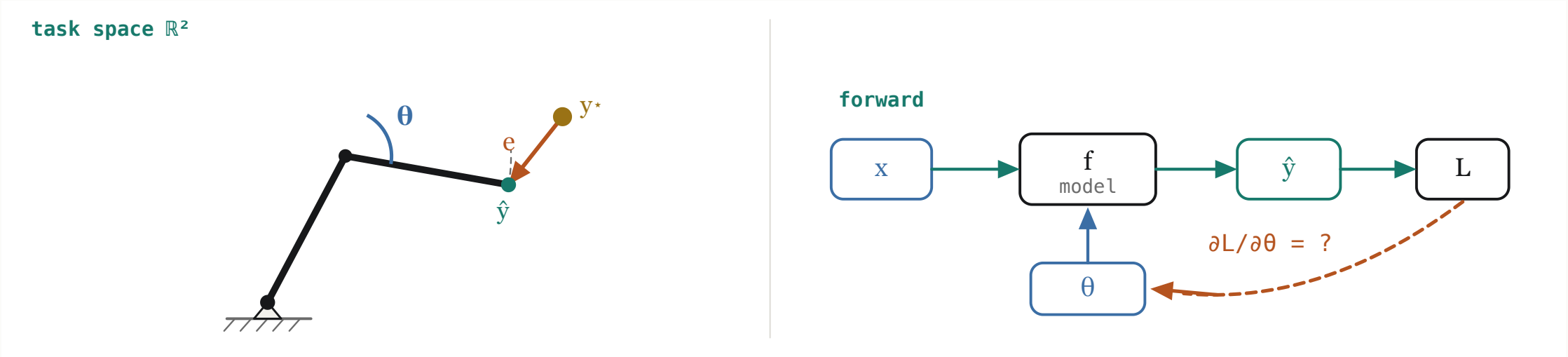
From Prediction Error to Parameter Sensitivity

How Does an Error Reach the Parameters?



$$\hat{y} = f_\theta(x), \qquad L = \tfrac{1}{2}\|\hat{y} - y^\star\|_2^2$$

- x, θ the input and the parameters
- $y^\star$ fixed target, not differentiated
- L one number, not a vector

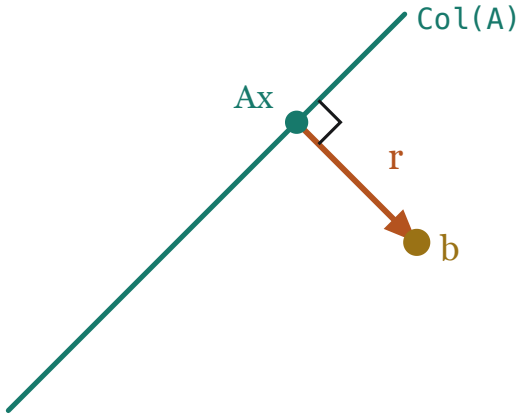
> What does L do if a parameter moves?

RECALL

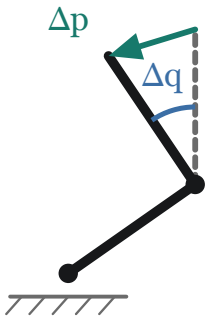
From Prediction Error to Parameter Sensitivity

What We Already Know, and What We Need Next

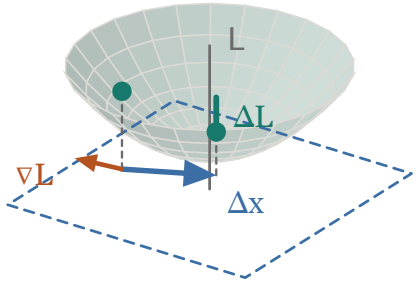
output space $\mathbb{R}^2$



joint space → task space



parameter space $\mathbb{R}^2$



$$r = b - Ax \qquad \Delta p \approx J(q)\,\Delta q \qquad \Delta L \approx \nabla L^{\mathsf{T}} \Delta x$$

r a projection's exact leftover $J(q)$ joint change to tip change

∇L one number per parameter $\approx$ first order, small steps only

> Backprop produces ∇L. Moving the parameters is a separate job.

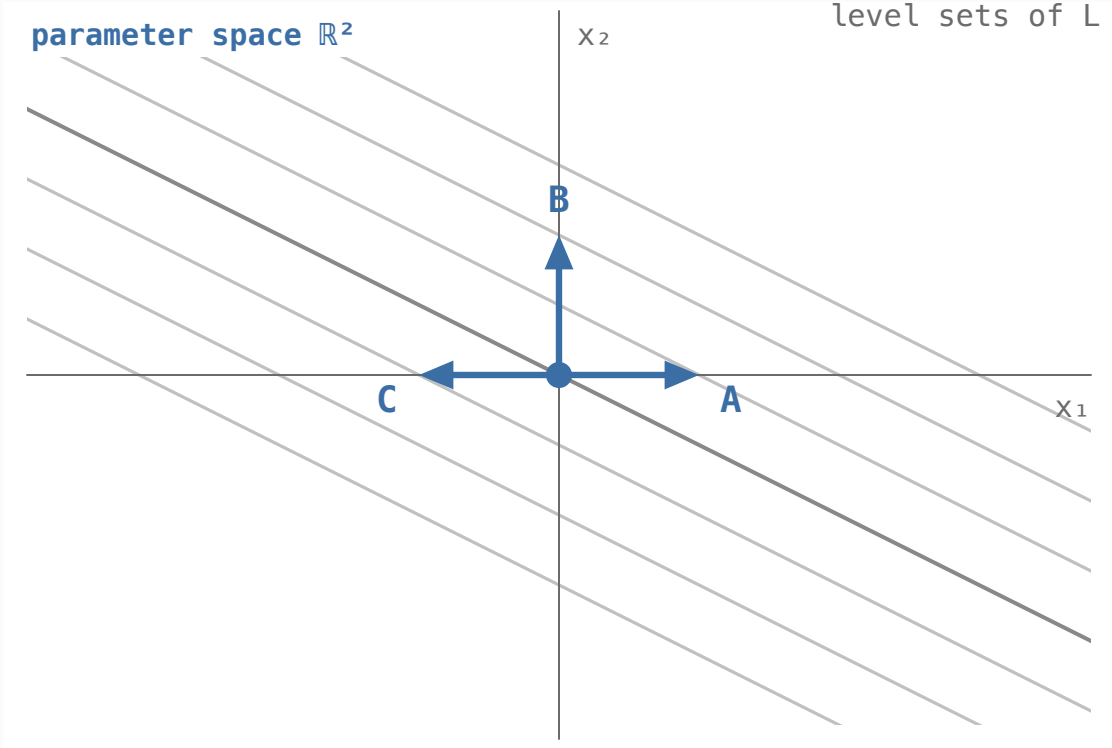
VOTE

From Prediction Error to Parameter Sensitivity

Which Small Move Reduces the Error?

$L = \frac{1}{2}(x_1 + 2x_2 - 3)^2$, now at $x = (0, 0)$. Which step of length $\varepsilon = 0.1$ lowers L the most?

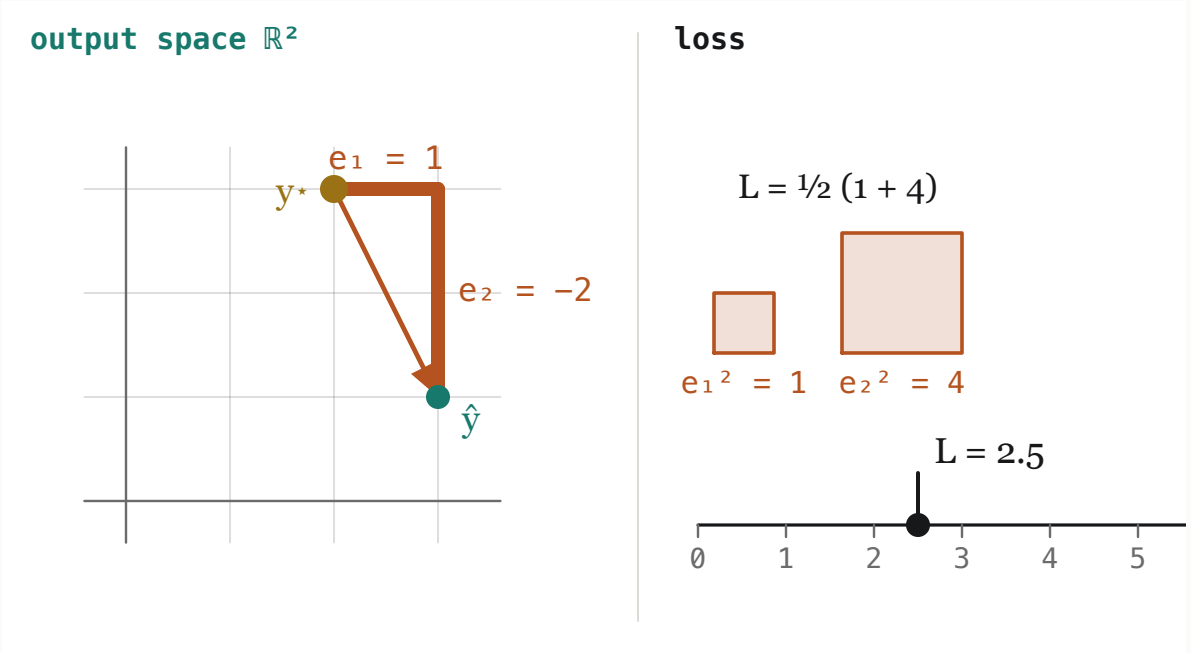
- **A** $(\varepsilon, 0)$
- **B** $(0, \varepsilon)$
- **C** $(-\varepsilon, 0)$



DEFINITION

From Prediction Error to Parameter Sensitivity

A Loss Turns Many Errors into One Number



$$e = \hat{y} - y^\star, \qquad L = \tfrac{1}{2}\|e\|_2^2 = \tfrac{1}{2} e^\mathsf{T} e = \tfrac{1}{2}\sum_i e_i^2$$

- $e \in \mathbb{R}^m$ prediction minus target
- $L \in \mathbb{R}$ one number, and never negative
- $\frac{1}{2}$ convenience: it moves values, not minimisers

WORKED EXAMPLE

$e = (1, -2)$: $L = \frac{1}{2}(1^2 + 2^2) = 2.5$.

Lecture 3 measured $r = y^\star - \hat{y}$, so $e = -r$.

VOTE

From Prediction Error to Parameter Sensitivity

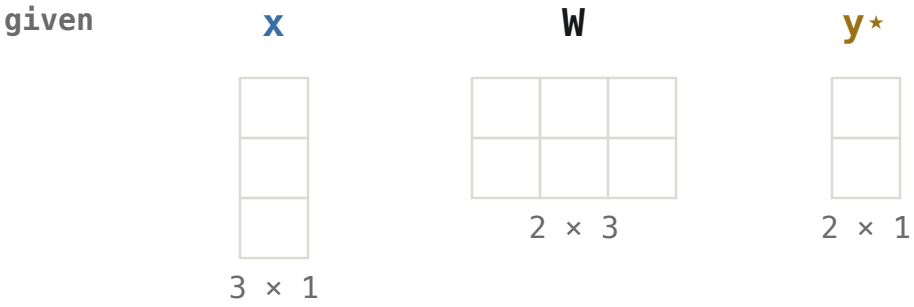
Scalar, Vector, or Matrix?

$x \in \mathbb{R}^3$, $W \in \mathbb{R}^{2\times 3}$, $\hat{y} = Wx$, $L = \frac{1}{2}\|\hat{y} - y^\star\|^2$. What shapes are L, $\nabla_x L$ and $\nabla_W L$?

A $1, 3 \times 1, 2 \times 3$

B $2 \times 1, 3 \times 1, 3 \times 2$

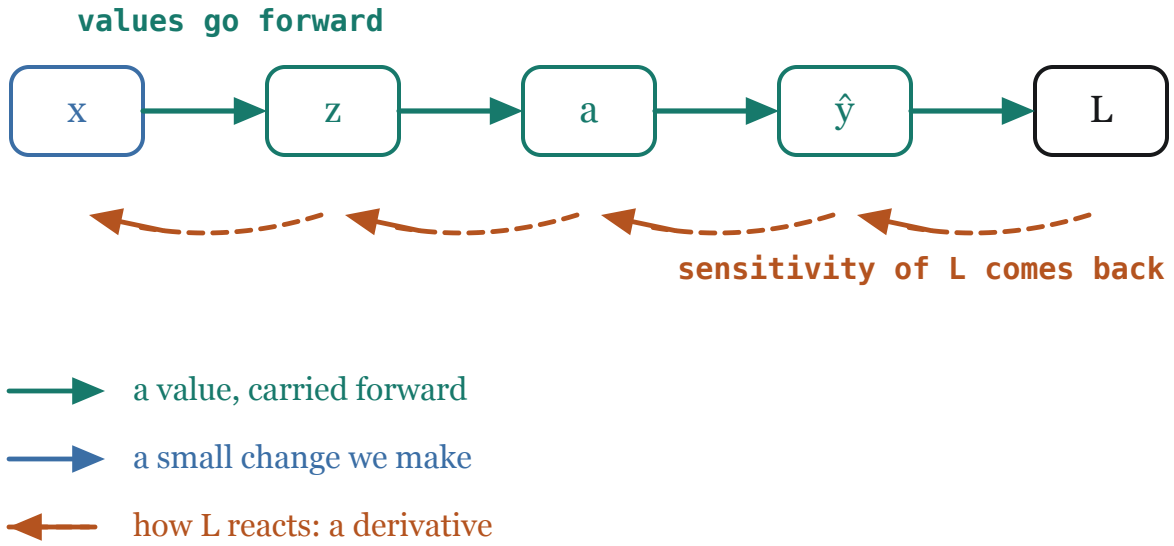
C $1, 1 \times 3, 3 \times 2$



ROADMAP

From Prediction Error to Parameter Sensitivity

One Question, Three Ways to Check the Answer



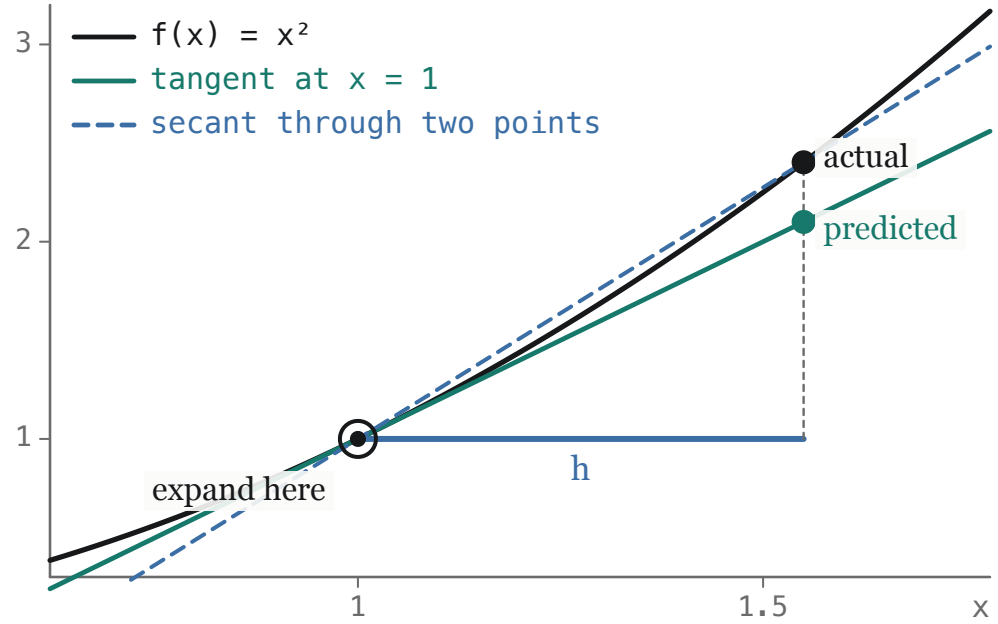
1. Local slopes, gradients, Jacobians
2. The chain rule, in both directions
3. Differentials for vectors and matrices
4. Graphs, then a whole small network

> One derivative, three checks: a tensor program, autograd, a finite difference.

DEFINITION

Derivatives as Local Linear Models

A Derivative Is a Local Slope



$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

$$f(x+h) = f(x) + f'(x)h + o(|h|)$$

- f differentiable at the expansion point
- h a small step, either sign
- $f'(x)$ the slope at the expansion point
- $o(|h|)$ the rest: divided by $|h|$, it tends to 0

VOTE

Derivatives as Local Linear Models

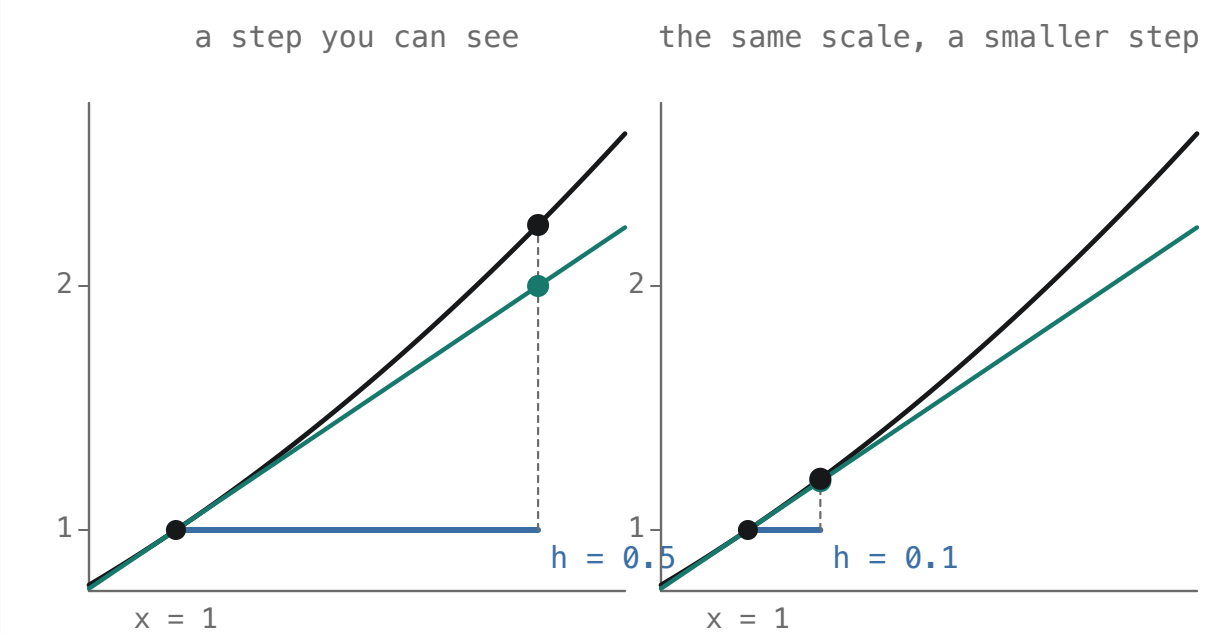
Does the Tangent Prediction Become Exact?

$f(x) = x^2$ at $x = 1$. The tangent predicts $1 + 2h$; the curve gives $(1 + h)^2$. How does actual minus predicted behave?

A like h

B like h^2

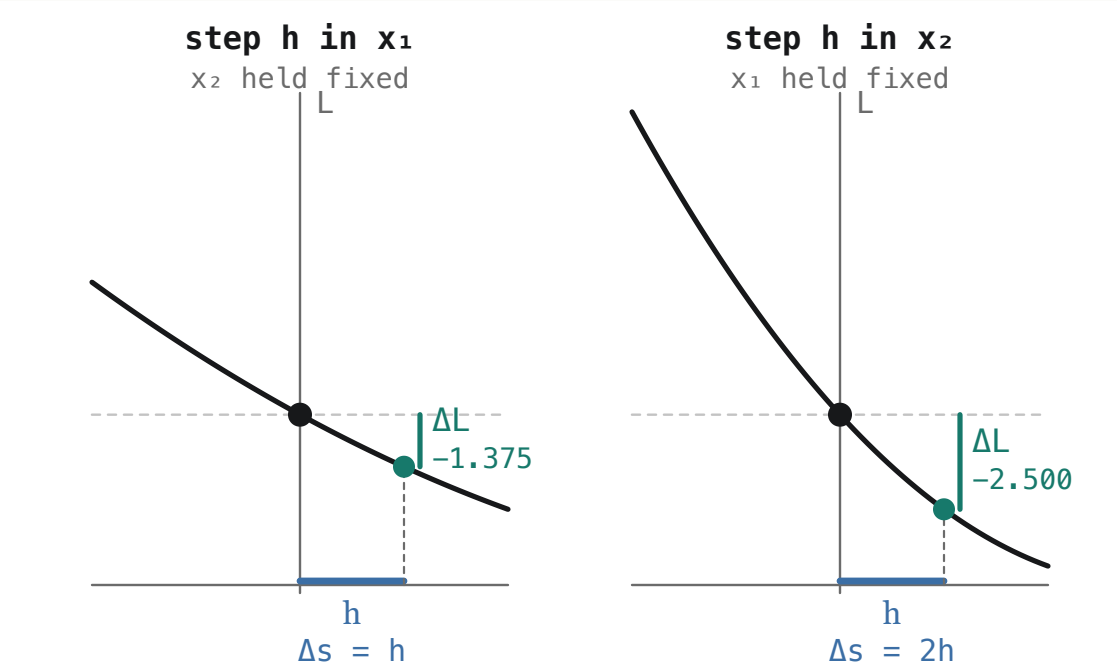
C zero for every h



DERIVATION

Derivatives as Local Linear Models

Partial Derivatives Hold Other Coordinates Fixed



$$\tfrac{1}{2}(s+\delta)^2 - \tfrac{1}{2}s^2 = s\,\delta + \tfrac{1}{2}\delta^2$$

s — $x_1 + 2x_2 - 3$, the value inside

$\delta = \Delta s$ — the change the step makes in s

`first-order term` — $s\delta$; it vanishes when $s = 0$

Step h in x_1: $\delta = h$, so $\partial_1 L = s$.

Step h in x_2: $\delta = 2h$, so $\partial_2 L = 2s$.

VOTE

Derivatives as Local Linear Models

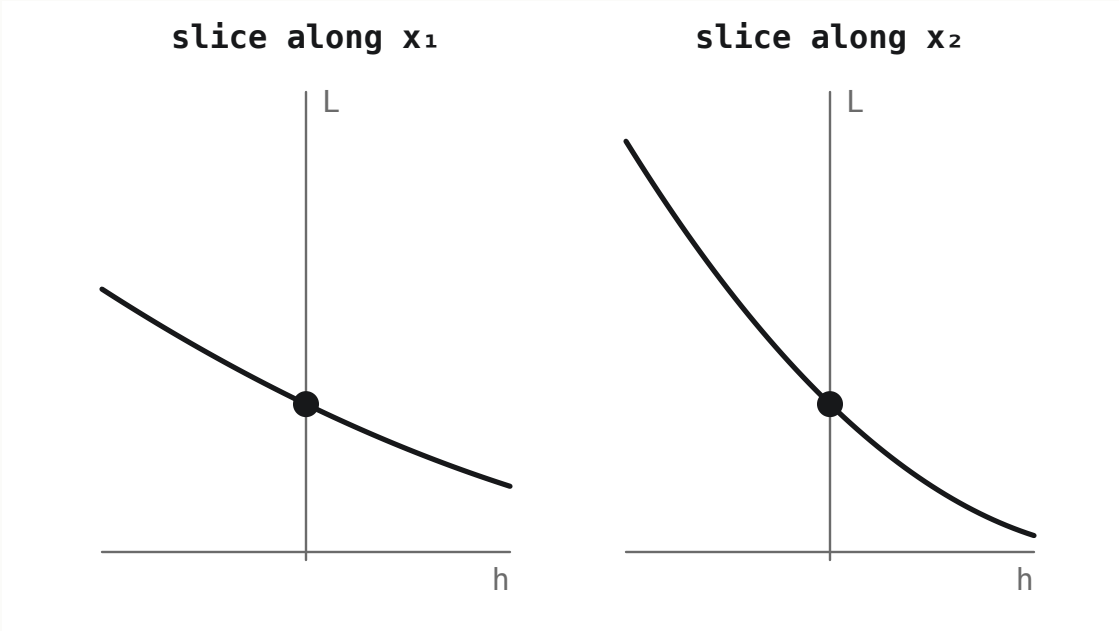
Which Coordinate Has Greater Local Sensitivity?

At $x = (0, 0)$, what is $\nabla L = (\partial_1 L,\ \partial_2 L)^\top$?

A $(-3, -6)$

B $(-3, -3)$

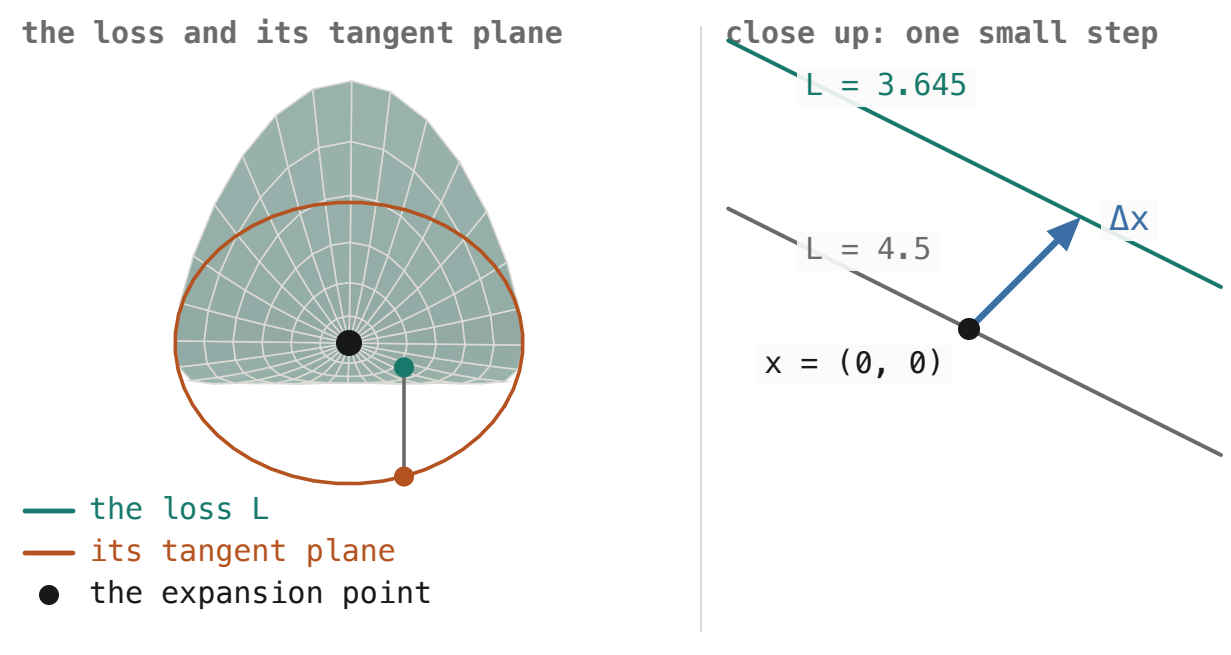
C $(3, 6)$



DERIVATION

Derivatives as Local Linear Models

The Gradient Predicts a Small Loss Change



$$dL = \nabla L(x)^\mathsf{T} dx$$

$$L(x + \Delta x) \approx L(x) + \nabla L(x)^\mathsf{T} \Delta x$$

$\nabla L(x)$	partials at x
dL	loss change predicted by the linear model
$\nabla L^\mathsf{T} \Delta x$	predicted change
$+ L(x)$	predicted new loss

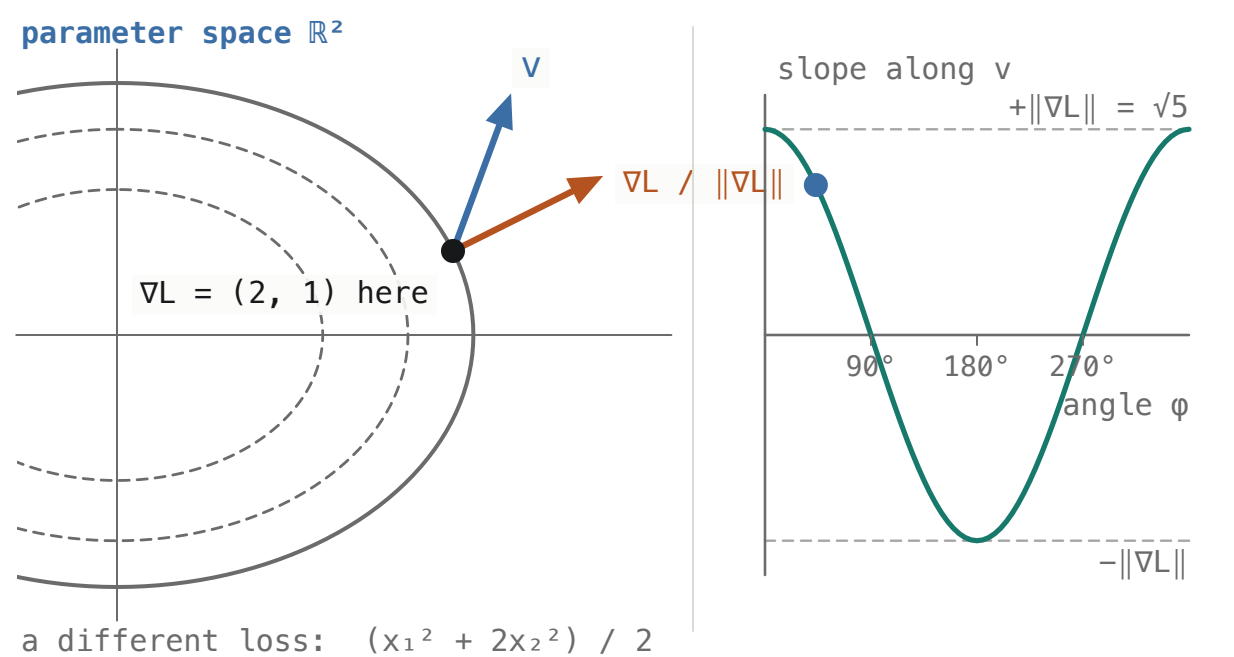
NUMERICAL CHECK

$\Delta x = (0.1, 0.1)$: predicted -0.9, actual -0.855.

