each eigendirection; accelerated gradient uses the second-order recurrence Eq. (12.49). The calculation explains the role of memory after the method has been derived. The related Chebyshev construction in Section 12.9.3 solves the exact finite-degree minimax polynomial problem for quadratics.

Nesterov and heavy-ball momentum. Nesterov's method evaluates the gradient at the extrapolated point y_t. Polyak's heavy-ball method instead uses

$$\theta_{t+1} = \theta_t - \eta_{\mathrm{HB}}\nabla L(\theta_t) + \gamma_{\mathrm{HB}}(\theta_t - \theta_{t-1}).$$

For a quadratic with spectrum in $[\mu, \beta]$, the quadratic-optimal fixed parameters and the resulting worst-case asymptotic spectral factor are

$$\eta_{\mathrm{HB}} = \frac{4}{(\sqrt{\beta} + \sqrt{\mu})^2}, \qquad \gamma_{\mathrm{HB}} = r^2, \qquad r = \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} = 1 - \Theta\left(\frac{1}{\sqrt{\kappa}}\right).$$

Thus heavy ball has accelerated $\sqrt{\kappa}$ dependence on strongly convex quadratics. The quadratic guarantee is not uniform over all smooth, strongly convex objectives: the same parameters can fail to converge on a nonquadratic objective. By contrast, Theorem 12.4.1 gives Nesterov's guarantee over the entire smooth, strongly convex class. Section 12.9.3 shows that Chebyshev semi-iteration has the heavy-ball form with coefficients that vary with t and converge to η_{HB} and γ_{HB}.

The gradient-evaluation comparison is

$$\underbrace{O\left(\kappa \log \frac{L(\theta_0) - L^\star}{\epsilon}\right)}_{\text{gradient descent}} \qquad \text{and} \qquad \underbrace{O\left(\sqrt{\kappa} \log \frac{L(\theta_0) - L^\star}{\epsilon}\right)}_{\text{acceleration}}. \tag{12.50}$$

Both methods require one gradient evaluation per iteration. Exercise 3 asks you to derive the same square-root dependence by restarting a smooth convex accelerated method.

The reduction from κ to $\sqrt{\kappa}$ in the worst-case number of first-order oracle calls is optimal in the standard first-order oracle model. In this model, the method queries a parameter θ and receives $L(\theta)$ and $\nabla L(\theta)$; each new query may depend on the preceding queries and replies. The lower bounds also explain why the assumptions must be stated carefully.

Proposition 12.4.2 (Relevant first-order lower bounds)**.** *Fix $\kappa > 1$ and consider dimensions large enough relative to the number of oracle calls. Let N_ϵ denote the number of oracle calls required to produce θ satisfying*

$$L(\theta) - L^\star \le \epsilon\big(L(\theta_0) - L^\star\big). \tag{12.51}$$

(a) For every deterministic first-order method, there is a smooth, strongly convex quadratic instance on which

$$N_\epsilon = \Omega\left(\sqrt{\kappa} \log \frac{1}{\epsilon}\right). \tag{12.52}$$

(b) For the larger class of smooth objectives satisfying the PL inequality, without a convexity assumption, every deterministic first-order method has an instance on which

$$N_\epsilon = \Omega\left(\kappa \log \frac{1}{\epsilon}\right). \tag{12.53}$$

Part (a) shows that the square-root dependence on κ is optimal over the smooth, strongly convex class. Part (b) concerns the general smooth PL class, which includes nonconvex objectives, and rules out a general $O\big(\big(\sqrt{\kappa}\log(1/\epsilon)\big)$ guarantee over the full smooth PL class. Smooth convex PL objectives form an intermediate class: the restart construction below attains the square-root dependence for that class.

The smooth convex $O(1/t^2)$ result is a different acceleration theorem. The main result here is the linear convergence bound Eq. (12.47); the smooth convex result and a restart argument are summarized in Section 12.9.2.

12.4.1 Accelerating when the condition number is unknown

The fixed parameters in Eqs. (12.44) and (12.46) use both β and μ. If certified bounds $B \geq \beta$ and $0 < m \leq \mu$ are available, one may use B and m in their place; the resulting guarantee depends on B/m. We now separate the two kinds of missing information.

First suppose that μ is known and β is unknown. Section 12.2 searched for a curvature estimate by testing the smooth quadratic upper bound at a proposed gradient step. The same test can be applied at the extrapolated point y_t. A trial curvature estimate $M > 0$ defines

$$\theta^+(M) = y_t - \frac{1}{M}\nabla L(y_t) \tag{12.54}$$

and is accepted when

$$L(\theta^+(M)) \leq L(y_t) + \langle \nabla L(y_t), \theta^+(M) - y_t \rangle + \frac{M}{2}\|\theta^+(M) - y_t\|_2^2. \tag{12.55}$$

The acceptance inequality is the smooth quadratic upper bound with M in place of β, evaluated only at y_t and $\theta^+(M)$. Every $M \geq \beta$ passes the test. After a rejection, replace M by ρM for a fixed $\rho > 1$ and test the shorter step. The search therefore terminates after finitely many objective evaluations.

Acceleration links the gradient step and the momentum coefficient in the same potential inequality. An *accelerated backtracking* method supplies the accepted curvature estimate to both parts of the accelerated update. With μ known and exact objective values available, standard accelerated backtracking schemes attain

$$N_\epsilon = O\left(\sqrt{\frac{\beta}{\mu}}\log\frac{L(\theta_0) - L^\star}{\epsilon}\right) \tag{12.56}$$

without receiving β as input. Changing the gradient-step denominator while keeping γ fixed leaves the potential calculation in Eq. (12.39) unjustified; accelerated backtracking recomputes the coupling coefficients when the curvature estimate changes.

Unknown μ requires a different procedure because the momentum coefficient cannot be obtained from the smoothness test. Suppose now that L is convex, its minimum is attained, and it is β-smooth and μ-PL for an unknown $\mu > 0$. Write $\Theta^\star = \arg\min L$ and define

$$\operatorname{dist}(z, \Theta^\star) = \inf_{\theta^\star \in \Theta^\star} \|z - \theta^\star\|_2. \tag{12.57}$$

The PL inequality implies the quadratic-growth bound

$$L(z) - L^\star \geq \frac{\mu}{2}\operatorname{dist}(z, \Theta^\star)^2. \tag{12.58}$$

Run the smooth-convex accelerated method for s iterations and then set both stored parameter iterates equal to the current parameter before continuing. The inner method can use a known β or the accelerated backtracking procedure above. One such run between resets is a *restart epoch*. If one epoch uses s gradients, Eq. (12.128) gives

$$L(\mathsf{A}_s(z)) - L^\star \leq \frac{4\kappa}{(s+1)^2}\big(L(z) - L^\star\big). \tag{12.59}$$

Thus an epoch length $s = \Theta(\sqrt{\kappa})$ reduces the objective gap by a constant factor, and repeated restarts attain

$$N_\epsilon = O\left(\sqrt{\kappa}\log\frac{L(\theta_0) - L^\star}{\epsilon}\right). \tag{12.60}$$

When κ is unknown, an adaptive restart rule searches for an epoch length on the scale $\sqrt{\kappa}$. For example, if $L^\star$ is available, one can try epoch lengths $1, 2, 4, \ldots$ from the same starting point until the observed objective gap decreases by one half. The successful search uses $O(\sqrt{\kappa})$ gradients, after which the accepted epoch length can be reused. Exercise 3 asks you to prove this claim. Choosing the epoch length without either μ or $L^\star$ requires a different test; the bibliographical notes cite provable rules and common practical heuristics. The optional reading derives the smooth-convex Lyapunov bound and the restart calculation.

12.5 Preconditioning changes the geometry

This section studies a different response to poor conditioning. Backtracking chooses one scalar step, whereas preconditioning changes the relative scale and direction of the parameter coordinates. A *preconditioned gradient update* uses a symmetric positive-definite matrix P:

$$\theta_{t+1} = \theta_t - \eta P \nabla L(\theta_t). \tag{12.61}$$

The matrix can rescale coordinates and mix them. The purpose of P is to make the objective better conditioned in the geometry used by the update.

Matrix preconditioning introduces a second meaning of adaptation. A line search changes one scalar according to the objective decrease observed at the current point. A preconditioner changes the relative scale and direction of the coordinates. The difference between scalar line search and matrix preconditioning matters in a large model: one scalar can control overall stability, while a matrix can assign substantially different effective stepsizes to different parameter directions.

Proposition 12.5.1 (Preconditioning as a change of coordinates)**.** *Let $\theta = P^{1/2} z$ and define $\widetilde{L}(z) = L(P^{1/2} z)$. If*

$$z_{t+1} = z_t - \eta \nabla \widetilde{L}(z_t), \tag{12.62}$$

then $\theta_t = P^{1/2} z_t$ satisfies

$$\theta_{t+1} = \theta_t - \eta P \nabla L(\theta_t). \tag{12.63}$$

If L is quadratic with Hessian H, then the Hessian in the z coordinates is

$$\widetilde{H} = P^{1/2} H P^{1/2}. \tag{12.64}$$

Proof. The chain rule gives $\nabla \widetilde{L}(z) = P^{1/2} \nabla L(P^{1/2} z)$. Thus

$$z_{t+1} = z_t - \eta \nabla \widetilde{L}(z_t)$$

maps, after multiplication by $P^{1/2}$, to

$$\theta_{t+1} = \theta_t - \eta P \nabla L(\theta_t).$$

Applying the chain rule twice gives Eq. (12.64). □

For the quadratic in Eq. (12.12), the ideal choice $P = H^{-1}$ gives $\widetilde{H} = I$ and condition number one. A dense global matrix, however, requires $\Theta(d^2)$ storage and arithmetic per matrix–vector product for a model with d parameters. Such a cost is not a realistic per-iteration option at modern model dimensions. A whole-model preconditioner must normally cost $O(d)$, or at most about $O(d \log d)$, to store and apply. More expensive factorizations can be useful only on small blocks or when the factorization cost is amortized over many gradient steps.

Restricting the cost also restricts the matrices that can be represented. A diagonal preconditioner costs $O(d)$ but cannot represent any cross-coordinate interaction. A block-diagonal preconditioner with blocks of size at most b costs $O(db)$ to store and apply; the block-diagonal form discards interactions between blocks. Factoring all such blocks can cost $O(db^2)$ when the factors are refreshed. Low-rank and Kronecker-factored forms retain other selected interactions, but neither type can represent an arbitrary dense inverse Hessian. If the discarded interactions contain the poorly conditioned directions, no general condition-number improvement follows. Exercise 4 gives a two-dimensional example in which diagonal Hessian scaling leaves the condition number unchanged while full preconditioning reduces it to one.

A common neural-network example makes the approximation explicit. The Fisher information matrix is the expected outer product of log-likelihood gradients. For a fully connected layer with weight matrix W, a per-example gradient has the outer-product form

$$\nabla_W \ell = \delta a^\top,$$

where a is the incoming activation and δ is the back-propagated derivative at the layer output. The corresponding Fisher block contains

$$\mathbb{E}\big[(aa^\top) \otimes (\delta\delta^\top)\big]. \tag{12.65}$$

Here $\otimes$ denotes the Kronecker product. A Kronecker-factored approximation replaces the expectation in Eq. (12.65) by

$$\mathbb{E}[aa^\top] \otimes \mathbb{E}[\delta\delta^\top]. \tag{12.66}$$

The factorization discards dependence between activations and back-propagated derivatives, and a layerwise block approximation also discards cross-layer interactions. Rigorous convergence results exist for restricted overparameterized or infinite-width network models, but no general theorem says that a low-rank or Kronecker-factored preconditioner improves the conditioning of a finite multilayer network.

Preconditioning also changes solution selection when the minimizer is not unique. Lecture 10 showed that the optimization geometry can determine which interpolating or separating solution is approached. Better conditioning and changed implicit bias are therefore separate consequences of the same matrix update.

12.6 AdaGrad adapts to observed gradient geometry

Backtracking adapts to curvature by evaluating the objective at trial points. That test is practical when reliable objective values are inexpensive. In large-scale minibatch training, evaluating the full empirical objective requires a pass over the data, and every trial on the current minibatch requires another forward pass. Trials evaluated on different minibatches also produce noisy values for different functions. These costs motivate adaptive rules that use the first-order observations already computed by the training procedure.

AdaGrad gives a clean example of such a rule. Its guarantee applies to subgradients of convex losses, including losses with no smoothness constant β. This section first studies scalar adaptation in online convex optimization: the method nearly matches the bound obtained by choosing the best constant stepsize after all gradients have been observed, without knowing the horizon or the future gradient norms. AdaGrad then applies the same idea coordinate by coordinate. The coordinatewise refinement can improve the scalar bound when gradient activity is sparse or uneven.

