The softmax Jacobian

Ten problems on softmax: its partial derivatives, the Jacobian diag(s) − ssᵀ and its rank, the s − y gradient of cross-entropy, log-softmax, soft labels, temperature and the batched version, with worked solutions and the mistakes that produce the wrong gradient.

BEFORE YOU START

Softmax turns a vector of scores into probabilities, and cross-entropy compares those probabilities with a target. The gradient of the pair with respect to the scores is the probabilities minus the target: short enough to memorize, and easy to misuse. These ten problems derive it three ways, work out the full Jacobian of the softmax on the way, and extend the result to batches, soft labels and temperature. The five mistakes at the end are the places the derivation goes wrong: a Jacobian treated as diagonal, a chain rule stopped one step early, a missing $1/N$, a formula used outside its hypothesis, and logits read as if they were unique.

- The conventions are those of the previous pages: vectors are columns, Jacobians are in numerator layout ($\partial y/\partial x$ is $m \times n$ for $y \in \mathbb{R}^m$, $x \in \mathbb{R}^n$), the gradient of a scalar is the transpose of its row derivative, a matrix gradient has the shape of the matrix, 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 logits are $z \in \mathbb{R}^n$, one per class, and $s = \mathrm{softmax}(z)$ has entries $s_i = e^{z_i} / \sum_k e^{z_k}$. Every $s_i > 0$ and $\sum_i s_i = 1$.
- $\mathbf{1}$ is the all-ones vector, so $\mathbf{1}^\top v = \sum_i v_i$ and $\mathbf{1}^\top s = s^\top \mathbf{1} = 1$.
- δ_{ij} is the Kronecker delta: 1 if $i = j$ and 0 otherwise, so δ_{ij} is entry (i, j) of I, the identity matrix.
- The cross-entropy loss is $L = -\sum_i y_i \log s_i$ (natural log). Unless a problem says otherwise, y is one-hot: one entry is 1, at the target class, and the rest are 0.
- Batched: $Z \in \mathbb{R}^{N \times n}$ has one example per row, $S = \mathrm{softmax}(Z)$ is taken row by row, Y holds the one-hot targets as rows, and the loss is the mean $L = \frac{1}{N} \sum_k L_k$ of the per-row losses.
- $\mathrm{diag}(v)$ and the elementwise product $\odot$ are as on the Jacobians page; y/s is elementwise division, $(y/s)_i = y_i/s_i$.
- These results are the backward pass of the last layer of every classifier trained with cross-entropy.

Builds on: Matrix calculus conventions, Jacobians and the chain rule

Problems

1. Show that $\mathrm{softmax}(z + c\mathbf{1}) = \mathrm{softmax}(z)$ for any scalar c. Why does an implementation subtract $\max_k z_k$ first? difficulty ·

2. Compute $\partial s_i / \partial z_j$, treating $i = j$ and $i \neq j$ separately, then write both cases as one formula. difficulty ··

3. Write $\partial s / \partial z$ as a matrix. Show it is symmetric and that $\big(\partial s / \partial z\big)\mathbf{1} = 0$. difficulty ··

4. What is the rank of $\partial s / \partial z$ when every $s_i > 0$? Describe its null space, and say what that means for how the logits determine the probabilities. difficulty ···

5. Let $L = -\sum_i y_i \log s_i$ with y one-hot. Compute $\nabla_z L$. difficulty ··

6. Let $\ell = \log \mathrm{softmax}(z)$, i.e. $\ell_i = z_i - \log \sum_k e^{z_k}$. Compute $\partial \ell / \partial z$. Use it to redo Problem 5 in one line. difficulty ··

7. Batched: $Z \in \mathbb{R}^{N \times n}$, $S = \mathrm{softmax}(Z)$ row-wise, Y one-hot rows, $L = -\frac{1}{N} \sum_{k,i} Y_{ki} \log S_{ki}$. Compute $\nabla_Z L$ and state its shape. difficulty ···

8. Repeat Problem 5 for soft labels y with $y_i \geq 0$, $\sum_i y_i = 1$. What changes if $\sum_i y_i \neq 1$? difficulty ··

9. With temperature, $s = \mathrm{softmax}(z / T)$. Compute $\partial s / \partial z$. What happens as $T \to \infty$ and as $T \to 0^+$? difficulty ···

10. Compute $\nabla_s L$ for $L = -\sum_i y_i \log s_i$, then apply the chain rule through the Jacobian of Problem 3 and show you recover the answer to Problem 5. difficulty ···

