One-hidden-layer backprop, the whole backward pass

Ten problems assembling the full backward pass of a one-hidden-layer classifier: the two deltas, every parameter gradient with its shape, the gradient to the input, the batched version with the 1/N, a ReLU variant and an L2 penalty, with worked solutions and the mistakes that swap shapes or drop factors.

BEFORE YOU START

The softmax and Jacobians pages computed the pieces of a backward pass: the gradient of softmax with cross-entropy, and the rule for pushing a gradient back through an affine map and an elementwise nonlinearity. These ten problems assemble them into the whole backward pass of a classifier with one hidden layer, first for one example and then for a batch, and then change the activation and the loss to see what moves. The five mistakes are the ones that survive a quick read: an outer product and a batched product transposed, the activation's derivative dropped, a bias gradient left unsummed, and a stray factor of 2 in weight decay.

- The conventions are those of the previous pages: vectors are columns, Jacobians are in numerator layout, a gradient has the shape of the variable it is taken with respect to, and for a scalar L of u with u a function of x, $\nabla_x L = (\partial u/\partial x)^\top \nabla_u L$.
- The network: $x \in \mathbb{R}^d$; $z_1 = W_1 x + b_1 \in \mathbb{R}^m$ and $h = \sigma(z_1)$, where σ is the logistic sigmoid applied elementwise, with $\sigma' = \sigma(1-\sigma)$; $z_2 = W_2 h + b_2 \in \mathbb{R}^k$; $\hat{y} = \mathrm{softmax}(z_2)$; and $L = -\sum_i y_i \log \hat{y}_i$ with y one-hot.
- The two deltas are the gradients at the pre-activations: $\delta_2 := \nabla_{z_2} L$ and $\delta_1 := \nabla_{z_1} L$.
- Entries: $W_{2,ij}$ is entry (i,j) of W_2, and $z_{2,i}$, $\delta_{2,i}$, h_j are entries of z_2, δ_2, h; likewise one layer down.
- $\odot$ is the elementwise product and $\mathbf{1}$ the all-ones vector.
- Batched: rows are examples. $X \in \mathbb{R}^{N \times d}$, $Z_1 = X W_1^\top + \mathbf{1} b_1^\top$, $H = \sigma(Z_1)$, $Z_2 = H W_2^\top + \mathbf{1} b_2^\top$, $\hat{Y}$ is the row-wise softmax of Z_2, Y holds the one-hot targets as rows, and $L = \frac{1}{N}\sum_n L_n$ is the mean of the per-example losses. $\Delta_2 := \nabla_{Z_2} L$ and $\Delta_1 := \nabla_{Z_1} L$.
- The softmax page gives $\nabla_{z_2} L = \hat{y} - y$, and the Jacobians page gives the pattern $\nabla_{\text{in}} = W^\top(\sigma'(z) \odot \nabla_{\text{out}})$; this page only assembles them, so every step is a citation plus a shape check.

Builds on: Jacobians and the chain rule, The softmax Jacobian

Problems

1. List the shapes of $W_1, b_1, z_1, h, W_2, b_2, z_2, \hat{y}$ and of $\nabla_{W_1} L, \nabla_{b_1} L, \nabla_{W_2} L, \nabla_{b_2} L, \nabla_x L$ in terms of d, m, k. difficulty ·

2. Compute $\delta_2 = \nabla_{z_2} L$. difficulty ·

3. Compute $\nabla_{W_2} L$ and $\nabla_{b_2} L$ in terms of δ_2 and h, with shapes. difficulty ··

4. Compute $\nabla_h L$. difficulty ··

5. Compute $\delta_1 = \nabla_{z_1} L$ in terms of δ_2, W_2 and z_1, without forming any Jacobian of σ. difficulty ··

6. Compute $\nabla_{W_1} L$ and $\nabla_{b_1} L$. difficulty ··

7. Compute $\nabla_x L$. Why would you want it? difficulty ··

8. Batched: with $X \in \mathbb{R}^{N \times d}$ (rows are examples), $Z_1 = X W_1^\top + \mathbf{1} b_1^\top$, $H = \sigma(Z_1)$, $Z_2 = H W_2^\top + \mathbf{1} b_2^\top$, $\hat{Y} = \mathrm{softmax}(Z_2)$ row-wise, Y one-hot rows and $L = \frac{1}{N} \sum_n L_n$, compute Δ_2, $\nabla_{W_2} L$, $\nabla_{b_2} L$, Δ_1, $\nabla_{W_1} L$, $\nabla_{b_1} L$ and $\nabla_X L$, with shapes. difficulty ···