Scope of the online model. The online model supplies the historical and mathematically clean setting for AdaGrad's adaptive subgradient guarantee. The regret theorem gives an exact comparison with a stepsize selected after the gradient sequence is known. Its formal scope is online convex optimization. Under an i.i.d. sampling model with $L(x) = \mathbb{E}[\ell_t(x)]$, convexity and the averaged decision $\overline{x}_T = T^{-1}\sum_{t=1}^{T} x_t$ give the online-to-batch relation

$$\mathbb{E}[L(\overline{x}_T) - L(u)] \le \frac{1}{T}\mathbb{E}[\mathrm{Reg}_T(u)]. \tag{12.67}$$

Language-model training is a nonconvex stochastic problem outside that scope. Here the theorem identifies the origin and precise meaning of AdaGrad's adaptive update.

The online problem supplies a sequence of convex losses instead of the one fixed objective L used above. To mark the change of setting, we write x_t for the online decisions. At round t, the learner chooses x_t before seeing the current loss ℓ_t and then receives a subgradient $g_t \in \partial\ell_t(x_t)$. Performance is compared with one fixed decision u chosen after the whole sequence is known. The regret is

$$\mathrm{Reg}_T(u) = \sum_{t=1}^{T}\big(\ell_t(x_t) - \ell_t(u)\big). \tag{12.68}$$

A sublinear regret bound means that, for every fixed comparator u,

$$\mathrm{Reg}_T(u) = o(T) \quad\Longleftrightarrow\quad \frac{1}{T}\sum_{t=1}^{T}\big(\ell_t(x_t) - \ell_t(u)\big) \longrightarrow 0. \tag{12.69}$$

The guarantee is deterministic and allows the loss sequence to be adversarial; no sampling model is required.

Let $\mathcal{X}$ be a closed convex set with Euclidean diameter at most $D > 0$. Projected online gradient descent with a constant stepsize $\eta > 0$ satisfies

$$\mathrm{Reg}_T(u) \le \frac{D^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\|g_t\|_2^2. \tag{12.70}$$

If the entire sequence were known before the run and at least one gradient were nonzero, the right side would be minimized by

$$\eta^\star = \frac{D}{\sqrt{\sum_{t=1}^{T}\|g_t\|_2^2}}. \tag{12.71}$$

The best value of the constant-stepsize upper bound is

$$\inf_{\eta>0}\left\{\frac{D^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\|g_t\|_2^2\right\} = D\left(\sum_{t=1}^{T}\|g_t\|_2^2\right)^{1/2}. \tag{12.72}$$

The tuned stepsize depends on future gradients and therefore cannot be supplied to the online algorithm.

Theorem 12.6.1 (Scalar adaptation to the observed gradient scale)**.** *Define*

$$S_t = \left(\sum_{s=1}^{t}\|g_s\|_2^2\right)^{1/2}$$

and use the projected update

$$x_{t+1} = \Pi_{\mathcal{X}}\left(x_t - \frac{D}{S_t} g_t\right), \tag{12.73}$$

with no change when $S_t = 0$. Then, for every $u \in \mathcal{X}$,

$$\mathrm{Reg}_T(u) \le \frac{3}{2} D \left(\sum_{t=1}^{T} \|g_t\|_2^2\right)^{1/2}. \tag{12.74}$$

Proof. If $S_T = 0$, then every $g_t = 0$, and convexity gives $\ell_t(x_t) - \ell_t(u) \le 0$ on every round. Suppose therefore that $S_T > 0$, and let t_0 be the first index for which $S_{t_0} > 0$. Rounds before t_0 again contribute nonpositive regret. For $t \ge t_0$, set $\eta_t = D/S_t$. Since $u \in \mathcal{X}$, the projection inequality gives

$$\|x_{t+1} - u\|_2^2 \le \|x_t - \eta_t g_t - u\|_2^2 = \|x_t - u\|_2^2 - 2\eta_t \langle g_t, x_t - u\rangle + \eta_t^2 \|g_t\|_2^2. \tag{12.75}$$

Rearranging Eq. (12.75) yields

$$\langle g_t, x_t - u\rangle \le \frac{\|x_t - u\|_2^2 - \|x_{t+1} - u\|_2^2}{2\eta_t} + \frac{\eta_t}{2}\|g_t\|_2^2. \tag{12.76}$$

Convexity gives

$$\ell_t(x_t) - \ell_t(u) \le \langle g_t, x_t - u\rangle.$$

Combining the last two inequalities gives the one-step bound

$$\ell_t(x_t) - \ell_t(u) \le \frac{\|x_t - u\|_2^2 - \|x_{t+1} - u\|_2^2}{2\eta_t} + \frac{\eta_t}{2}\|g_t\|_2^2.$$

We next sum the distance terms. Summation by parts is the discrete analogue of integration by parts. The mnemonic

$$\int Fg = FG - \int fG, \qquad F' = f, \quad G' = g,$$

says to move the derivative—or, for a sum, the finite difference—from one factor to the other and retain the boundary terms. Put $A_t = \|x_t - u\|_2^2$. Directly expanding the finite sum gives

$$\sum_{t=t_0}^{T} \frac{A_t - A_{t+1}}{2\eta_t} = \frac{A_{t_0}}{2\eta_{t_0}} + \sum_{t=t_0+1}^{T} \left(\frac{1}{2\eta_t} - \frac{1}{2\eta_{t-1}}\right) A_t - \frac{A_{T+1}}{2\eta_T}. \tag{12.77}$$

The sequence S_t is nondecreasing, so $1/\eta_t = S_t/D$ is also nondecreasing. Every coefficient in the middle sum of Eq. (12.77) is therefore nonnegative. Using $A_t \le D^2$ and discarding the final nonpositive term yields

$$\sum_{t=t_0}^{T} \frac{A_t - A_{t+1}}{2\eta_t} \le D^2 \left[\frac{1}{2\eta_{t_0}} + \sum_{t=t_0+1}^{T} \left(\frac{1}{2\eta_t} - \frac{1}{2\eta_{t-1}}\right)\right] = \frac{D^2}{2\eta_T} = \frac{DS_T}{2}. \tag{12.78}$$

For the gradient terms, $S_t^2 - S_{t-1}^2 = \|g_t\|_2^2$ and hence

$$\sum_{t=t_0}^{T} \frac{\|g_t\|_2^2}{S_t} = \sum_{t=t_0}^{T} (S_t - S_{t-1}) \frac{S_t + S_{t-1}}{S_t} \le 2 \sum_{t=t_0}^{T} (S_t - S_{t-1}) = 2S_T. \tag{12.79}$$

Multiplying Eq. (12.79) by $D/2$ bounds the gradient sum in the one-step inequality by DS_T. Combining the two sums proves Eq. (12.74). □

Combining Eqs. (12.72) and (12.74) gives

$$\mathrm{Reg}_T(u) \le \frac{3}{2} \inf_{\eta>0} \left\{ \frac{D^2}{2\eta} + \frac{\eta}{2} \sum_{t=1}^{T} \|g_t\|_2^2 \right\}. \tag{12.80}$$

Thus the adaptive guarantee is within a factor $3/2$ of the best value of the constant-stepsize upper bound Eq. (12.70). The procedure does not need the horizon or a bound on the gradient norms, but the procedure still uses the domain diameter D. Adaptation is always relative to a specified set of unknown and known quantities.

Diagonal AdaGrad applies the same argument separately to each coordinate. Let $\mathcal{X} = \prod_{i=1}^{d} [a_i, b_i]$, with coordinate diameter $D_i = b_i - a_i$.

Define the accumulated coordinate magnitude

$$G_{t,i} = \left(\delta^2 + \sum_{s=1}^{t} g_{s,i}^2 \right)^{1/2}, \qquad \delta \ge 0. \tag{12.81}$$

Diagonal AdaGrad uses

$$x_{t+1,i} = \Pi_{[a_i,b_i]} \left(x_{t,i} - \frac{D_i}{G_{t,i}} g_{t,i} \right). \tag{12.82}$$

When $\delta = 0$ and $G_{t,i} = 0$, the coordinate is left unchanged. A small positive δ is commonly used to avoid division by zero.

Theorem 12.6.2 (Data-dependent regret of diagonal AdaGrad)**.** *With $\delta = 0$, the iterates in Eq. (12.82) satisfy, for every $u \in \mathcal{X}$,*

$$\sum_{t=1}^{T} \big(\ell_t(x_t) - \ell_t(u) \big) \le \frac{3}{2} \sum_{i=1}^{d} D_i \left(\sum_{t=1}^{T} g_{t,i}^2 \right)^{1/2}. \tag{12.83}$$

Exercise 5 asks you to prove the theorem. To compare the diagonal and scalar guarantees, define

$$D_2 = \left(\sum_{i=1}^{d} D_i^2 \right)^{1/2}, \qquad S_i = \left(\sum_{t=1}^{T} g_{t,i}^2 \right)^{1/2}.$$

Cauchy–Schwarz gives

$$\sum_{i=1}^{d} D_i S_i \le D_2 \left(\sum_{t=1}^{T} \|g_t\|_2^2 \right)^{1/2}. \tag{12.84}$$

Thus the diagonal AdaGrad bound is never worse than the corresponding scalar adaptive bound with the box's Euclidean diameter. The comparison is between the two upper bounds; a comparison of realized losses requires separate analysis. If all D_i are equal and only k coordinates ever receive gradients of comparable cumulative magnitude, then, in the exactly equal case,

$$\frac{\sum_{i=1}^{d} D_i S_i}{D_2 \big(\sum_{t=1}^{T} \|g_t\|_2^2 \big)^{1/2}} = \sqrt{\frac{k}{d}}. \tag{12.85}$$

The diagonal regret upper bound is therefore smaller by the factor $\sqrt{d/k}$, with the largest factor $\sqrt{d}$ attained when $k = 1$.

The theorem identifies the specific sense in which AdaGrad is adaptive. AdaGrad records the gradient magnitudes observed in each coordinate, and the regret bound depends on the same recorded quantities. A sparse or uneven gradient sequence can therefore yield a smaller comparator bound than scalar adaptation.

The smooth-objective and online analyses use gradient information differently. For one fixed smooth objective, the quadratic upper bound converts a gradient step into objective descent, giving an $O(1/T)$ convex objective-gap bound or geometric objective-gap decay under the PL inequality. In online convex optimization, adversarially changing losses prevent telescoping of objective values from different rounds. The online proof uses a squared-distance recursion and retains the cumulative squared gradients. Under bounded gradients, the theorem gives the usual average-regret bound: if $\|g_t\|_2 \le G$, then

$$\frac{1}{T}\sum_{t=1}^{T}\big(\ell_t(x_t) - \ell_t(u)\big) \le \frac{3DG}{2\sqrt{T}}. \tag{12.86}$$

AdaGrad pays for long-term memory in another way. The denominator $G_{t,i}$ never decreases. A large gradient early in training permanently reduces later steps in that coordinate. Permanent dependence on early gradients is appropriate for the adversarial online regret theorem but can be too conservative when gradient scales change during nonconvex training.

The statistic $G_{t,i}^2$ is a sum of squared gradients. A diagonal entry of the Hessian is a derivative of a gradient coordinate. Particular probabilistic models can relate these quantities, but their definitions and general roles are different. AdaGrad constructs an adaptive diagonal geometry from the observed gradients.

12.7 Adapting to changing gradient scales

AdaGrad retains every past squared gradient in its denominator. A large gradient therefore reduces every later step in the same coordinate, even after the relevant gradient scale has changed. Neural-network gradients can also differ greatly across coordinates and across stages of training. Adam uses recent squared gradients to set coordinatewise scales and uses recent gradients to set the update direction. This section first explains that design and then studies an exact logistic-regression example.