Answers

1. $\mathrm{softmax}(z + c\mathbf{1}) = \mathrm{softmax}(z)$; subtracting $\max_k z_k$ keeps every exponent ≤ 0, so nothing overflows and the largest term is exactly 1
2. $\partial s_i / \partial z_j = s_i(\delta_{ij} - s_j)$
3. $\partial s / \partial z = \mathrm{diag}(s) - ss^\top$; symmetric because both terms are; $(\mathrm{diag}(s) - ss^\top)\mathbf{1} = s - s(s^\top \mathbf{1}) = s - s = 0$
4. rank $= n - 1$; null space $\mathrm{span}\{\mathbf{1}\}$: logits are determined by the probabilities only up to a common shift, which is Problem 1 again
5. $\nabla_z L = s - y$
6. $\partial \ell / \partial z = I - \mathbf{1}s^\top$; then $\nabla_z L = -(\partial \ell / \partial z)^\top y = -(y - s\,\mathbf{1}^\top y) = s - y$
7. $\nabla_Z L = \frac{1}{N}(S - Y)$, $N \times n$
8. $\nabla_z L = (\sum_i y_i)\, s - y$, which is $s - y$ when the labels sum to 1
9. $\partial s / \partial z = \frac{1}{T}\big(\mathrm{diag}(s) - ss^\top\big)$ with $s = \mathrm{softmax}(z/T)$; as $T \to \infty$, $s \to \mathbf{1}/n$ and the Jacobian $\to 0$; as $T \to 0^+$, when the largest logit is unique the softmax approaches a hard argmax and the Jacobian $\to 0$, while at a tie for the largest logit entries of the Jacobian grow without bound
10. $\nabla_z L = J^\top \nabla_s L = s - y$, the same as Problem 5

Worked solutions

Problem 1

> Show that $\mathrm{softmax}(z + c\mathbf{1}) = \mathrm{softmax}(z)$ for any scalar c. Why does an implementation subtract $\max_k z_k$ first?

$\mathrm{softmax}(z + c\mathbf{1})_i = \dfrac{e^{z_i + c}}{\sum_k e^{z_k + c}}.$	Adding $c\mathbf{1}$ adds the same c to every logit: the one in the numerator and every one in the sum.
$= \dfrac{e^{c}\, e^{z_i}}{e^{c} \sum_k e^{z_k}}.$	$e^{a+b} = e^a e^b$, and e^c is the same factor in every term of the sum, so it comes out of the sum.
$= \dfrac{e^{z_i}}{\sum_k e^{z_k}} = s_i.$	$e^c > 0$, so it cancels. This holds for every i and every real c.
Take $c = -\max_k z_k$. Then every exponent $z_k - \max_k z_k$ is ≤ 0, and it is 0 for the largest logit.	Step 3 allows any c, so choose the one that makes the exponentials safe. Unshifted, e^{z_k} overflows to infinity once z_k exceeds about 709 in 64-bit floats (about 88 in 32-bit), and the ratio becomes ∞/∞, which is NaN.
$\mathrm{softmax}(z + c\mathbf{1}) = \mathrm{softmax}(z)$; **subtracting $\max_k z_k$ keeps every exponent ≤ 0, so nothing overflows and the largest term is exactly 1**	Every shifted exponential lies in $(0, 1]$, and because one of them is exactly 1 the denominator is at least 1: it cannot overflow, and it cannot underflow to 0 however negative the other logits are.

Problem 2

> Compute $\partial s_i / \partial z_j$, treating $i = j$ and $i \neq j$ separately, then write both cases as one formula.

Let $\Sigma = \sum_k e^{z_k}$, so $s_i = e^{z_i}/\Sigma$.	Naming the denominator separates the two places z appears: z_i in the numerator, and every z_k in Σ.
$\partial \Sigma / \partial z_j = e^{z_j}$.	Only the j-th term of Σ depends on z_j, and e^t is its own derivative.