9. Replace σ by $\mathrm{ReLU}(z) = \max(z, 0)$. What changes in Problems 5–7? Write δ_1 and $\nabla_x L$. difficulty ···

10. Add an L2 penalty: $L' = L + \frac{\lambda}{2} \|W_2\|_F^2$. Compute $\nabla_{W_2} L'$. difficulty ···

Answers

1. $W_1 : m \times d$, $b_1, z_1, h : m$; $W_2 : k \times m$, $b_2, z_2, \hat{y} : k$; each gradient has the shape of its variable: $\nabla_{W_1} L : m \times d$, $\nabla_{b_1} L : m$, $\nabla_{W_2} L : k \times m$, $\nabla_{b_2} L : k$, $\nabla_x L : d$
2. $\delta_2 = \hat{y} - y$
3. $\nabla_{W_2} L = \delta_2 h^\top$ $(k \times m)$; $\nabla_{b_2} L = \delta_2$
4. $\nabla_h L = W_2^\top \delta_2$
5. $\delta_1 = \sigma'(z_1) \odot (W_2^\top \delta_2)$
6. $\nabla_{W_1} L = \delta_1 x^\top$ $(m \times d)$; $\nabla_{b_1} L = \delta_1$
7. $\nabla_x L = W_1^\top \delta_1$
8. $\Delta_2 = \frac{1}{N}(\hat{Y} - Y)$ $(N \times k)$; $\nabla_{W_2} L = \Delta_2^\top H$ $(k \times m)$; $\nabla_{b_2} L = \Delta_2^\top \mathbf{1}$; $\Delta_1 = \sigma'(Z_1) \odot (\Delta_2 W_2)$ $(N \times m)$; $\nabla_{W_1} L = \Delta_1^\top X$; $\nabla_{b_1} L = \Delta_1^\top \mathbf{1}$; $\nabla_X L = \Delta_1 W_1$ $(N \times d)$
9. $\delta_1 = [z_1 > 0] \odot (W_2^\top \delta_2)$; $\nabla_x L = W_1^\top \delta_1$; nothing else changes
10. $\nabla_{W_2} L' = \delta_2 h^\top + \lambda W_2$

Worked solutions

Problem 1

> List the shapes of $W_1, b_1, z_1, h, W_2, b_2, z_2, \hat{y}$ and of $\nabla_{W_1} L, \nabla_{b_1} L, \nabla_{W_2} L, \nabla_{b_2} L, \nabla_x L$ in terms of d, m, k.

$W_1 x \in \mathbb{R}^m$ for $x \in \mathbb{R}^d$, so W_1 is $m \times d$; b_1 is added to $W_1 x$, so b_1 and z_1 have m entries.	A matrix takes d-vectors to m-vectors only if it has d columns and m rows, and vectors can be added only if their shapes agree.
$h = \sigma(z_1)$ has m entries.	σ is elementwise, so it keeps the shape of its input.
$W_2 h \in \mathbb{R}^k$, so W_2 is $k \times m$; b_2, z_2 and $\hat{y}$ have k entries.	Step 1 one layer up, with h as the input; the softmax gives one probability per class, so it keeps the shape of z_2.
Entry (i, j) of $\nabla_{W_1} L$ is $\partial L / \partial W_{1,ij}$, and likewise for every other variable.	A gradient has one entry per entry of its variable, which is why $W_1 - \eta \nabla_{W_1} L$ is defined.
$W_1 : m \times d$, $b_1, z_1, h : m$; $W_2 : k \times m$, $b_2, z_2, \hat{y} : k$; **each gradient has the shape of its variable:** $\nabla_{W_1} L : m \times d$, $\nabla_{b_1} L : m$, $\nabla_{W_2} L : k \times m$, $\nabla_{b_2} L : k$, $\nabla_x L : d$	Every answer below is checked against this list.