DERIVATION

Derivatives as Local Linear Models

Directional Derivatives and Steepest Ascent



$$D_v L = \frac{d}{dt} L(x + tv)\Big|_{t=0} = \nabla L^\mathsf{T} v = \|\nabla L\|_2 \cos\phi$$

- v $\|v\|_2 = 1$
- ϕ angle from v to ∇L
- $\nabla L \neq 0$ largest at $v = \nabla L / \|\nabla L\|_2$

WORKED EXAMPLE

New loss: $\frac{1}{2}(x_1^2 + 2x_2^2)$ at $(2, \frac{1}{2})$, $\nabla L = (2, 1)$, $D_v L = \sqrt{5}\cos 60^\circ \approx 1.118$.

VOTE

Derivatives as Local Linear Models

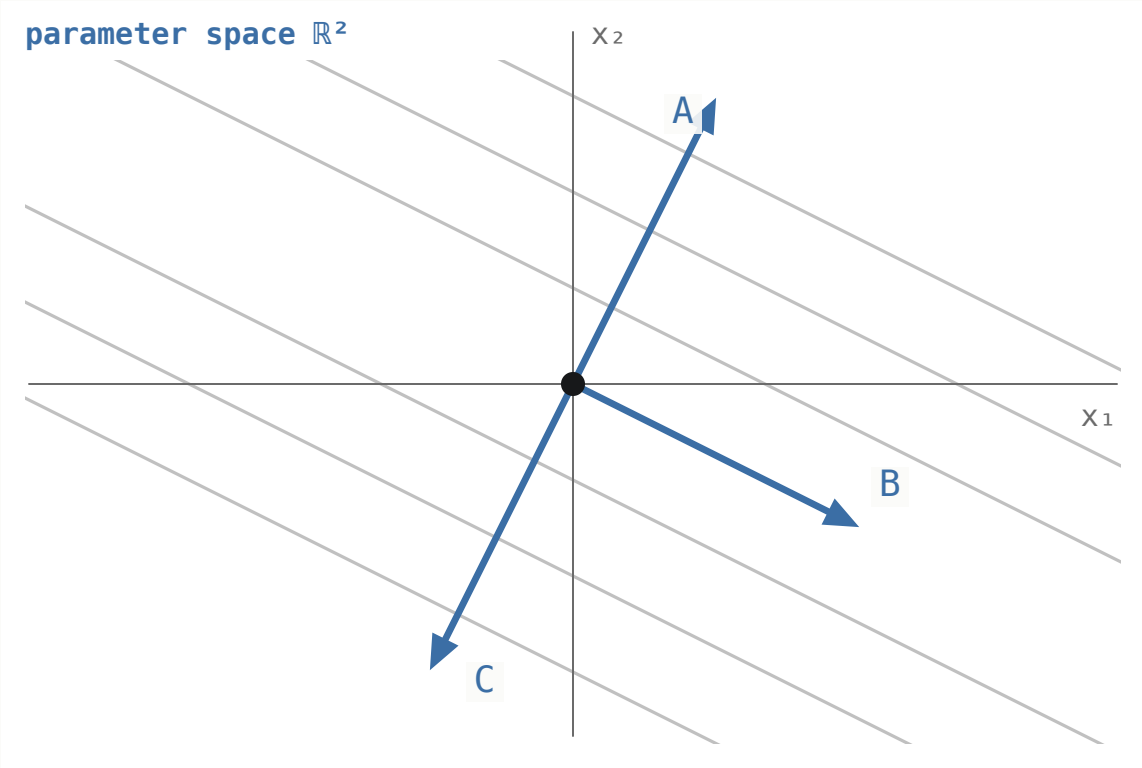
Which Direction Is Locally Flat?

Back at $x = (0, 0)$. Which unit direction has $D_v L = 0$?

A $(1, 2)/\sqrt{5}$

B $(2, -1)/\sqrt{5}$

C $(-1, -2)/\sqrt{5}$

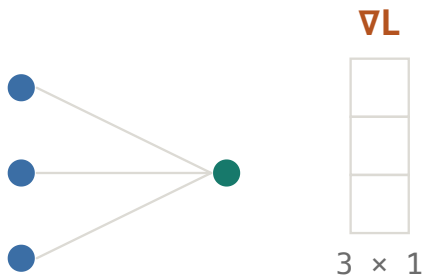


SYNTHESIS

Derivatives as Local Linear Models

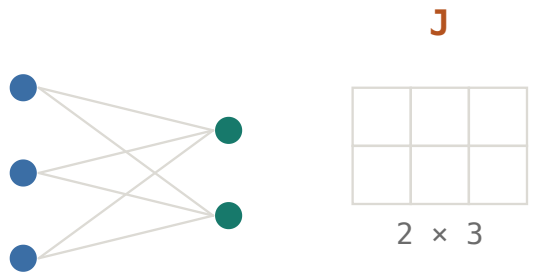
Gradient, Jacobian, and Hessian Have Different Jobs

$\mathbb{R}^n \to \mathbb{R}$



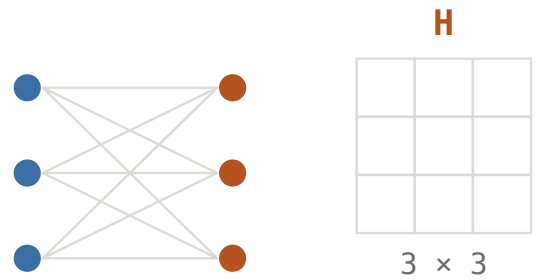
one number out: a sensitivity per input

$\mathbb{R}^n \to \mathbb{R}^m$



each output against each input

$\nabla L : \mathbb{R}^n \to \mathbb{R}^n$



how the gradient itself changes

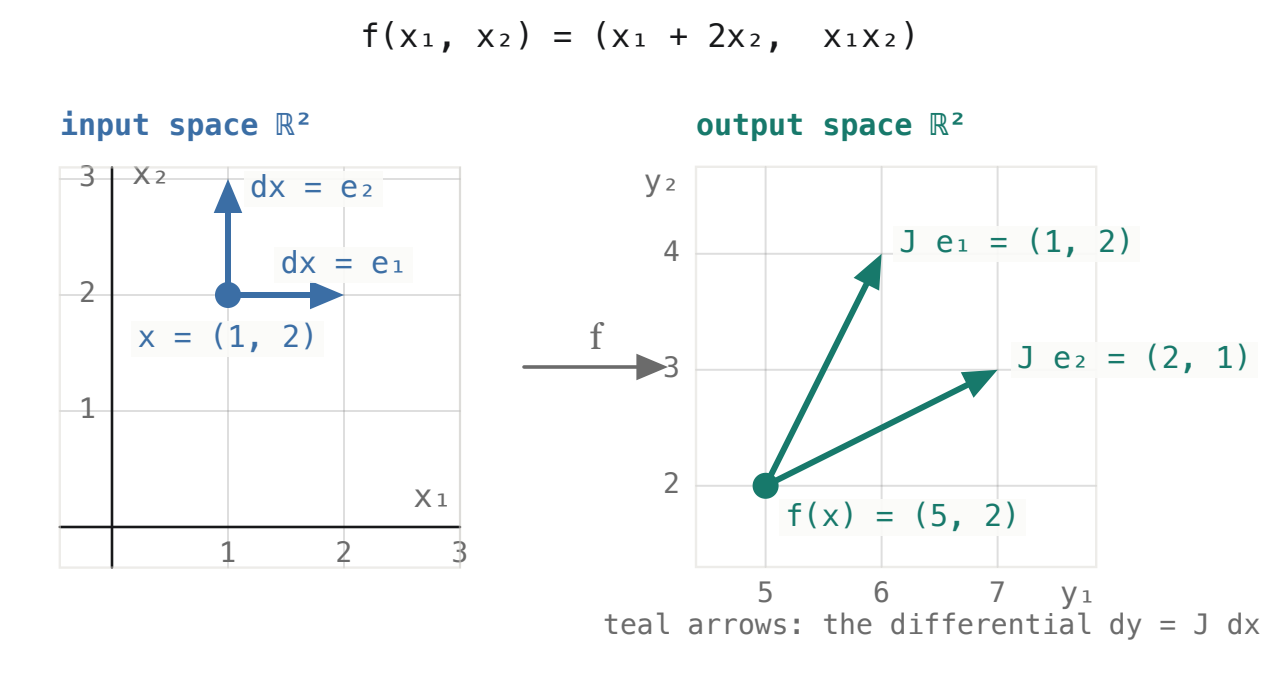
$$J_L = (\nabla L)^\mathsf{T}, \qquad H = \partial(\nabla L)/\partial x$$

- ∇L $n \times 1$: the shape of x
- H $n \times n$: the gradient map differentiated
- J_f $m \times n$ for $f : \mathbb{R}^n \to \mathbb{R}^m$
- $\nabla L = 0$ stationary, and not always minimal

DEFINITION

Jacobians and the Chain Rule

Columns of the Jacobian Are Input Effects



$$dy = J_f\, dx, \qquad (J_f)_{ij} = \frac{\partial f_i}{\partial x_j}$$

x, dx	input and its increment, in $\mathbb{R}^n$
y, dy	output and its increment, in $\mathbb{R}^m$
J_f	$m \times n$, and it changes with x

WORKED EXAMPLE

At $x = (1, 2)$: $f(x) = (5, 2)$, $J_f = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$, $dy = J_f e_1 = (1, 2)$.

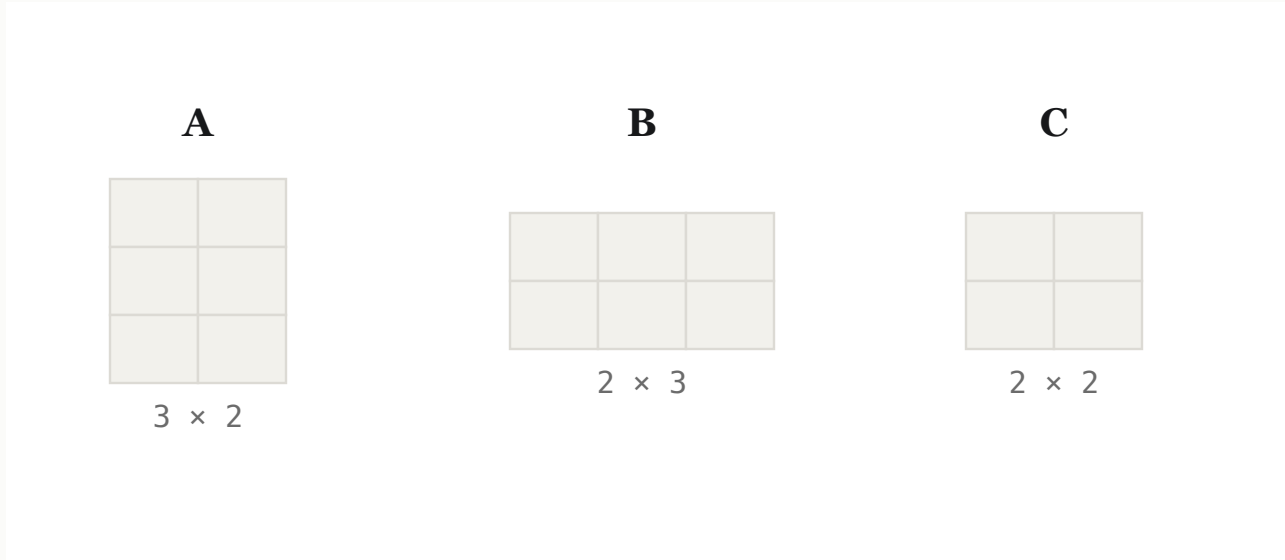
VOTE

Jacobians and the Chain Rule

Which Matrix Has the Right Shape?

$f : \mathbb{R}^3 \to \mathbb{R}^2$. Which shape must J_f have?

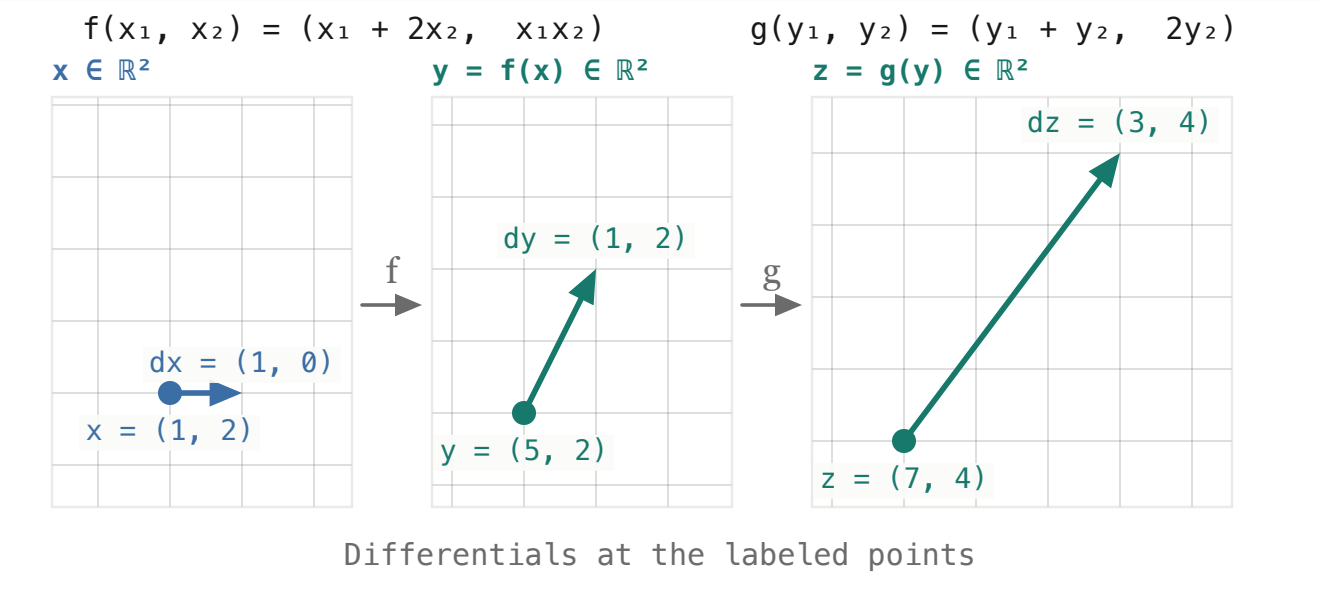
- (A) 3×2
- (B) 2×3
- (C) 2×2



$J_f = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$

Jacobians and the Chain Rule

Follow a Perturbation Through Two Maps



$$dy = J_f\, dx$$

$$dz = J_g\, dy = J_g(J_f\, dx) = (J_g J_f)\, dx$$

J_f at $x = (1, 2)$

J_g from $g = (y_1 + y_2, 2y_2)$ at $y = (5, 2)$

$J_g J_f$ dx to dz

$dx = e_1$ gives $dy = (1, 2)$

and then $dz = (3, 4)$

$$J_g J_f = \begin{bmatrix} 1 & 1 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 3 \\ 4 & 2 \end{bmatrix}$$

DERIVATION

Jacobians and the Chain Rule

Matrix Multiplication Is the Multivariable Chain Rule

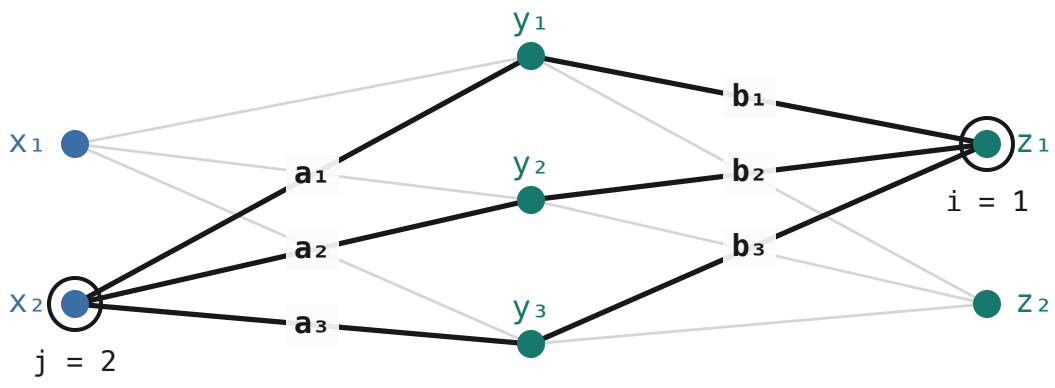
$$\frac{\partial z_i}{\partial x_j} = \sum_k \frac{\partial z_i}{\partial y_k} \frac{\partial y_k}{\partial x_j}$$

k runs over every intermediate coordinate

shapes $J_g : p \times m$, $J_f : m \times n$, product $p \times n$

order g after f gives $J_g J_f$, not the reverse

Multiply along a route, add across routes.



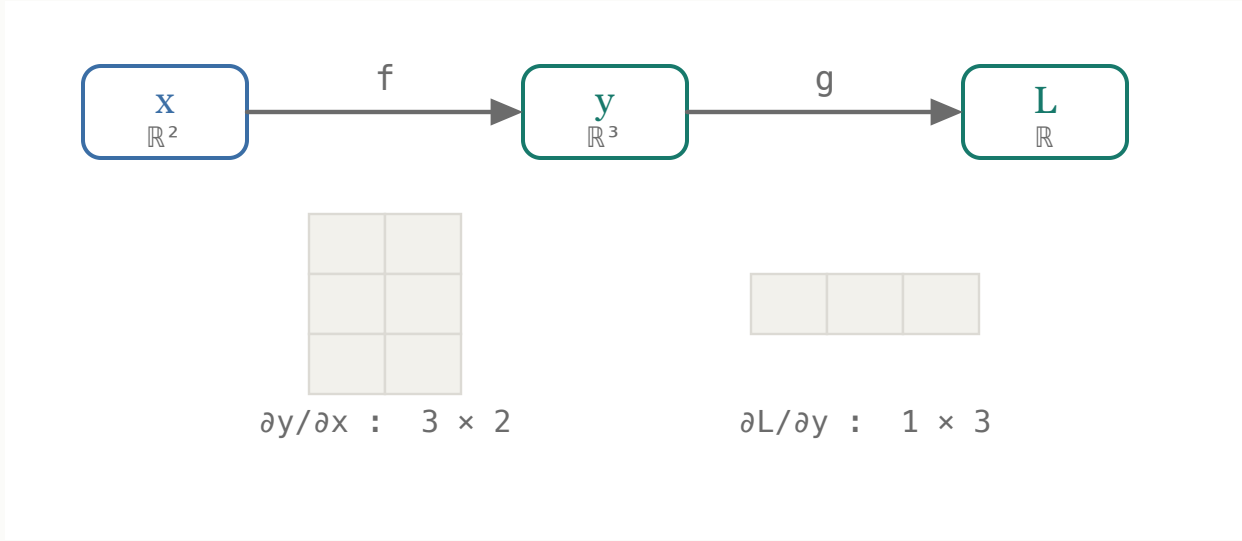
VOTE

Jacobians and the Chain Rule

Which Order Propagates the Input Change?

$f : \mathbb{R}^2 \to \mathbb{R}^3$ then $g : \mathbb{R}^3 \to \mathbb{R}$, with $L = g(f(x))$.
Which product is $\partial L / \partial x$?

- (A) $J_f J_g$
- (B) $J_g J_f$
- (C) $J_g + J_f$



DERIVATION

Jacobians and the Chain Rule

A Scalar Loss Sends Sensitivity Backward

$$dL = (\nabla_y L)^\top dy$$

dy an output increment, $\mathbb{R}^m$

$\nabla_y L$ output sensitivity, $\mathbb{R}^m$

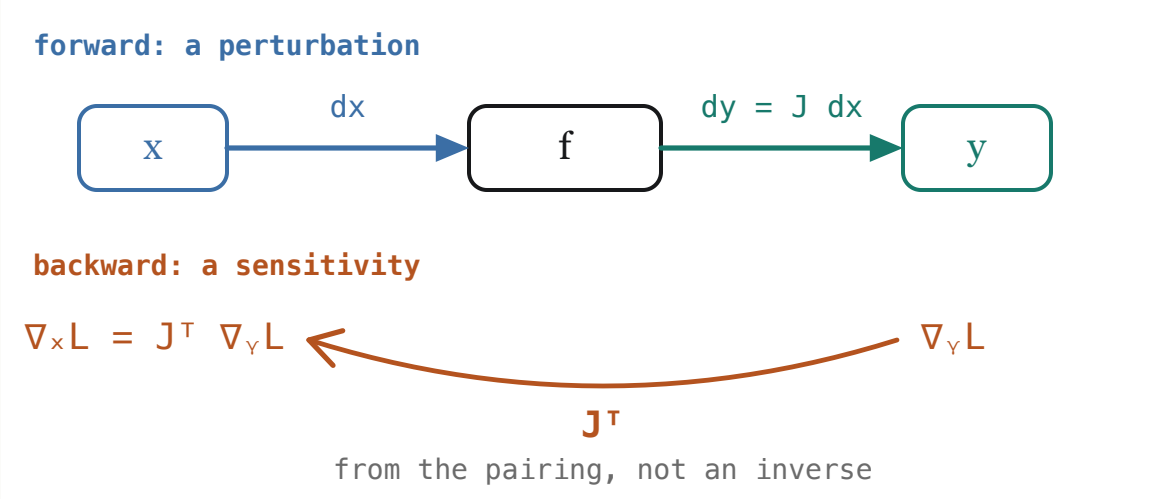
$\nabla_x L$ input sensitivity, $\mathbb{R}^n$

$= (\nabla_y L)^\top J_f\, dx$: put $dy = J_f\, dx$ in

$= (J_f^\top \nabla_y L)^\top dx$: move J_f across the pairing

since $(J_f^\top \nabla_y L)^\top = (\nabla_y L)^\top J_f$

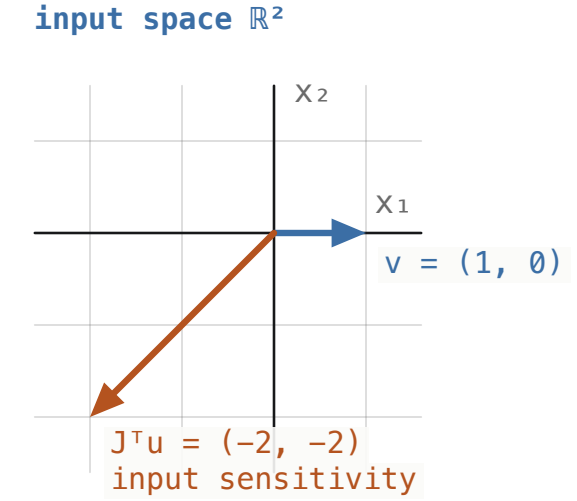
$\nabla_x L = J_f^\top \nabla_y L$: the coefficient of dx



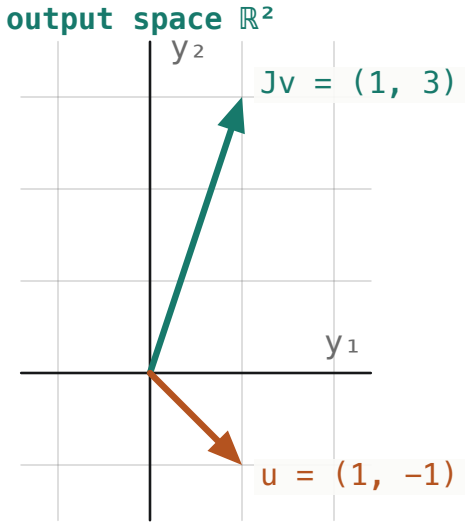
DEFINITION

Jacobians and the Chain Rule

JVP and VJP Answer Different Questions



JVP: an input direction forward.



VJP: an output sensitivity back.

$$Jv \in \mathbb{R}^m, \qquad u^\mathsf{T} J = (J^\mathsf{T} u)^\mathsf{T}$$

Jv	JVP: v, an input direction, pushed forward
$u^\mathsf{T} J$	VJP: $u = \nabla_y L$, an output loss sensitivity, pulled back
both	available without ever forming J

WORKED EXAMPLE

$J = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$: $Jv = (1, 3)$ for $v = (1, 0)$, and $J^\mathsf{T} u = (-2, -2)$ for $u = (1, -1)$.

VOTE

Jacobians and the Chain Rule

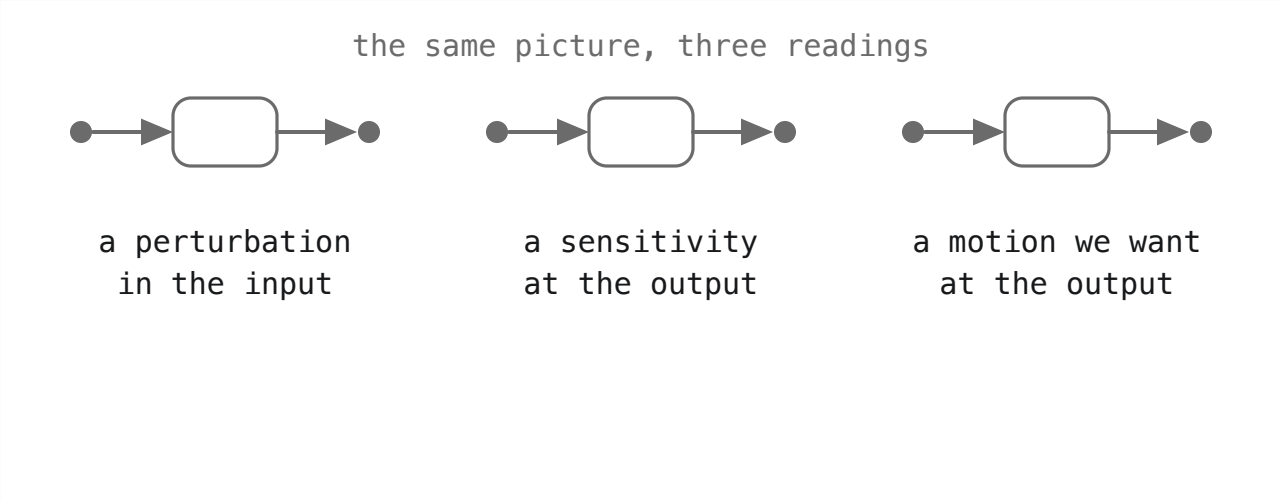
Transpose or Pseudoinverse?

