

---

# Convexity and Numerical Optimization through Adam

ECE 57000 — October 5, 2026

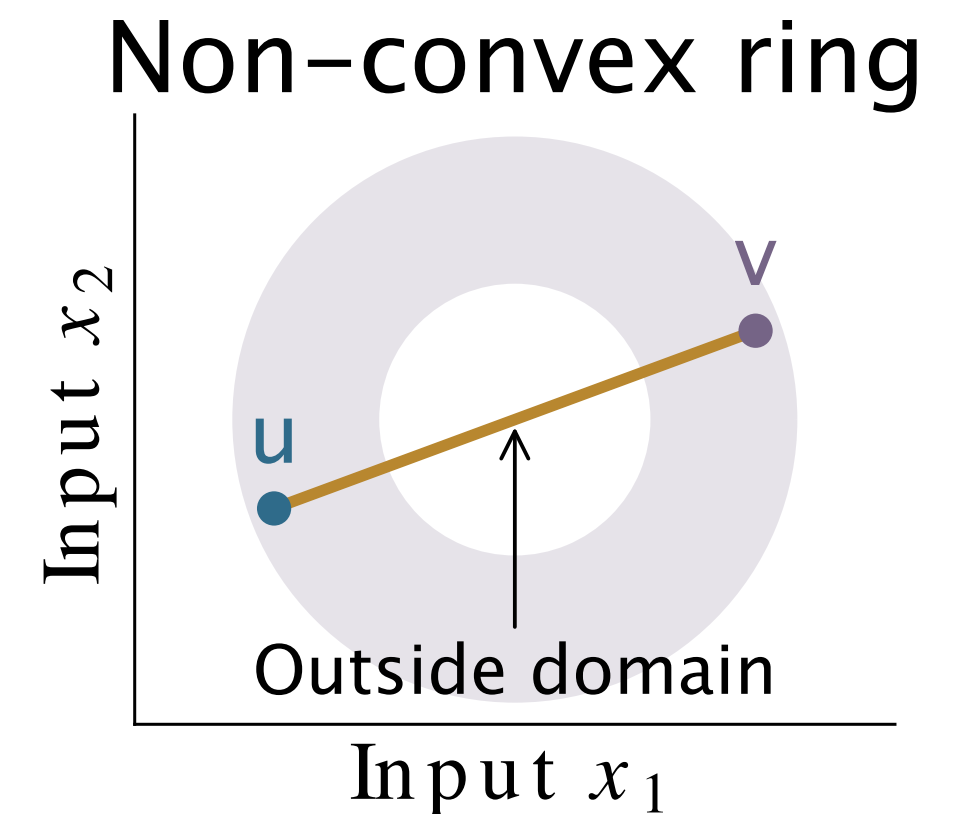
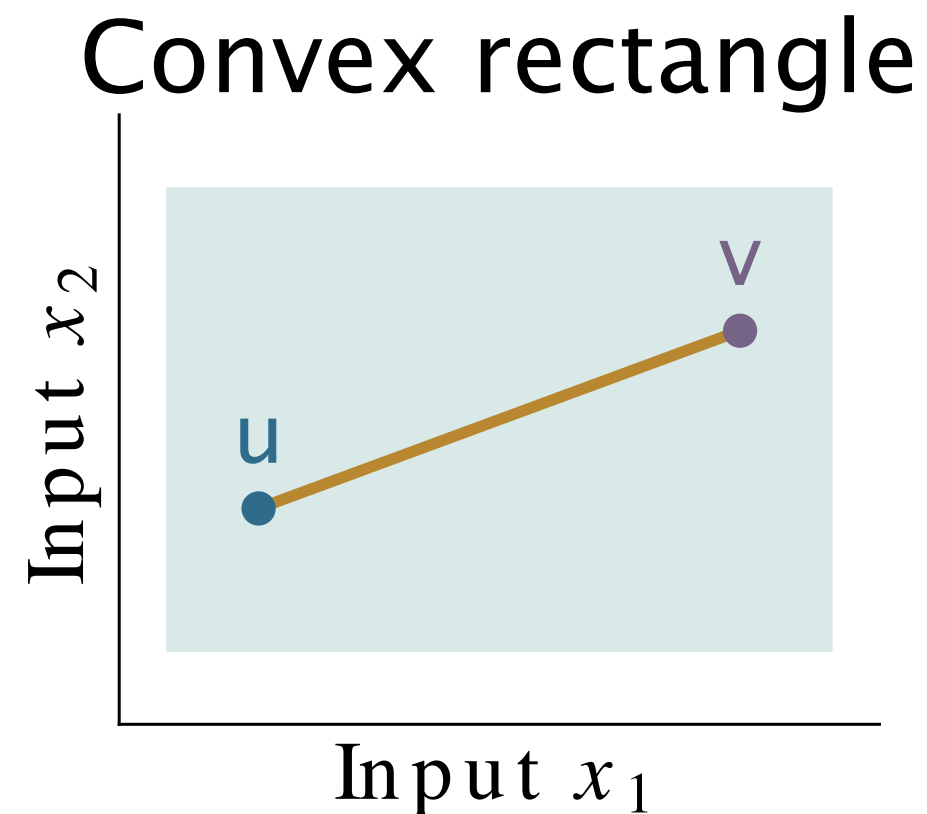
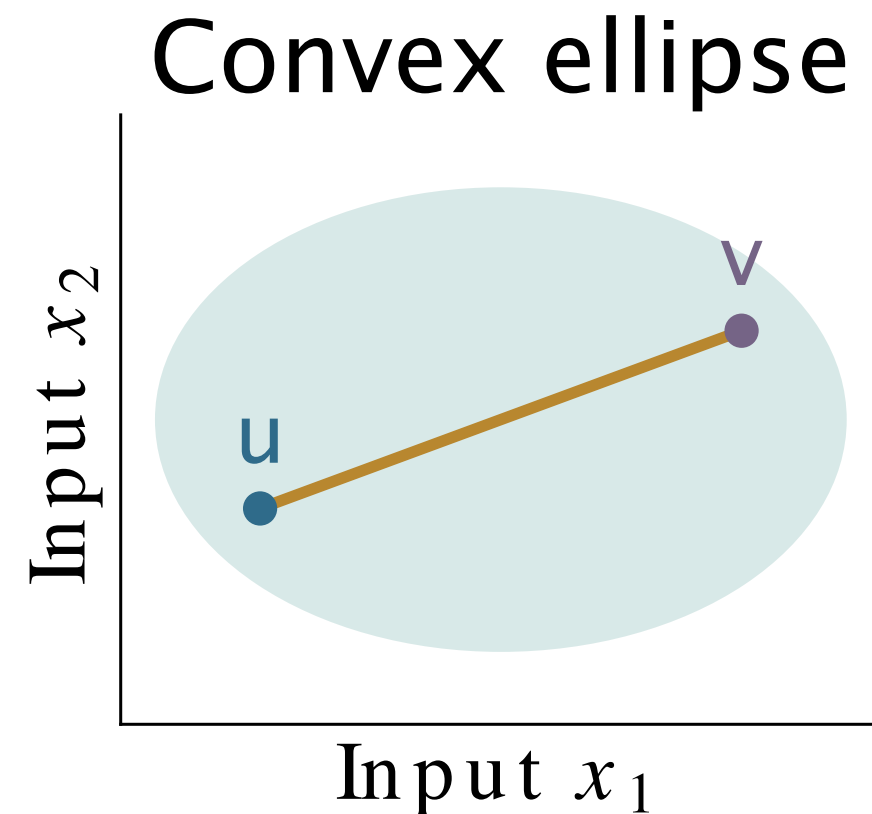
David I. Inouye