Problem 2

> Compute $\delta_2 = \nabla_{z_2} L$.

As a function of z_2, $L = -\sum_i y_i \log \operatorname{softmax}(z_2)_i$.	$\hat{y} = \operatorname{softmax}(z_2)$, and δ_2 holds everything below z_2 fixed.
Let c be the index with $y_c = 1$. Then $L = -z_{2,c} + \log \sum_j e^{z_{2,j}}$.	Only the target term of the sum survives, and $\log \hat{y}_c = z_{2,c} - \log \sum_j e^{z_{2,j}}$.

$\partial L/\partial z_{2,j} = -y_j + \hat{y}_j.$	$z_{2,c}$ contains $z_{2,j}$ only when $j = c$, which is y_j; the log-sum-exp term differentiates to $e^{z_{2,j}} / \sum_{j'} e^{z_{2,j'}} = \hat{y}_j$. This is the softmax page's Problem 5 with $z = z_2$.
$\delta_2 = \hat{y} - y$	$k \times 1$, the shape of z_2. Its entries sum to 0: the target entry is $\hat{y}_c - 1 < 0$ and the others are positive.

Problem 3

> Compute $\nabla_{W_2} L$ and $\nabla_{b_2} L$ in terms of δ_2 and h, with shapes.

$z_{2,i} = \sum_j W_{2,ij} h_j + b_{2,i}.$	Index form shows which entry of z_2 each entry of W_2 and b_2 builds.
$W_{2,ij}$ appears only in $z_{2,i}$, with coefficient h_j.	It sits in row i of W_2, and row i builds only $z_{2,i}$; within that sum, only term j contains it.
$\partial L/\partial W_{2,ij} = \sum_{i'} \dfrac{\partial L}{\partial z_{2,i'}} \dfrac{\partial z_{2,i'}}{\partial W_{2,ij}} = \delta_{2,i}\, h_j.$	L depends on W_2 only through z_2; by step 2 only the $i' = i$ term is nonzero.
$(\delta_2 h^\top)_{ij} = \delta_{2,i} h_j$, with shapes $(k \times 1)(1 \times m) = k \times m$.	An outer product $ab^\top$ has entries $a_i b_j$: the row index comes from δ_2 and the column index from h, matching step 3.
$\partial L/\partial b_{2,i} = \delta_{2,i}.$	$b_{2,i}$ appears only in $z_{2,i}$, with coefficient 1: $\partial z_2/\partial b_2 = I_k$ (the Jacobians page, Problem 2).
$\nabla_{W_2} L = \delta_2 h^\top$ **(**$k \times m$**)**; $\nabla_{b_2} L = \delta_2$	$k \times m$ and k, the shapes of W_2 and b_2 from Problem 1. It is the Jacobians page's $(Wx - y)\, x^\top$ with δ_2 as the incoming gradient: the gradient at the output of the affine map times its input, transposed.

Problem 4

> Compute $\nabla_h L$.

$\partial z_2/\partial h = W_2,\ k \times m.$	$z_2 = W_2 h + b_2$ is affine in h (the Jacobians page, Problem 2).
$\nabla_h L = (\partial z_2/\partial h)^\top \nabla_{z_2} L.$	L depends on h only through z_2, so the chain rule for gradients applies.
$\nabla_h L = W_2^\top \delta_2$	Shapes $(m \times k)(k \times 1) = m \times 1$, the shape of h. Entry j is $\sum_i W_{2,ij}\,\delta_{2,i}$: hidden unit j collects the output deltas weighted by its outgoing weights, column j of W_2.

Problem 5

> Compute $\delta_1 = \nabla_{z_1} L$ in terms of δ_2, W_2 and z_1, without forming any Jacobian of σ.