Given a stochastic gradient $g_t \in \mathbb{R}^d$ at θ_t and initial states $m_0 = v_0 = 0$, Adam uses

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \tag{12.87}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)(g_t \odot g_t), \tag{12.88}$$

$$\widehat{m}_t = \frac{m_t}{1-\beta_1^t}, \qquad \widehat{v}_t = \frac{v_t}{1-\beta_2^t}, \tag{12.89}$$

$$\theta_{t+1} = \theta_t - \alpha \frac{\widehat{m}_t}{\sqrt{\widehat{v}_t} + \varepsilon}. \tag{12.90}$$

Products, square roots, and division are coordinatewise. The numerator $\widehat{m}_t$ averages recent gradients and supplies the direction. The denominator supplies one scale per coordinate. Unrolling its recursion gives

$$v_t = (1-\beta_2)\sum_{s=1}^{t} \beta_2^{t-s}(g_s \odot g_s). \tag{12.91}$$

Thus a squared gradient observed $t-s$ iterations ago receives weight β_2^{t-s}. AdaGrad assigns weight one to every past squared gradient; Adam assigns geometrically decreasing weights. The parameters

$\beta_1, \beta_2 \in [0, 1)$ determine the two memory timescales, $\alpha > 0$ is a base stepsize, and $\varepsilon \geq 0$ controls the update when the recent squared gradients are small. When $\varepsilon = 0$, the coordinate i update in Eq. (12.90) is defined whenever $\widehat{v}_{t,i} > 0$; the scalar examples below have a nonzero gradient at every finite iterate.

The factors $1 - \beta_1^t$ and $1 - \beta_2^t$ correct the initialization at zero. For a constant gradient sequence, the corrected quantities equal the corresponding first and squared gradients at every iteration. The quantities α, β_1, β_2, and ε remain algorithmic choices.

RMSProp is the earlier method that uses the same recent-square scaling with the current gradient in the numerator. A common version uses

$$\theta_{t+1} = \theta_t - \alpha \frac{g_t}{\sqrt{v_t} + \varepsilon}, \qquad v_t = \beta_2 v_{t-1} + (1 - \beta_2)(g_t \odot g_t). \tag{12.92}$$

Setting $\beta_1 = 0$ in Adam gives the bias-corrected version of this update. Convergence-to-stationarity guarantees for RMSProp require explicit smoothness, noise, and hyperparameter conditions. Convex counterexamples rule out an unconditional convergence guarantee.

We next examine Adam's behavior on a simple separable logistic problem. For the special case $\beta_1 = \beta_2 = 0$, the calculation gives an exact comparison between the normalized update and constant-stepsize gradient descent, and shows that normalization can produce faster loss decay in this setting.

Example 12.7.1 (Loss decay on a one-dimensional logistic problem)**.** Let

$$L(\theta) = \log(1 + e^{-\theta}), \qquad L'(\theta) = -\frac{1}{1 + e^{\theta}}. \tag{12.93}$$

Constant-stepsize gradient descent satisfies

$$\theta_{t+1} = \theta_t + \frac{\eta}{1 + e^{\theta_t}}. \tag{12.94}$$

For every $\eta > 0$, the iterates obey

$$\lim_{t \to \infty} t L(\theta_t) = \frac{1}{\eta}. \tag{12.95}$$

The normalized update with $\alpha > 0$ gives

$$\theta_{t+1} = \theta_t - \alpha \frac{L'(\theta_t)}{|L'(\theta_t)|} = \theta_t + \alpha, \qquad \lim_{t \to \infty} e^{\alpha t} L(\theta_t) = e^{-\theta_0}. \tag{12.96}$$

Thus the loss decays as $1/t$ for the constant gradient-descent stepsize and exponentially for the normalized update.

To verify Eq. (12.95), set $u_t = e^{\theta_t}$. The iterates in Eq. (12.94) increase without bound, and

$$u_{t+1} - u_t = u_t \left[\exp\left(\frac{\eta}{1 + u_t} \right) - 1 \right] \longrightarrow \eta.$$

The Stolz–Cesàro theorem gives $u_t / t \to \eta$, and $\log(1 + 1/u_t) \sim 1/u_t$ proves Eq. (12.95). The identity $\theta_t = \theta_0 + \alpha t$ proves Eq. (12.96).

The normalized update in the example is the RMSProp and Adam special case $\beta_1 = \beta_2 = 0$ and $\varepsilon = 0$. The calculation proves exponential loss decay for this scalar problem. Comparisons on other objectives require their own analysis.

The same special case gives an exact interpretation of $\varepsilon > 0$. The update becomes

$$\theta_{t+1} = \theta_t + \alpha \frac{|L'(\theta_t)|}{|L'(\theta_t)| + \varepsilon}. \tag{12.97}$$

For $0 < \varepsilon < 1$, the gradient threshold $|L'(\theta)| = \varepsilon$ corresponds exactly to the loss level

$$L(\theta) = -\log(1 - \varepsilon). \tag{12.98}$$

When $|L'(\theta_t)| \gg \varepsilon$, the update in Eq. (12.97) is approximately the normalized update. When $|L'(\theta_t)| \ll \varepsilon$, the update is approximately a gradient step with stepsize α/ε. In fact,

$$\lim_{t\to\infty} tL(\theta_t) = \frac{\varepsilon}{\alpha}. \tag{12.99}$$

To obtain the limit, again set $u_t = e^{\theta_t}$ and observe that $u_{t+1} - u_t \to \alpha/\varepsilon$. Thus $\varepsilon = 0$ preserves exponential decay, whereas every fixed $\varepsilon > 0$ eventually returns to $1/t$ decay after a potentially long normalized phase. For general Adam, the transition occurs when $\sqrt{\widehat{v}_{t,i}}$ is comparable to ε. Each coordinate has its own transition, determined by the recent gradient history.

Exponential forgetting removes the monotonicity used in AdaGrad's regret proof. An old large gradient can cease to constrain the effective step even when that gradient remains important for the optimization objective.

Example 12.7.2 (An extreme forgetting failure)**.** On $[-1, 1]$, repeat the three linear losses

$$\ell_{3k+1}(x) = Cx, \qquad \ell_{3k+2}(x) = \ell_{3k+3}(x) = -x, \qquad C > 2.$$

The three losses sum to $(C-2)x$ over each block, so the best fixed point is $x = -1$. Take the projected Adam special case $\beta_1 = \beta_2 = 0$ and $\varepsilon = 0$, and fix $0 < \alpha \le 2$. For $g_t = \ell_t'(x_t)$, the update is

$$x_{t+1} = \Pi_{[-1,1]}\left(x_t - \alpha \frac{g_t}{|g_t|}\right). \tag{12.100}$$

The update normalizes every nonzero gradient to its sign. Starting from $x = 1$, the iterates used for one block are $1, 1-\alpha, 1$. The next block again begins at 1. Every block therefore incurs the same positive regret. After K blocks,

$$\sum_{t=1}^{3K} \big(\ell_t(x_t) - \ell_t(-1)\big) = K\big(2(C-2) + \alpha\big). \tag{12.101}$$

Exercise 6 asks you to calculate the iterates and the exact block regret.

The example proves an online convex optimization failure:

$$\frac{\mathrm{Reg}_{3K}(-1)}{3K} = \frac{2(C-2)+\alpha}{3} > 0. \tag{12.102}$$

Failures also occur in stochastic convex optimization and in deterministic batch optimization. Separate counterexamples establish the following two statements.

(a) Let Ξ be a random data example. There is a convex loss $\ell(\theta; \Xi)$ for which Adam applied to the stochastic objective

$$L(\theta) = \mathbb{E}[\ell(\theta; \Xi)] \tag{12.103}$$

produces iterates satisfying

$$\operatorname{dist}\big(\theta_t, \arg\min_\theta L(\theta)\big) \not\longrightarrow 0. \tag{12.104}$$

(b) A deterministic batch version of Adam with bias correction has trajectories that approach a nonoptimal period-two limit cycle on the strongly convex quadratic

$$L(\theta) = \frac{c}{2}\theta^2, \qquad c > 0. \tag{12.105}$$

For such a trajectory, there are two distinct states $s_+ = (a, m_+, v_+)$ and $s_- = (-a, m_-, v_-)$ with $a \neq 0$ such that

$$(\theta_{2t}, m_{2t}, v_{2t}) \longrightarrow s_+, \qquad (\theta_{2t+1}, m_{2t+1}, v_{2t+1}) \longrightarrow s_-. \tag{12.106}$$

The deterministic construction uses $\sqrt{v_t + \varepsilon}$ in the denominator; the update in Eq. (12.90) uses $\sqrt{v_t} + \varepsilon$. The construction rules out a global convergence guarantee for the former batch Adam variant. Its conclusion concerns the existence of particular trajectories and leaves behavior from standard initializations and default hyperparameters open. Long-memory variants such as AMSGrad restore a nondecreasing second-moment envelope for the corresponding convex analysis.

Positive convergence theorems for Adam provide basic reassurance under restricted assumptions. An advantage over scalar adaptation or SGD requires a sharper comparison. Recent results identify settings in which the coordinatewise normalization can improve the bound. Examples include objectives with nonuniform smoothness of the schematic form

$$\|\nabla L(u) - \nabla L(v)\|_2 \lesssim \big(L_0 + L_1 \|\nabla L(u)\|_2\big)\|u - v\|_2$$

locally, and problems whose gradients or curvature are favorable in ℓ_∞ or blockwise geometry. These results give problem-dependent comparisons with scalar methods. A universal ordering of Adam and tuned SGD and a hindsight-best diagonal guarantee comparable to AdaGrad's theorem remain unavailable.

Adam remains a practically important optimizer, including for transformer training. Existing theory and empirical evidence support particular mechanisms and regimes. The denominator is an exponential average of squared gradients. Default hyperparameters still require empirical validation, and the available theorems apply to specified objective and noise classes.

12.7.1 Decoupling weight decay from adaptive scaling

The logistic example sends the parameter to infinity along the separating direction. Practical training often includes weight decay to control parameter growth. For an adaptive method, the placement of this shrinkage changes the update. This subsection derives two common updates and then returns to the scalar logistic example.

Suppose an adaptive method uses a positive diagonal matrix D_t and applies an ℓ_2 penalty to the objective:

$$F(\theta) = L(\theta) + \frac{\lambda}{2}\|\theta\|_2^2.$$

The update is

$$\theta_{t+1} = \theta_t - \alpha D_t^{-1}(\nabla L(\theta_t) + \lambda\theta_t) = (I - \alpha\lambda D_t^{-1})\theta_t - \alpha D_t^{-1}\nabla L(\theta_t). \tag{12.107}$$

The adaptive matrix scales the penalty gradient. Coordinates with different entries of D_t receive different amounts of shrinkage.

AdamW applies *decoupled weight decay*:

$$\theta_{t+1} = (1 - \alpha\lambda)\theta_t - \alpha D_t^{-1}\nabla L(\theta_t). \tag{12.108}$$

The factor multiplying θ_t is independent of the adaptive gradient geometry. For ordinary unpreconditioned gradient descent, $D_t = I$ and the two updates coincide. For a general adaptive diagonal matrix, the two updates define different algorithms. Exercise 7 asks you to verify the distinction.

For the normalized scalar logistic update in Example 12.7.1, AdamW gives

$$\theta_{t+1} = (1 - \alpha\lambda)\theta_t + \alpha. \tag{12.109}$$

If $0 < \alpha\lambda < 2$, then

$$\theta_t \longrightarrow \frac{1}{\lambda}, \qquad L(\theta_t) \longrightarrow \log(1 + e^{-1/\lambda}) > 0. \tag{12.110}$$

The limiting parameter is finite, and the limiting unregularized logistic loss is positive. The weight-decayed and unregularized procedures therefore optimize different training criteria.

The algebra also clarifies the design choice. An ℓ_2 penalty belongs to the objective, so an adaptive optimizer preconditions the penalty gradient together with the loss gradient. Decoupled weight decay prescribes uniform multiplicative shrinkage in the chosen parameterization. The two updates encode different parameter preferences.

12.7.2 How acceleration and adaptation enter neural-network training

The methods in this lecture solve different problems, and practical training often combines several of them. Momentum SGD and Nesterov-style momentum are used for neural networks. The smooth strongly convex acceleration theorem assumes a deterministic objective with global constants μ and β. General neural-network training has a nonconvex objective and noisy minibatch gradients, so the theorem supplies no rate for that setting. The momentum coefficient and the base learning rate therefore remain empirical choices in such applications.

AdaGrad and Adam modify a different part of the update. AdaGrad assigns coordinatewise scales from all past squared gradients. The permanent accumulator is useful when rare or sparse coordinates should receive larger updates, but the same permanence can make later updates too small when the gradient scale changes. Adam uses exponential memory so that the coordinatewise scales can change. AdamW combines that normalization with uniform weight decay. Transformer training provides extensive empirical use of AdamW. Available theory gives guarantees in restricted models and leaves a universal comparison with momentum SGD open.