# | Friday connected **model, loss, and objective shape**

- OLS gives an optimal training fit; redundant features can make coefficients nonunique.
- Logistic regression maps a linear score to a probability and fits it with log loss.
- Convexity means the loss at an averaged input lies below the average endpoint loss.

**Today:** use that definition to understand optimality, then compute useful updates.

# A convex domain contains every segment between its points

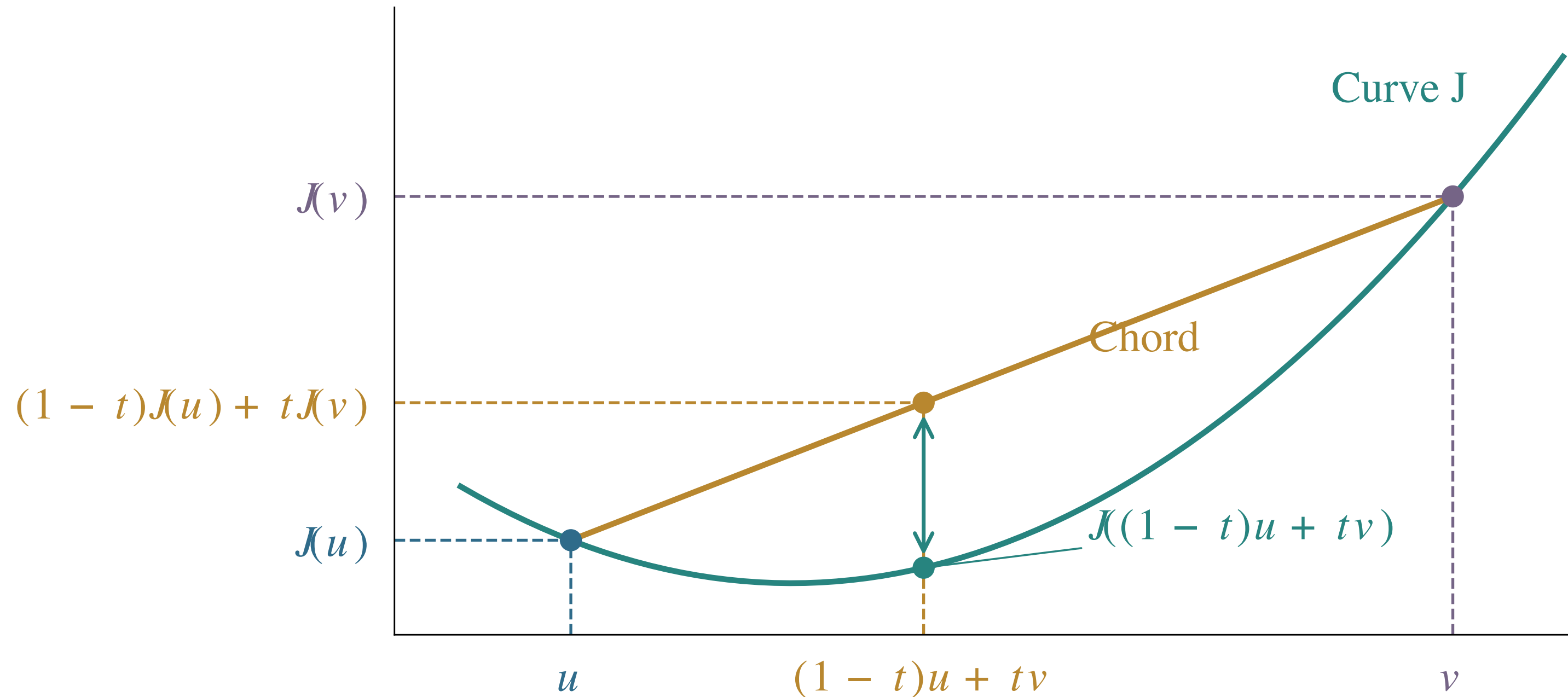


A set  $\mathcal{D} \subseteq \mathbb{R}^d$  is **convex** if

$$\forall u, v \in \mathcal{D}, \quad \forall t \in [0, 1], \quad (1 - t)u + tv \in \mathcal{D}.$$

**Visual segment test:** the entire segment must stay inside the domain for **every pair**.

# | Definition: a convex function lies below every chord



$J$  is **convex** on a convex domain if, for **every**  $u, v$  in the domain and  $0 \leq t \leq 1$ ,

$$J\left((1-t)u + tv\right) \leq (1-t)J(u) + tJ(v).$$

**Loss at the averaged input  $\leq$  average of the endpoint losses.**

# | Convexity makes **every local minimum global**

**Proof by contradiction.** Let  $J$  be convex on a convex domain  $\mathcal{D}$ .

Local minimality at  $\mathbf{u}$  means: for some  $r > 0$ ,

$$J(\mathbf{x}) \geq J(\mathbf{u}) \quad \text{whenever } \mathbf{x} \in \mathcal{D}, \|\mathbf{x} - \mathbf{u}\| < r.$$

Suppose  $\mathbf{u}$  is **not global**: some  $\mathbf{v} \in \mathcal{D}$  has  $J(\mathbf{v}) < J(\mathbf{u})$ .

Choose  $0 < t < \min\{1, r/\|\mathbf{v} - \mathbf{u}\|\}$  and set  $\mathbf{x}_t = (1 - t)\mathbf{u} + t\mathbf{v}$ .

The segment stays in  $\mathcal{D}$ , and  $\|\mathbf{x}_t - \mathbf{u}\| = t\|\mathbf{v} - \mathbf{u}\| < r$ .

$$J(\mathbf{x}_t) \leq (1 - t)J(\mathbf{u}) + tJ(\mathbf{v}) \quad (\text{convexity})$$

$$< (1 - t)J(\mathbf{u}) + tJ(\mathbf{u}) = J(\mathbf{u}) \quad (\text{strictly lower loss, } t > 0).$$

**Contradiction:** a lower value lies inside the local-minimum neighborhood. Thus  $\mathbf{u}$  is global.

# For a twice differentiable function, **convexity means nonnegative curvature**

Let  $J$  be twice continuously differentiable on an **open convex domain**.

| Parameters          | Equivalent criterion for convexity                    |
|---------------------|-------------------------------------------------------|
| One scalar $\theta$ | $J''(\theta) \geq 0$ everywhere                       |
| A vector $\theta$   | $H(\theta) = \nabla^2 J(\theta) \succeq 0$ everywhere |