$\partial h/\partial z_1 = \operatorname{diag}(\sigma'(z_1)),\ m \times m.$	$h = \sigma(z_1)$ is elementwise (the Jacobians page, Problem 1).
$\delta_1 = (\partial h/\partial z_1)^\top \nabla_h L = \operatorname{diag}(\sigma'(z_1))\, W_2^\top \delta_2.$	L depends on z_1 only through h; a diagonal matrix is its own transpose; $\nabla_h L$ is Problem 4.
$\operatorname{diag}(\sigma'(z_1))\, v = \sigma'(z_1) \odot v$ for any $v \in \mathbb{R}^m$.	A diagonal matrix acting on a vector scales entry i by diagonal entry i: the vector-Jacobian product of the Jacobians page, Problem 6, so the $m \times m$ matrix is never built.
$\delta_1 = \sigma'(z_1) \odot (W_2^\top \delta_2)$	$(m \times 1) \odot (m \times 1) = m \times 1$, the shape of z_1. For the sigmoid, $\sigma'(z_1) = h \odot (1 - h)$, computed from the h saved in the forward pass.

Problem 6

> Compute $\nabla_{W_1} L$ and $\nabla_{b_1} L$.

$z_1 = W_1 x + b_1$ has the form of $z_2 = W_2 h + b_2$, with x in place of h and $\delta_1 = \nabla_{z_1} L$ in place of δ_2.	Problem 3 used only that form and the gradient at its output, so its argument applies unchanged.
$\partial L / \partial W_{1,ij} = \delta_{1,i}\, x_j$ and $\partial L / \partial b_{1,i} = \delta_{1,i}$.	$W_{1,ij}$ appears only in $z_{1,i}$, with coefficient x_j, and $b_{1,i}$ only in $z_{1,i}$, with coefficient 1 (Problem 3, steps 2 to 5).
$\nabla_{W_1} L = \delta_1 x^\top$ **(**$m \times d$**)**; $\nabla_{b_1} L = \delta_1$	$(m \times 1)(1 \times d) = m \times d$ and m, the shapes of W_1 and b_1 from Problem 1.

Problem 7

> Compute $\nabla_x L$. Why would you want it?

$\partial z_1 / \partial x = W_1$, $m \times d$.	$z_1 = W_1 x + b_1$ is affine in x.
$\nabla_x L = (\partial z_1 / \partial x)^\top \delta_1 = W_1^\top \delta_1$, with shapes $(d \times m)(m \times 1) = d \times 1$.	L depends on x only through z_1; the result has the shape of x.
No parameter update uses it.	x is data, not a parameter, so gradient descent never changes it.
$\nabla_x L = W_1^\top \delta_1$	You want it for three things. Adversarial examples: a small step along $\operatorname{sign}(\nabla_x L)$ raises the loss the most, to first order, among changes of the same maximum entry size. Saliency: the size of entry j says how sensitive the loss is to input feature j. And if x is itself the output of a layer below, $\nabla_x L$ is that layer's incoming gradient, from which it gets its own δ_0 exactly as Problem 5 got δ_1 from $W_2^\top \delta_2$.

Problem 8

> Batched: with $X \in \mathbb{R}^{N \times d}$ (rows are examples), $Z_1 = XW_1^\top + \mathbf{1}b_1^\top$, $H = \sigma(Z_1)$, $Z_2 = HW_2^\top + \mathbf{1}b_2^\top$, $\hat{Y} = \text{softmax}(Z_2)$ row-wise, Y one-hot rows and $L = \frac{1}{N}\sum_n L_n$, compute Δ_2, $\nabla_{W_2} L$, $\nabla_{b_2} L$, Δ_1, $\nabla_{W_1} L$, $\nabla_{b_1} L$ and $\nabla_X L$, with shapes.

Write $x^{(n)}$ for example n as a column, so row n of X is $x^{(n)\top}$, and likewise $h^{(n)}$, $\hat{y}^{(n)}$, $y^{(n)}$, and $\delta_2^{(n)} = \hat{y}^{(n)} - y^{(n)}$, $\delta_1^{(n)}$ for the single-example deltas of Problems 2 and 5.