A learning-rate schedule controls the global multiplier left outside these direction and normalization choices. For a known training budget T, a single cosine decay has the form

$$\alpha_t = \alpha_{\min} + \frac{\alpha_{\max} - \alpha_{\min}}{2}\left(1 + \cos\frac{\pi t}{T}\right), \qquad 0 \le t \le T. \tag{12.111}$$

The schedule makes relatively large updates early and reduces the global step scale smoothly near the end of a fixed-budget run. A *warmup phase* increases the learning rate from a smaller initial value to $\alpha_{\max}$ during a prescribed number of initial iterations and is often placed before the decay. Cosine decay is fixed before training: α_t depends only on t and the chosen schedule parameters. The choices of $\alpha_{\max}$, $\alpha_{\min}$, the warmup length, and T still require specification. When cosine decay is used with Adam, the coefficients β_1 and β_2 usually remain fixed. The weights β_1^{t-s} and β_2^{t-s} assigned to past gradients therefore decay at the same rate throughout training. Near the end of training, however, the smaller α_t means that the parameters move less during one fixed-length memory window, so the remembered gradients come from more similar parameter values. Forgetting per iteration is constant; the memory window becomes longer relative to the rate at which the parameters move. The original cosine-annealing proposal used repeated cycles with warm restarts; a single decay uses one cycle without restarting.

12.8 How much tuning can be removed?

Each method studied above removes a particular tuning requirement. Backtracking selects a safe stepsize without being supplied with β; scalar and diagonal AdaGrad select stepsizes without knowing the future gradient scales. RMSProp and Adam track recent coordinatewise scales while retaining a base learning rate and memory parameters. Exercise 8 asks you to recover the exact comparisons with the corresponding methods tuned in advance. This section addresses the broader question: can one algorithm match a tuned method when several properties of the problem are unknown?

We use the phrase *tuning-free* for a comparison between a self-tuning method and a specified reference method. A tuning-free claim must specify the quantities supplied to the reference method, the looser information supplied to the self-tuning method, and the performance criterion. Two important results show both the scope and the limits of such claims.

12.8.1 Adapting to the regularity of a deterministic objective

Universal convex methods address an entire family of regularity regimes. Suppose that L is convex and that, for some unknown $\nu \in [0,1]$ and $H_\nu > 0$, the first-order oracle obeys

$$\|g(u) - g(v)\|_2 \le H_\nu \|u - v\|_2^\nu, \qquad g(u) \in \partial L(u). \tag{12.112}$$

Let $R = \|\theta_0 - \theta^\star\|_2$. Given a target objective error $\epsilon > 0$, a universal accelerated method can attain

$$L(\theta_N) - L^\star \le \epsilon \quad \text{after} \quad N = O\left(\inf_{\nu \in [0,1]} \left(\frac{H_\nu}{\epsilon} \right)^{\frac{2}{1+3\nu}} R^{\frac{2+2\nu}{1+3\nu}} \right) \tag{12.113}$$

first-order oracle calls, without being supplied with ν or H_ν. For $\nu = 1$, the bound becomes $O(\sqrt{H_1 R^2/\epsilon})$, the accelerated smooth-convex rate. For $\nu = 0$, it becomes $O(H_0^2 R^2/\epsilon^2)$, the nonsmooth rate. The same method selects the appropriate rate from the observed first-order behavior.

12.8.2 Matching tuned SGD with several unknown scales

Stochastic optimization makes the comparison more demanding because the observations used for tuning are noisy. Consider a convex, β-smooth objective on a convex set of diameter D. At a query θ, suppose the oracle returns $G(\theta)$ satisfying

$$\mathbb{E}[G(\theta) \mid \theta] = \nabla L(\theta), \qquad \|G(\theta) - \nabla L(\theta)\|_2 \le \sigma \quad \text{almost surely.} \tag{12.114}$$

With suitable tuning, variants of SGD attain the high-probability rate

$$L(\widehat{\theta}_T) - L^\star = \widetilde{O}\left(\frac{\beta D^2}{T} + \frac{\sigma D}{\sqrt{T}} \right), \tag{12.115}$$

where $\widetilde{O}$ suppresses logarithmic factors in the failure probability and the number of oracle calls. A tuning-free counterpart of SGD receives loose ranges containing β, D, and σ, and must match Eq. (12.115) up to logarithmic factors in the widths of those ranges.

Proposition 12.8.1 (Scope of tuning-free stochastic convex optimization)**.** *For the bounded-domain problem above, algorithms exist that match Eq. (12.115) up to logarithmic factors without the exact values of β, D, and σ. On the unbounded domain $\mathbb{R}^d$, replace D by the initial distance*

$$D_\star = \|\theta_0 - \theta^\star\|_2. \tag{12.116}$$

No algorithm supplied only with loose ranges for the problem scales can guarantee

$$L(\widehat{\theta}_T) - L^\star = \widetilde{O}\left(\frac{\beta D_\star^2}{T} + \frac{\sigma D_\star}{\sqrt{T}}\right) \tag{12.117}$$

uniformly over all smooth convex objectives and bounded-noise stochastic oracles. The impossibility already holds for one-dimensional objectives.

The logarithmic factors depend on the loose parameter ranges, T, and the requested failure probability.

The bounded domain supplies a finite upper bound on how far a minimizer can be from the initial point. On an unbounded domain, two stochastic problems can produce the same observations with substantial probability while having different values of $D_\star$ and different optimal tuning. An algorithm then makes the same choices on both problems, although matching Eq. (12.117) would require different choices. Extra assumptions on the noise can make its scale estimable and restore positive results. The bibliographical notes give the precise theorems and references.

For neural-network training, the result identifies the information that a strong tuning claim would have to address. The parameter domain is usually unbounded, the objective is nonconvex, and the minibatch oracle is noisy. Adam's coordinatewise normalization leaves the base learning-rate schedule, memory parameters, numerical stabilizer, and weight decay to be selected. A theorem comparing an automatically selected training recipe with the best tuned AdamW or SGD recipe would require a specified model class, stochastic oracle, training budget, and performance measure.

12.9 Optional reading*

This optional section proves the smooth-convex accelerated rate with a time-varying Lyapunov function, uses that rate to obtain linear convergence by restarting, and gives a spectral explanation for quadratic acceleration.

12.9.1 A Lyapunov proof of smooth-convex acceleration

The smooth-convex accelerated method uses a growing coefficient in its Lyapunov function. The growth of that coefficient produces the $1/t^2$ rate that will be converted into geometric decay by restarting.

Let $A_0 = 0$ and initialize $\theta_0 = v_0 = z$. For the state (θ_t, v_t), consider

$$E_t = A_t\big(L(\theta_t) - L^\star\big) + \frac{1}{2}\|v_t - \theta^\star\|_2^2. \tag{12.118}$$

The design objective is to keep E_t nonincreasing while making A_t grow. For $v_{t+1} = v_t - a_{t+1}g_t$, expansion gives

$$\frac{1}{2}\|v_{t+1} - \theta^\star\|_2^2 = \frac{1}{2}\|v_t - \theta^\star\|_2^2 - a_{t+1}\langle g_t, v_t - \theta^\star\rangle + \frac{a_{t+1}^2}{2}\|g_t\|_2^2. \tag{12.119}$$

For $\theta_{t+1} = y_t - g_t/\beta$, smoothness gives

$$A_{t+1}\big(L(\theta_{t+1}) - L^\star\big) \le A_{t+1}\big(L(y_t) - L^\star\big) - \frac{A_{t+1}}{2\beta}\|g_t\|_2^2. \tag{12.120}$$

The two squared-gradient terms cancel when

$$\frac{A_{t+1}}{2\beta} = \frac{a_{t+1}^2}{2} \qquad \Longleftrightarrow \qquad A_{t+1} = \beta a_{t+1}^2.$$

Use $a_{t+1} > 0$ also as the increment $A_{t+1} - A_t$. The two design requirements are therefore

$$A_{t+1} = A_t + a_{t+1} = \beta a_{t+1}^2, \tag{12.121}$$

and define

$$\begin{aligned} y_t &= \frac{A_t\theta_t + a_{t+1}v_t}{A_{t+1}}, \\ g_t &= \nabla L(y_t), \\ v_{t+1} &= v_t - a_{t+1}g_t, \\ \theta_{t+1} &= \frac{A_t\theta_t + a_{t+1}v_{t+1}}{A_{t+1}} = y_t - \frac{1}{\beta}g_t. \end{aligned} \tag{12.122}$$

Proposition 12.9.1 (Smooth-convex acceleration)**.** *Suppose that L is convex and β-smooth and that the minimizer set $\Theta^\star$ is nonempty. The iterates in Eq. (12.122) satisfy $E_{t+1} \le E_t$. Consequently, for every $t \ge 1$,*

$$L(\theta_t) - L^\star \le \frac{2\beta\,\mathrm{dist}(z, \Theta^\star)^2}{(t+1)^2}. \tag{12.123}$$

Proof. The smoothness and auxiliary-state terms have already been expanded in Eqs. (12.119) and (12.120). After substituting $A_{t+1} = \beta a_{t+1}^2$ into the smoothness bound, it remains to control the term $L(y_t) - L^\star$. Convexity at y_t, applied to θ_t and $\theta^\star$ with weights A_t and a_{t+1}, respectively, gives

$$\begin{aligned} &A_t\big(L(\theta_t) - L(y_t)\big) + a_{t+1}\big(L^\star - L(y_t)\big) \\ &\quad \ge \langle g_t, A_t(\theta_t - y_t) + a_{t+1}(\theta^\star - y_t)\rangle = -a_{t+1}\langle g_t, v_t - \theta^\star\rangle, \end{aligned}$$

where the equality uses $A_{t+1}y_t = A_t\theta_t + a_{t+1}v_t$. Rearranging yields

$$A_{t+1}\big(L(y_t) - L^\star\big) \le A_t\big(L(\theta_t) - L^\star\big) + a_{t+1}\langle g_t, v_t - \theta^\star\rangle. \tag{12.124}$$

Adding Eqs. (12.119), (12.120) and (12.124) cancels the inner-product and squared-gradient terms and proves $E_{t+1} \le E_t$.

The rate now follows from a lower bound on A_t. If $q_t = \sqrt{\beta A_t}$, then Eq. (12.121) implies $q_{t+1}^2 = q_t^2 + q_{t+1}$, and hence $q_{t+1} \ge q_t + 1/2$. Since $q_1 = 1$,

$$A_t \ge \frac{(t+1)^2}{4\beta} \qquad (t \ge 1). \tag{12.125}$$

Finally, choose a minimizer $\theta^\star$ closest to z. Then $E_t \le E_0 = \mathrm{dist}(z, \Theta^\star)^2/2$ and $A_t(L(\theta_t) - L^\star) \le E_t$ give Eq. (12.123). □

The cancellation in the proof explains the coefficient equation $A_{t+1} = \beta a_{t+1}^2$. The method is a standard coupled representation of Nesterov's smooth-convex accelerated gradient method; eliminating v_t gives its familiar two-step form.

12.9.2 Restarting acceleration under convex PL

This subsection derives the accelerated linear rate for smooth convex PL objectives. The argument combines smooth-convex acceleration with the quadratic-growth consequence of PL. Strong convexity is one sufficient condition for this result.

Let $\mathsf{A}_s(z)$ denote the output θ_s of Eq. (12.122) after s gradient evaluations, starting from z. By Proposition 12.9.1,

$$L(\mathsf{A}_s(z)) - L^\star \le \frac{2\beta \operatorname{dist}(z, \Theta^\star)^2}{(s+1)^2}. \tag{12.126}$$

The quadratic-growth bound Eq. (12.58) gives

$$\operatorname{dist}(z, \Theta^\star)^2 \le \frac{2}{\mu}\big(L(z) - L^\star\big). \tag{12.127}$$

Combining the two inequalities yields

$$L(\mathsf{A}_s(z)) - L^\star \le \frac{4\kappa}{(s+1)^2}\big(L(z) - L^\star\big). \tag{12.128}$$

Choose $s = \lceil \sqrt{8\kappa} \rceil - 1$ and restart the method from the preceding output after every s gradients. If z_r is the parameter after r epochs, then

$$L(z_r) - L^\star \le 2^{-r}\big(L(z_0) - L^\star\big). \tag{12.129}$$

The total number of gradient evaluations needed to obtain $L(z_r) - L^\star \le \epsilon$ is therefore

$$O\left(\sqrt{\kappa} \log \frac{L(z_0) - L^\star}{\epsilon}\right). \tag{12.130}$$

The additional symbols A_s, s, and r describe the inner method, the epoch length, and the epoch index. They are needed only for this restart construction.