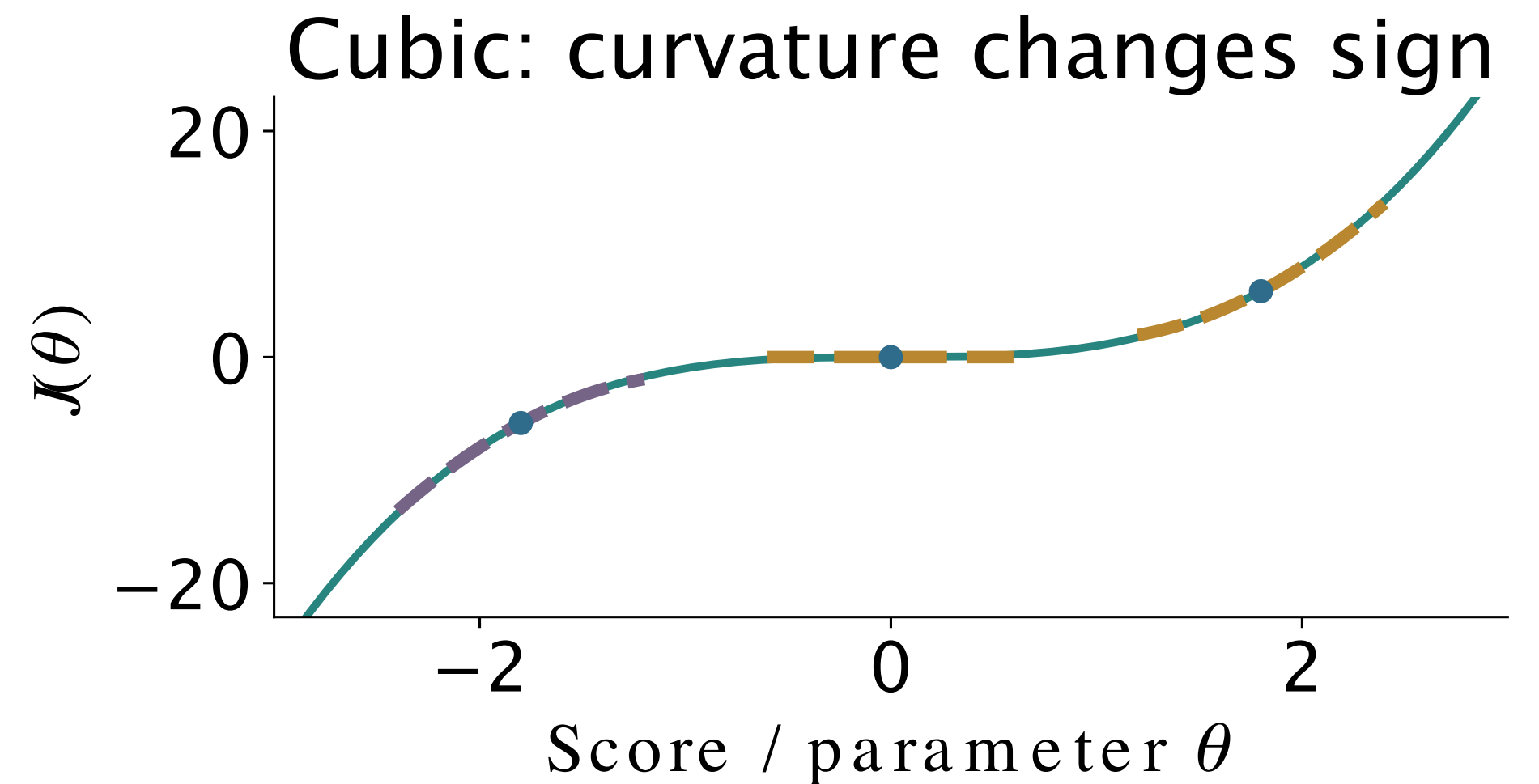
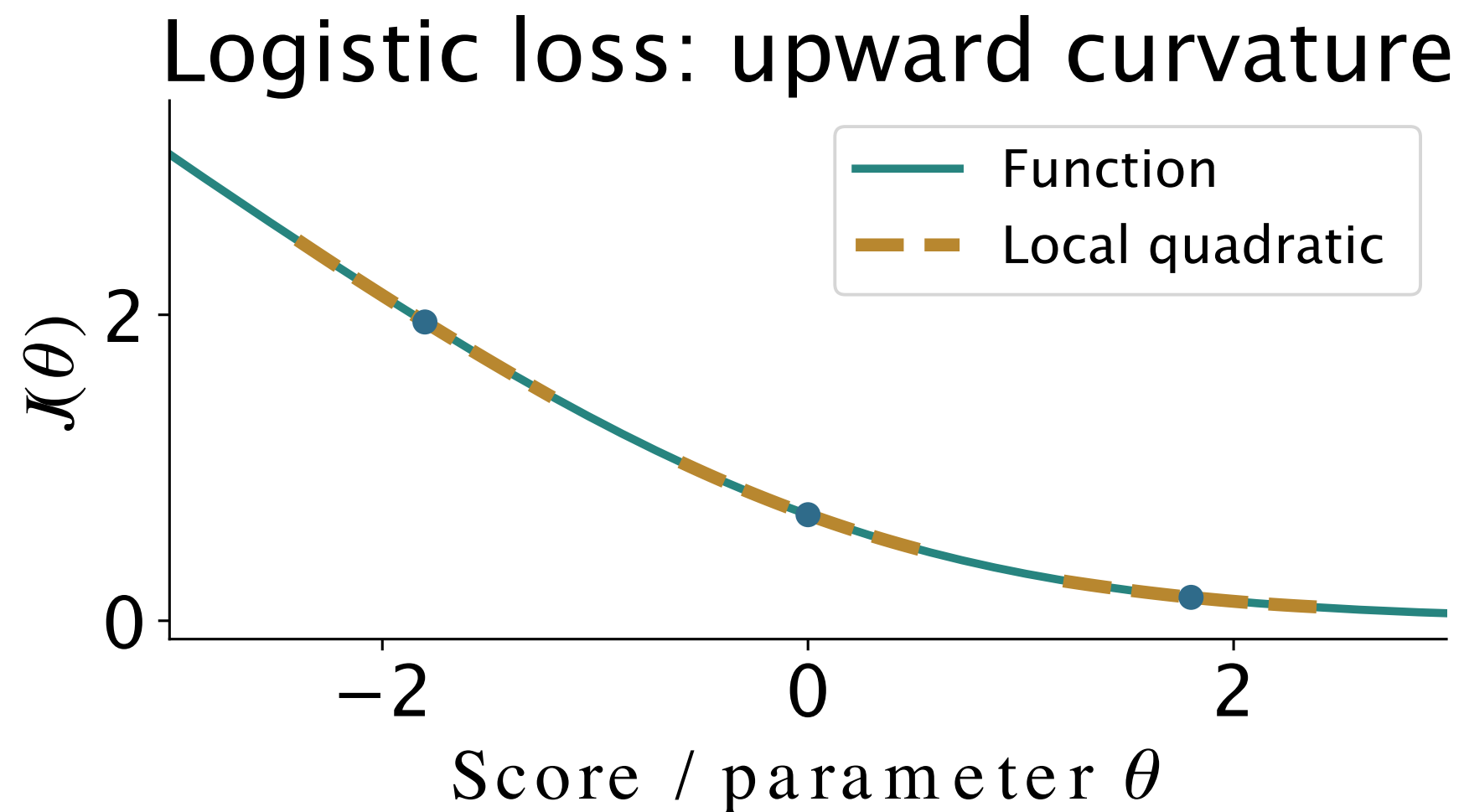
The **Hessian** is the matrix of second partial derivatives:  $H_{jk} = \frac{\partial^2 J}{\partial \theta_j \partial \theta_k}$ .