Row n of Z_1 is $x^{(n)\top}W_1^\top + b_1^\top = (W_1 x^{(n)} + b_1)^\top$, and likewise down the network.	Row n of $XW_1^\top$ is row n of X times $W_1^\top$, and $\mathbf{1}b_1^\top$ adds $b_1^\top$ to every row. So the batch is N copies of the single-example network, one per row, sharing the parameters.
L_n depends only on row n of X, Z_1, H, Z_2 and $\hat{Y}$, so row n of Δ_2 is $\frac{1}{N}\,\delta_2^{(n)\top}$.	Only the n-th term of the mean contains row n of Z_2, and it carries the factor $\frac{1}{N}$; the rest is Problem 2 for example n (the softmax page, Problem 7).
$\Delta_2 = \frac{1}{N}(\hat{Y} - Y)$, $N \times k$.	Stack the rows of step 2.
$\nabla_{W_2} L = \sum_n \frac{1}{N}\,\delta_2^{(n)} h^{(n)\top}$.	W_2 is used by every row, so the loss reaches it through all N copies and its gradient is the sum of their contributions; each is Problem 3 for example n, scaled by $\frac{1}{N}$.
For matrices A and B with rows $a^{(n)\top}$ and $b^{(n)\top}$, $\sum_n a^{(n)} b^{(n)\top} = A^\top B$.	$(A^\top B)_{ij} = \sum_n A_{ni} B_{nj} = \sum_n a_i^{(n)} b_j^{(n)}$, the (i, j) entry of the sum of outer products.
$\nabla_{W_2} L = \Delta_2^\top H$, with shapes $(k \times N)(N \times m) = k \times m$.	Step 5 with $A = \Delta_2$, whose rows are $\frac{1}{N}\delta_2^{(n)\top}$, and $B = H$. The shape is that of W_2.
$\nabla_{b_2} L = \sum_n \frac{1}{N}\,\delta_2^{(n)} = \Delta_2^\top \mathbf{1}$, with shapes $(k \times N)(N \times 1) = k \times 1$.	b_2 is added to every row, so its gradient is the sum over rows of Problem 3's δ_2; $\Delta_2^\top \mathbf{1}$ sums the rows of Δ_2.
$\nabla_H L = \Delta_2 W_2$, with shapes $(N \times k)(k \times m) = N \times m$.	Row n is Problem 4 for example n, transposed: $(W_2^\top \frac{1}{N}\delta_2^{(n)})^\top = \frac{1}{N}\delta_2^{(n)\top} W_2$.

$\Delta_1 = \sigma'(Z_1) \odot (\Delta_2 W_2)$, $N \times m$, with σ' applied to every entry of Z_1.	Problem 5 row by row: an elementwise product is taken entry by entry, so it does not care whether the entries are laid out as columns or rows. Row n is $\frac{1}{N}\delta_1^{(n)\top}$, Problem 5 for example n.
$\nabla_{W_1} L = \Delta_1^\top X$, $(m \times N)(N \times d) = m \times d$; $\nabla_{b_1} L = \Delta_1^\top \mathbf{1}$, $(m \times N)(N \times 1) = m \times 1$.	Steps 4 to 7 one layer down, with Δ_1 in place of Δ_2 and X in place of H.
$\nabla_X L = \Delta_1 W_1$, $(N \times m)(m \times d) = N \times d$.	Step 8 one layer down: row n is $\frac{1}{N}(W_1^\top \delta_1^{(n)})^\top$, Problem 7 for example n scaled by $\frac{1}{N}$.
$\Delta_2 = \frac{1}{N}(\hat{Y} - Y)$ $(N \times k)$; $\nabla_{W_2} L = \Delta_2^\top H$ $(k \times m)$; $\nabla_{b_2} L = \Delta_2^\top \mathbf{1}$; $\Delta_1 = \sigma'(Z_1) \odot (\Delta_2 W_2)$ $(N \times m)$; $\nabla_{W_1} L = \Delta_1^\top X$; $\nabla_{b_1} L = \Delta_1^\top \mathbf{1}$; $\nabla_X L = \Delta_1 W_1$ $(N \times d)$	Every gradient has the shape of its variable. The $\frac{1}{N}$ enters once, in Δ_2, and everything after it is linear in Δ_2, so it is carried through and never applied again.

Problem 9

> Replace σ by $\mathrm{ReLU}(z) = \max(z, 0)$. What changes in Problems 5–7? Write δ_1 and $\nabla_x L$.