J is 2×3 and $g = \nabla_y L \in \mathbb{R}^2$. Which product is $\nabla_x L$?

A Jg

B $J^\top g$

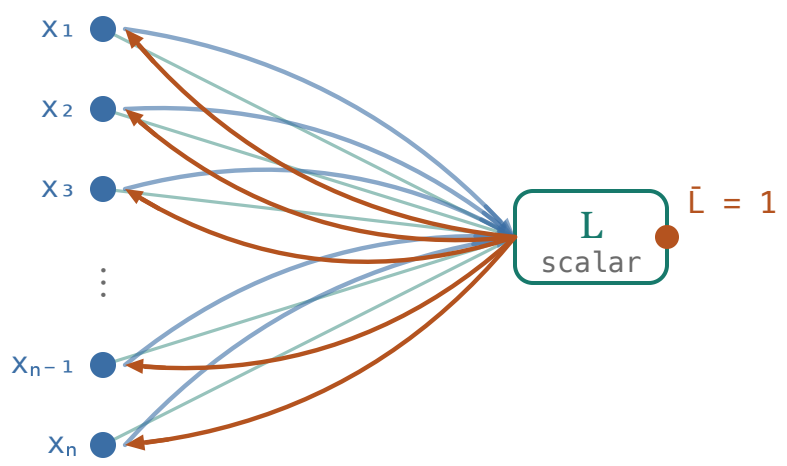
C $J^+ g$



IN PRACTICE

Jacobians and the Chain Rule

Why a Scalar Loss Favors Reverse Mode



forward mode
one seed per input
sweeps: n

reverse mode
one seed at the loss
sweeps: 1
after a forward pass

$$\bar{L} = 1 \;\Rightarrow\; \nabla_x L \text{ in one reverse sweep}$$

$\bar{L}$	the seed $\partial L / \partial L = 1$
cost	one forward sweep per input direction, one reverse sweep per output direction
memory	reverse mode saves or recomputes the forward values its local rules need

Both modes are the same chain rule.

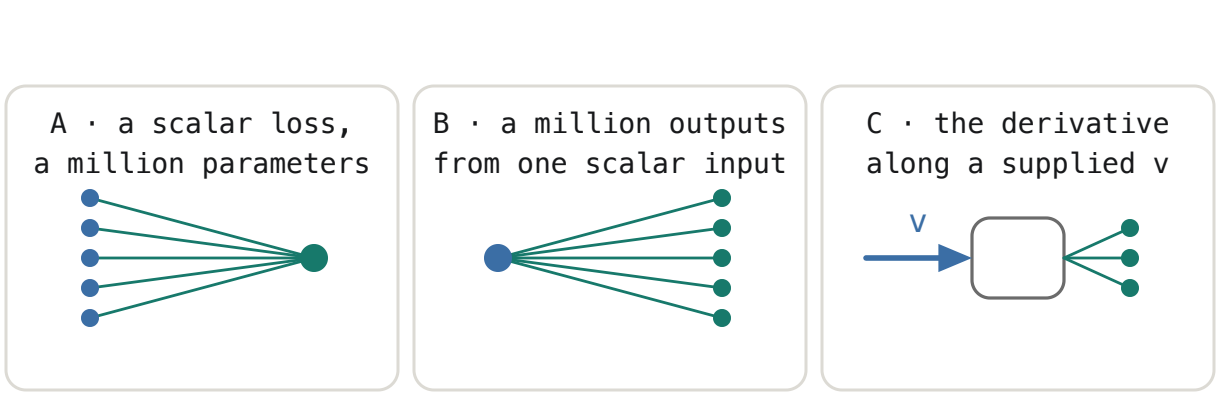
VOTE

Jacobians and the Chain Rule

Choose a Differentiation Direction

Assign reverse mode or one forward JVP to jobs A, B and C.

1. A reverse · B forward · C forward
2. A forward · B reverse · C reverse
3. A reverse · B reverse · C reverse



METHOD

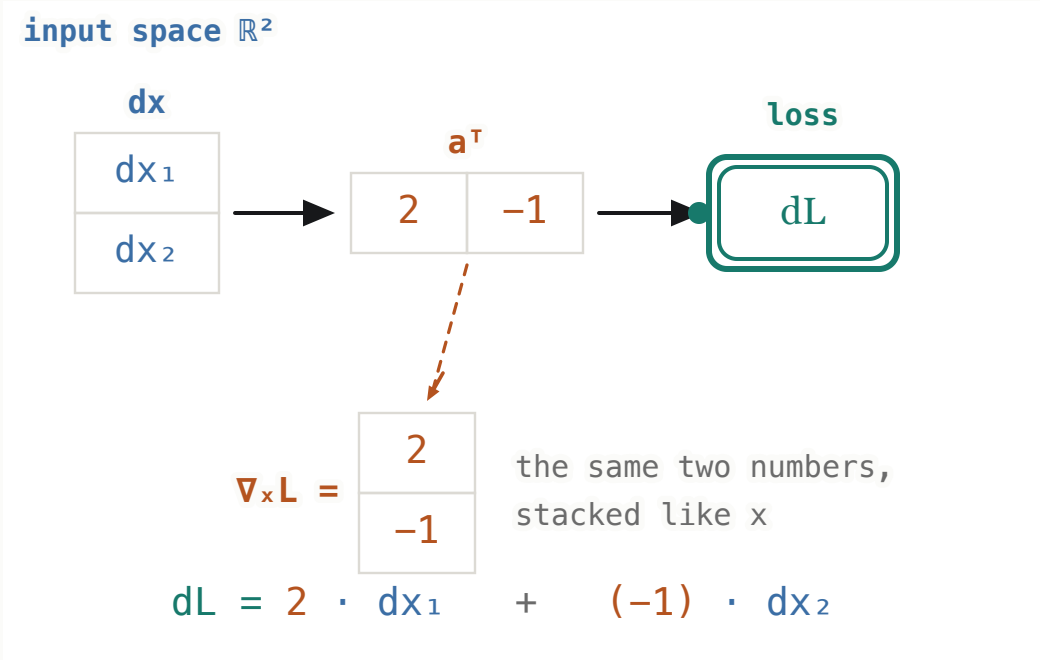
Matrix Calculus Through Differentials

Find the Coefficient of the Input Change

$$L = a^\top x \;\Rightarrow\; dL = a^\top dx \;\Rightarrow\; \nabla_x L = a$$

- $x,\ dx \in \mathbb{R}^2$ — the variable and a linear increment
- $a = (2, -1)$ — fixed coefficients
- $dL \in \mathbb{R}$ — the matching change in the loss
- $\nabla_x L$ — shaped like x

Match $dL = (\nabla_x L)^\top dx$ for every dx: $dx = e_j$ reads off entry j. d means differential, not a number to cancel.



DEFINITION

Matrix Calculus Through Differentials

Matrices Have an Inner Product Too

$$\langle A, B \rangle_F = \sum_{ij} A_{ij} B_{ij} = \operatorname{tr}(A^\top B)$$

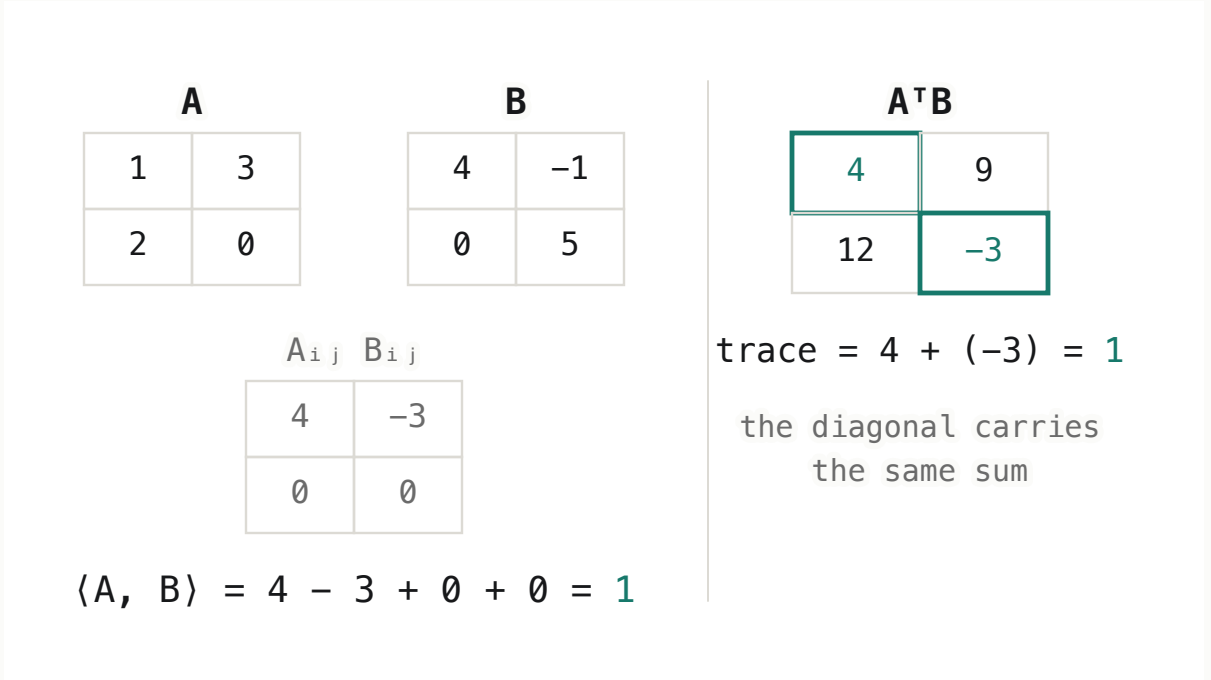
$A, B \in \mathbb{R}^{m \times n}$ — same shape, paired cell by cell

tr — sum of the diagonal entries

$dL = \langle \nabla_W L, dW \rangle_F$ — the matrix version of $a^\top dx$, shaped like W

WORKED EXAMPLE

$\langle A, B \rangle_F = 4 - 3 + 0 + 0 = 1 = \operatorname{tr}(A^\top B).$



VOTE

Matrix Calculus Through Differentials

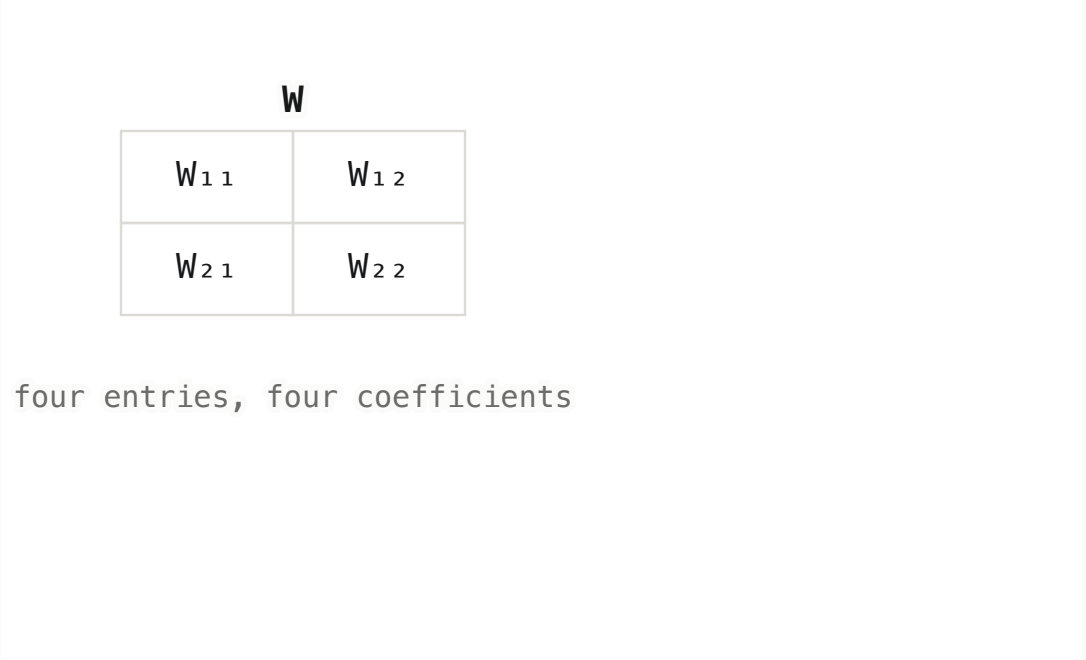
Which Matrix Is the Gradient?

$L(W) = 2W_{11} - W_{12} + 3W_{21} + 0 \cdot W_{22}$. Which matrix is $\nabla_W L$?

A $\begin{bmatrix} 2 & 3 \\ -1 & 0 \end{bmatrix}$

B $\begin{bmatrix} 2 & -1 \\ 3 & 0 \end{bmatrix}$

C $2W$



DERIVATION

Matrix Calculus Through Differentials

Differentiate a Squared Length

$$L = \tfrac{1}{2} x^\top x, \quad dL = \tfrac{1}{2}[(dx)^\top x + x^\top dx] = x^\top dx$$

$x \in \mathbb{R}^n$ the variable

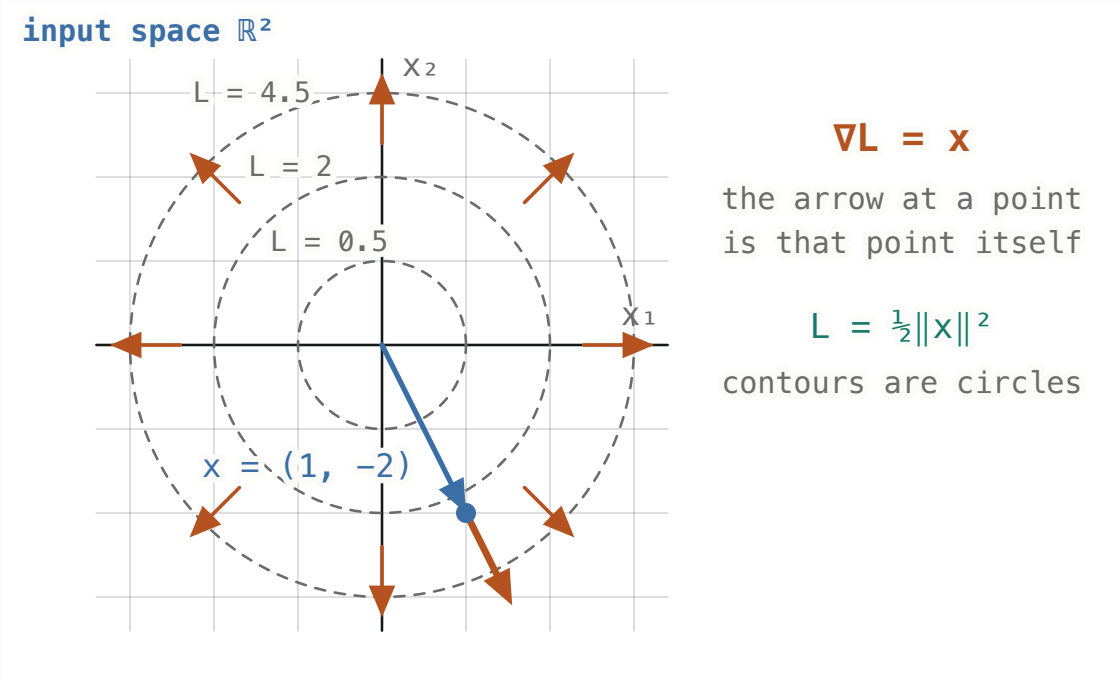
$(dx)^\top x = x^\top dx$ each term is a number

$\nabla L = x$ shaped like x

WORKED EXAMPLE

$x = (1, -2)$: $L = 2.5$ and $\nabla L = (1, -2)$.

Without the $\frac{1}{2}$ the gradient would be $2x$.



DERIVATION

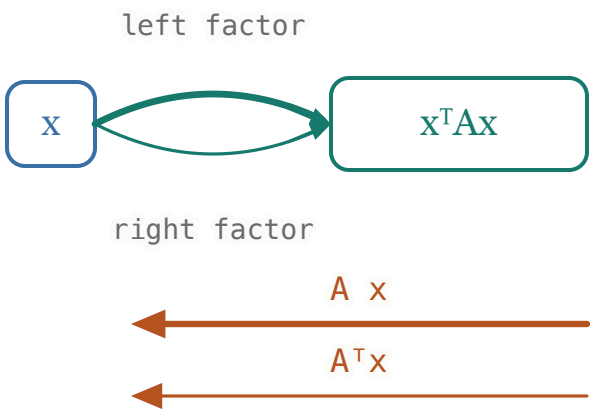
Matrix Calculus Through Differentials

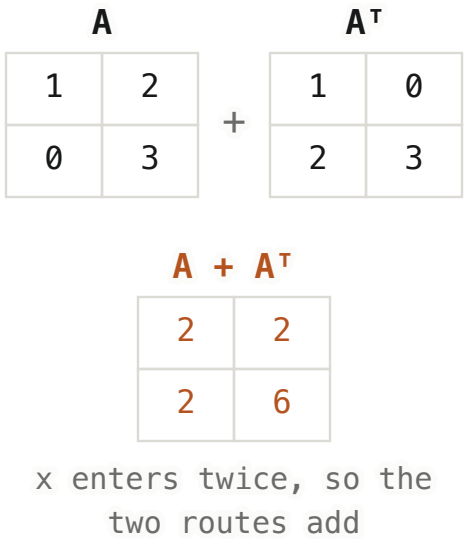
Differentiate a Quadratic Form

$$d(x^\mathsf{T}Ax) = (dx)^\mathsf{T}Ax + x^\mathsf{T}A\,dx$$
$$= [(A + A^\mathsf{T})x]^\mathsf{T}dx$$

$A \in \mathbb{R}^{n\times n}$	constant in x
$(dx)^\mathsf{T}Ax$	x as the left factor
$x^\mathsf{T}A\,dx$	x as the right factor
∇	two routes, added

So $\nabla(x^\mathsf{T}Ax) = (A + A^\mathsf{T})x$; A need not be symmetric, and $\nabla(\frac{1}{2}x^\mathsf{T}Hx) = Hx$ when $H = H^\mathsf{T}$.





VOTE

Matrix Calculus Through Differentials

Where Did the Factor of Two Go?

$L = 2x_1^2 + \frac{1}{2}x_2^2 = \frac{1}{2}x^\top H x$ with H diagonal. Which H?

A $\mathrm{diag}(2, \frac{1}{2})$

B $\mathrm{diag}(4, 1)$

C $\mathrm{diag}(1, 4)$

½ xᵀHx = ½ h₁x₁² + ½ h₂x₂²

H

h₁	0
0	h₂

match the two squared terms

DERIVATION

Matrix Calculus Through Differentials

The Least-Squares Gradient, One Step at a Time

$$e = Ax - b, \qquad L = \tfrac{1}{2} e^{\mathsf{T}} e$$

$A \in \mathbb{R}^{3\times 2}$ constant

$e \in \mathbb{R}^3$ prediction minus target

$\nabla_x L$ shaped like x

$$de = A\,dx$$

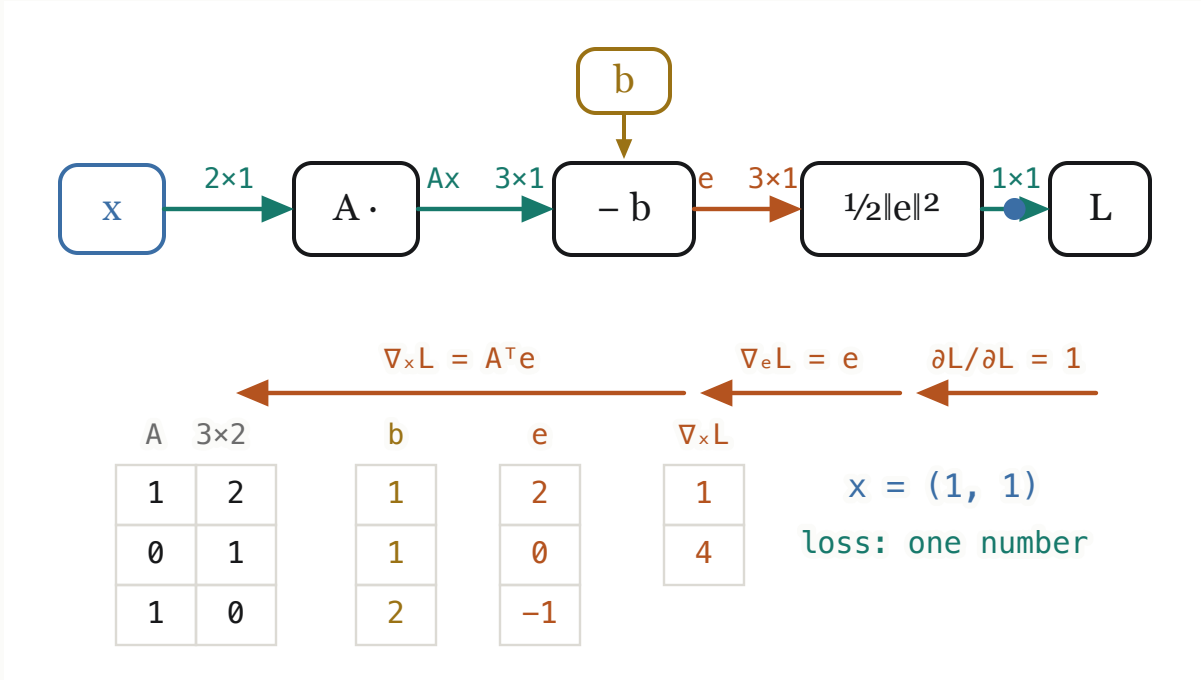
$$dL = e^{\mathsf{T}} de = e^{\mathsf{T}} A\,dx$$

$$\nabla_x L = A^{\mathsf{T}}(Ax - b)$$

WORKED EXAMPLE

$x = (1,1)$: $e = (2, 0, -1)$, $\nabla_x L = (1, 4)$.

Lecture 3's $r = b - Ax$ is $-e$, so $\nabla_x L = -A^{\mathsf{T}} r$.



RECALL

Matrix Calculus Through Differentials

Calculus Recovers the Orthogonality Condition

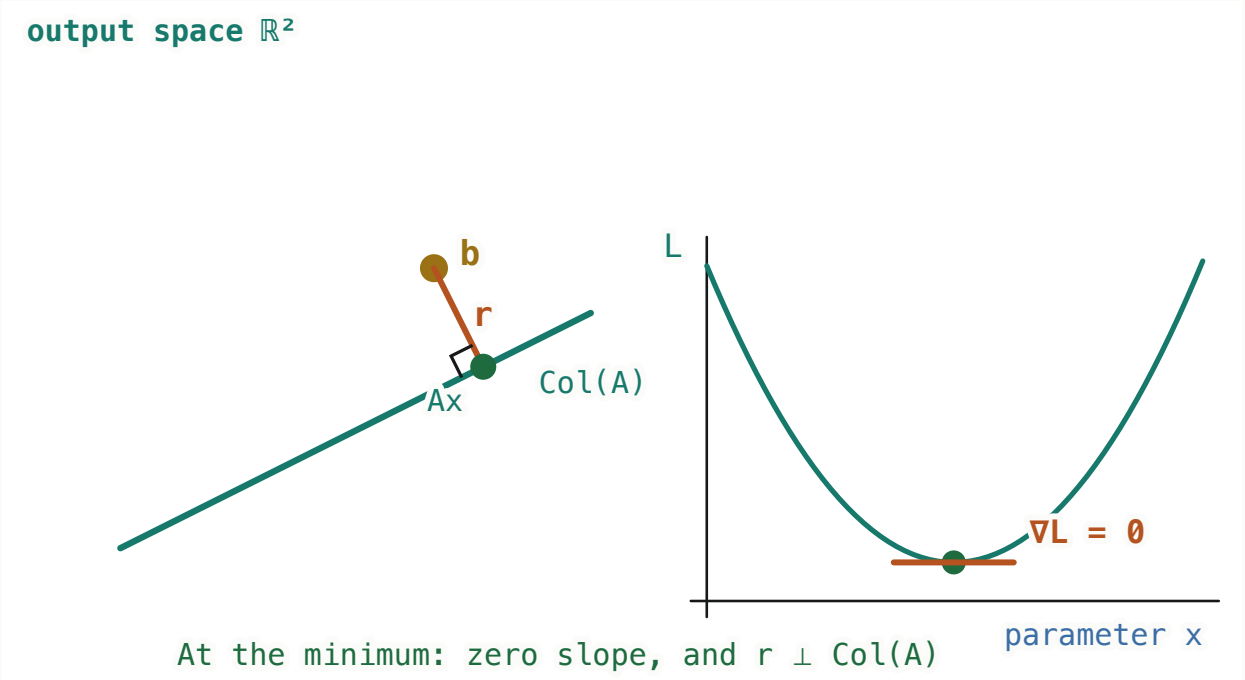
$$\nabla L = 0 \iff A^\top (Ax - b) = 0 \iff A^\top r = 0$$

$r = b - Ax$ the residual of Lecture 3

$A^\top r = 0$ $r \perp$ every column, and $r \neq 0$ in general

$\nabla L = 0$ a stationary point

Curvature along any direction v is $v^\top A^\top A v = \|Av\|^2 \geq 0$, so L is convex and a stationary point is a global minimum. Unique unless some $v \neq 0$ has $Av = 0$.



$A = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ $b = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

VOTE

Matrix Calculus Through Differentials

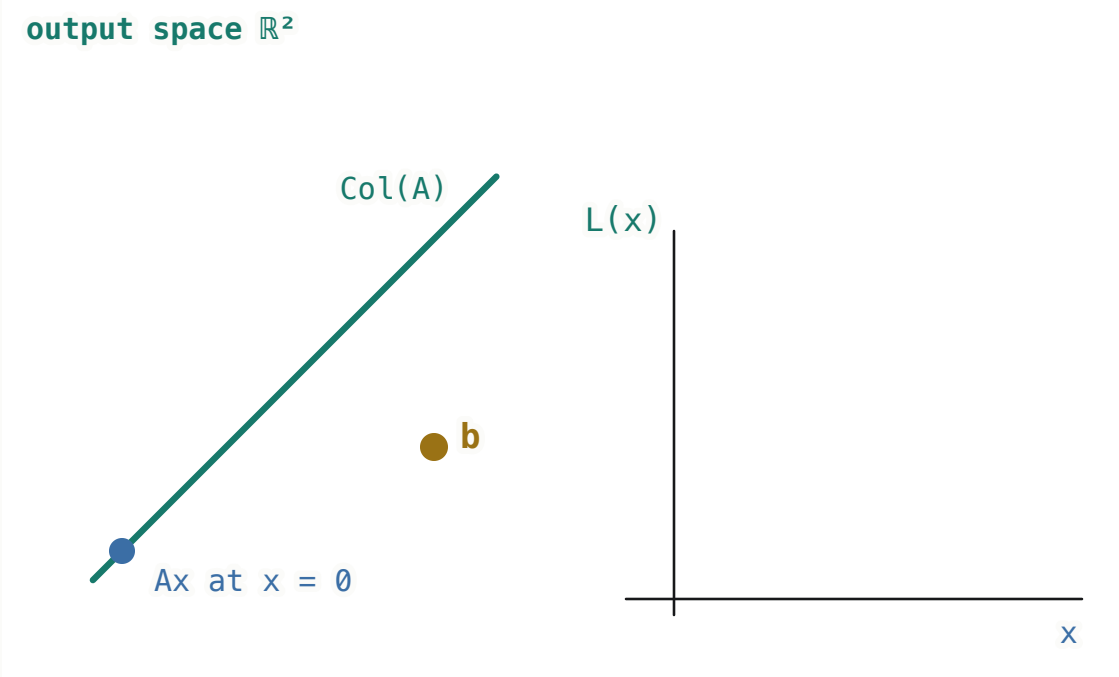
Match the Gradient to the Geometry

$L(x) = \frac{1}{2}[(x-3)^2 + (x-1)^2]$ with scalar x. What is $L'(0)$?