**Positive semidefinite (PSD)** means  $\mathbf{v}^\top H \mathbf{v} \geq 0$  for every direction  $\mathbf{v}$ . Equivalently, **all eigenvalues of  $H$  are  $\geq 0$** : no eigenvector direction curves downward; zero allows a flat direction at that point.

# Local quadratic approximations reveal the sign of curvature

Near any point  $\theta_0$ , with  $h = \theta - \theta_0$ ,

$$J(\theta_0 + h) = J(\theta_0) + J'(\theta_0)h + \underbrace{\frac{1}{2}J''(\theta_0)h^2 + o(h^2)}_{\text{local quadratic approximation}}.$$



# | OLS and logistic log loss have **nonnegative directional curvature**

Append the intercept to  $\widetilde{X}$  and the coefficients to  $\theta$ .

| Objective         | Hessian $H = \nabla^2 J$                                                               |
|-------------------|----------------------------------------------------------------------------------------|
| OLS               | $\frac{1}{n} \widetilde{X}^\top \widetilde{X}$                                         |
| Logistic log loss | $\frac{1}{n} \widetilde{X}^\top W \widetilde{X}$ , with $W_{ii} = a_i(1 - a_i) \geq 0$ |

For logistic loss,  $\ell(z, y) = \log(1 + e^z) - yz$  gives  $\ell' = \sigma(z) - y$  and  $\ell'' = a(1 - a)$ .

$$\boldsymbol{v}^\top H \boldsymbol{v} = \frac{1}{n} \sum_i a_i(1 - a_i) (\widetilde{\boldsymbol{x}}_i^\top \boldsymbol{v})^2 \geq 0.$$

For OLS, replace the weights by 1. **Both objectives are convex in the coefficients.**

# | Convexity makes a local optimum **globally meaningful**

- **Local minimum** → **global minimum**. Zero gradient also certifies a global minimum for a differentiable convex objective.
- **Uniqueness needs more**: strict convexity excludes multiple minimizers, if a minimizer exists.
- **Existence needs more**: a convex loss can approach its infimum without attaining it.

**Next:** a useful shape does not itself tell us how to compute the solution.

---

# Fitting models with local information

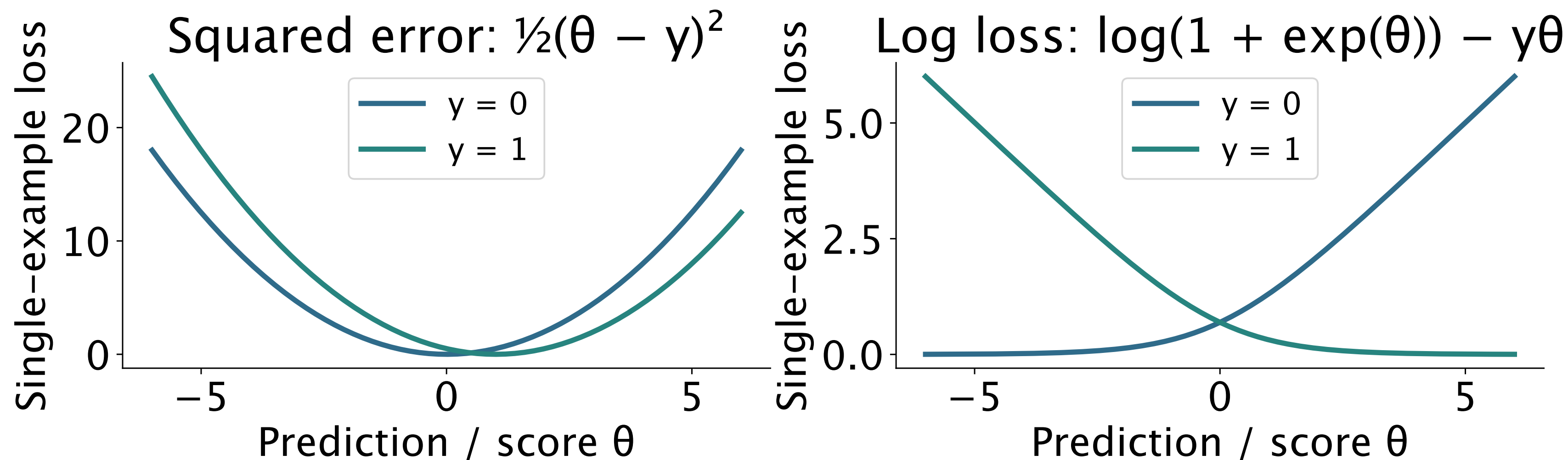
| A well-behaved shape still requires **a numerical route to the solution**

**OLS:** solve linear normal equations.

**Logistic regression:** repeatedly evaluate and improve the coefficients.

The objective is convex; our next task is to turn **local information into useful steps**.

## | Logistic loss has flatter tails than squared error



- Each curve is the loss for **one example**, with its label fixed at  $y = 0$  or  $y = 1$ .
- Squared error has constant curvature; log loss flattens toward a confident correct prediction.
- For  $y = 1$ , log loss decreases as the score grows; for  $y = 0$ , it decreases as the score falls.