$\mathrm{ReLU}'(t) = 1$ for $t > 0$ and 0 for $t < 0$; write $[z_1 > 0]$ for the vector whose entry i is 1 if $z_{1,i} > 0$ and 0 otherwise.	ReLU is t to the right of 0 and 0 to the left. At $t = 0$ the one-sided slopes are 0 and 1, so the derivative is undefined; it is taken as 0, and a pre-activation exactly at 0 is rare with real-valued data.
δ_2, $\nabla_{W_2} L$, $\nabla_{b_2} L$ and $\nabla_h L$ keep their formulas, with $h = \mathrm{ReLU}(z_1)$.	Problems 2 to 4 used h only as a vector feeding z_2; none of them differentiated the activation. The values change, the formulas do not.
$\delta_1 = \mathrm{ReLU}'(z_1) \odot (W_2^\top \delta_2)$.	Problem 5 used only that the activation is elementwise, so its derivative $\sigma'(z_1)$ is replaced by $\mathrm{ReLU}'(z_1)$, which is $[z_1 > 0]$ by step 1.

$\nabla_{W_1} L = \delta_1 x^\top$, $\nabla_{b_1} L = \delta_1$, $\nabla_x L = W_1^\top \delta_1$.	Problems 6 and 7 start from δ_1 and never look at the activation; the new δ_1 flows into them.
$\delta_1 = [z_1 > 0] \odot (W_2^\top \delta_2)$; $\nabla_x L = W_1^\top \delta_1$; **nothing else changes**	Only the elementwise factor in Problem 5 changes. A hidden unit with $z_{1,i} \le 0$ passes back nothing: entry i of δ_1 is 0, so row i of $\nabla_{W_1} L$ and entry i of $\nabla_{b_1} L$ are zero for this example. A unit that is negative on every input never updates, which is a dead ReLU.

Problem 10

> Add an L2 penalty: $L' = L + \frac{\lambda}{2}\|W_2\|_F^2$. Compute $\nabla_{W_2} L'$.

$\frac{\lambda}{2}\|W_2\|_F^2 = \frac{\lambda}{2}\sum_{i,j} W_{2,ij}^2$.	The squared Frobenius norm is the sum of the squares of the entries.
$\partial\big(\frac{\lambda}{2}\|W_2\|_F^2\big)/\partial W_{2,ij} = \lambda W_{2,ij}$, so the penalty's gradient is λW_2, $k \times m$.	Only one term of the sum contains $W_{2,ij}$, and $\frac{\lambda}{2}t^2$ has derivative λt. This is the matrix-calculus page's $\nabla_X \|X\|_F^2 = 2X$, times $\frac{\lambda}{2}$.
$\nabla_{W_2} L = \delta_2 h^\top$, $k \times m$.	Problem 3; the penalty does not involve z_2, so δ_2 is unchanged.
$\nabla_{W_2} L' = \delta_2 h^\top + \lambda W_2$	The gradient of a sum is the sum of the gradients, and both terms are $k \times m$. A gradient step becomes $W_2 \leftarrow (1 - \eta\lambda) W_2 - \eta\, \delta_2 h^\top$: the update shrinks the weights by $1 - \eta\lambda$ and adds the data term, which is why this is called weight decay. The other gradients are unchanged, because the penalty contains only W_2.

Where this goes wrong

1. Outer product the wrong way round

The weight gradient is built from two column vectors, δ_2 and h, and a product of two columns

needs one of them transposed.

$\nabla_{W_2} L$ is an outer product of δ_2 ($k \times 1$) and h ($m \times 1$)	Right so far: Problem 3, step 3 gives $\partial L / \partial W_{2,ij} = \delta_{2,i}\, h_j$.
"h goes into the layer and δ_2 comes back out of it, so h comes first."	The analogy that causes the mistake: writing the factors in the order the data flows, input then output, as if the gradient were read along the forward pass.
$\nabla_{W_2} L = h\, \delta_2^\top$ ✗	It is $m \times k$, the transpose of W_2's shape, so $W_2 - \eta\, h\, \delta_2^\top$ is undefined unless $m = k$, and wrong even then. The index form fixes which index is which: in $\delta_{2,i}\, h_j$ the row index i is the output unit and belongs to δ_2, and the column index j is the input unit and belongs to h, so the gradient is $\delta_2 h^\top$.

2. Passing the delta through the activation unchanged

Backprop is often summarized as "multiply the delta by the transposed weights and pass it down", and the summary leaves out half of each layer.