12.9.3 Implementing the minimax polynomial by Chebyshev iteration

This subsection identifies the degree-t polynomial with the smallest worst-case residual on $[\mu, \beta]$ and derives a first-order iteration that generates that polynomial. Chebyshev semi-iteration has the algebraic form of a heavy-ball update with time-varying coefficients. It is neither the fixed-coefficient Polyak heavy-ball method nor the Nesterov method in Section 12.4.

For the quadratic Eq. (12.12), a linear first-order method whose iterates use past gradients and linear combinations has an error of the form

$$e_t = p_t(H) e_0, \qquad p_t(0) = 1,$$

for a polynomial p_t of degree at most t. The condition $p_t(0) = 1$ has a direct meaning: when $\lambda = 0$, every gradient vanishes in that eigendirection, so the error remains equal to its initial value. For every H satisfying $\mu I \preceq H \preceq \beta I$, the Euclidean error satisfies

$$\|e_t\|_2 \le \max_{\lambda \in [\mu, \beta]} |p_t(\lambda)| \, \|e_0\|_2. \tag{12.131}$$

Proposition 12.9.2 (Polynomials generated by gradient descent)**.** *Constant-step gradient descent generates*

$$p_t(\lambda) = (1 - \eta\lambda)^t.$$

Gradient descent with predetermined steps $\eta_0, \dots, \eta_{t-1}$ generates

$$p_t(\lambda) = \prod_{j=0}^{t-1} (1 - \eta_j \lambda).$$

Thus repeated constant-step gradient descent restricts p_t to a power of one linear factor.

Proof. Each eigencoordinate satisfies $e_{j+1}(\lambda) = (1 - \eta_j \lambda) e_j(\lambda)$. Iterating the scalar recursion gives both formulas. □

Let T_t be the degree-t Chebyshev polynomial, characterized by $T_t(\cos\phi) = \cos(t\phi)$. Assume $0 < \mu < \beta$ throughout this subsection. If $\mu = \beta$, then $H = \beta I$, and one gradient step with stepsize $1/\beta$ reaches $\theta^\star$.

Theorem 12.9.3 (Minimax residual polynomial)**.** *For every integer $t \geq 1$,*

$$\min_{\substack{p \in \mathbb{R}[\lambda],\ \deg(p) \leq t \\ p(0)=1}} \max_{\lambda \in [\mu,\beta]} |p(\lambda)| = \frac{1}{\left| T_t\left(\frac{\beta+\mu}{\beta-\mu}\right)\right|}. \tag{12.132}$$

A minimizer is

$$p_t^\star(\lambda) = \frac{T_t\left(\frac{\beta+\mu-2\lambda}{\beta-\mu}\right)}{T_t\left(\frac{\beta+\mu}{\beta-\mu}\right)}. \tag{12.133}$$

Proof. The affine map

$$x = \frac{\beta + \mu - 2\lambda}{\beta - \mu}$$

sends $[\mu, \beta]$ to $[-1, 1]$ and sends $\lambda = 0$ to $a = (\beta + \mu)/(\beta - \mu) > 1$. Under the displayed affine change of variable, a feasible polynomial p becomes a degree-t polynomial q satisfying $q(a) = 1$. The displayed polynomial is feasible, and $|T_t(x)| \leq 1$ on $[-1, 1]$, so the maximum magnitude of the displayed polynomial is the right side of Eq. (12.132).

To prove optimality, suppose that another transformed feasible polynomial q had a strictly smaller maximum. At the $t+1$ alternating extrema $x_j = \cos(j\pi/t)$, the polynomial $T_t(x) - T_t(a)q(x)$ has the sign of $T_t(x_j) = (-1)^j$. The difference polynomial therefore has at least one root between every consecutive pair of extrema and also vanishes at $x = a$ because $q(a) = 1$. A nonzero polynomial of degree at most t cannot have these $t + 1$ distinct roots. □

Proposition 12.9.4 (Chebyshev iteration implements the minimax polynomial)**.** *Put*

$$a = \frac{\beta + \mu}{\beta - \mu}, \qquad C_t = T_t(a).$$

Starting from θ_0, set

$$\theta_1 = \theta_0 - \frac{2}{\beta + \mu} \nabla L(\theta_0). \tag{12.134}$$

For every $t \geq 1$, compute

$$\theta_{t+1} = \theta_t - \frac{4C_t}{(\beta - \mu)C_{t+1}} \nabla L(\theta_t) + \frac{C_{t-1}}{C_{t+1}}(\theta_t - \theta_{t-1}). \tag{12.135}$$

Then

$$e_t = p_t^\star(H) e_0 \qquad \text{for every } t \geq 0. \tag{12.136}$$

The scalar coefficients can be computed from

$$C_0 = 1, \qquad C_1 = a, \qquad C_{t+1} = 2aC_t - C_{t-1}. \tag{12.137}$$

Thus each iteration uses one gradient evaluation, the current iterate, the preceding iterate, and scalar coefficients determined by μ and β.

Proof. The recurrence $T_{t+1}(x) = 2xT_t(x) - T_{t-1}(x)$ and Eq. (12.133) give

$$p^\star_{t+1}(\lambda) = \frac{2C_t}{C_{t+1}}\left(a - \frac{2\lambda}{\beta-\mu}\right)p^\star_t(\lambda) - \frac{C_{t-1}}{C_{t+1}}p^\star_{t-1}(\lambda).$$

The scalar recurrence in Eq. (12.137) implies

$$\frac{2aC_t}{C_{t+1}} = 1 + \frac{C_{t-1}}{C_{t+1}}.$$

Substituting H for λ, applying the resulting polynomial identity to e_0, and using $\nabla L(\theta_t) = He_t$ gives Eq. (12.135). Finally, $p^\star_1(\lambda) = 1 - 2\lambda/(\beta+\mu)$ gives Eq. (12.134). Induction proves Eq. (12.136). □

Corollary 12.9.5 (Euclidean-error bound for the minimax polynomial)**.** *The minimax value in Eq. (12.132) satisfies*

$$\max_{\lambda\in[\mu,\beta]} |p^\star_t(\lambda)| \le 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^t. \tag{12.138}$$

Proof. Put

$$r = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}.$$

Direct calculation gives

$$\frac{\beta+\mu}{\beta-\mu} = \frac{\kappa+1}{\kappa-1} = \frac{r+r^{-1}}{2}.$$

The identity $T_t((u+u^{-1})/2) = (u^t+u^{-t})/2$ therefore gives

$$T_t\left(\frac{\beta+\mu}{\beta-\mu}\right) = \frac{r^t+r^{-t}}{2}.$$

Taking the reciprocal yields

$$\frac{2r^t}{1+r^{2t}} \le 2r^t,$$

which proves Eq. (12.138). □

Corollary 12.9.6 (Relation between Chebyshev iteration and heavy ball)**.** *Let*

$$r = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}.$$

The Chebyshev update Eq. (12.135) has the heavy-ball form

$$\theta_{t+1} = \theta_t - \eta^{\mathrm{Ch}}_t \nabla L(\theta_t) + \gamma^{\mathrm{Ch}}_t(\theta_t - \theta_{t-1}), \tag{12.139}$$

where

$$\eta^{\mathrm{Ch}}_t = \frac{4C_t}{(\beta-\mu)C_{t+1}}, \qquad \gamma^{\mathrm{Ch}}_t = \frac{C_{t-1}}{C_{t+1}}. \tag{12.140}$$

The coefficients satisfy

$$\lim_{t\to\infty}\eta^{\mathrm{Ch}}_t = \frac{4}{(\sqrt{\beta}+\sqrt{\mu})^2}, \qquad \lim_{t\to\infty}\gamma^{\mathrm{Ch}}_t = r^2. \tag{12.141}$$

Thus the limits are the quadratic-optimal Polyak heavy-ball coefficients. Chebyshev iteration uses the time-varying coefficients in Eq. (12.140) to attain the finite-degree minimax polynomial; fixed-coefficient heavy ball does not attain that polynomial in general.

Proof. The calculation in the proof of Corollary 12.9.5 gives

$$C_t = \frac{r^t + r^{-t}}{2}.$$

Taking the two ratios in Eq. (12.140) gives

$$\frac{C_t}{C_{t+1}} \longrightarrow r, \qquad \frac{C_{t-1}}{C_{t+1}} \longrightarrow r^2.$$

The identity

$$\frac{4r}{\beta - \mu} = \frac{4}{(\sqrt{\beta} + \sqrt{\mu})^2}$$

proves Eq. (12.141). For comparison, fixed-coefficient heavy ball initialized by $\theta_{-1} = \theta_0$ has characteristic polynomial $(z + r)^2$ at $\lambda = \beta$ and initial values $p_0^{\mathrm{HB}}(\beta) = 1$ and $p_1^{\mathrm{HB}}(\beta) = -r(2 + r)$. Hence

$$p_t^{\mathrm{HB}}(\beta) = (-r)^t\big(1 + t(1 + r)\big). \tag{12.142}$$

The linear factor in Eq. (12.142) shows directly that the fixed-coefficient method is not the finite-degree minimax method. □

The square root of κ comes from uniform polynomial approximation over the spectral interval. Proposition 12.9.4 shows that the minimax polynomial is also an algorithm: its three-term recurrence becomes the time-varying heavy-ball update Eq. (12.139). The implementation does not compute an eigendecomposition, although it does require the spectral bounds μ and β.

Nesterov acceleration generates a third polynomial sequence. On the quadratic objective, Eq. (12.49) implies

$$p_{t+1}^{\mathrm{NAG}}(\lambda) = \left(1 - \frac{\lambda}{\beta}\right)\left((1 + \gamma)p_t^{\mathrm{NAG}}(\lambda) - \gamma p_{t-1}^{\mathrm{NAG}}(\lambda)\right). \tag{12.143}$$

with

$$p_0^{\mathrm{NAG}}(\lambda) = 1, \qquad p_1^{\mathrm{NAG}}(\lambda) = 1 - \frac{\lambda}{\beta}. \tag{12.144}$$

Proposition 12.9.7 (Worst-case Nesterov residual on quadratics)**.** *Set $\tau = 1/\sqrt{\kappa}$ and initialize $\theta_{-1} = \theta_0$. The residual polynomials defined by Eq. (12.143) satisfy, for every $t \geq 0$,*

$$\max_{\lambda \in [\mu, \beta]} |p_t^{\mathrm{NAG}}(\lambda)| = \left(1 + \frac{t}{\sqrt{\kappa}}\right)\left(1 - \frac{1}{\sqrt{\kappa}}\right)^t. \tag{12.145}$$

Proof. The claim is immediate for $t = 0$. For $t \geq 1$, put

$$x = \sqrt{\frac{1 - \lambda/\beta}{1 - \tau^2}} \in [0, 1].$$

Let U_j denote the Chebyshev polynomial of the second kind, characterized by $U_j(\cos\phi) = \sin((j + 1)\phi)/\sin\phi$ with the endpoint values defined by continuity. Substitution in Eq. (12.143), together with $\gamma = (1 - \tau)/(1 + \tau)$, gives

$$p_t^{\mathrm{NAG}}(\lambda) = \big((1 - \tau)x\big)^t\left(T_t(x) + \tau x U_{t-1}(x)\right). \tag{12.146}$$

Indeed, after division by $((1-\tau)x)^t$, the recurrence becomes the Chebyshev recurrence, and $p_1^{\mathrm{NAG}}(\lambda) = (1-\tau)(1+\tau)x^2$. The representation at $x = 0$ follows by continuity. For $x \in [-1, 1]$,

$$|T_t(x)| \le 1, \qquad |U_{t-1}(x)| \le t.$$

Therefore

$$|p_t^{\mathrm{NAG}}(\lambda)| \le (1-\tau)^t x^t (1+\tau t x) \le (1-\tau)^t (1+\tau t).$$

At $\lambda = \mu$, one has $x = 1$, $T_t(1) = 1$, and $U_{t-1}(1) = t$. The upper bound is therefore attained. □

Equations (12.138) and (12.145) both give $\sqrt{\kappa}$ iteration dependence, but they are different finite-time statements. Chebyshev semi-iteration generates the minimax polynomial Eq. (12.133); Nesterov acceleration generates Eq. (12.146). The Euclidean-error comparison in this subsection is also distinct from the objective-gap potential guarantee in Theorem 12.4.1.

12.10 Take-home messages

- Backtracking line search avoids advance knowledge of the smoothness constant by testing the finite-step decrease inequality needed in the proof. Under the PL inequality and with exact objective values, the number of gradient evaluations needed to reach a prescribed objective gap increases by at most the search multiplier ρ; the additional cost is objective evaluations at rejected trial points.