# | A derivative predicts the effect of a **small parameter change**

**First-order Taylor expansion centered at  $\theta$** , for a small change  $\Delta$ :

$$J(\theta + \Delta) \approx J(\theta) + J'(\theta)\Delta.$$

- If  $J'(\theta) > 0$ , a small **negative** change reduces the objective.
- If  $J'(\theta) < 0$ , a small **positive** change reduces it.
- This is a **local approximation**, not a prediction for an arbitrary step.

# | A gradient collects the **partial derivatives**

Bundle the parameters into  $\boldsymbol{\theta} = (w_1, \dots, w_d, b)^\top$ .

$$\nabla J(\boldsymbol{\theta}) = [\partial J / \partial w_1 \quad \dots \quad \partial J / \partial w_d \quad \partial J / \partial b]^\top.$$

For a small vector change  $\boldsymbol{\Delta}$ :

$$J(\boldsymbol{\theta} + \boldsymbol{\Delta}) \approx J(\boldsymbol{\theta}) + \nabla J(\boldsymbol{\theta})^\top \boldsymbol{\Delta}.$$

The gradient has **one component per parameter**.

# | Moving against the gradient gives **local descent**

For  $\alpha > 0$ , choose  $\Delta = -\alpha \nabla J(\theta)$ .

$$\nabla J(\theta)^\top \Delta = -\alpha \|\nabla J(\theta)\|_2^2 \leq 0.$$

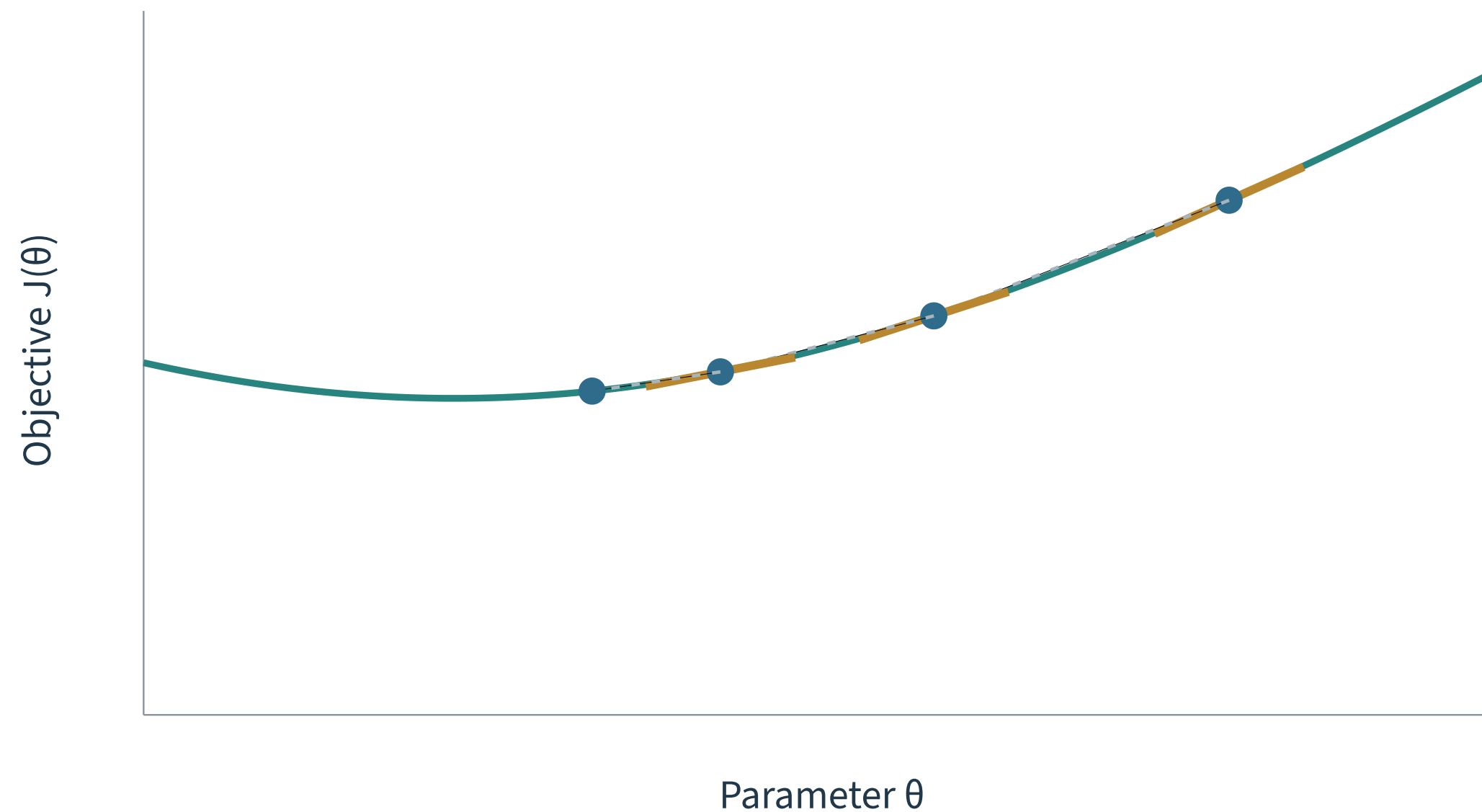
This motivates the repeated update

$$\theta_{t+1} = \theta_t - \alpha \nabla J(\theta_t).$$

**Next:** inspect what these finite steps do when we can see only the local slope.

# | Small steps follow the local downhill direction

Same objective, same starting point. **Learning rate:**  $\alpha = 2.0$ .