A -4

B $+4$

C 0



RECAP

Matrix Calculus Through Differentials

A Small Derivative Toolkit

Each row comes from writing dL as a pairing with the increment.

$$\nabla(a^\top x) = a \quad n \times 1$$

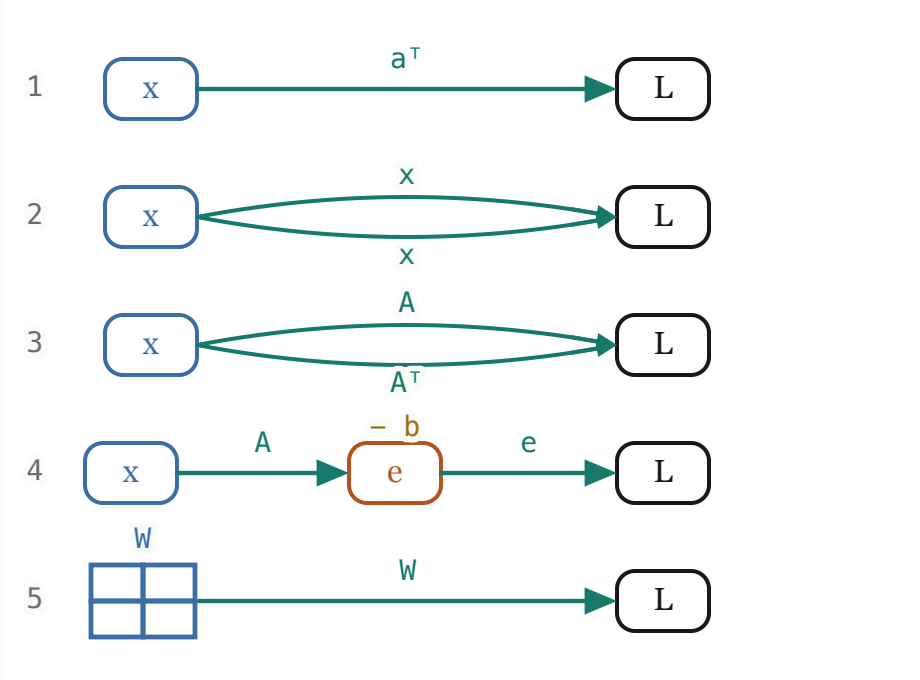
$$\nabla(\tfrac{1}{2}\|x\|^2) = x \quad n \times 1$$

$$\nabla(x^\top A x) = (A + A^\top)x \quad n \times 1$$

$$\nabla(\tfrac{1}{2}\|Ax - b\|^2) = A^\top(Ax - b) \quad n \times 1$$

$$\nabla_W(\tfrac{1}{2}\|W\|_F^2) = W \quad m \times n$$

a, A, b fixed. Row 3 holds for any A; $\nabla(\frac{1}{2}x^\top Hx) = Hx$ needs $H = H^\top$.



DEFINITION

Computation Graphs and Reverse Accumulation

Split a Formula into Small Operations

$$L = (ab + c)^2$$

$$u = ab, \quad v = u + c, \quad L = v^2$$

a, b, c the inputs, here $2, 3, -1$

u, v values the graph stores

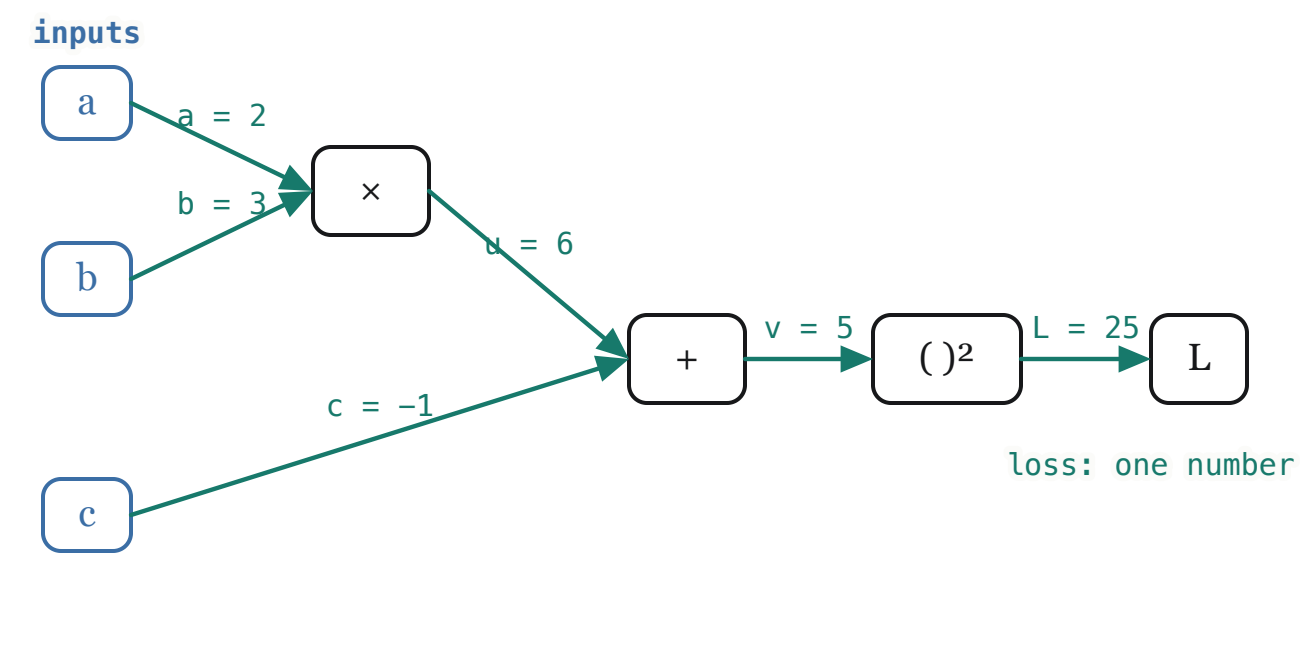
L one number at the end

$u = ab = 6$

$v = u + c = 5$

$L = v^2 = 25$

The graph records dependencies, and the backward pass reuses exactly these stored values.



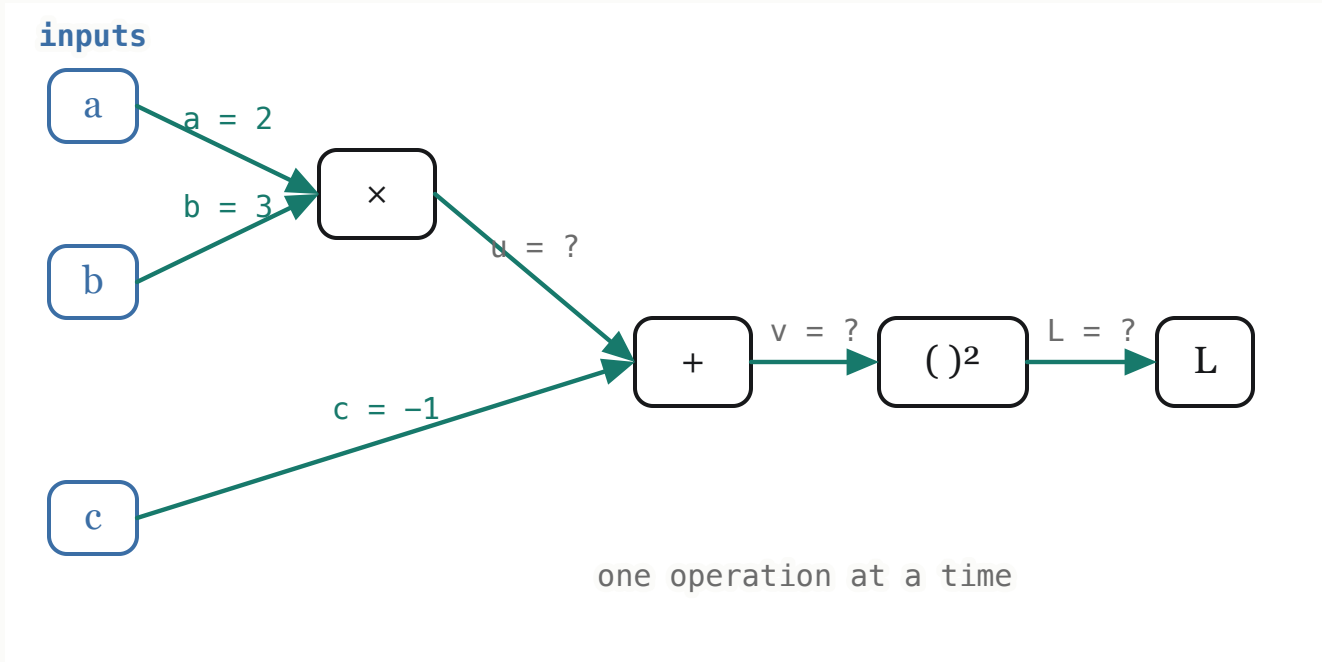
VOTE

Computation Graphs and Reverse Accumulation

Fill the Forward Pass

Which triple is (u, v, L), and at which node do the other two break?

- A $(6, 5, 25)$
- B $(6, 7, 49)$
- C $(5, 6, 36)$



DEFINITION

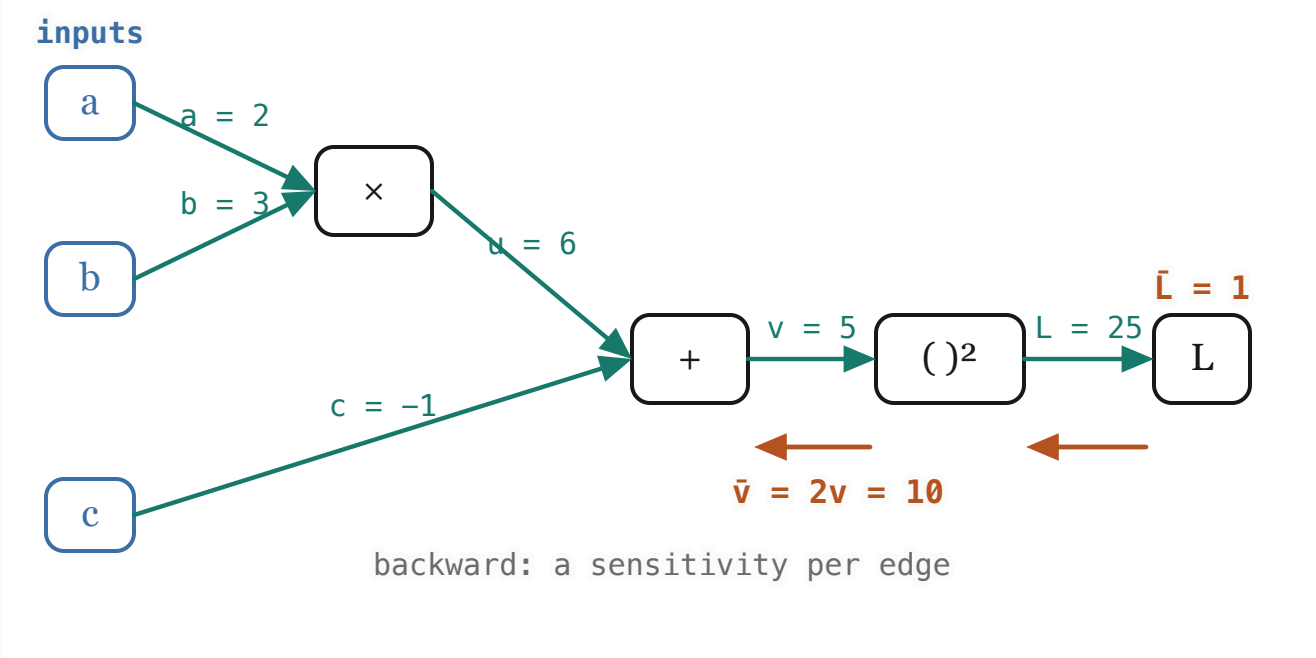
Computation Graphs and Reverse Accumulation

Start Backward with One

$$\bar{u} := \frac{\partial L}{\partial u}, \qquad \bar{L} = 1, \qquad \bar{v} = 2v\,\bar{L} = 10$$

- $\bar{v}$ how much L moves per unit of v
- $\bar{L} = 1$ the seed, not the value $L = 25$
- $2v$ the local derivative of $L = v^2$

$\bar{x}$ and g_x name the same object. Upstream means nearer the loss along the backward path.



RULES

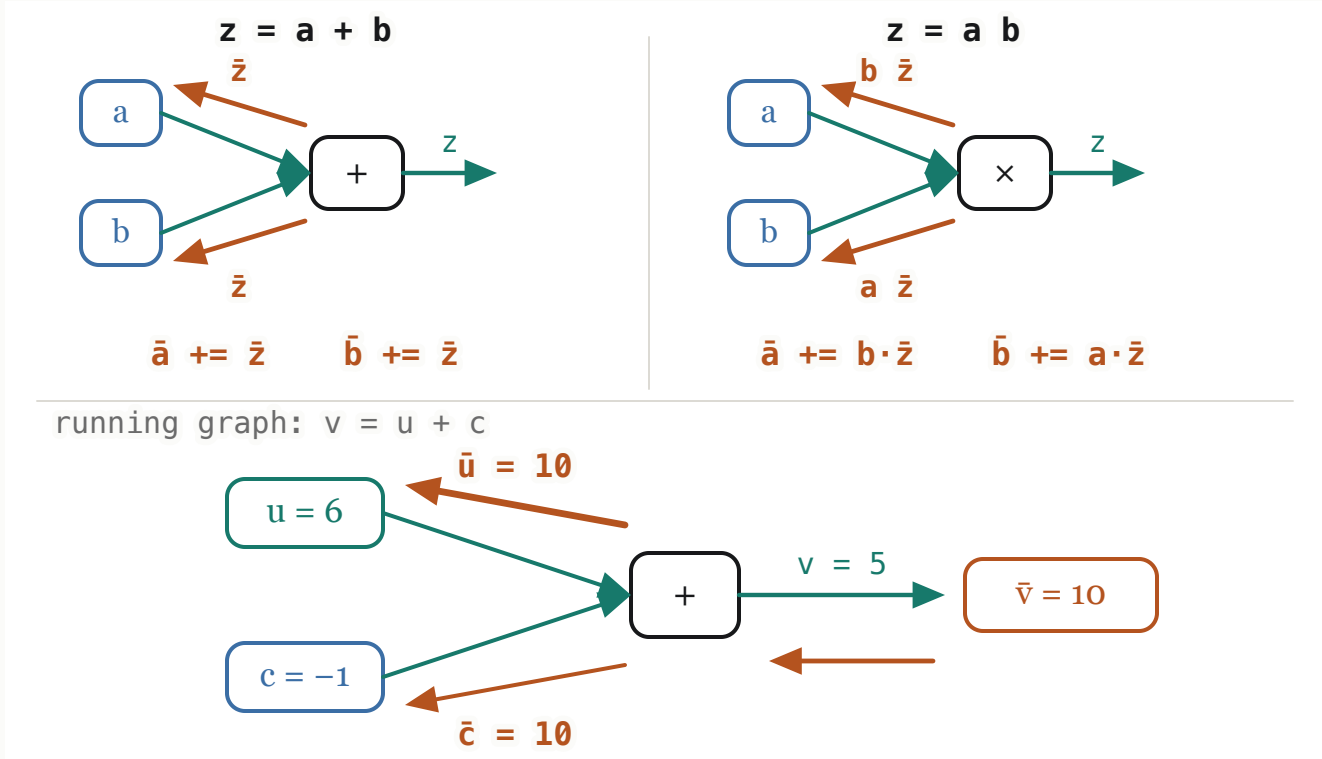
Computation Graphs and Reverse Accumulation

Addition Copies, Multiplication Scales

$$z = a + b:\ \bar{a} \mathrel{+}= \bar{z},\ \bar{b} \mathrel{+}= \bar{z}$$
$$z = ab:\ \bar{a} \mathrel{+}= b\,\bar{z},\ \bar{b} \mathrel{+}= a\,\bar{z}$$

- $\bar{z}$ already known at this node
- $+=$ accumulate, never overwrite
- b, a the other input's stored value

In the running graph $\bar{v} = 10$ gives $\bar{u} = \bar{c} = 10$.



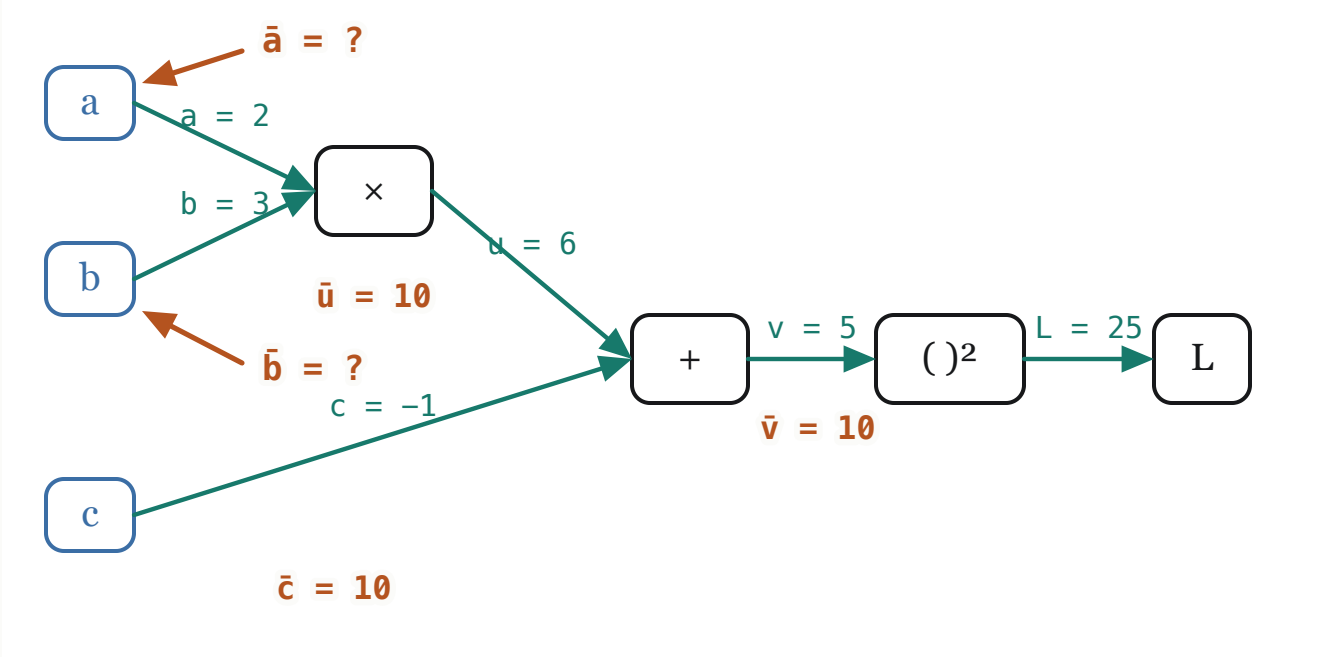
VOTE

Computation Graphs and Reverse Accumulation

Which Number Returns to a?

$\bar{v} = 10$ is already known. Which number returns to a?

- (A) 10
- (B) 20
- (C) 30



DERIVATION

Computation Graphs and Reverse Accumulation

Shared Variables Receive Contributions from Every Path

$$\frac{dL}{dx} = \frac{\partial L}{\partial u}\frac{du}{dx} + \frac{\partial L}{\partial v}\frac{dv}{dx} = 2 + 2x$$

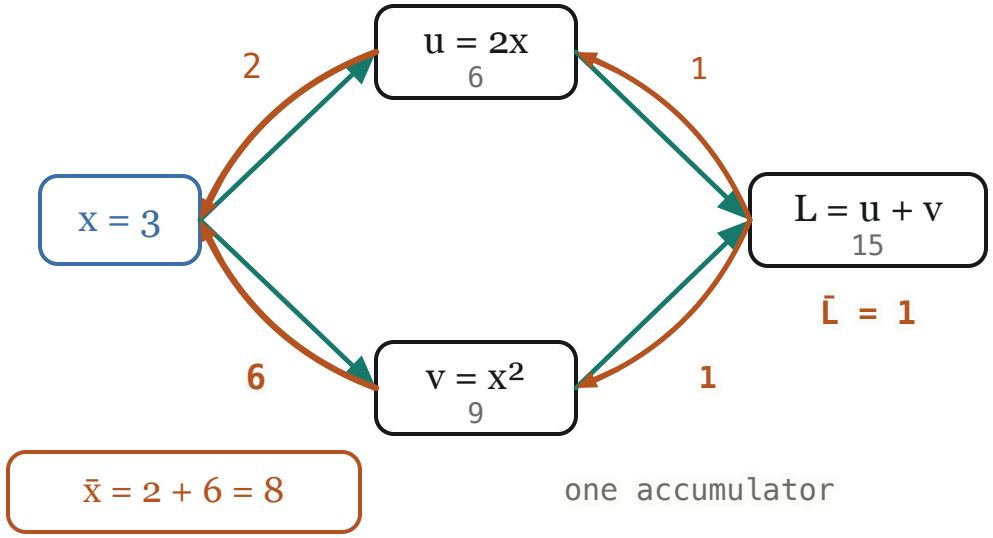
$u = 2x,\ v = x^2$ two paths out of one x

$L = u + v$ where they meet

$\bar{x}$ one accumulator: start at 0 and add

WORKED EXAMPLE

At $x = 3$: 2 from u and 6 from v add to 8 in the accumulator.



VOTE · PAIRS

Computation Graphs and Reverse Accumulation

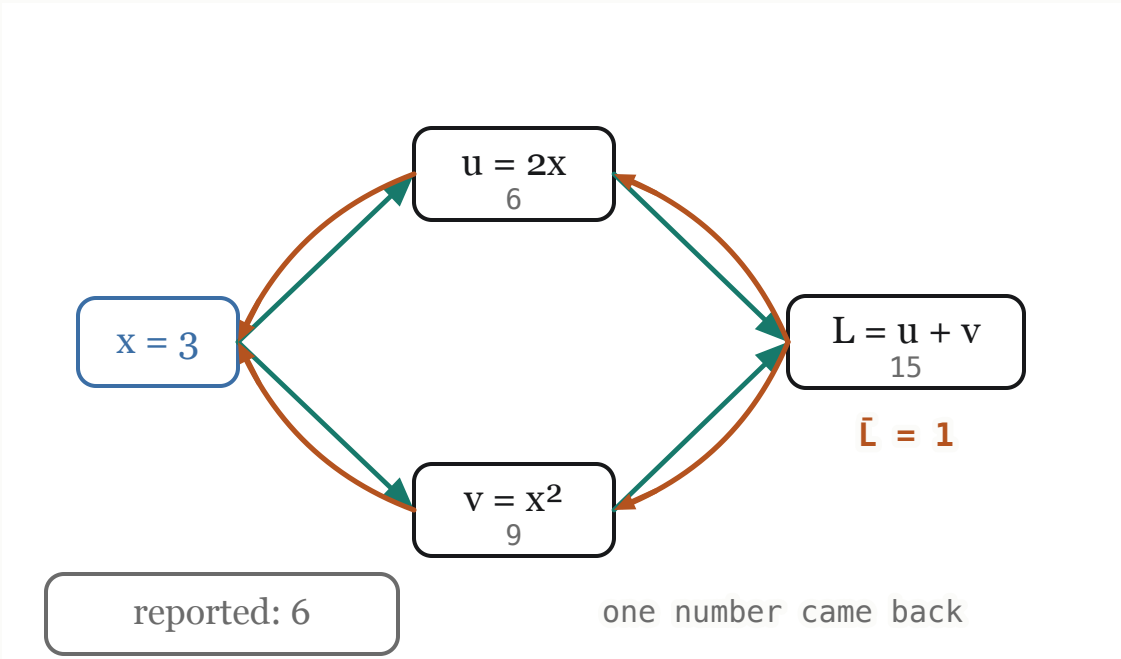
Find the Missing Path

At $x = 3$ a program reports $dL/dx = 6$. What is wrong?

A the result is correct

B the path through u is missing

C the path through v is missing



WORKED EXAMPLE

Computation Graphs and Reverse Accumulation

The Same Operand Can Enter Twice

$$z = x \cdot x,\ L = z: \qquad \bar{x} = x\,\bar{z} + x\,\bar{z} = 2x$$

$\bar{z} = 1$ the seed

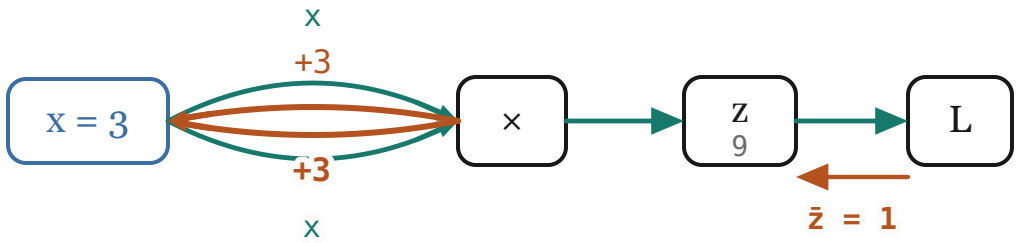
$x\,\bar{z}$ what each edge sends back

$\bar{x}$ the sum over edges

WORKED EXAMPLE

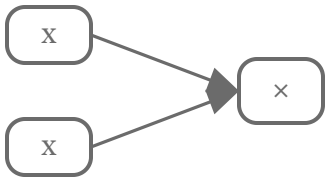
At $x = 3$: $\bar{x} = 3 + 3 = 6$.

One shared node is not two copies. This is the product rule, and where the factor 2 on a squared length came from.



$\bar{x} = 3 + 3 = 6$

one node, two edges



two copies would be two variables

METHOD

Computation Graphs and Reverse Accumulation

Reverse Order Makes the Local Rules Work

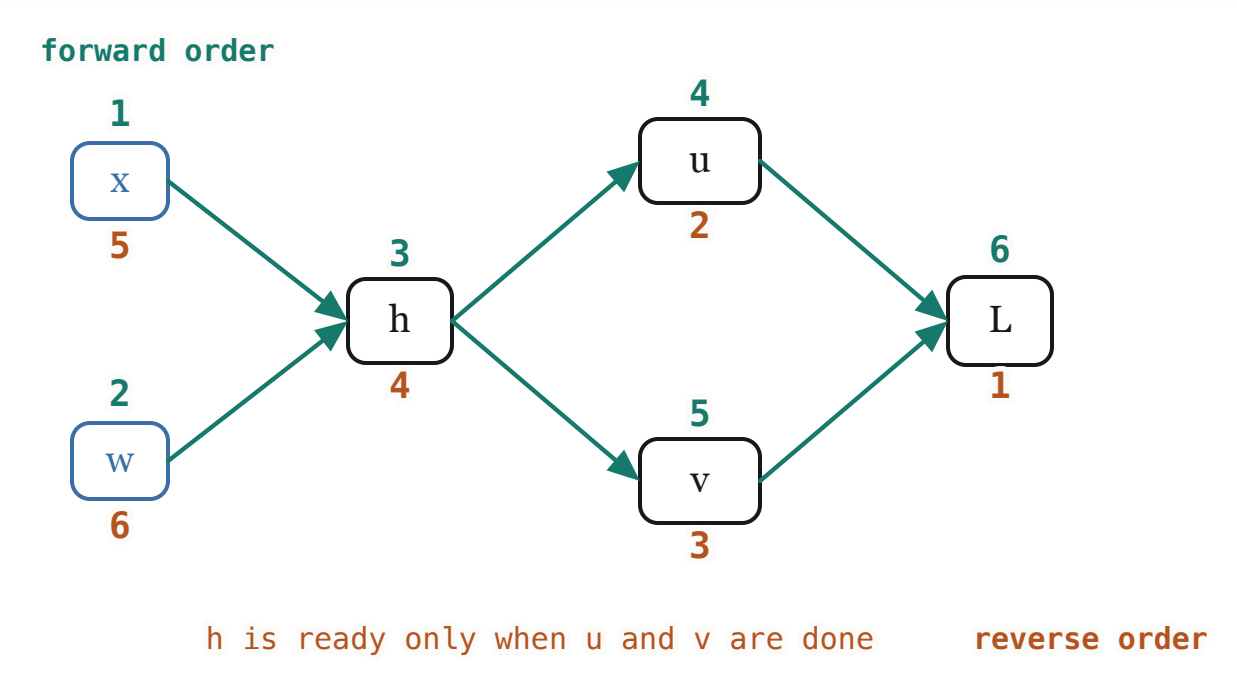
$$\bar{x}_i = \sum_{s:\, x_i \to x_j} (\partial x_j / \partial s)^{\mathsf{T}}\, \bar{x}_j$$

$s : x_i \to x_j$ one edge per argument slot

$\partial x_j / \partial s$ vary slot s, hold the rest

$\bar{x}_j$ already finished

1. Store the forward values.
2. Seed $\bar{L} = 1$.
3. Visit in reverse topological order.
4. Add one term per slot: $x \cdot x$ gives $x\bar{z} + x\bar{z}$.