$\nabla_h L = W_2^\top \delta_2$	Right so far: Problem 4.
"The delta goes back through the transposed weights, so the hidden layer's delta is $W_2^\top \delta_2$."	The analogy that causes the mistake: the rule for a linear layer, applied to a layer that also has a nonlinearity, so that h and z_1 are treated as the same vector.
$\delta_1 = W_2^\top \delta_2$ ✗	That is $\nabla_h L$, not $\nabla_{z_1} L$. The chain rule through $h = \sigma(z_1)$ contributes $\sigma'(z_1)$ elementwise; dropping it is right only for an identity activation. For the sigmoid $\sigma' \le \frac{1}{4}$, so every nonzero entry comes out at least 4 times too large, and a saturated unit, whose true delta is near 0, keeps receiving a full-sized one.

3. Batched weight gradient with the data matrix on the wrong side

Linear regression's gradient, $X^\top(Xw - y)$, puts the transposed data matrix on the left, and the batched backward pass looks like the same kind of product.

$\nabla_{W_2} L = \sum_n \frac{1}{N} \delta_2^{(n)} h^{(n)\top}$, summed over the batch	Right so far: Problem 8, step 4.
"A sum over examples is a product with the transposed data matrix on the left, as in $X^\top(Xw - y)$."	The analogy that causes the mistake: in linear regression the weights are a column vector, so the data matrix goes on the left; here the weights are a $k \times m$ matrix and the order is fixed by W_2's shape.
$\nabla_{W_2} L = H^\top \Delta_2$ ✗	It is $(m \times N)(N \times k) = m \times k$. With rows as examples the gradient is $\Delta_2^\top H$, $k \times m$. The entries are the same numbers transposed, $(H^\top \Delta_2)^\top = \Delta_2^\top H$, which is why it "looks right" until the update fails to broadcast; when $m = k$ the update runs and silently applies the transpose.

4. Bias gradient that is still a matrix

For one example the bias gradient is the delta itself, $\nabla_{b_2} L = \delta_2$, and in the batch the delta is a matrix.

$\Delta_2 = \frac{1}{N}(\hat{Y} - Y)$, $N \times k$	Right so far: Problem 8, step 3.
"The bias gradient equals the delta, as in Problem 3."	The analogy that causes the mistake: a single-example identity carried to the batch without asking how the bias enters. $\mathbf{1} b_2^\top$ copies one bias into every row.

$\nabla_{b_2} L = \Delta_2$ ✗	It is $N \times k$, and b_2 has k entries. The bias is shared across the batch, so its gradient sums the rows: $\Delta_2^\top \mathbf{1}$. In array code $b_2 - \eta\,\Delta_2$ may broadcast without complaint and turn the bias into an $N \times k$ matrix, one bias per example.

5. Doubling the weight-decay gradient

The matrix-calculus page's $\nabla_X \|X\|_F^2 = 2X$ is short enough to memorize, and the 2 in it is easy to carry across without the $\frac{1}{2}$ it is meant to meet.

$L' = L + \frac{\lambda}{2}\|W_2\|_F^2$, and $\nabla_X \|X\|_F^2 = 2X$	Right so far: both are true.
"The gradient of a squared Frobenius norm is twice the matrix, so the penalty contributes $2\lambda W_2$."	The analogy that causes the mistake: $\nabla \|X\|_F^2 = 2X$ applied to $\lambda\|W_2\|_F^2$ instead of $\frac{\lambda}{2}\|W_2\|_F^2$, as if the prefactor were λ.
$\nabla_{W_2} \frac{\lambda}{2}\|W_2\|_F^2 = 2\lambda W_2$ ✗	The $\frac{1}{2}$ is there to cancel the 2: $\frac{\lambda}{2} \cdot 2W_2 = \lambda W_2$ (Problem 10). The slip survives because training still works: the model is simply regularized as if λ were twice what was set, so a λ tuned with this code must be doubled in any correct implementation.

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

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

This set is also Chapter 9 of *Deep Learning Math by Hand*, a 250-problem workbook with worked solutions: eulerpad.app/workbooks

eulerpad.app/practice/one-hidden-layer-backprop