Full unregularized logistic loss — revealed last

The algorithm sees **this point and this slope**.

A negative-gradient step uses that local information.

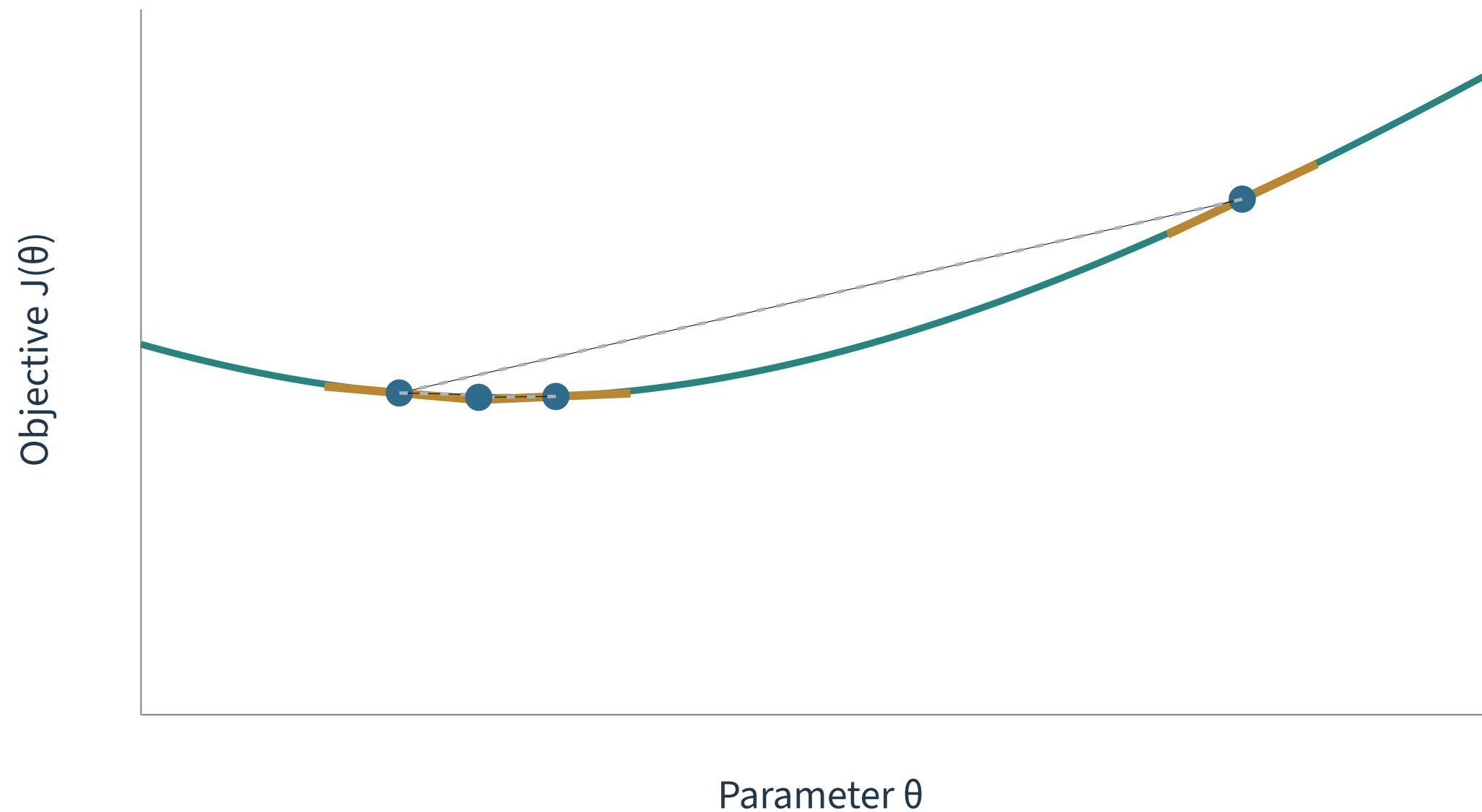
Gold segments show only the slopes we have evaluated.

The objective falls; the slope keeps its sign.

Three steps approach the minimum.

# | What happens after this larger step?

Same objective, same starting point. **Learning rate:**  $\alpha = 6.0$ .