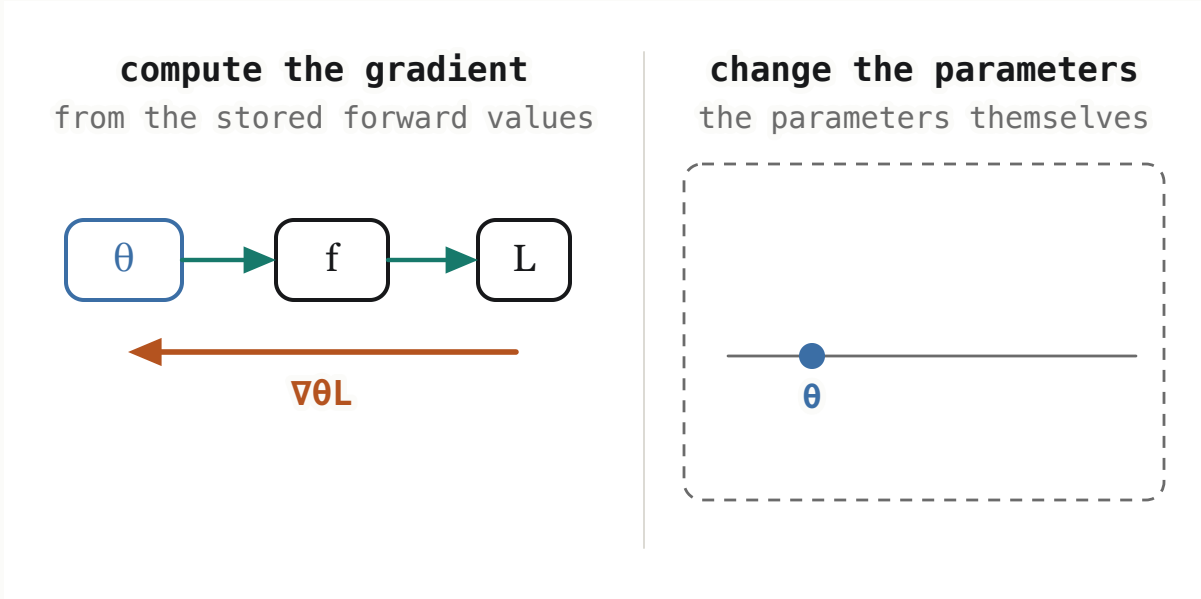
VOTE

Computation Graphs and Reverse Accumulation

Backpropagation or Optimization?

Which of these is backpropagation itself?

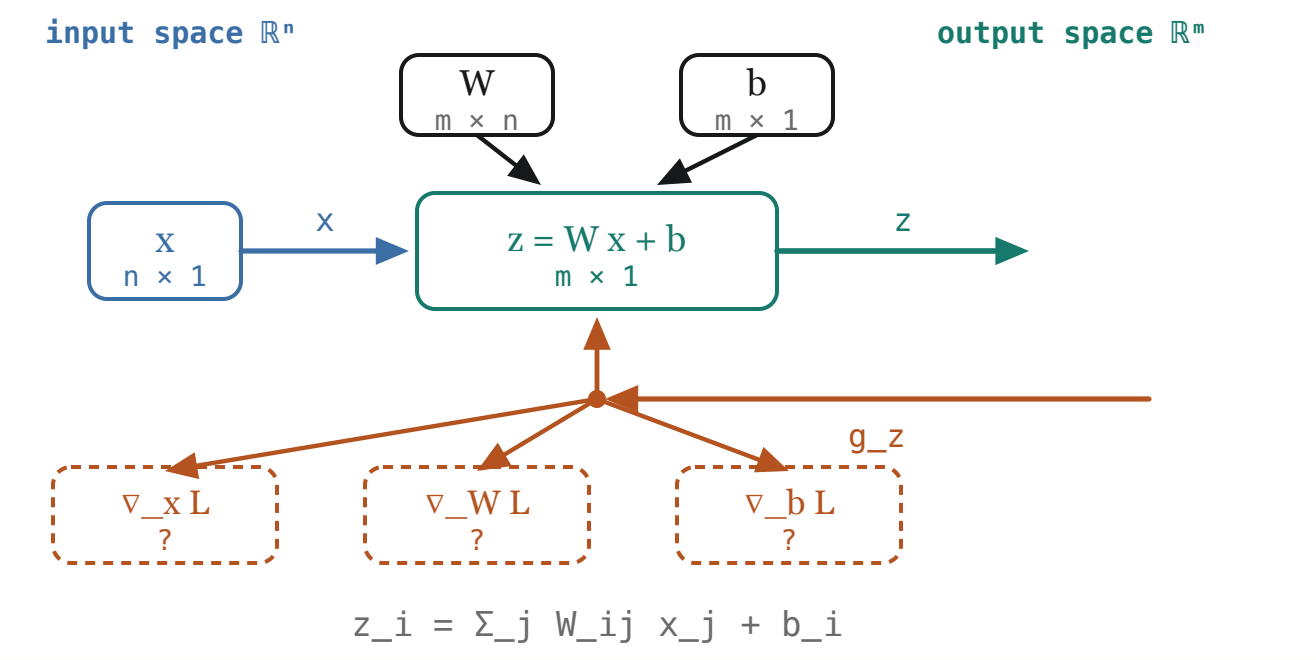
- (A) compute $\nabla_\theta L$ from the stored values
- (B) change θ with an update rule
- (C) both together



DEFINITION

Backpropagation Through Neural Networks

A Linear Layer Has Three Differentiable Inputs



$$z = Wx + b, \qquad g_z = \nabla_z L$$

- x the layer's input, $n \times 1$
- W, b parameters, $m \times n$ and $m \times 1$
- g_z sensitivity arriving from above, $m \times 1$

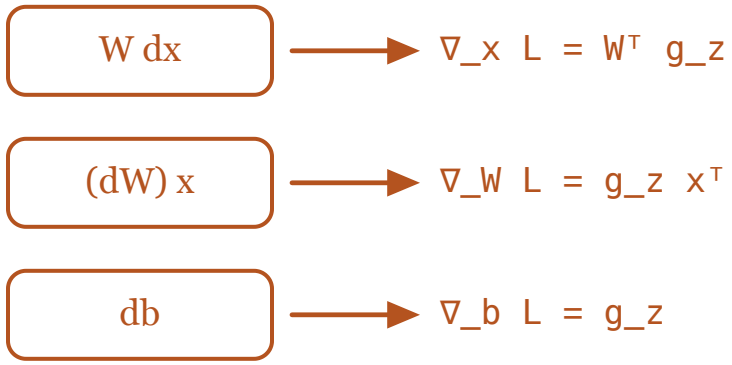
Three arrows enter, so three gradients leave. With a bias the map is affine; "linear layer" is the usual name.

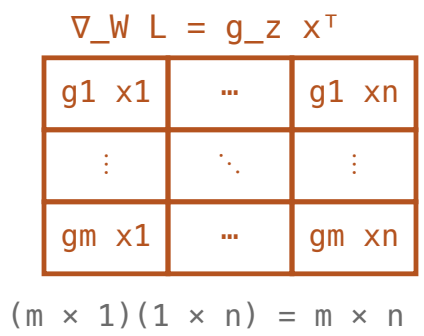
DERIVATION

Backpropagation Through Neural Networks

Derive the Three Affine Backward Rules

$$dz = (dW)x + W\,dx + db, \qquad dL = g_z^{\mathsf{T}} dz$$





dz	the linear part of a change in z
$g_z^{\mathsf{T}} dz$	all of dL when x, W, b reach L only via z
shape	$(m \times 1)(1 \times n) = m \times n$

$$W\,dx \;\Rightarrow\; \nabla_x L = W^{\mathsf{T}} g_z$$

$$(dW)x \;\Rightarrow\; \nabla_W L = g_z x^{\mathsf{T}}$$

$$db \;\Rightarrow\; \nabla_b L = g_z$$

VOTE

Backpropagation Through Neural Networks

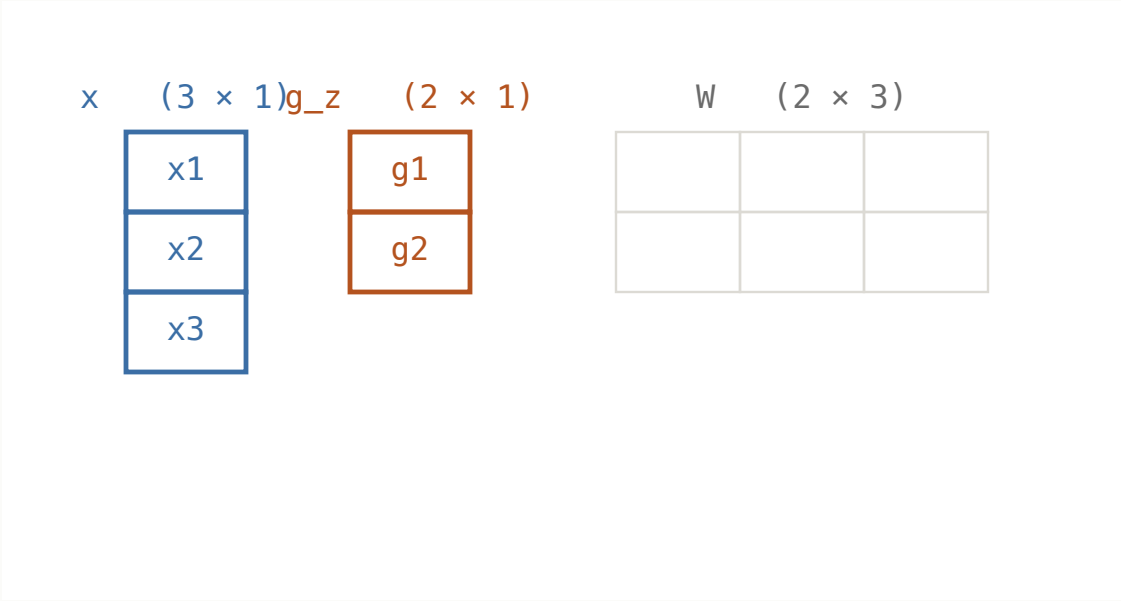
Which Product Has the Weight's Shape?

x has 3 entries and z has 2, so W is 2×3. Which product has W's shape?

A $x g_z^\top$

B $g_z x^\top$

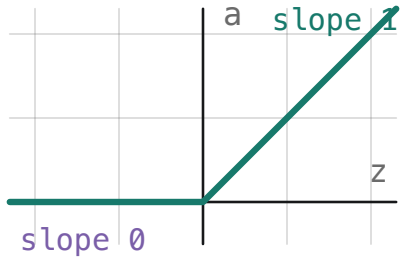
C $g_z^\top x$



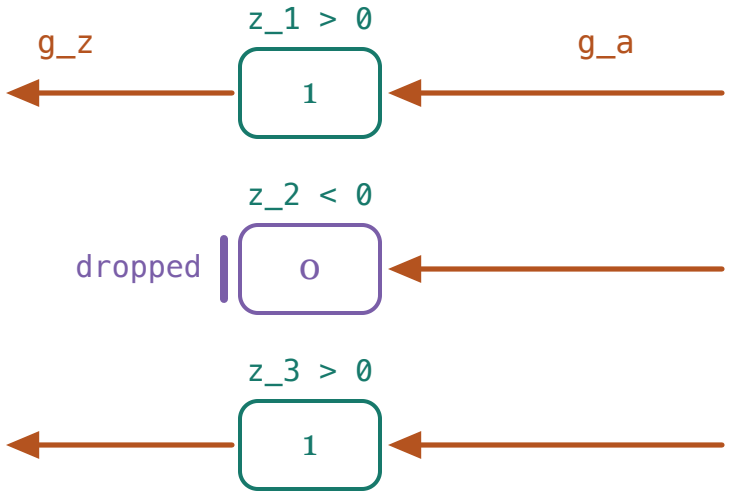
DEFINITION

Backpropagation Through Neural Networks

ReLU Passes Some Local Sensitivities



kink at z = 0



g_z = g_a ⊙ 1[z > 0]

$$a_i = \max(0, z_i), \quad g_z = g_a \odot \mathbf{1}_{z>0}$$

- $\odot$ entrywise product
- $\partial a / \partial z$ $\operatorname{diag}(\mathbf{1}_{z>0})$, $m \times m$, where $z_i \neq 0$
- $z_i = 0$ no ordinary derivative; 0 is our backward rule

ReLU clamps the value. Backward, each entry is multiplied by 0 or 1 — the diagonal Jacobian, applied entrywise.

VOTE

Backpropagation Through Neural Networks

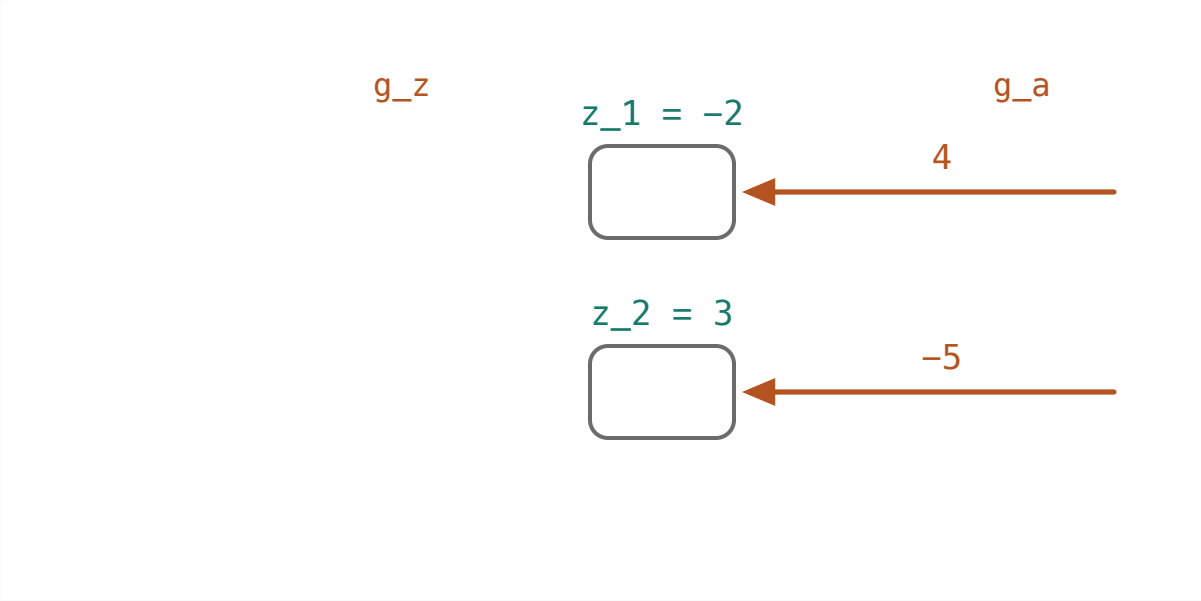
Which Gradient Entries Survive?

A ReLU layer had $z = (-2, 3)$ on the forward pass, and $g_a = (4, -5)$ arrives. Which is g_z?

A $(0, -5)$

B $(4, 0)$

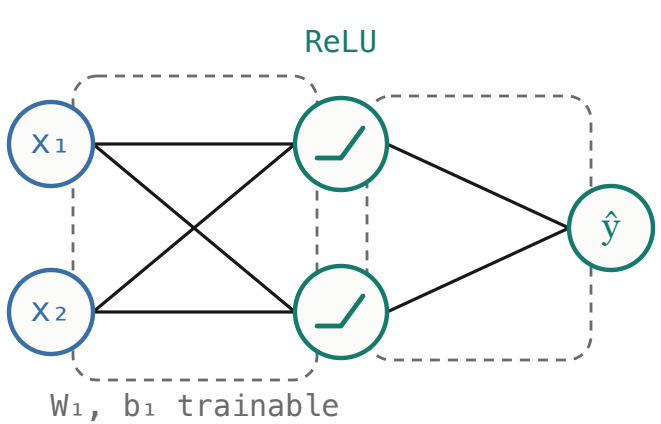
C $(0, 5)$

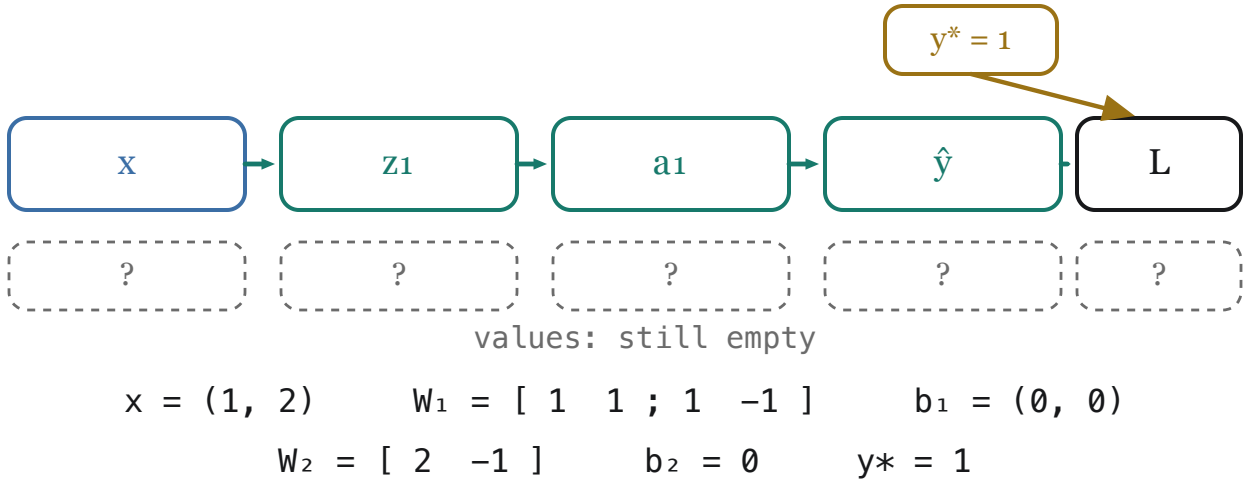


WORKED SETUP

Backpropagation Through Neural Networks

One Network We Can Calculate Completely





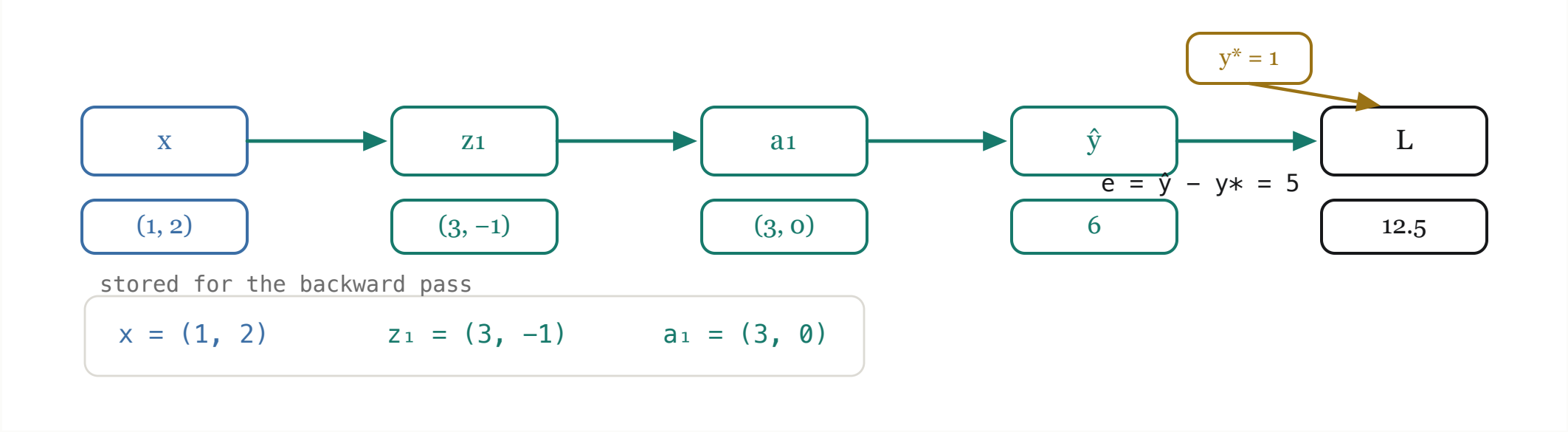
$$z_1 = W_1 x + b_1,\ a_1 = \mathrm{ReLU}(z_1),\ \hat{y} = W_2 a_1 + b_2,\ L = \tfrac{1}{2}(\hat{y} - y^\star)^2$$

- x one input, 2×1; W_1 is 2×2 and W_2 is 1×2
- $y^\star$ the fixed target; it is never differentiated

$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ $y* = \begin{bmatrix} 1 \end{bmatrix}$

Backpropagation Through Neural Networks

Forward Values Become Backward Ingredients



$$\hat{y} = W_2\,\mathrm{ReLU}(W_1 x + b_1) + b_2$$

$e = \hat{y} - y^\star$ prediction minus target

cache x, z_1, a_1 are kept; W_1, W_2 stay in the layers

WORKED EXAMPLE

$z_1 = (1{+}2,\ 1{-}2)$, and the second unit is clamped, so a_1 keeps 3 and drops the negative entry. Then $\hat{y} = 2 \cdot 3 - 1 \cdot 0$.

$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ $y* = \begin{bmatrix} 1 \end{bmatrix}$

VOTE

Backpropagation Through Neural Networks

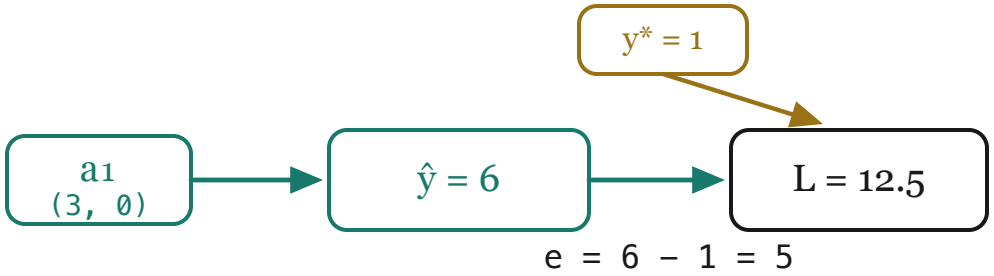
Start at the Loss

Three numbers sit at the end of the forward pass.
Which one is $g_{\hat{y}} = \partial L / \partial \hat{y}$?

A 6

B 5

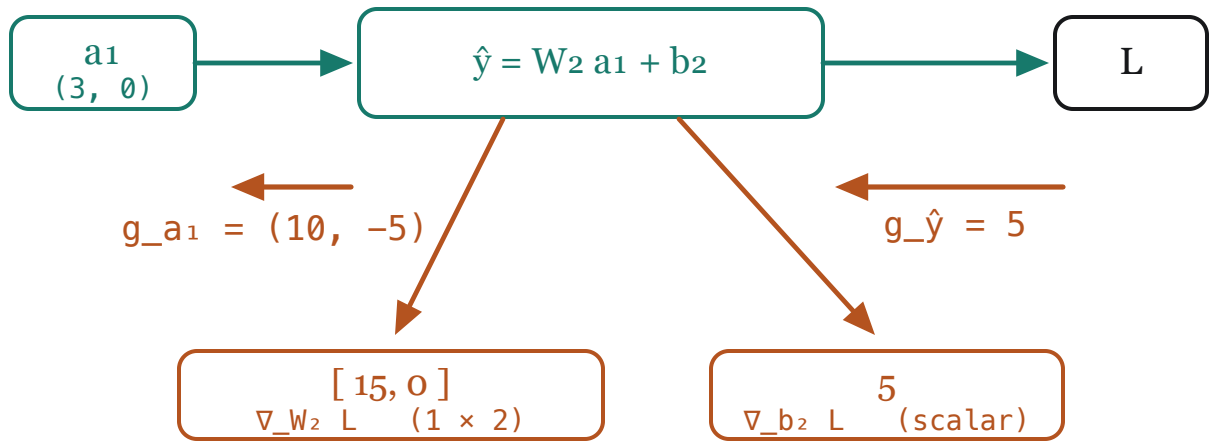
C 12.5



$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ $y* = \begin{bmatrix} 1 \end{bmatrix}$

Backpropagation Through Neural Networks

Backpropagate Through the Output Layer



$$\nabla_{W_2} L = g_{\hat{y}} a_1^\top, \quad \nabla_{b_2} L = g_{\hat{y}}, \quad g_{a_1} = W_2^\top g_{\hat{y}}$$

$\nabla_{W_2} L$ 1×2, the shape of W_2

g_{a_1} 2×1, one entry per hidden unit

a_1 from the forward cache

The second weight's gradient is zero because $a_{1,2} = 0$, not because $g_{a_1,2}$ is.

VOTE

Backpropagation Through Neural Networks

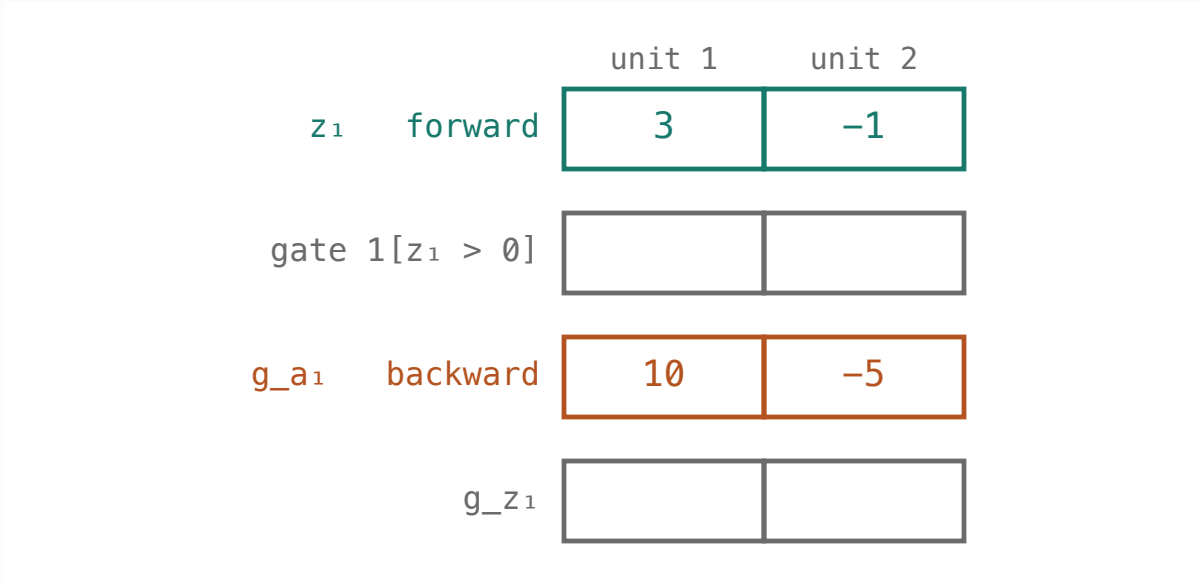
Backward Through the Hidden Gate

The forward pass left $z_1 = (3, -1)$, and $g_{a_1} = (10, -5)$ now arrives. Which is g_{z_1}?