- For smooth, strongly convex objectives, acceleration reduces the number of gradient evaluations needed for a fixed relative objective-gap reduction from order κ to order $\sqrt{\kappa}$. A matching first-order lower bound makes the square-root dependence optimal. The stated one-step potential proof uses the lower model supplied by strong convexity. The accelerated update can be derived by combining the smooth upper step with that lower model and choosing the coupling coefficient so that the squared-gradient terms cancel. The fixed momentum coefficient uses κ. For smooth convex objectives satisfying the PL inequality, restarting the smooth-convex accelerated method also gives square-root condition-number dependence; a geometric search over epoch lengths handles an unknown κ when $L^\star$ is available.

- Preconditioning changes coordinates. The relevant quadratic curvature is $P^{1/2} H P^{1/2}$, and a structured approximation trades geometric fidelity for near-linear computation and memory.

- Scalar online adaptation matches the best constant-stepsize regret bound in hindsight within a factor 3/2. Diagonal AdaGrad is no worse at the level of the scalar and diagonal upper bounds and can improve the scalar bound when gradient activity is concentrated in a small number of coordinates.

- RMSProp replaces cumulative memory by exponential squared-gradient memory. Adam adds an exponential average of the gradients and bias correction. Both updates remain dependent on a base learning rate and memory parameters. Exponential forgetting prevents a general AdaGrad-style guarantee. Counterexamples exist in online and stochastic convex optimization, and a deterministic batch Adam variant can approach a nonoptimal limit cycle. Restricted positive theory identifies nonuniform smoothness and favorable coordinate or block geometry as possible benefits. On the scalar separable logistic example, exact gradient normalization improves loss decay from $\Theta(1/t)$ to $\Theta(e^{-\alpha t})$. The conclusion applies to the scalar example; comparisons on other objectives require separate analysis.

- AdamW applies uniform weight decay outside the adaptive gradient scaling. An ℓ_2 penalty inside the objective receives coordinatewise scaling from the adaptive matrix. The two formulas define different updates.

- Universal deterministic methods can attain the rate associated with an unknown Hölder-smoothness regime. In stochastic convex optimization, methods can match tuned SGD up to logarithmic factors on bounded domains. The analogous guarantee over an unbounded domain is impossible under the general bounded-noise oracle model. Every tuning-free claim must specify the retained hints, oracle, domain, reference method, and performance measure.

Bibliographical notes

The sufficient-decrease line search goes back to classical work including Armijo [1966]. The presentation here uses the quadratic upper-bound test that appears in proximal-gradient and accelerated proximal-gradient methods. Beck and Teboulle [2009] analyze both ordinary and accelerated backtracking and prove the $O(1/t)$ and $O(1/t^2)$ convex objective-gap bounds used here.

Polyak's heavy-ball method was introduced by Polyak [1964]. Its quadratic-optimal parameters give accelerated asymptotic spectral convergence on strongly convex quadratics. Nesterov's accelerated method was introduced by Nesterov [1983]; Nesterov [2004] gives a systematic treatment of its guarantees over the full smooth convex and smooth strongly convex classes, together with the standard first-order lower bounds. The construction in Section 12.4 is an elementary estimate-sequence argument: it combines a smooth upper step with accumulated strongly convex lower models and determines the coupling coefficient from the desired potential contraction. The smooth-convex construction in Section 12.9.1 expresses the estimate-sequence mechanism as a time-varying Lyapunov inequality.

Restarting converts the smooth-convex accelerated rate into a linear rate under quadratic growth. The implication from PL to quadratic growth, with the same constant, is proved by Karimi et al. [2016]. O'Donoghue and Candès [2015] propose objective- and gradient-based adaptive restart rules motivated by oscillation of the accelerated iterates. Fercoq and Qu [2019] give a provable adaptive restart scheme under a local quadratic-growth condition; their test uses the norm of the gradient mapping and adjusts the restart frequency from observed progress. Roulet and d'Aspremont [2020] analyze scheduled and adaptive restarts under more general Hölderian growth conditions. For convex objectives, such growth conditions are related to the nonlinear gradient-dominance inequalities in Lecture 11 and lead to exponent-dependent restart schedules and rates. The main text uses the quadratic-growth case because it gives the clean convex-PL result in Eq. (12.60).

The distinction between quadratic and general objectives is essential for heavy ball. Lessard et al. [2016] give a smooth, strongly convex nonquadratic example on which the quadratic-optimal heavy-ball parameters produce a nonconvergent cycle. In dimension at least two, Goujaud et al. [2023] prove a stronger fixed-parameter non-acceleration result: every fixed heavy-ball parameter choice either has a nonaccelerated worst-case rate on strongly convex quadratics or fails to converge on some smooth, strongly convex objective.

The polynomial construction in Section 12.9.3 is the classical Chebyshev semi-iterative method. Barrett et al. [1994] present the three-term implementation for linear systems and discuss its dependence on spectral bounds. Chebyshev semi-iteration is a nonstationary heavy-ball iteration: its coefficients converge to the quadratic-optimal Polyak coefficients, but the time-varying coefficients are what generate the finite-degree minimax polynomial. Nesterov acceleration generates a different polynomial recurrence. All three methods exhibit square-root condition-number dependence in their respective quadratic guarantees. The lower bound separating smooth PL objectives from strongly

convex objectives is proved by Yue et al. [2023]: over the general smooth PL class, first-order methods require order $(\beta/\mu)\log(1/\epsilon)$ oracle calls in the worst case.

The Kronecker-factored calculation in Eqs. (12.65) and (12.66) is the structural approximation used by K-FAC. Zhang et al. [2019] prove global convergence for natural gradient and K-FAC under Jacobian rank and stability conditions, including an overparameterized two-layer ReLU setting. Karakida and Osawa [2020] analyze deep networks in an infinite-width limit and show that several approximate Fisher matrices have the same function-space dynamics as exact natural gradient under the assumptions of their analysis. The two sets of results justify specific structured approximations in restricted regimes; the results do not give a general finite-width conditioning theorem for multilayer networks.

The adaptive subgradient framework and data-dependent regret guarantees for AdaGrad are due to Duchi et al. [2011]. The diagonal theorem in Theorem 12.6.2 is a direct coordinatewise specialization with a simple box constraint. Exercise 5 asks you to prove the specialization and identify exactly which observed geometry enters.

RMSProp was introduced in Geoffrey Hinton's 2012 neural-network course [Hinton, 2012]. Its update has several implementation variants, including placement of ε and optional correction of the initial second-moment bias. Zou et al. [2019] give sufficient conditions for convergence of RMSProp and Adam in nonconvex stochastic optimization. The conditions involve the base stepsizes and the accumulated second-moment quantities. The result supplies a conditional convergence guarantee. A tuning-free comparison with SGD requires a different theorem.

Adam was introduced by Kingma and Ba [2015]. The original convergence analysis was later shown to contain a problem. Reddi et al. [2018] prove separate online and stochastic convex nonconvergence results and propose AMSGrad. The complete-forgetting example in Example 12.7.2 is an elementary limiting case that isolates the online mechanism. Bock and Weiß [2022] construct trajectories of a deterministic bias-corrected batch Adam variant that approach a nonoptimal period-two limit cycle on a scalar strongly convex quadratic. Their denominator uses $\sqrt{v_t + \varepsilon}$, and the result rules out global convergence without asserting failure from every initialization. Défossez et al. [2022] give a convergence analysis for Adam and AdaGrad on smooth, possibly nonconvex objectives with bounded gradients. Results aimed at explaining an advantage beyond basic convergence include the nonuniform-smoothness comparison of Wang et al. [2024], the ℓ_∞ and blockwise analysis of Xie et al. [2025], and the block-heterogeneity analysis and experiments of Zhang et al. [2024b]. The assumptions and conclusions of the three analyses differ, so none provides a universal ordering of Adam and SGD.

The original motivation for Adam combines AdaGrad's behavior on sparse gradients with RMSProp's responsiveness to nonstationary gradient scales; see Kingma and Ba [2015]. The original paper also reports experiments on ℓ_2-regularized multiclass logistic regression and sparse bag-of-words classification. In the logistic experiment, Adam and momentum SGD have similar convergence and both improve on AdaGrad under the reported tuning. The scalar calculation in Example 12.7.1 isolates the flattening mechanism without attributing every feature of Adam to that example. For separable logistic regression, Soudry et al. [2018] prove $1/t$ loss decay and convergence toward the Euclidean max-margin direction for gradient descent with an admissible constant stepsize. Zhang et al. [2024a] prove that full-batch Adam with a class of diminishing learning rates instead approaches an ℓ_∞ max-margin direction. The latter result concerns implicit bias. Loss-decay rates require a separate analysis. Axiotis and Sviridenko [2023] obtain linear logistic-loss convergence with a variable gradient-descent stepsize, showing that the faster rate can also arise from gradient descent. Recent nonuniform-smoothness results give sharper comparisons for normalized methods and restricted Adam variants; see Vaswani et al. [2026].

Decoupled weight decay and AdamW are due to Loshchilov and Hutter [2019]. Zhuang et al.

[2022] interpret AdamW through an approximate proximal update and study the scale-freeness of the update. Convergence-rate results for AdamW are also available under explicit smoothness and noise assumptions; see, for example, Li et al. [2025]. The main text retains only the algebraic distinction between an adaptively scaled ℓ_2 penalty and decoupled decay because the distinction is needed to understand the update.

Momentum methods have also been used directly in neural-network training. Sutskever et al. [2013] study classical and Nesterov-style momentum for deep and recurrent networks, while the ResNet training recipe of He et al. [2016] uses momentum SGD. Loshchilov and Hutter [2017] introduce cosine annealing with warm restarts. A modern large-language-model example is the Llama 2 recipe of Touvron et al. [2023], which combines AdamW, learning-rate warmup, and cosine decay. These references document successful uses of the methods; they do not extend the smooth strongly convex acceleration theorem to neural networks or establish a universal ordering of optimizers.

The universal accelerated method of Nesterov [2015] is given the target accuracy while the Hölder exponent and its constant remain unknown. Its complexity gives both endpoint rates in Eq. (12.113). Khaled and Jin [2024] define a tuning-free counterpart of SGD by comparison with optimally tuned SGD under loose parameter hints. They identify existing positive results on bounded domains and prove the unbounded-domain impossibility summarized in Proposition 12.8.1. They also give positive unbounded-domain results under stronger noise assumptions and treat a separate nonconvex stationarity question. These results ask for simultaneous adaptation to smoothness, noise, and distance scales, which is a stronger requirement than backtracking with exact objective evaluations.

Glossary

Acceptable curvature estimate
A number $M > 0$ for which the proposed point $\theta^+(M)$ satisfies the finite-step acceptance test used by a line search.

Acceleration
An improvement in worst-case iteration complexity obtained from state carried across iterations. For β-smooth, μ-strongly convex objectives, the gradient complexities are $O(\kappa \log(1/\epsilon))$ for gradient descent and $O(\sqrt{\kappa} \log(1/\epsilon))$ for accelerated gradient descent. For β-smooth convex objectives with no PL or strong-convexity assumption, the corresponding objective-gap bounds are $O(\beta R^2/t)$ and $O(\beta R^2/t^2)$, where $R = \|\theta_0 - \theta^\star\|_2$.

Adam
An adaptive stochastic first-order method that keeps exponential averages m_t and v_t of the gradients and squared gradients. The update Eq. (12.90) divides the bias-corrected first-moment estimate $\widehat{m}_t$ coordinatewise by $\sqrt{\widehat{v}_t} + \varepsilon$.

AdamW and decoupled weight decay
An Adam update in which the parameter shrinkage is applied separately from the adaptively scaled gradient: $\theta_{t+1} = (1 - \alpha\lambda)\theta_t - \alpha D_t^{-1} g_t$.

Backtracking line search
A procedure that increases a trial curvature estimate, equivalently decreases the trial stepsize, until an evaluated finite-step inequality holds. Accelerated backtracking also updates the coupling coefficients that depend on the accepted curvature estimate.

Chebyshev semi-iteration
A quadratic-optimization method whose time-varying two-step recurrence implements the degree-t polynomial minimizing $\max_{\lambda\in[\mu,\beta]} |p_t(\lambda)|$ subject to $p_t(0) = 1$.

Cosine learning-rate decay
A fixed-budget schedule that changes the global learning rate according to Eq. (12.111), from $\alpha_{\max}$ at the start of the decay to $\alpha_{\min}$ at its end.

Diagonal AdaGrad
An online first-order method whose coordinate-i denominator is the square root of the cumulative squared gradients observed in that coordinate.