Full unregularized logistic loss — revealed last

**After the next update, what happens?**

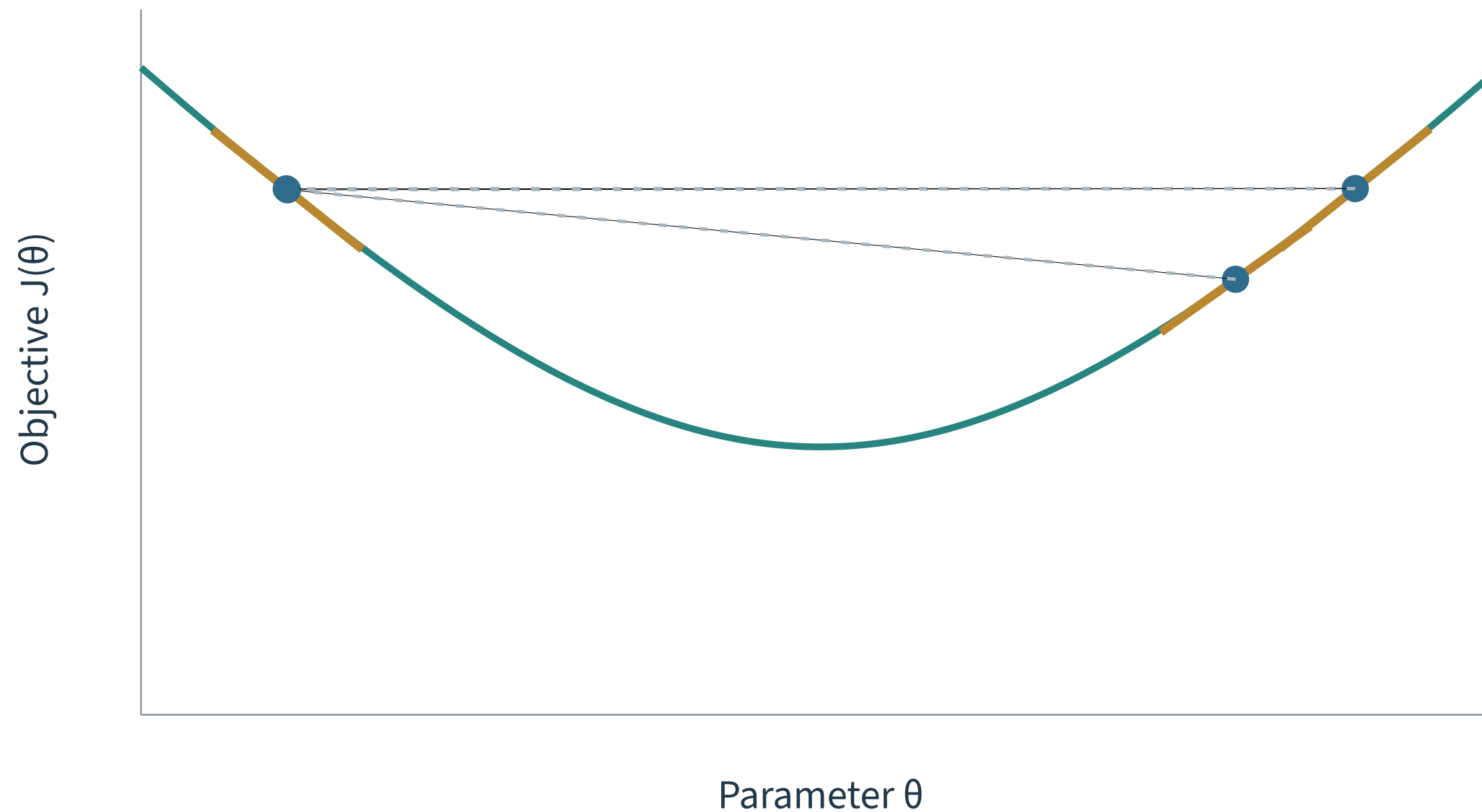
- A. Objective decreases; slope keeps its sign.
- B. Objective decreases; slope reverses sign.
- C. Objective increases; slope keeps its sign.
- D. Objective increases; slope reverses sign.

**B.** The objective falls, but the slope reverses: we jumped across the valley.

Three steps approach the minimum.

# | What happens when we take an even larger step?

Same objective, same starting point. **Learning rate:**  $\alpha = 12.0$ .



Full unregularized logistic loss — revealed last

## After the next update, what happens?

- A. Objective decreases; slope keeps its sign.
- B. Objective decreases; slope reverses sign.
- C. Objective increases; slope keeps its sign.
- D. Objective increases; slope reverses sign.

**D.** The objective rises and the slope reverses: we jumped too far.

Continuing these updates settles into an oscillation instead of reaching the minimum.

# | The gradient is an **average of example contributions**

If  $J(\boldsymbol{\theta}) = \frac{1}{n} \sum_i \ell_i(\boldsymbol{\theta})$ , then

$$\nabla J(\boldsymbol{\theta}) = \frac{1}{n} \sum_i \nabla \ell_i(\boldsymbol{\theta}).$$

| Update             | Examples used for one gradient estimate |
|--------------------|-----------------------------------------|
| Full-batch GD      | All $n$ examples                        |
| Single-example SGD | One randomly sampled example            |
| Mini-batch SGD     | A small randomly sampled group $B$      |

All use  $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha_t \mathbf{g}_t$ .

# | Sampling trades exact direction for **cheaper, noisy updates**

For uniform sampling at the current parameters:

$$\mathbf{g}_B = \frac{1}{|B|} \sum_{i \in B} \nabla \ell_i(\boldsymbol{\theta}), \quad \mathbb{E}_B[\mathbf{g}_B \mid \boldsymbol{\theta}] = \nabla J(\boldsymbol{\theta}).$$

- One stochastic step can increase the full training objective.
- Larger batches usually reduce gradient noise but cost more per update.
- Compare runs by **examples processed or time**, not only update count.

A fixed learning rate can leave persistent fluctuations near the minimum.

# Adam averages directions and adapts each coordinate scale

For the current gradient estimate  $\mathbf{g}_t$ :

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \mathbf{g}_t, \quad \mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2) \mathbf{g}_t^2.$$

The hats correct the initial bias from  $\mathbf{m}_0 = \mathbf{v}_0 = \mathbf{0}$ :

$$\widehat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1 - \beta_1^t}, \quad \widehat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1 - \beta_2^t}.$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \frac{\widehat{\mathbf{m}}_t}{\sqrt{\widehat{\mathbf{v}}_t} + \epsilon}.$$

**Average the recent directions; rescale coordinates with large recent squared gradients.**

The base learning rate still matters. Adaptation is not a convergence guarantee.