A $(10, -5)$

B $(10, 0)$

C $(0, -5)$



$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ $y* = \begin{bmatrix} 1 \end{bmatrix}$

Backpropagation Through Neural Networks

Finish the First Layer

$$\nabla_{W_1} L = g_{z_1} x^\top$$

	$x_1 = 1$	$x_2 = 2$
$(g_{z_1})_1 = 10$	$10 \cdot 1 = 10$	$10 \cdot 2 = 20$
$(g_{z_1})_2 = 0$	$0 \cdot 1 = 0$	$0 \cdot 2 = 0$

$$\nabla_{b_1} L = g_{z_1} = (10, 0)$$

$$\nabla_x L = W_1^\top g_{z_1} = (10, 10)$$

$$\nabla_{W_1} L = g_{z_1} x^\top, \qquad \nabla_x L = W_1^\top g_{z_1}$$

$g_{z_1} x^\top$	$(2 \times 1)(1 \times 2) = 2 \times 2$
$\nabla_{b_1} L$	$= g_{z_1}$, one entry per unit
$\nabla_x L$	2×1, how L responds to a change in x

No parameter has moved: changing them is a separate job.

VOTE · PAIRS

Backpropagation Through Neural Networks

Diagnose a Zero Gradient

Why is the second row of $\nabla_{W_1} L$ zero here?

- (A) that row of W_1 is zero
- (B) $z_{1,2} < 0$, so the path is blocked
- (C) a two-layer network always has one

x = (1, 2)

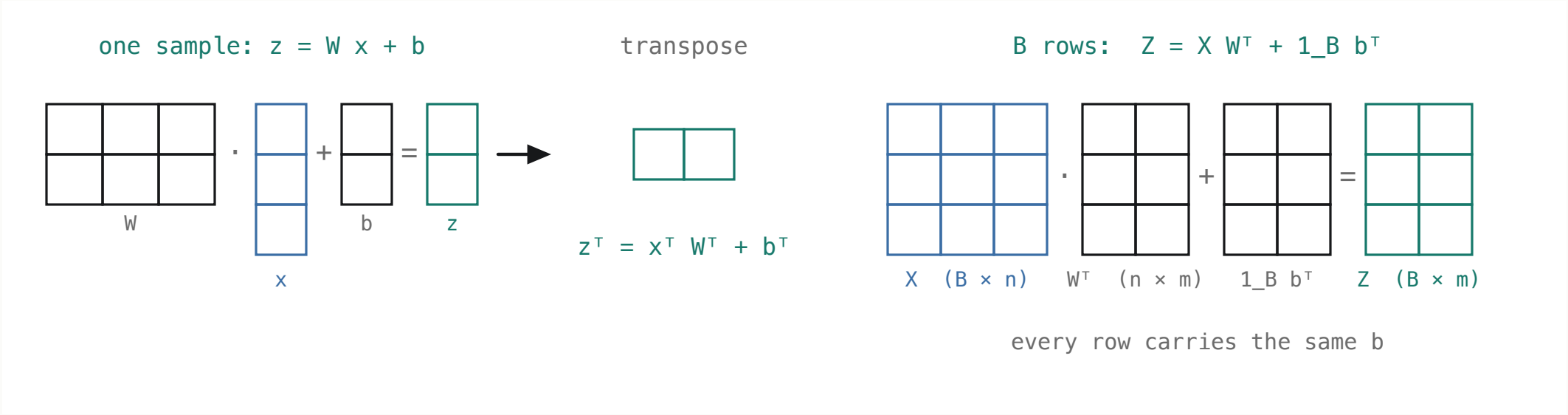
∇_W₁ L

10	20
0	0

CONVENTION

Batched Gradients and Broadcasting

Put Samples in Rows



$$Z = XW^{\mathsf{T}} + \mathbf{1}_B b^{\mathsf{T}}$$

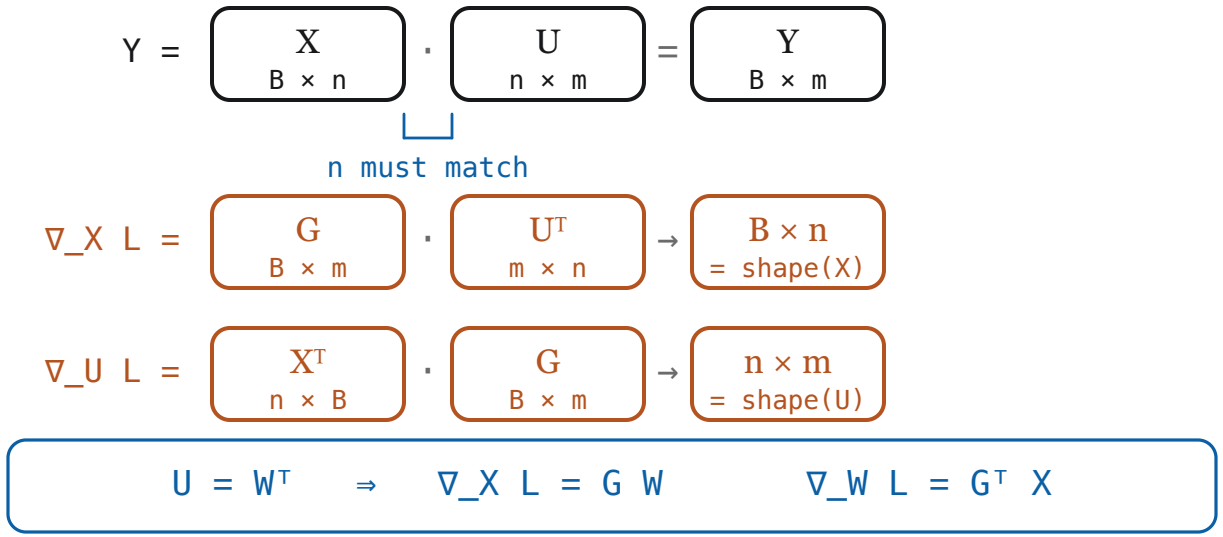
One sample gives $z = Wx + b$. Transposing turns Wx into $x^{\mathsf{T}}W^{\mathsf{T}}$, so a row of X goes through W^{T}. In code b holds shape $(m,)$ and broadcasts.

- X $B \times n$: one sample per row
- W $m \times n$, exactly as for one sample
- b a column, $m \times 1$

DERIVATION

Batched Gradients and Broadcasting

Matrix Multiplication Backward



$$dL = \operatorname{tr}(G^\top [dX\, U + X\, dU])$$

$Y = XU$ $(B \times n)(n \times m)$

G $\nabla_Y L, B \times m$

$\operatorname{tr}(A^\top B)$ $\langle A, B \rangle_F$

$\nabla_X L = GU^\top$: the dX coefficient.

$\nabla_U L = X^\top G$; entrywise $\partial L / \partial U_{kj} = \sum_i X_{ik} G_{ij}$.

With $U = W^\top$: $\nabla_X L = GW$ and $\nabla_W L = G^\top X$.

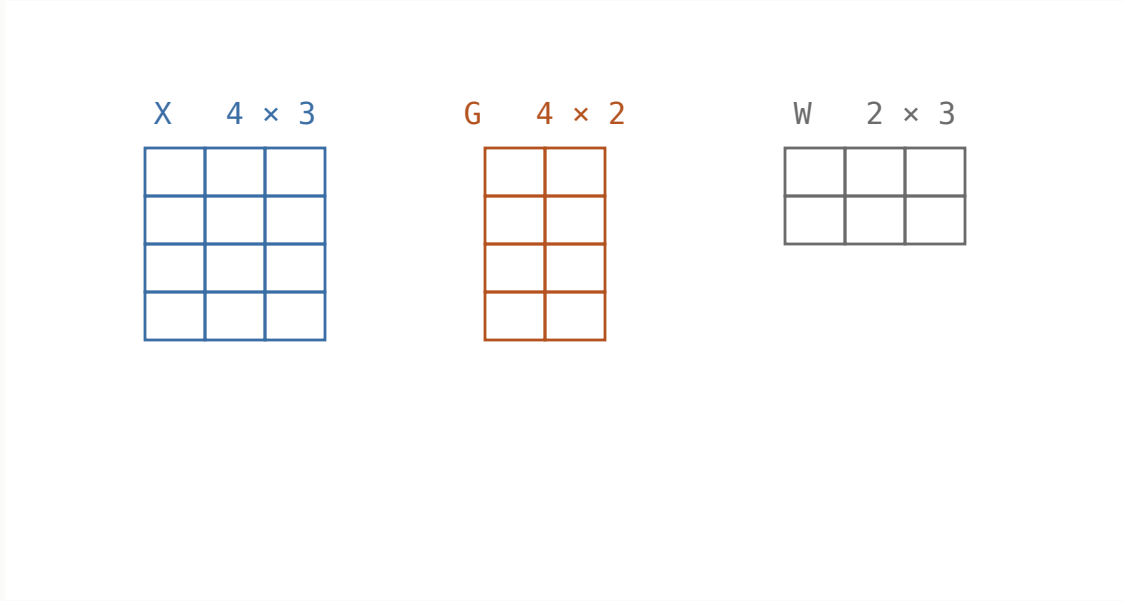
VOTE

Batched Gradients and Broadcasting

Choose the Batch Weight Gradient

$B = 4, n = 3, m = 2$ and $Z = XW^\top$, with $G = \nabla_Z L$.
Which is $\nabla_W L$?

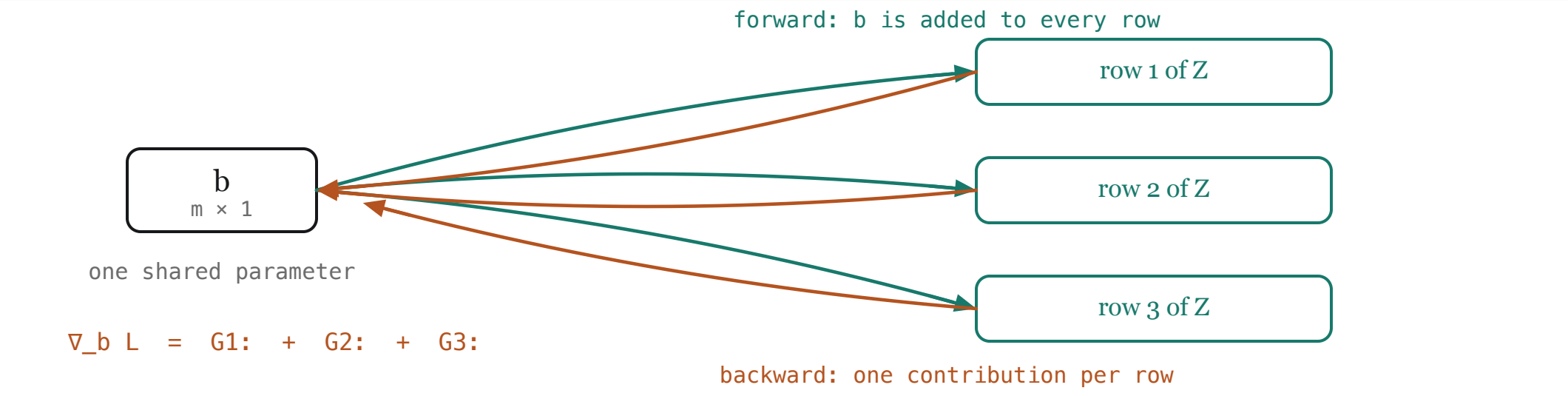
- (A) $X^\top G$
- (B) $G^\top X$
- (C) $GX^\top$



DERIVATION

Batched Gradients and Broadcasting

Broadcasting Forward Means Summing Backward



$$Z_{ij} = \sum_k X_{ik} W_{jk} + b_j \;\Rightarrow\; \nabla_b L = G^\top \mathbf{1}_B$$

A variable used in many places collects one contribution from each.

b_j one number, shared by all B rows

$\partial L / \partial b_j$ $\sum_i G_{ij}$, with $G = \nabla_Z L$

$\mathbf{1}_B$ the all-ones column, $B \times 1$

VOTE

Batched Gradients and Broadcasting

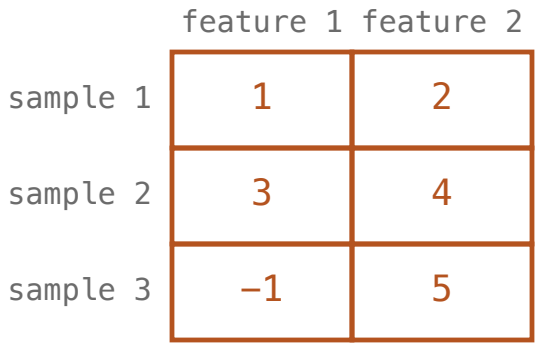
Which Axis Must Be Summed?

$G = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ -1 & 5 \end{bmatrix}$ with $B = 3$ and $m = 2$. Which is $\nabla_b L$?

A $(3, 7, 4)$

B $(1, 2)$

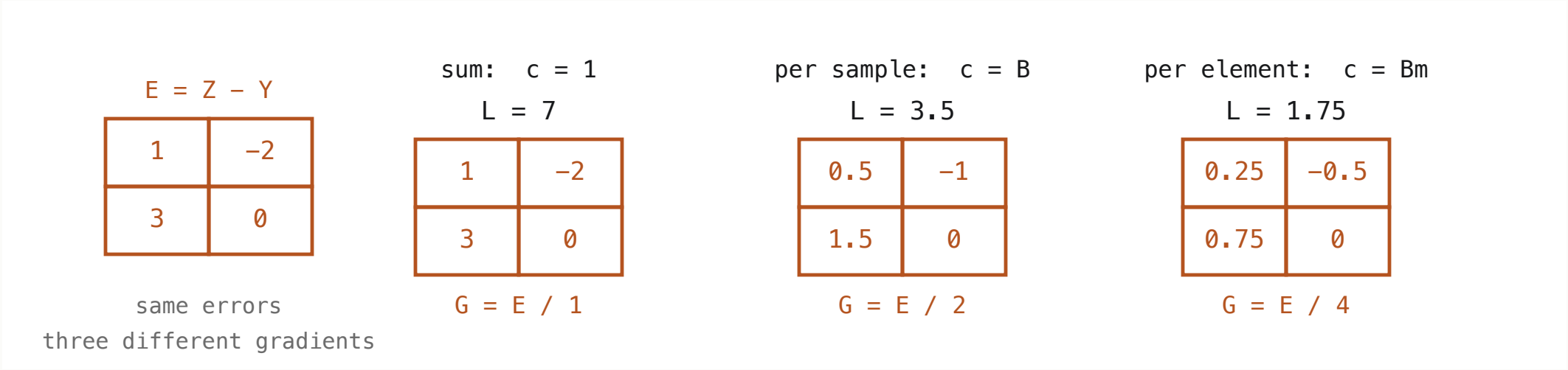
C $(3, 11)$



DEFINITION

Batched Gradients and Broadcasting

Define the Loss Reduction Before Differentiating



$$L = \frac{1}{2c}\sum_{ij} E_{ij}^2 \;\Rightarrow\; G = \nabla_Z L = E/c$$

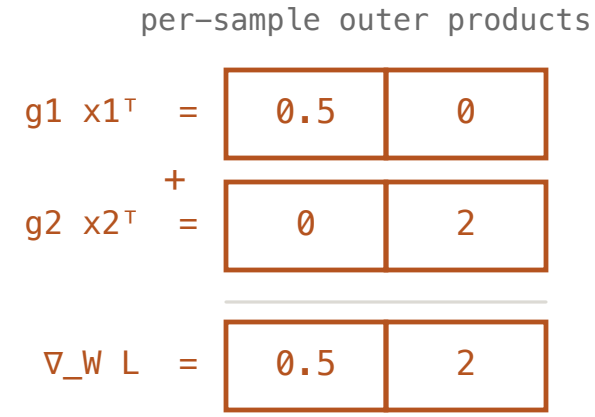
- E $Z - Y$, the error grid, $B \times m$
- c 1, B or Bm — every exercise states which
- G already carries that scale

“MSE” alone is ambiguous: at the same denominator, a framework MSE without the one-half has twice this gradient.

WORKED EXAMPLE

Batched Gradients and Broadcasting

Per-Sample Contributions Add into a Matrix Gradient




```
Z = X Wᵀ = (1, 2)ᵀ
L = ‖Z − Y‖²_F / (2B) = 1.25
G = (Z − Y) / B = (0.5, 1)ᵀ
∇_b L = Gᵀ 1_B = 1.5
∇_X L = G W = [ 0.5  0.5 ; 1  1 ]
the 1/B is inside G already
```

$$\nabla_W L = G^\top X = \sum_{i=1}^{B} g_i \, x_i^\top$$

- $x_i^\top$ row i of X
- g_i row i of G, transposed: $m \times 1$, $1/B$ inside
- sum over samples; do not divide a second time

WORKED EXAMPLE

$X = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$, $W = [1, 1]$, $b = 0$, $Y = 0$, $B = 2$, $L = \frac{1}{2B}\|Z - Y\|_F^2$: then $Z = (1, 2)^\top$ and $L = 1.25$.

VOTE

Batched Gradients and Broadcasting

Find the Double-Averaging Bug

G is already $(Z - Y)/B$ for the same batch. The next line writes `dW = G.T @ X / B`. That result is:

A. correct

B. too small by a factor B

C. too large by a factor B

```
G  = (Z - Y) / B

dW = G.T @ X / B
```

METHOD

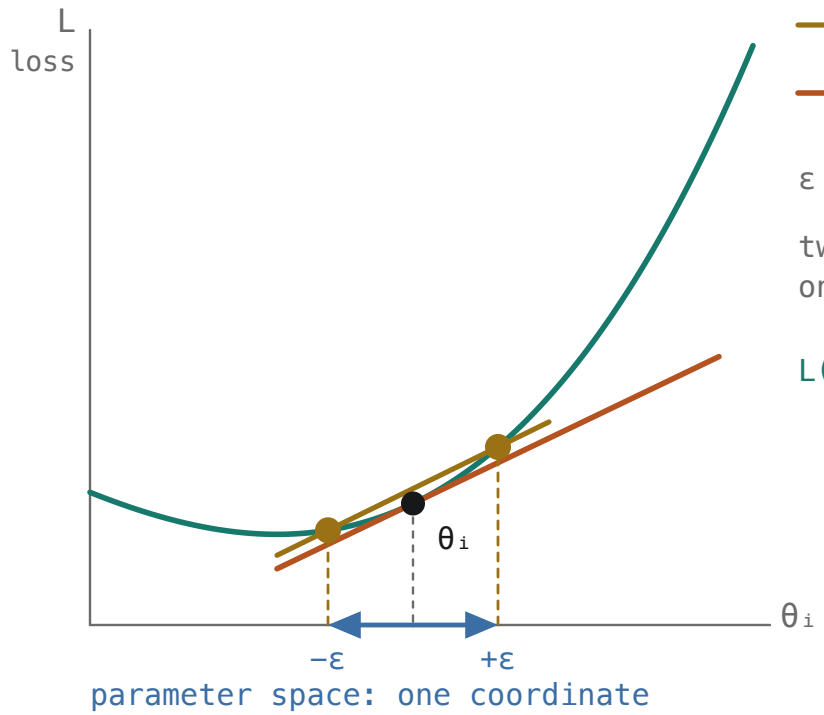
Checking Derivatives

Compare a Gradient with a Measured Change

$$\partial_i L(\theta) \approx \frac{L(\theta + \varepsilon e_i) - L(\theta - \varepsilon e_i)}{2\varepsilon}$$

- e_i only θ_i moves
- ε a finite step: two more evaluations
- $\partial_i L$ the entry the backward pass returns

> A finite difference **measures**. Reverse mode applies the chain rule to the operations as written.



$W_1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

VOTE

Checking Derivatives

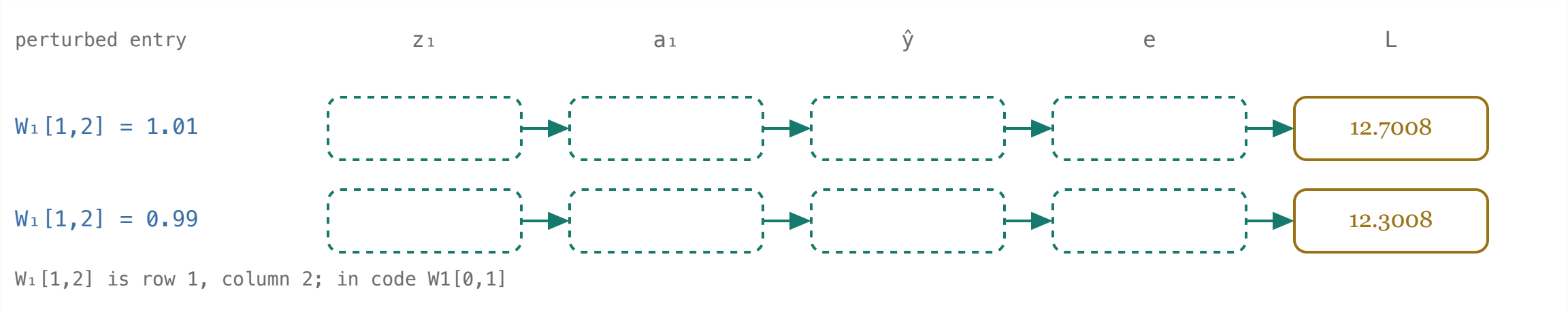
Check One Weight Numerically

Move $W_1[1,2]$ by $\varepsilon = 0.01$, nothing else. Given $L_+ = 12.7008$ and $L_- = 12.3008$, what is $\dfrac{L_+ - L_-}{2\varepsilon}$?

A 10

B 20

C 40



METHOD

Checking Derivatives

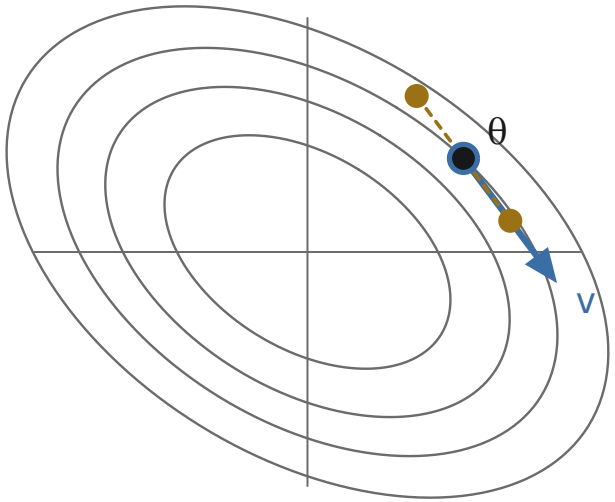
Check a Direction Without Building a Jacobian

$$\phi'(0) = \sum_j \partial_j L \, v_j = \nabla L(\theta)^\top v \approx \frac{\phi(\varepsilon) - \phi(-\varepsilon)}{2\varepsilon}$$

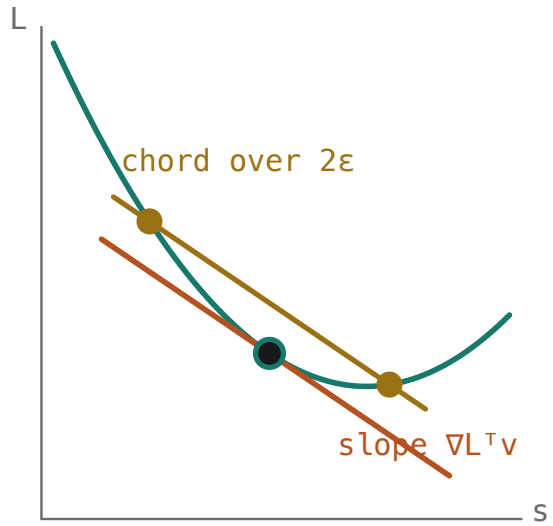
$\phi(s)$	$L(\theta + sv)$; $\phi(0) = L(\theta)$
v_j	the rate of θ_j in s
`cost`	two evaluations, any n

> A coordinate sweep costs two evaluations per coordinate. One direction that agrees is evidence, not proof.

parameter space



φ(s) = L(θ + sv)



‖v‖ = 1 here, so φ'(0) is a projection

θ − εv and θ + εv, ‖v‖ = 1 no Jacobian is formed

VOTE

Checking Derivatives

Is a Smaller Epsilon Always Better?

Does a smaller ε always give a better check?

- **A** yes, without limit
- **B** no — truncation and cancellation pull opposite ways
- **C** ε does not matter

truncation

ε	chord slope
0.80	0.586667
0.20	0.486667
0.05	0.480417

exact 0.480000

the gap falls like ε^2

cancellation

ε = 1e−10

0.75733333338133324

0.75733333328533337

9 leading digits agree;
the difference keeps the rest

$L_+ - L_- = 9.600e{-}11$

WARNING

Checking Derivatives

Nondifferentiability Needs a Different Check

$$\frac{\max(0, \varepsilon) - \max(0, -\varepsilon)}{2\varepsilon} = \frac{1}{2}$$

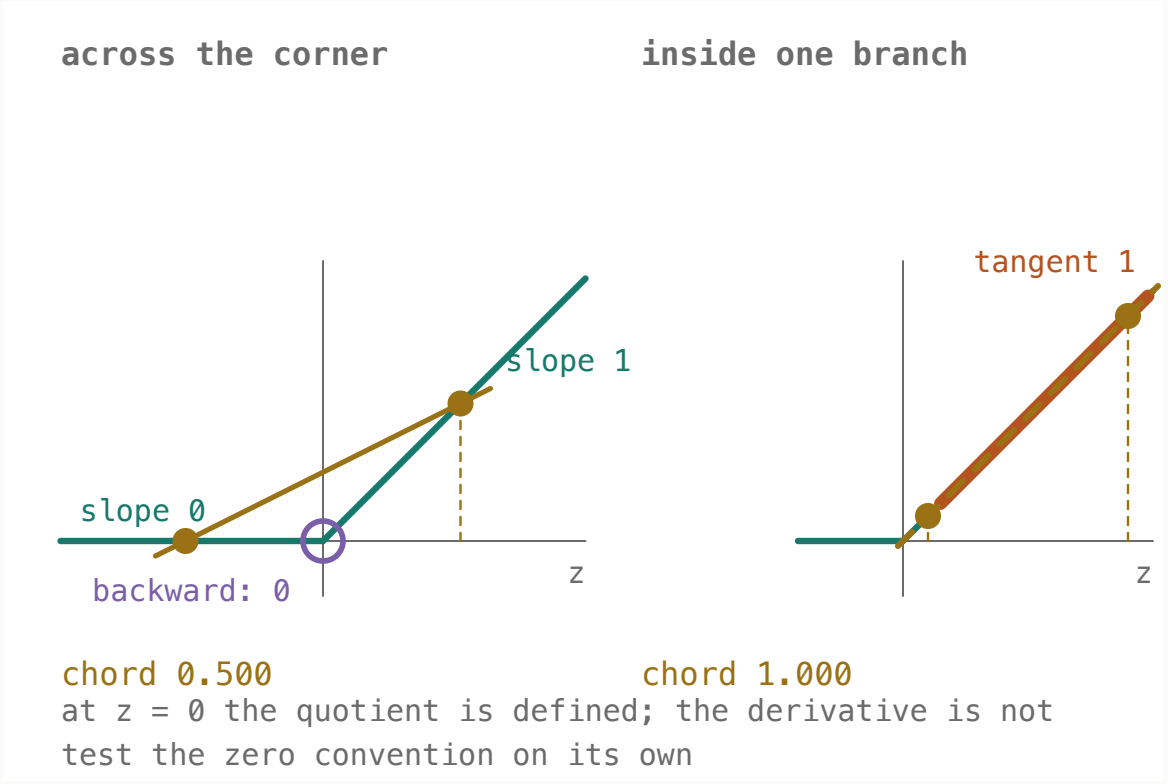
centred quotient $\frac{1}{2}$ for every $\varepsilon > 0$

$z = 0$ no ordinary derivative; our backward gives 0

WORKED EXAMPLE

$\varepsilon = 0.01$: $(0.01 - 0)/0.02 = \frac{1}{2}$, while the backward pass returns 0.