Exponential memory
A moving average in which an observation from s iterations ago receives a weight proportional to a power such as β_2^s.

First-order oracle
An interface that returns an objective value and gradient at a queried parameter. In the lower-bound comparison in this lecture, each query may depend on all preceding queries and oracle replies.

Heavy-ball method
The two-step update $\theta_{t+1} = \theta_t - \eta\nabla L(\theta_t) + \gamma(\theta_t - \theta_{t-1})$, which evaluates the gradient at the current iterate. Quadratic-optimal fixed coefficients need not converge on every smooth, strongly convex objective.

Hindsight-tuned stepsize
A constant stepsize chosen after the complete gradient sequence is known to minimize a proved upper bound such as Eq. (12.70). The comparison concerns the upper bound, not necessarily the realized losses.

Nesterov accelerated gradient
The two-step method $y_t = \theta_t + \gamma(\theta_t - \theta_{t-1})$ and $\theta_{t+1} = y_t - \nabla L(y_t)/\beta$. For the strongly convex theorem in this lecture, $\gamma = (\sqrt{\kappa} - 1)/(\sqrt{\kappa} + 1)$.

Online convex optimization and regret
At round t, the learner chooses x_t before observing the convex loss ℓ_t. Its regret relative to a fixed u is $\mathrm{Reg}_T(u) = \sum_{t=1}^{T}(\ell_t(x_t) - \ell_t(u))$.

Preconditioner
A positive-definite matrix P multiplying the gradient. On a quadratic with Hessian H, the conditioning effect of P is determined by $P^{1/2}HP^{1/2}$.

Quadratic growth
For a nonempty minimizer set $\Theta^\star$, the inequality $L(\theta) - L^\star \geq \mu\,\mathrm{dist}(\theta, \Theta^\star)^2/2$, where $\mathrm{dist}(\theta, \Theta^\star) = \inf_{z\in\Theta^\star} \|\theta - z\|_2$. The PL inequality implies quadratic growth with the same constant. In the convex setting, this bound converts the distance-dependent accelerated rate into a gap-dependent restart guarantee.

Restart
Reinitialization of an accelerated method's memory after a fixed or data-dependent epoch while retaining the current parameter. Restarting a smooth-convex accelerated method every $\Theta(\sqrt{\kappa})$

iterations gives linear convergence under convexity and the PL inequality; adaptive restart rules search for this epoch length when κ is unknown.

Scalar online adaptation
A stepsize rule based on the cumulative Euclidean gradient magnitude. The rule in Theorem 12.6.1 matches the best constant-stepsize bound in hindsight within a factor $3/2$ without knowing the horizon or future gradient magnitudes.

RMSProp
An adaptive stochastic first-order method that divides the current gradient coordinatewise by the square root of an exponential average of recent squared gradients, as in Eq. (12.92).

Tuning-free guarantee
A theorem comparing an algorithm that lacks specified problem parameters with a benchmark that is optimally tuned using those parameters. The omitted parameters, allowed hints, oracle, and domain are part of the definition.

Universal convex method
A method that attains the rate associated with the unknown Hölder parameters ν and H_ν in Eq. (12.112). The guarantee Eq. (12.113) includes the nonsmooth and smooth-convex endpoint rates.

Warmup phase
An initial part of a learning-rate schedule during which the global learning rate increases from a smaller value to its prescribed peak value.

12.11 Exercises

General instructions. Justify every answer, including yes/no answers, even when the question does not explicitly ask for a justification.

Exercise profiles. Filled mountains show difficulty on a four-mountain scale: one is a direct application, two uses a familiar method, three combines ideas, and four is a substantial challenge. Filled cogs show technical load on a three-cog scale: one is light, two is sustained, and three is dense. Filled hourglasses show estimated time: one is under 20 minutes, two is 20–45 minutes, three is 45–90 minutes, and four is more than 90 minutes. Pale pictograms show the unused part of each scale. The work row uses the canonical order proof, calculation, construction, computation, and interpretation; pale icons mark work types not used. ◎ marks the intended learning outcome. Time estimates cover the non-optional parts. Ratings are not prerequisites or grading weights. A superscript asterisk marks an exercise based on supplementary lecture material; it is not a difficulty rating and does not make the exercise optional when assigned.

Unknown smoothness and acceleration

Exercise 1: Backtracking avoids conservative curvature scales.

◎ Quantify the work saved by testing a locally sufficient curvature scale.

Let

$$L(\theta) = \frac{\beta}{2}\theta^2, \qquad \beta > 0,$$

and consider the trial point $\theta^+(M) = \theta - L'(\theta)/M$ for $M > 0$. The sufficient-decrease test is

$$L(\theta^+(M)) \le L(\theta) - \frac{1}{2M}|L'(\theta)|^2. \tag{12.147}$$

(a) For $\theta \neq 0$, prove that Eq. (12.147) holds if and only if $M \ge \beta$. State what happens when $\theta = 0$.

(b) Let $0 < a < b$ and define

$$\widetilde{L}(\theta) = \begin{cases} a\theta^2/2, & \theta \le 0, \\ b\theta^2/2, & \theta > 0. \end{cases}$$

Prove that $\widetilde{L}$ is differentiable and b-smooth. Starting from $\theta_0 < 0$, show that the globally certified step $1/b$ gives

$$\theta_t = \left(1 - \frac{a}{b}\right)^t \theta_0,$$

whereas the trial curvature $M = a$ passes Eq. (12.147) with $\widetilde{L}$ in place of L and reaches the minimizer in one step. Deduce that the globally certified method requires

$$t \ge \frac{\log(1/\epsilon)}{-\log(1 - a/b)}$$

iterations to obtain $|\theta_t| \le \epsilon|\theta_0|$, for $0 < \epsilon < 1$.

(c) Return to $L(\theta) = \beta\theta^2/2$. Starting from an arbitrary $M_0 > 0$, repeatedly replace M by $2M$ until the test holds. Prove that the accepted value satisfies

$$M_{\mathrm{acc}} \le \max\{M_0, 2\beta\}$$

and that the number of rejected trials is at most

$$\max\left\{0, \left\lceil \log_2 \frac{\beta}{M_0} \right\rceil\right\}.$$

Hints For (a), expand $(1 - \beta/M)^2$. For (b), check pairs of points on the same side of zero and pairs that straddle zero separately.

Exercise 2: Eliminating the auxiliary state and proving the accelerated rate.

Eliminate the auxiliary state and use the potential contraction to prove the accelerated linear rate.

Let $L : \mathbb{R}^d \to \mathbb{R}$ be β-smooth and μ-strongly convex, let $\theta^\star$ be its minimizer, and define

$$\kappa = \frac{\beta}{\mu}, \qquad \tau = \frac{1}{\sqrt{\kappa}}, \qquad \gamma = \frac{1 - \tau}{1 + \tau}.$$

Initialize $v_0 = \theta_0$ and $\theta_{-1} = \theta_0$.

(a) Prove that the coupled update

$$y_t = \frac{\theta_t + \tau v_t}{1 + \tau}, \qquad \theta_{t+1} = y_t - \frac{1}{\beta}\nabla L(y_t),$$
$$v_{t+1} = (1 - \tau)v_t + \tau\left(y_t - \frac{1}{\mu}\nabla L(y_t)\right)$$

and the two-step update

$$y_t = \theta_t + \gamma(\theta_t - \theta_{t-1}), \qquad \theta_{t+1} = y_t - \frac{1}{\beta}\nabla L(y_t)$$

generate the same sequences (θ_t) and (y_t).

(b) Suppose the coupled update satisfies

$$V(\theta_t, v_t) \le (1-\tau)^t V(\theta_0, v_0), \qquad V(\theta, v) = L(\theta) - L^\star + \frac{\mu}{2}\|v - \theta^\star\|_2^2.$$

Prove

$$L(\theta_t) - L^\star \le 2\left(1 - \frac{1}{\sqrt{\kappa}}\right)^t \big(L(\theta_0) - L^\star\big),$$

and, for $0 < \epsilon < 2(L(\theta_0) - L^\star)$, show that $L(\theta_t) - L^\star \le \epsilon$ whenever

$$t \ge \sqrt{\kappa}\log\frac{2(L(\theta_0) - L^\star)}{\epsilon}.$$

Hints For part (a), use induction with

$$v_t = \frac{\theta_t - (1-\tau)\theta_{t-1}}{\tau}.$$

For part (b), use strong convexity at θ_0 and $1 - \tau \le e^{-\tau}$.

Exercise 3: Restarting acceleration under convex PL.

Convert the accelerated convex rate into square-root condition-number dependence and adapt the restart length by geometric search.

Let $L : \mathbb{R}^d \to \mathbb{R}$ be β-smooth, convex, and μ-PL, let $\kappa = \beta/\mu$, and suppose that its minimizer set $\Theta^\star$ is nonempty. You may use the quadratic-growth consequence of the PL inequality,

$$\operatorname{dist}(z, \Theta^\star)^2 \le \frac{2}{\mu}\big(L(z) - L^\star\big). \tag{12.148}$$

Here $\operatorname{dist}(z, \Theta^\star) = \inf_{\theta^\star \in \Theta^\star} \|z - \theta^\star\|_2$. Suppose that an accelerated method started from any z produces, after $s \ge 1$ iterations, an output $\mathsf{A}_s(z)$ satisfying

$$L(\mathsf{A}_s(z)) - L^\star \le \frac{2\beta \operatorname{dist}(z, \Theta^\star)^2}{(s+1)^2}. \tag{12.149}$$

(a) Prove

$$L(\mathsf{A}_s(z)) - L^\star \le \frac{4\kappa}{(s+1)^2}\big(L(z) - L^\star\big).$$

(b) Choose an integer epoch length s that makes the multiplicative factor at most $1/2$. Starting each epoch from the preceding epoch's output, prove that r epochs give objective gap at most $2^{-r}(L(\theta_0) - L^\star)$.

(c) Derive a total iteration bound of order

$$\sqrt{\kappa}\log\frac{L(\theta_0) - L^\star}{\epsilon}$$

for reaching objective gap at most ϵ. State which of the two inequalities used in part (a) comes from smoothness and convexity and which comes from the PL inequality.

(d) Suppose that κ is unknown but $L^\star$ is available. From a fixed point z, try epoch lengths $s = 1, 2, 4, \ldots$, restarting every trial from z, until

$$L(\mathsf{A}_s(z)) - L^\star \le \frac{1}{2}\big(L(z) - L^\star\big).$$

Prove that the first successful trial has $s < 2\sqrt{8\kappa}$ and that all trials together use fewer than $2s$ gradient evaluations. If the successful epoch length is then reused, with each later epoch starting from the preceding epoch's output, derive the total bound

$$O\left(\sqrt{\kappa}\left(1 + \log\frac{L(\theta_0) - L^\star}{\epsilon}\right)\right).$$

HINTS For part (b), require $(s+1)^2 \ge 8\kappa$.

Adaptive geometry

Exercise 4: A rotated quadratic defeats diagonal preconditioning.

Distinguish coordinatewise rescaling from correction of rotated curvature.

Let $\kappa > 1$ and

$$Q = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}, \qquad H = Q\begin{pmatrix} 1 & 0 \\ 0 & \kappa \end{pmatrix}Q^\top, \qquad L(\theta) = \frac{1}{2}\theta^\top H\theta.$$

For a positive-definite preconditioner P, the transformed Hessian is $P^{1/2}HP^{1/2}$.

(a) Compute H and prove that its condition number is κ.

(b) Let $P_{\text{diag}} = \operatorname{diag}(H)^{-1}$. Compute $P_{\text{diag}}^{1/2}HP_{\text{diag}}^{1/2}$ and prove that its condition number remains κ.

(c) Let $P_{\text{full}} = H^{-1}$. Compute the transformed Hessian and its condition number. Explain precisely which geometric information is absent from P_{diag}.

HINTS The two diagonal entries of H are equal. Orthogonal similarity preserves eigenvalues.

Exercise 5: The diagonal AdaGrad bound.[*]

Derive a regret bound whose coordinate scales are determined by the observed gradients.

Let $\mathcal{X} = \prod_{i=1}^d [a_i, b_i]$ with $D_i = b_i - a_i > 0$. On round t, let $\ell_t : \mathcal{X} \to \mathbb{R}$ be convex and let $g_t \in \partial\ell_t(x_t)$. Define

$$G_{t,i} = \left(\sum_{s=1}^{t} g_{s,i}^2\right)^{1/2}, \qquad \eta_{t,i} = \frac{D_i}{G_{t,i}}$$

when $G_{t,i} > 0$. If $G_{t,i} = 0$, leave coordinate i unchanged. Otherwise update it by

$$x_{t+1,i} = \Pi_{[a_i,b_i]}(x_{t,i} - \eta_{t,i}g_{t,i}).$$

(a) For nonnegative $a_1, \ldots, a_T$ and $A_t = \sum_{s=1}^t a_s$, prove

$$\sum_{t=1}^{T} \frac{a_t}{\sqrt{A_t}} \le 2\sqrt{A_T},$$

where a term with $A_t = 0$ is defined to be zero.