A mismatch at the corner is not evidence of a broken smooth derivative.



VOTE

Checking Derivatives

What Counts as a Gradient Check?

Which test shows a manual backward returns the right numbers?

- (A) the gradient is not `None`
- (B) values against AD and an independent finite difference
- (C) the shape only

what the function returned

2	1	3
−4	−2	−6

2 × 3, finite, not None
another one returned a different 2 × 3 grid

1 not None
2 the right shape
3 the right values

three different questions

Inspected 2026-09-30: ***TorchCode*** `torch_judge/tasks/linear.py` blob `82fe0b4d`: one task at one version, not every TorchCode test.

IN PRACTICE

From Formulas to Small Tensor Programs

Read the Exercise as a Mathematical Contract

$$Z = XW^\top + \mathbf{1}_B b^\top, \quad G = \nabla_Z L$$

X, W, b $B \times n, m \times n, m \times 1$

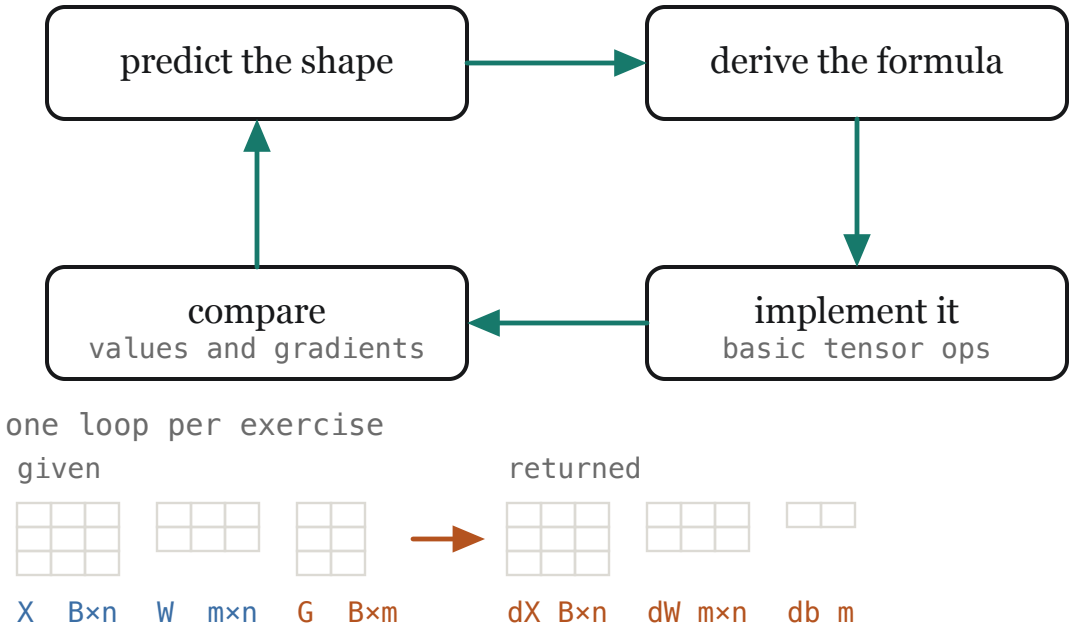
G $B \times m$: the reduction is already in it

CONTRACT

`affine_backward(X, W, G)`

→ `dX` $B \times n$, `dW` $m \times n$, `db` m

Exercise shape after ***TorchCode***, github.com/duoan/TorchCode; the fixtures and the checker are ours.



IN PRACTICE

From Formulas to Small Tensor Programs

Translate the Affine Backward Rules

$$\nabla_X L = GW, \qquad \nabla_W L = G^\top X, \qquad \nabla_b L = G^\top \mathbf{1}_B$$

GW $(B \times m)(m \times n)$: m is summed away

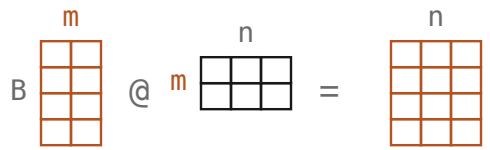
$G^\top X$ $(m \times B)(B \times n)$: B is summed away

$G^\top \mathbf{1}_B$ adds the B rows: $m \times 1$ in maths

A manual backward uses basic tensor operations; it never calls `.backward()`. AD is the checker's job.

```
dX = G @ W
```

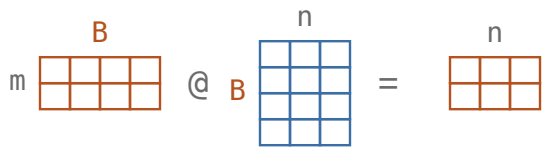
(B×m) @ (m×n) → (B×n)



the m axis is summed away

```
dW = G.T @ X
```

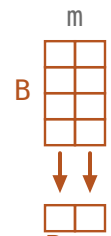
(m×B) @ (B×n) → (m×n)



the B axis is summed away

```
db = G.sum(dim=0)
```

(B×m) → (m)



the B rows add

maths m×1 column, code shape (m,)

VOTE

From Formulas to Small Tensor Programs

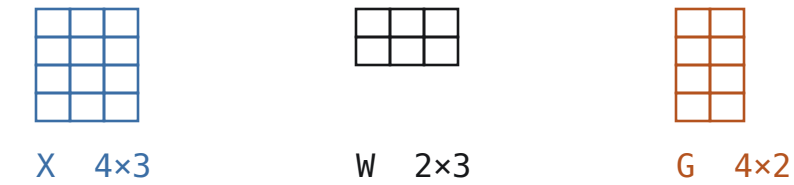
Find the Wrong Transpose

X is 4×3, W is 2×3, G is 4×2. Which line gives dX?

A `G @ W`

B `G @ W.T`

C `W @ G`



IN PRACTICE

From Formulas to Small Tensor Programs

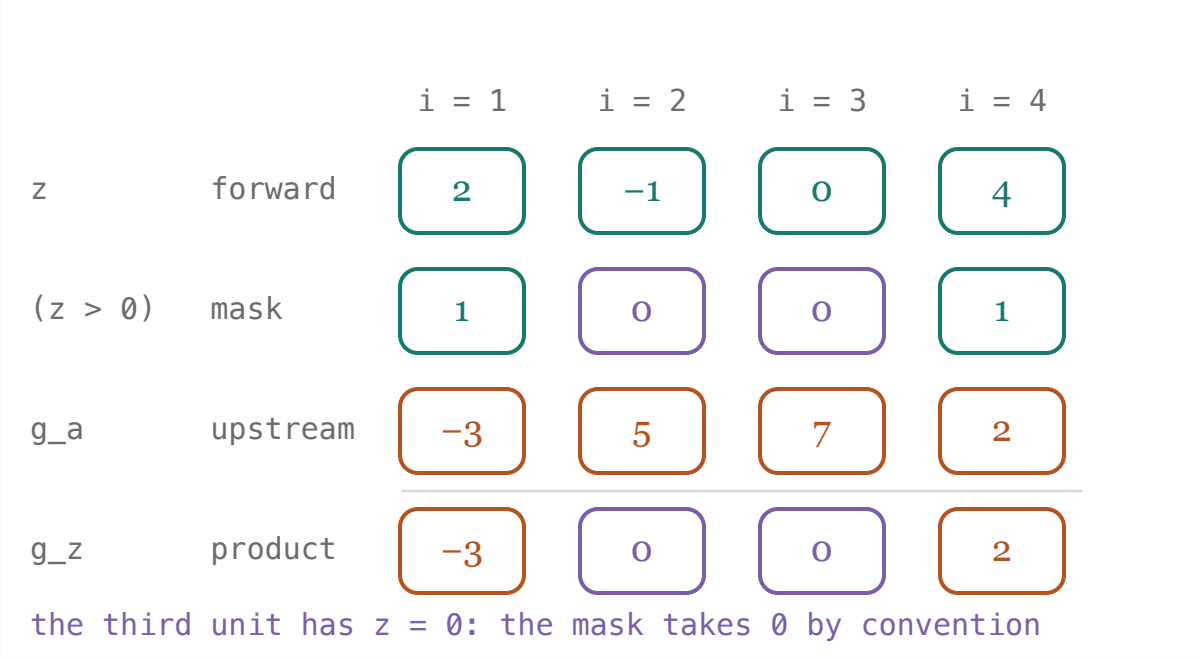
Translate ReLU Backward

$$g_z = g_a \odot \mathbf{1}_{z>0}$$

- $\odot$ entrywise product
- $\mathbf{1}_{z>0}$ the ReLU slope; Boolean, read as 1 and 0
- z the cached forward value; at $z_i = 0$ we use 0

`g_z = g_a * (z > 0)` — the mask needs the cached forward z.

> A test that only checks nonnegative forward outputs says nothing about the backward pass.



VOTE

From Formulas to Small Tensor Programs

Which Mask Computes the Correct Derivative?

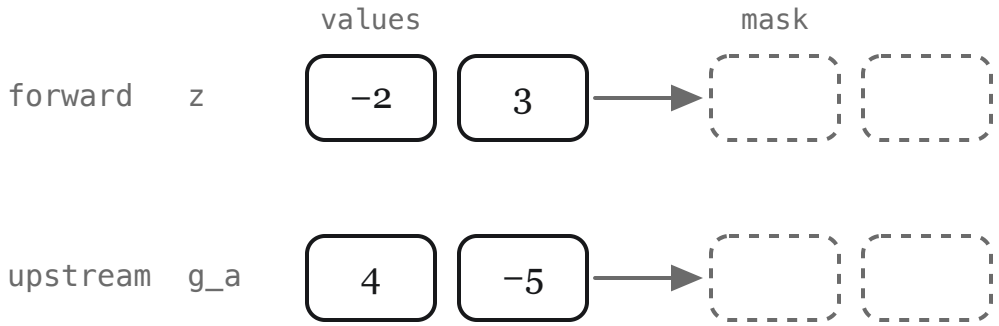
$z = (-2, 3)$ and $g_a = (4, -5)$. Which line gives g_z?

A `g_a * (g_a > 0)`

B `g_a * (z > 0)`

C `relu(g_a)`

a mask could be read from either row



$$W_1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

IN PRACTICE

From Formulas to Small Tensor Programs

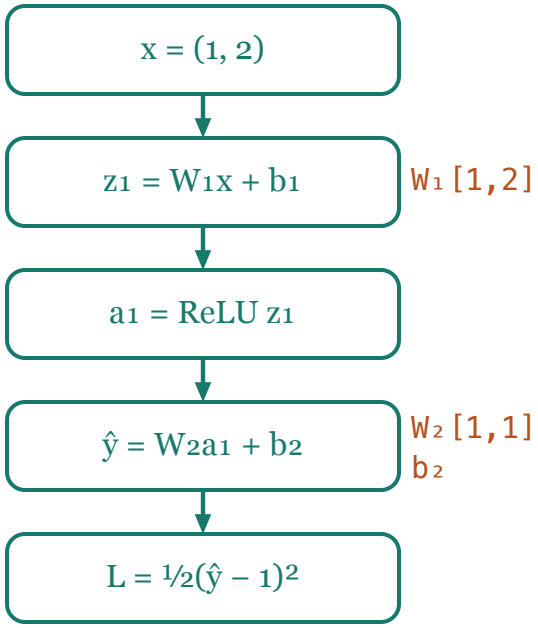
Three Checks of the Same Small Network

ENTRY	BY HAND	PROGRAM	AUTOGRAD	CENTRED DIFFERENCE
$W_1[1,2]$	20	20	20	20.000
$W_2[1,1]$	15	15	15	15.000
b_2	5	5	5	5.000

> Four columns, three checks of the hand column. `ε = 1e-4 · autograd 0 (tol 1e-9) · fd ≤ 2.406e-12 (tol 2e-5)`

> Scaled error $|a-b|/\max(1,|a|,|b|)$: absolute near zero, not relative. No torch: autograd rows read `skip`, not pass.

Measured by `python3 build/exercises/m5/check.py --verbose --require-autograd` at `0b3feee`, float64: 85 comparisons executed, 0 failed, 0 skipped. Transcript in `build/exercises/m5/EVIDENCE.md`.



the entries the table checks

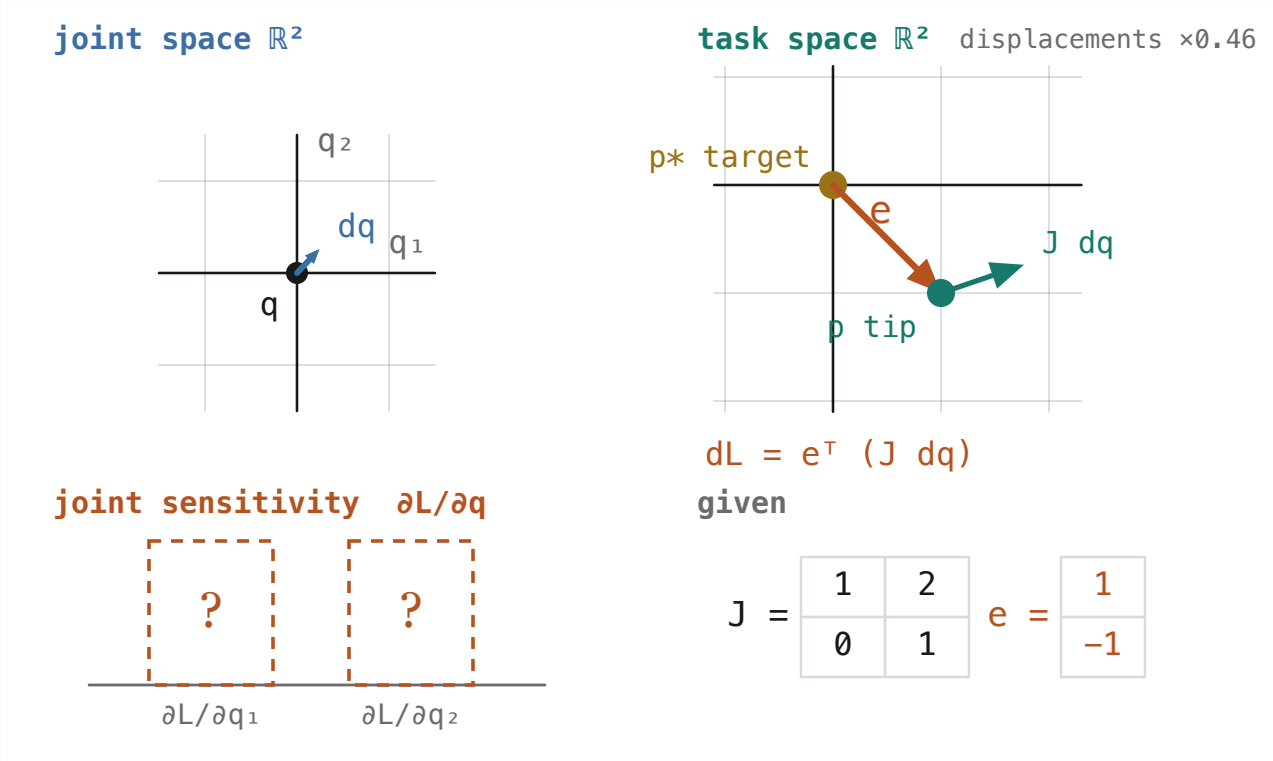
APPLICATION

Reading and Debugging Derivatives

A Robot Error Uses the Same Chain Rule

$$dL = e^\top dp = e^\top J\, dq \;\Rightarrow\; \nabla_q L = J^\top e$$

- q — the differentiated variable, $p = f(q)$ the tip
- $e = p - p^\star$ — $L = \frac{1}{2}\|e\|^2$, so $dL = e^\top de$
- J — $m \times n$: the local map $dq \mapsto dp$
- $\nabla_q L$ — $n \times 1$, shaped like q; a loss gradient, not a joint velocity



VOTE

Reading and Debugging Derivatives

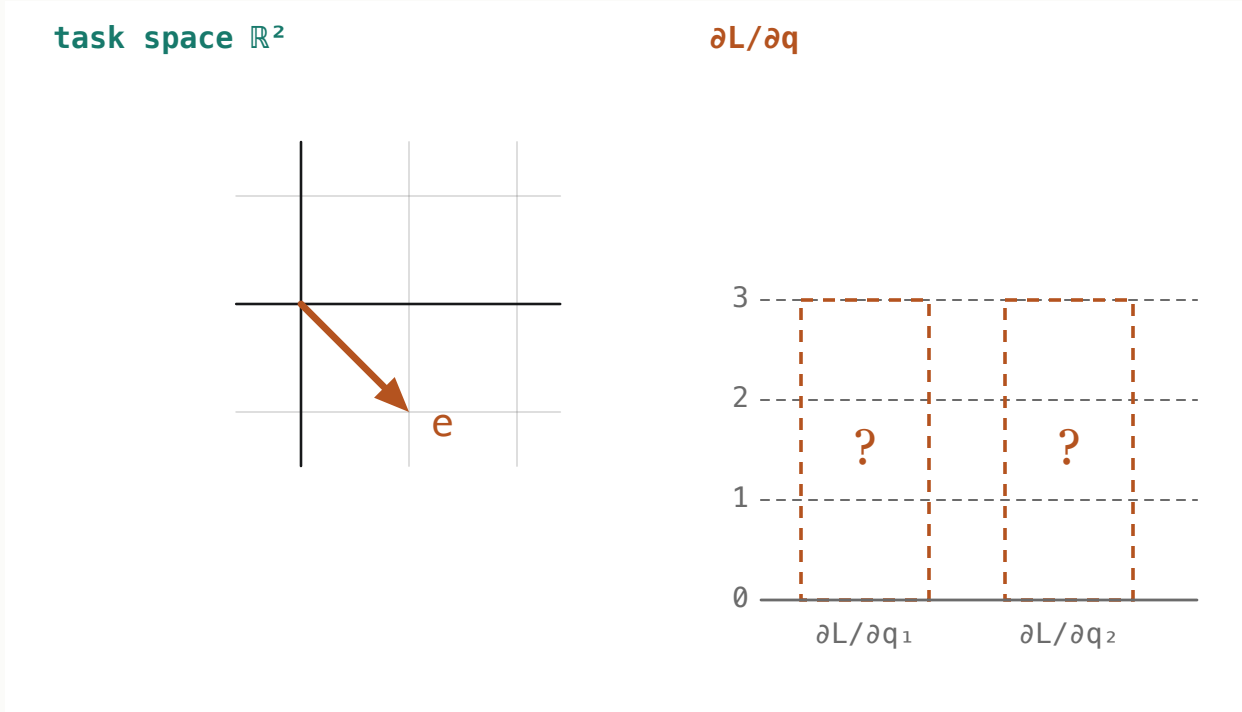
Which Joint Has the Larger Signed Derivative?

$J = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}$, $e = (1, -1)$. What is $\nabla_q L = J^\top e$?

A $(1, 1)$

B $(-1, -1)$

C $(1, 3)$



COMPARISON

Reading and Debugging Derivatives

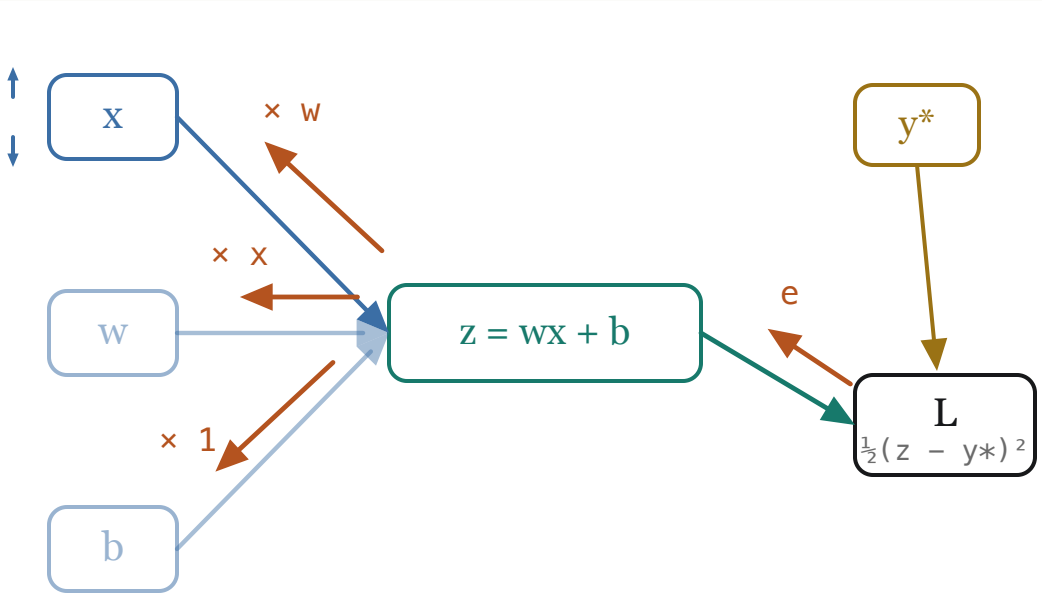
Input Gradients and Parameter Gradients Are Different Questions

$$\frac{\partial L}{\partial x} = ew, \qquad \frac{\partial L}{\partial w} = ex, \qquad \frac{\partial L}{\partial b} = e$$

$z = wx + b$ $L = \frac{1}{2}(z - y^\star)^2$ and $e = z - y^\star$

x, w, b one varies, the other two are held fixed

`all three` are scalars: a shape check cannot tell them apart



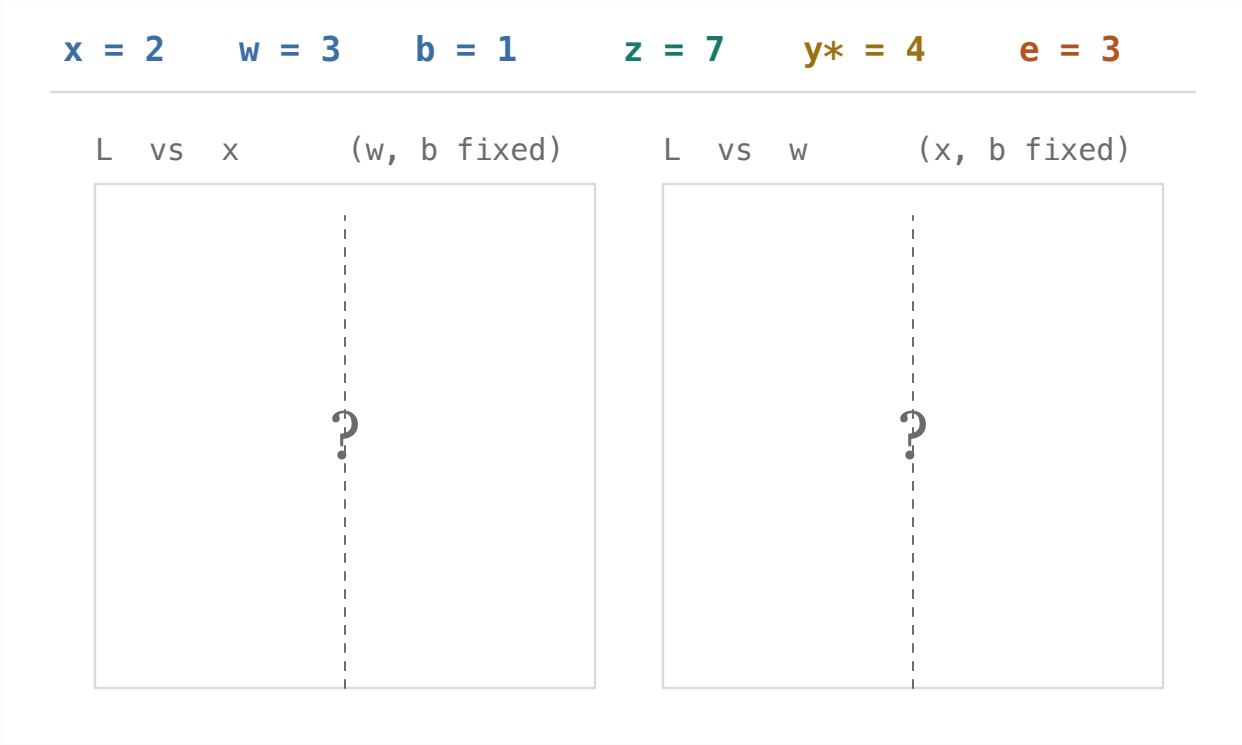
VOTE

Reading and Debugging Derivatives

Same Forward Value, Different Derivative

$x = 2,\ w = 3,\ b = 1,\ y^\star = 4$, so $z = 7$ and $e = 3$.
What is $(\partial L/\partial x,\ \partial L/\partial w)$?

- **A** $(6, 9)$
- **B** $(9, 6)$
- **C** $(3, 3)$



WARNING

Reading and Debugging Derivatives

Passing a Shape Check Is Not Enough

$$Z = XW^{\mathsf{T}}, \qquad \nabla_W L = G^{\mathsf{T}} X$$

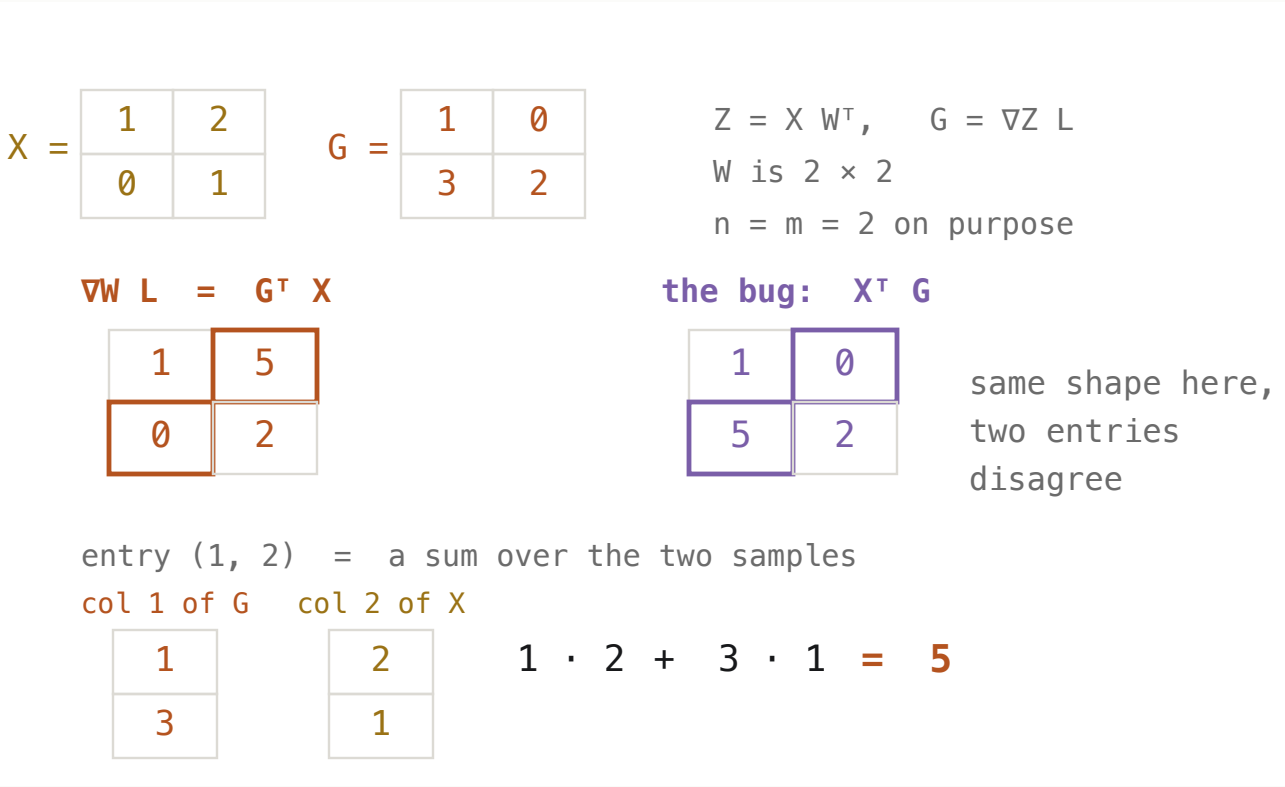
X — $B \times n$ sample rows

$G = \nabla_Z L$ — $B \times m$ upstream

$\nabla_W L$ — $m \times n$; $X^{\mathsf{T}} G$ is $n \times m$, equal here only because $n = m = 2$

WORKED EXAMPLE

Entry $(1, 2)$ sums over the two samples: $1 \cdot 2 + 3 \cdot 1 = 5$. The transposed product puts 0 there.