(b) Fix $u \in \mathcal{X}$. Prove for each coordinate that

$$\sum_{t=1}^{T} g_{t,i}(x_{t,i} - u_i) \le \frac{3}{2} D_i G_{T,i}.$$

(c) Use convexity to prove the regret bound

$$\sum_{t=1}^{T} \big(\ell_t(x_t) - \ell_t(u)\big) \le \frac{3}{2} \sum_{i=1}^{d} D_i \left(\sum_{t=1}^{T} g_{t,i}^2 \right)^{1/2}.$$

State how the right side differs from a bound based only on one worst-case gradient norm.

Hints For (a), use $a_t/\sqrt{A_t} \le 2(\sqrt{A_t} - \sqrt{A_{t-1}})$. For (b), use nonexpansiveness of interval projection and telescope with the nonincreasing stepsizes.

Changing gradient scales and weight decay

Exercise 6: Complete forgetting can give linear regret.

Verify a convex failure mechanism caused by normalizing only with the current gradient magnitude.

Let $\mathcal{X} = [-1, 1]$, choose $C > 2$ and $\alpha \in (0, 2]$, and repeat the three linear losses

$$\ell_{3k+1}(x) = Cx, \qquad \ell_{3k+2}(x) = \ell_{3k+3}(x) = -x, \qquad k \ge 0.$$

Consider the projected Adam special case with no momentum, complete second-moment forgetting, and no stabilizer:

$$m_t = g_t, \qquad v_t = g_t^2, \qquad x_{t+1} = \Pi_{[-1,1]}\left(x_t - \alpha \frac{m_t}{\sqrt{v_t}} \right), \qquad x_1 = 1,$$

where $g_t = \ell_t'(x_t)$.

(a) Prove that every three-round block begins at $x_{3k+1} = 1$ and list the three iterates used to evaluate its losses.

(b) Prove that the minimizer of the sum of the three losses over $[-1, 1]$ is $u = -1$. Compute the three-round regret against u and show that it equals $2C - 4 + \alpha > 0$.

(c) Conclude that the regret after $3K$ rounds grows linearly in K. Explain why the squared-gradient statistic in this example is not a reliable curvature estimate.

Hints The normalized update uses only $\operatorname{sign}(g_t)$. Track the projection after each of the two gradients equal to -1.

Exercise 7: Adaptive penalties and decoupled weight decay.

Distinguish an adaptively preconditioned ℓ_2 penalty from uniform weight decay.

Let D_t be a positive diagonal matrix, let $g_t = \nabla L(x_t)$, and let $\alpha, \lambda > 0$.

(a) Apply the adaptive update $x_{t+1} = x_t - \alpha D_t^{-1} \nabla F(x_t)$ to

$$F(x) = L(x) + \frac{\lambda}{2} \|x\|_2^2.$$

Show that

$$x_{t+1} = (I - \alpha\lambda D_t^{-1}) x_t - \alpha D_t^{-1} g_t.$$

(b) AdamW instead uses

$$x_{t+1} = (1 - \alpha\lambda)x_t - \alpha D_t^{-1} g_t.$$

State the necessary and sufficient condition on D_t under which the two shrinkage maps coincide for every x_t. Explain what is decoupled in the AdamW update.

HINTS Differentiate the quadratic penalty. Compare the two matrices multiplying x_t.

Comparing adaptive and tuned guarantees



Exercise 8: Comparing adaptive methods with tuned methods.*

Compare the guarantees proved in the lecture with the corresponding bounds obtained by tuning the stepsizes in advance.

(a) Suppose $M_{\text{init}} \le \beta$. Compare Theorem 12.2.3 and Eq. (12.9) with gradient descent using the tuned stepsize $1/\beta$. Show that the guaranteed contraction factor changes from $1 - \mu/\beta$ to $1 - \mu/(\rho\beta)$ and that the search uses at most

$$\left\lceil \log_\rho \frac{\beta}{M_{\text{init}}} \right\rceil$$

additional objective evaluations.

(b) Define

$$S_T = \left(\sum_{t=1}^{T} \|g_t\|_2^2 \right)^{1/2}, \qquad S_i = \left(\sum_{t=1}^{T} g_{t,i}^2 \right)^{1/2}.$$

Prove, for $a, b > 0$, that

$$\inf_{\eta > 0} \left\{ \frac{a^2}{2\eta} + \frac{\eta b^2}{2} \right\} = ab, \qquad \eta^\star = \frac{a}{b}.$$

Apply the identity to Eq. (12.70) and coordinatewise to prove

$$\inf_{\eta > 0} \left\{ \frac{D^2}{2\eta} + \frac{\eta S_T^2}{2} \right\} = D S_T, \qquad \inf_{\eta_1, \dots, \eta_d > 0} \sum_{i=1}^{d} \left(\frac{D_i^2}{2\eta_i} + \frac{\eta_i S_i^2}{2} \right) = \sum_{i=1}^{d} D_i S_i.$$

Use Theorems 12.6.1 and 12.6.2 to show that each adaptive upper bound is at most $3/2$ times its corresponding hindsight-tuned upper bound. Interpret a term with $S_i = 0$ as an infimum over $\eta_i > 0$.

References

Larry Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, 16(1):1–3, 1966. https://projecteuclid.org/euclid.pjm/1102995080.

Kyriakos Axiotis and Maxim Sviridenko. Gradient descent converges linearly for logistic regression on separable data. In *Proceedings of the 40th International Conference on Machine Learning*, pages 1302–1319, 2023. https://proceedings.mlr.press/v202/axiotis23a.html.

Richard Barrett, Michael Berry, Tony F. Chan, James Demmel, June Donato, Jack Dongarra, Victor Eijkhout, Roldan Pozo, Charles Romine, and Henk van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, second edition, 1994. https://www.netlib.org/templates/templates.pdf.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009. https://doi.org/10.1137/080716542.

Sebastian Bock and Martin Georg Weiß. Non-convergence and limit cycles in the Adam optimizer. 2022. https://arxiv.org/abs/2210.02070.

Alexandre Défossez, Léon Bottou, Francis Bach, and Nicolas Usunier. A simple convergence proof of Adam and AdaGrad. *Transactions on Machine Learning Research*, 2022. https://arxiv.org/abs/2003.02395.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011. https://www.jmlr.org/papers/v12/duchi11a.html.

Olivier Fercoq and Zheng Qu. Adaptive restart of accelerated gradient methods under local quadratic growth condition. *IMA Journal of Numerical Analysis*, 39(4):2069–2095, 2019. https://doi.org/10.1093/imanum/drz007.

Baptiste Goujaud, Adrien B. Taylor, and Aymeric Dieuleveut. Provable non-accelerations of the heavy-ball method. 2023. https://arxiv.org/abs/2307.11291.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.

Geoffrey Hinton. RMSProp: Divide the gradient by a running average of its recent magnitude. Lecture 6e, *Neural Networks for Machine Learning*, 2012. https://www.cs.utoronto.ca/~hinton/coursera_lectures.html.

Ryo Karakida and Kazuki Osawa. Understanding approximate Fisher information for fast convergence of natural gradient descent in wide neural networks. In *Advances in Neural Information Processing Systems 33*, 2020. https://papers.neurips.cc/paper_files/paper/2020/hash/7b41bfa5085806dfa24b8c9de0ce567f-Abstract.html.

Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the Polyak–Łojasiewicz condition. In *Machine Learning and Knowledge Discovery in Databases*, pages 795–811, 2016. https://arxiv.org/abs/1608.04636.

Ahmed Khaled and Chi Jin. Tuning-free stochastic optimization. In *Proceedings of the 41st International Conference on Machine Learning*, pages 23622–23661, 2024. https://proceedings.mlr.press/v235/khaled24a.html.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. https://arxiv.org/abs/1412.6980.

Huan Li, Yiming Dong, and Zhouchen Lin. On the $O(\sqrt{d}/K^{1/4})$ convergence rate of AdamW measured by ℓ_1 norm. 2025. https://arxiv.org/abs/2505.11840.

Laurent Lessard, Benjamin Recht, and Andrew Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 26(1):57–95, 2016. https://doi.org/10.1137/15M1009597.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017. https://arxiv.org/abs/1608.03983.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. https://openreview.net/forum?id=Bkg6RiCqY7.

Yurii E. Nesterov. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Doklady Akademii Nauk SSSR*, 269:543–547, 1983.

Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic Publishers, 2004. https://doi.org/10.1007/978-1-4419-8853-9.

Yurii Nesterov. Universal gradient methods for convex optimization problems. *Mathematical Programming*, 152:381–404, 2015. https://doi.org/10.1007/s10107-014-0790-0.

Brendan O'Donoghue and Emmanuel J. Candès. Adaptive restart for accelerated gradient schemes. *Foundations of Computational Mathematics*, 15(3):715–732, 2015. https://doi.org/10.1007/s10208-013-9150-3.

Boris T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964. https://doi.org/10.1016/0041-5553(64)90137-5.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of Adam and beyond. In *International Conference on Learning Representations*, 2018. https://openreview.net/forum?id=ryQu7f-RZ.

Vincent Roulet and Alexandre d'Aspremont. Sharpness, restart and acceleration. *SIAM Journal on Optimization*, 30(1):262–289, 2020. https://arxiv.org/abs/1702.03828.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *Journal of Machine Learning Research*, 19(70):1–57, 2018. https://www.jmlr.org/papers/v19/18-188.html.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1139–1147, 2013. https://proceedings.mlr.press/v28/sutskever13.html.

Hugo Touvron et al. Llama 2: Open foundation and fine-tuned chat models. 2023. https://arxiv.org/abs/2307.09288.

Sharan Vaswani, Yifan Sun, and Reza Babanezhad. Convergence of steepest descent and Adam under non-uniform smoothness. 2026. https://arxiv.org/abs/2605.30648.

Bohan Wang, Yushun Zhang, Huishuai Zhang, Qi Meng, Ruoyu Sun, Zhi-Ming Ma, Tie-Yan Liu, Zhi-Quan Luo, and Wei Chen. Provable adaptivity of Adam under non-uniform smoothness. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024. https://arxiv.org/abs/2208.09900.

Shuo Xie, Mohamad Amin Mohamadi, and Zhiyuan Li. Adam exploits ℓ_∞ geometry of loss landscape via coordinate-wise adaptivity. In *International Conference on Learning Representations*, 2025. https://proceedings.iclr.cc/paper_files/paper/2025/file/1700ad4e6252e8f2955909f96367b34d-Paper-Conference.pdf.

Pengyun Yue, Cong Fang, and Zhouchen Lin. On the lower bound of minimizing Polyak–Łojasiewicz functions. 2023. https://arxiv.org/abs/2212.13551.

Guodong Zhang, James Martens, and Roger B. Grosse. Fast convergence of natural gradient descent for over-parameterized neural networks. In *Advances in Neural Information Processing Systems 32*, 2019. https://proceedings.neurips.cc/paper_files/paper/2019/hash/1da546f25222c1ee710cf7e2f7a3ff0c-Abstract.html.

Chenyang Zhang, Difan Zou, and Yuan Cao. The implicit bias of Adam on separable data. In *Advances in Neural Information Processing Systems 37*, 2024. https://arxiv.org/abs/2406.10650.

Yushun Zhang, Congliang Chen, Tian Ding, Ziniu Li, Ruoyu Sun, and Zhi-Quan Luo. Why transformers need Adam: A Hessian perspective. In *Advances in Neural Information Processing Systems 37*, 2024. https://proceedings.neurips.cc/paper_files/paper/2024/hash/ee0e45ff4de76cbfdf07015a7839f339-Abstract-Conference.html.

Zhenxun Zhuang, Mingrui Liu, Ashok Cutkosky, and Francesco Orabona. Understanding AdamW through proximal methods and scale-freeness. *Transactions on Machine Learning Research*, 2022. https://arxiv.org/abs/2202.00089.

Fangyu Zou, Li Shen, Zequn Jie, Weizhong Zhang, and Wei Liu. A sufficient condition for convergences of Adam and RMSProp. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11127–11135, 2019. https://arxiv.org/abs/1811.09358.