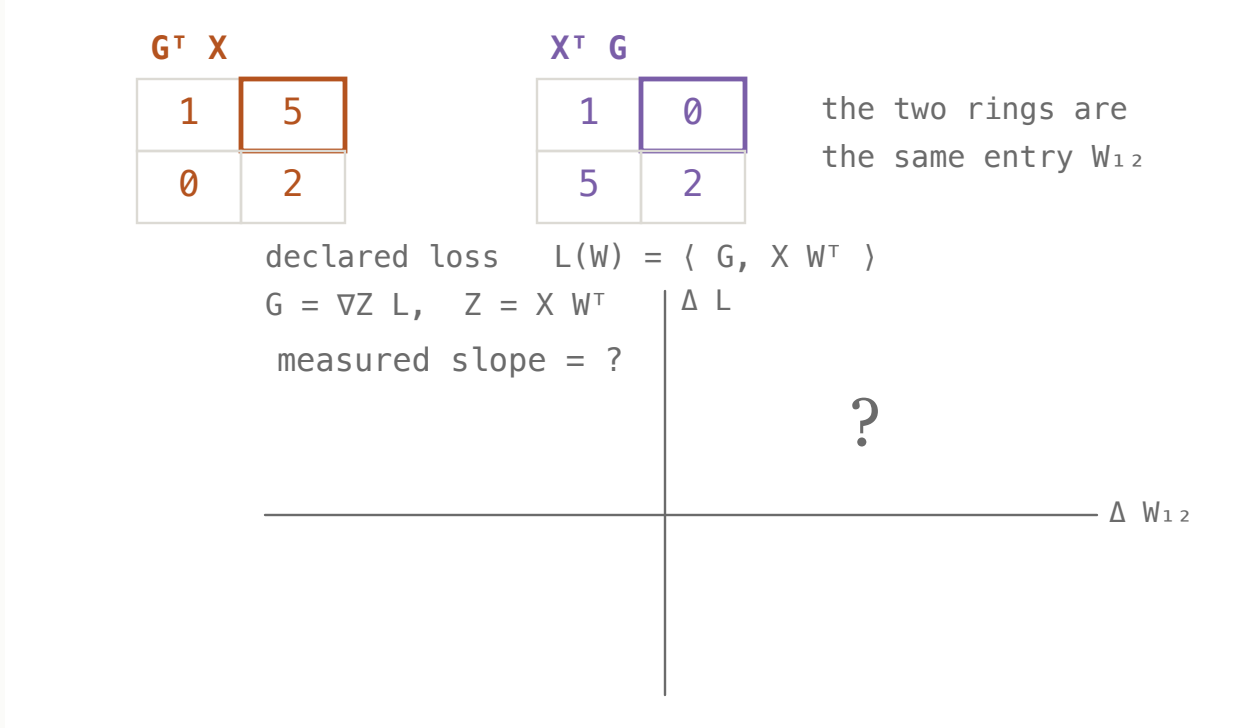
VOTE

Reading and Debugging Derivatives

Which Test Detects the Hidden Bug?

Both candidates are 2×2. Which test separates them?

- **A** assert the shape is 2×2
- **B** check that every value is finite
- **C** compare an entrywise derivative with the declared loss



SETUP

Putting the Derivatives Together

A Complete Regression Graph

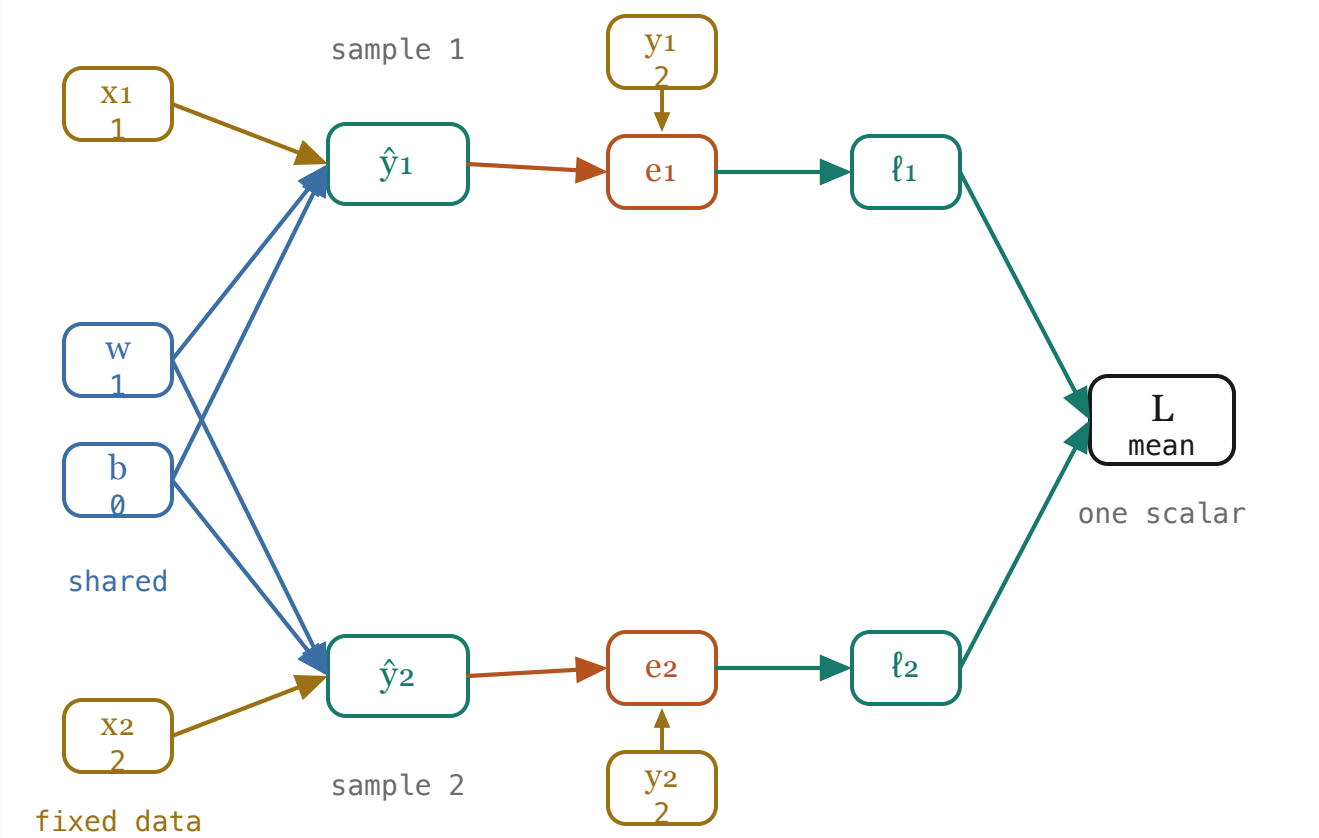
$$\hat{y}_i = w x_i + b, \qquad e_i = \hat{y}_i - y_i$$

$$L = \frac{1}{2B} \sum_{i=1}^{B} e_i^2$$

$B = 2$ the declared reduction is a mean over samples

x_i, y_i fixed data, not differentiated

w, b shared by both branches: paths will add



WORKED FORWARD

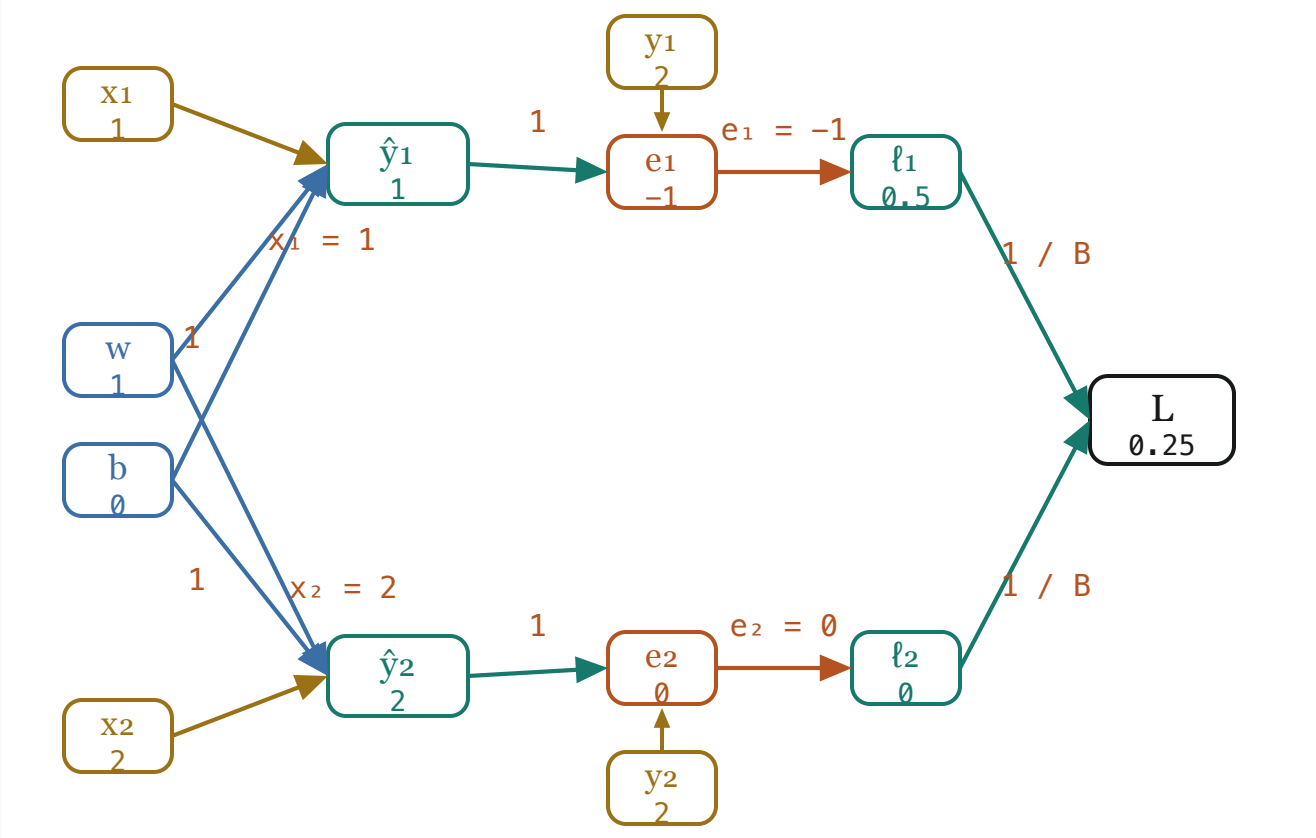
Putting the Derivatives Together

Forward Values and Local Derivatives

$$\partial \hat{y}_i / \partial w = x_i, \qquad \partial \hat{y}_i / \partial b = 1$$

$$\partial L / \partial e_i = e_i / B$$

$\hat{y} = (1, 2)$	errors $e = (-1, 0)$ at $w = 1, b = 0$
$\ell_i = \frac{1}{2} e_i^2$	0.5 and 0; the mean $L = 0.25$
e_i / B	combined sensitivity
`edges`	one factor each, combined nowhere yet



VOTE · PAIRS

Putting the Derivatives Together

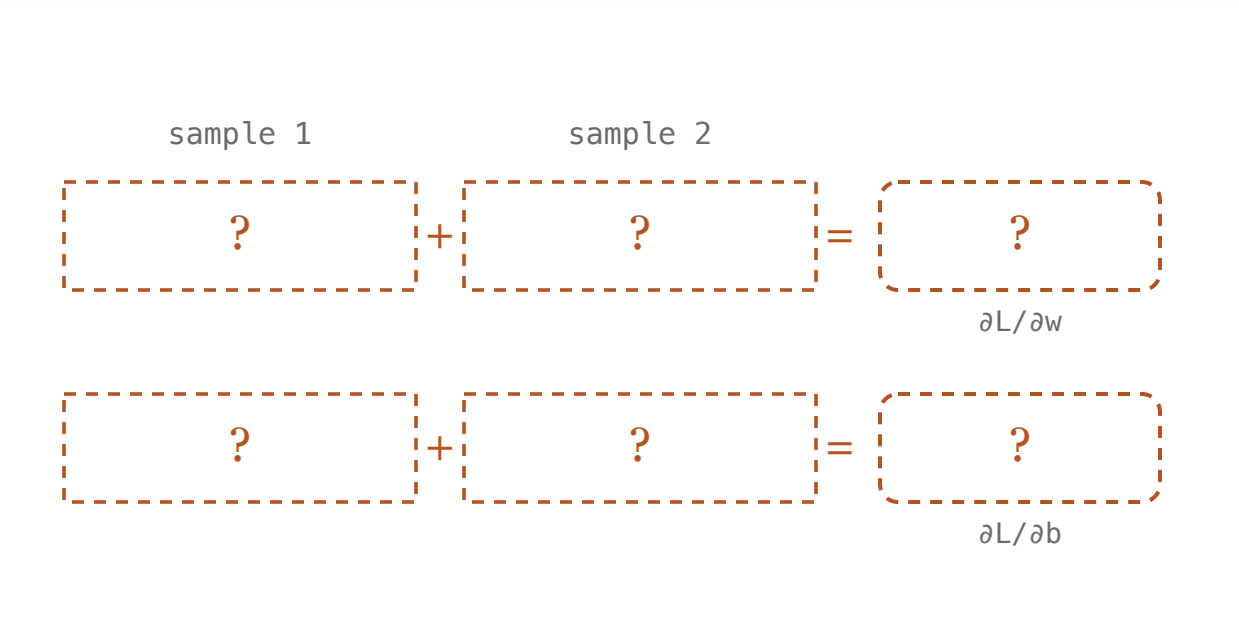
Accumulate the Parameter Gradients

With $\hat{y} = (1, 2)$, $e = (-1, 0)$ and $B = 2$: what is $(\partial L/\partial w,\ \partial L/\partial b)$?

A $(-1, -1)$

B $(-0.5, -0.5)$

C $(0.5, 0.5)$



SYNTHESIS

Putting the Derivatives Together

Write the Same Calculation as Matrix Calculus

$$L = \frac{1}{2B}\|A\theta - y\|^2, \qquad \nabla_\theta L = \frac{1}{B}A^\top(A\theta - y)$$

- A — $B \times 2$, row i is $(x_i, 1)$
- $\theta = (w, b)^\top$ — the parameters as one 2×1 vector
- $A\theta - y$ — $= (-1, 0)$: the same errors as the graph

two scalar branches

$\hat{y}_1 = 1 \cdot w + 1 \cdot b$

$\hat{y}_2 = 2 \cdot w + 1 \cdot b$

$$\underset{A}{\begin{bmatrix}1 & 1\\ 2 & 1\end{bmatrix}} \underset{\substack{\theta\\ (w,\ b)}}{\begin{bmatrix}1\\ 0\end{bmatrix}} - \underset{y}{\begin{bmatrix}2\\ 2\end{bmatrix}} = \begin{bmatrix}-1\\ 0\end{bmatrix}$$

$\nabla\theta\ L \ = \ (1/B)\ A^\top\ (A\theta - y)$

$$\underset{A^\top}{\begin{bmatrix}1 & 2\\ 1 & 1\end{bmatrix}} \cdot \begin{bmatrix}-1\\ 0\end{bmatrix} / 2 = \begin{bmatrix}-0.5\\ -0.5\end{bmatrix}$$

the graph route same pair

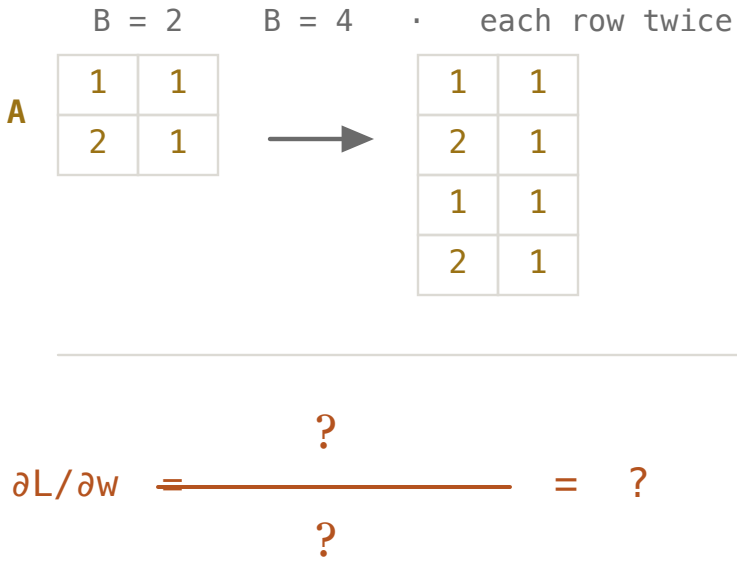
VOTE

Putting the Derivatives Together

What If We Duplicate Every Sample?

Duplicate both samples, so $B = 4$, with the same mean loss and the same w, b. The gradient…

- (A) doubles
- (B) stays $(-0.5, -0.5)$
- (C) halves



CHECK

Putting the Derivatives Together

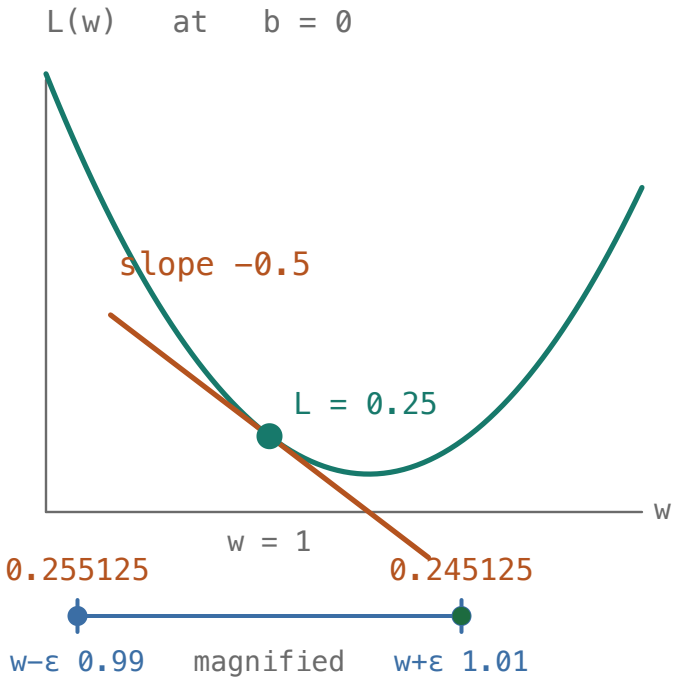
Verify the Capstone Independently

$$L(w) = \tfrac{1}{4}\left[(w-2)^2 + (2w-2)^2\right]$$
$$L'(1) = -\tfrac{1}{2}$$

- $b = 0$ held fixed, so this is a 1-D slice
- $\varepsilon = 0.01$ the probe step, so the probes sit 0.02 apart; a quotient is evidence

WORKED EXAMPLE

Probes 0.245125 and 0.255125 differ by -0.01; over 0.02 that is -0.5.



graph
Σ eᵢxᵢ / B = −0.5

matrix
[Aᵀ(Aθ − y)/B]₁ = −0.5

probe
−0.01 / 0.02 = −0.5

all three agree

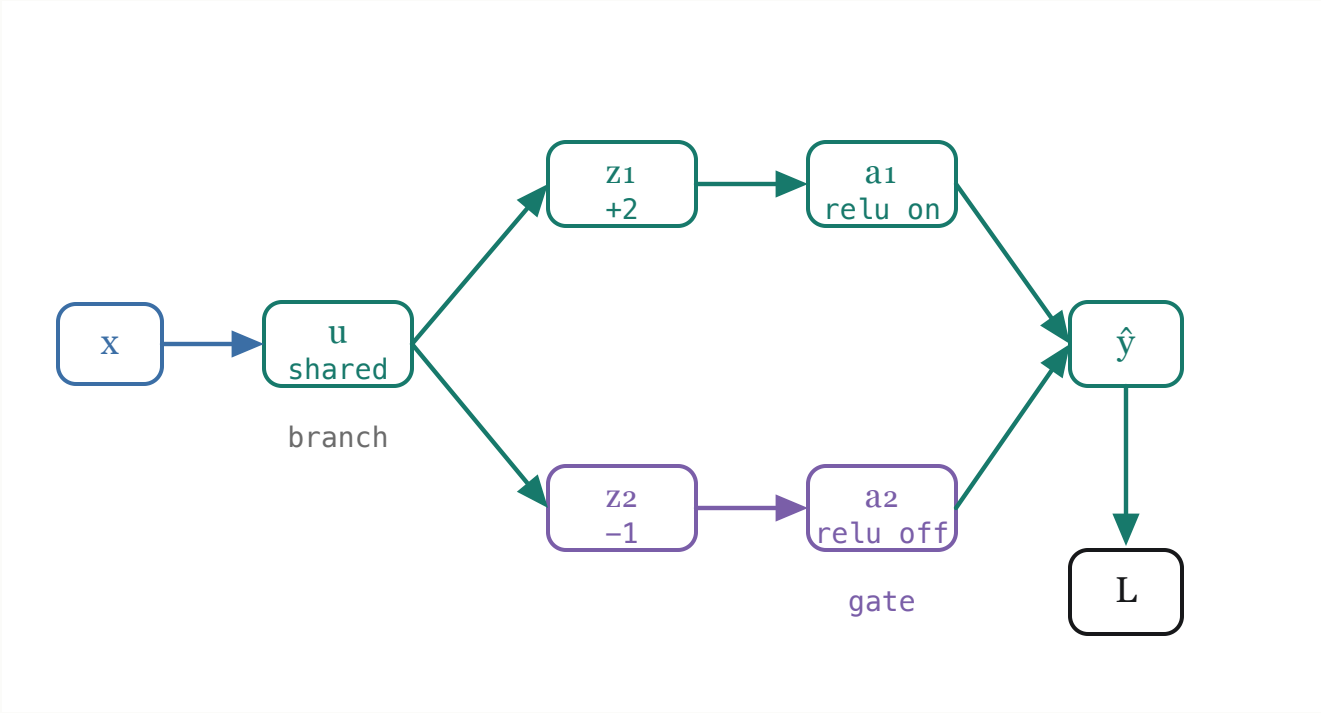
VOTE

Putting the Derivatives Together

Exit Vote: What Actually Travels Backward?

What actually travels backward?

- **A** the prediction error, copied through every layer
- **B** downstream sensitivity × local derivatives, summed over paths
- **C** inverse matrices that undo the forward pass



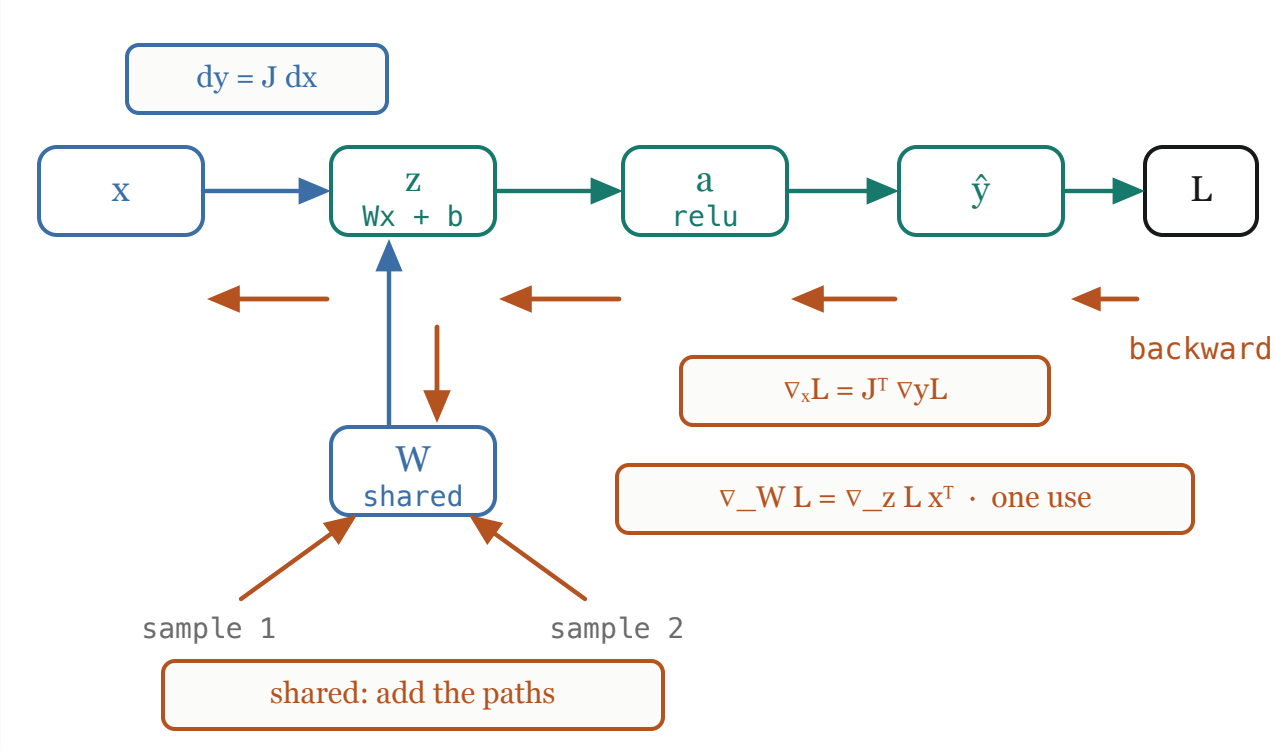
RECAP

Putting the Derivatives Together

Four Rules to Take Away

1. $dy = J\,dx$ — a derivative is a local linear map
2. $\nabla_x L = J^\top \nabla_y L$ — the transpose, on column gradients
3. $\nabla_W L = g_z x^\top$ — one affine use: an outer product, shaped like W
4. a shared variable, or a batch, sums its paths

> Next: new data, your choice of method.



VOTE · PAIRS

Putting the Derivatives Together

New Data, Same Method?

$$\hat{y}_i = wx_i + b, \quad L = \frac{1}{2B}\sum_i (\hat{y}_i - y_i)^2, \quad B = 2$$

(x_i, y_i) $(-1, 1), (2, 1)$: input, target

(w, b) $(1, 0)$: shared parameters

Find L and both gradients. Choose a method; check one slope.

A $(2, -\frac{1}{2})$

B $(4, -1)$

C $(1, -\frac{1}{4})$

References

Where the rules come from

1. Goodfellow, Bengio & Courville. *Deep Learning*, ch. 6, §6.5. MIT Press, 2016.
2. Stanford CS231n. *Backpropagation, Intuitions*. Course notes.
3. Stanford CS231n. *Gradient checks*. Course notes.
4. Baydin, Pearlmutter, Radul & Siskind. *Automatic Differentiation in Machine Learning: a Survey*. JMLR, 2018.
5. Parr & Howard. *The Matrix Calculus You Need For Deep Learning*. arXiv, 2018.

References

Practice, and the course standards

6. *TorchCode*, duoan/TorchCode. Auto-graded tensor exercises: practice, not a source of notation.
7. ROB-GY 6013. *Notation and Conventions*. Columns, the gradient row, the batch layout, r against e.
8. ROB-GY 6013. *Textbooks and Reading Guide*. What each source is for, and where its notation differs.

Practice-sheet pattern after TorchCode, MIT licence. Gradient existence: its ***torch_judge/tasks/linear.py***, blob 82fe0b4d81b5. Kink convention: ***tasks/relu.py***, blob aea98fe24001. Both inspected 2026-09-30.