$i = j$: $\dfrac{\partial s_i}{\partial z_i} = \dfrac{e^{z_i}\,\Sigma - e^{z_i}\,e^{z_i}}{\Sigma^2} = s_i - s_i^2.$	Both the numerator and the denominator contain z_i, so each contributes a term; then $e^{z_i}/\Sigma = s_i$ in both.
$i \neq j$: $\dfrac{\partial s_i}{\partial z_j} = -\dfrac{e^{z_i}\,e^{z_j}}{\Sigma^2} = -s_i s_j.$	The numerator e^{z_i} does not contain z_j, so only the denominator contributes, through step 2.
$s_i - s_i^2 = s_i(1 - s_i)$ and $-s_i s_j = s_i(0 - s_j)$.	Factoring out s_i leaves two expressions that differ only in the 1 or 0 in front of s_j, and that 1 or 0 is δ_{ij}.
$\partial s_i / \partial z_j = s_i(\delta_{ij} - s_j)$	δ_{ij} is 1 exactly in the case $i = j$. Sanity check: summing over i gives $s_j - s_j \sum_i s_i = 0$, as it must, because $\sum_i s_i = 1$ for every z, so no change in z_j can change the total.

Problem 3

> Write $\partial s / \partial z$ as a matrix. Show it is symmetric and that $\left(\partial s / \partial z\right)\mathbf{1} = 0$.

Entry (i, j) of $\partial s / \partial z$ is $\partial s_i / \partial z_j = s_i \delta_{ij} - s_i s_j$.	Numerator layout puts s_i on the rows and z_j on the columns; the value is Problem 2, multiplied out.
$s_i \delta_{ij}$ is entry (i, j) of $\operatorname{diag}(s)$, and $s_i s_j$ is entry (i, j) of the outer product $s s^\top$.	$\operatorname{diag}(s)$ has s_i at (i, i) and zeros elsewhere; $(a b^\top)_{ij} = a_i b_j$, here with $a = b = s$.
Let $J = \partial s / \partial z = \operatorname{diag}(s) - s s^\top$, $n \times n$.	Steps 1 and 2 agree entry by entry.
$J^\top = \operatorname{diag}(s)^\top - (s s^\top)^\top = \operatorname{diag}(s) - s s^\top = J$.	A diagonal matrix is its own transpose, and $(s s^\top)^\top = (s^\top)^\top s^\top = s s^\top$.
$J\mathbf{1} = \operatorname{diag}(s)\,\mathbf{1} - s\,(s^\top \mathbf{1}) = s - s$.	$\operatorname{diag}(s)\,\mathbf{1} = s \odot \mathbf{1} = s$; associativity lets $s^\top \mathbf{1}$ be computed first, and it is $\sum_i s_i = 1$.

$\partial s/\partial z = \operatorname{diag}(s) - ss^\top$; **symmetric because both terms are;** $(\operatorname{diag}(s) - ss^\top)\mathbf{1} = s - s(s^\top \mathbf{1}) = s - s = 0$	$J\mathbf{1} = 0$ is Problem 1 in derivative form: moving z along $\mathbf{1}$ does not change s, so the directional derivative in that direction is zero.

Problem 4

> What is the rank of $\partial s/\partial z$ when every $s_i > 0$? Describe its null space, and say what that means for how the logits determine the probabilities.

Let $J = \operatorname{diag}(s) - ss^\top$ (Problem 3), and for any $v \in \mathbb{R}^n$ let $\bar{v} = \sum_i s_i v_i$. Then $v^\top J v = \sum_i s_i v_i^2 - \bar{v}^2$.	$v^\top \operatorname{diag}(s)\, v = \sum_i s_i v_i^2$, and $v^\top s s^\top v = (s^\top v)^2 = \bar{v}^2$ by associativity.
$v^\top J v = \sum_i s_i (v_i - \bar{v})^2 = \operatorname{Var}_s(v)$, the variance of the entries of v when entry i is drawn with probability s_i.	Expanding the square gives $\sum_i s_i v_i^2 - 2\bar{v}^2 + \bar{v}^2$, because $\sum_i s_i v_i = \bar{v}$ and $\sum_i s_i = 1$.
$\operatorname{Var}_s(v) \geq 0$, with equality if and only if $v_i = \bar{v}$ for every i, that is, v is a multiple of $\mathbf{1}$.	Each term is ≥ 0, and because every $s_i > 0$ the sum is 0 only if every $v_i - \bar{v}$ is 0. Conversely, $v = a\mathbf{1}$ gives $\bar{v} = a$.
If $Jv = 0$ then $v^\top J v = 0$, so $v \in \operatorname{span}\{\mathbf{1}\}$; and $J\mathbf{1} = 0$. So the null space is exactly $\operatorname{span}\{\mathbf{1}\}$, and $\operatorname{rank} J = n - 1$.	Multiply by $v^\top$ and apply step 3; Problem 3 gives the reverse inclusion; then rank–nullity for an $n \times n$ matrix.
If $\operatorname{softmax}(z) = \operatorname{softmax}(z')$, then $z' = z + (\log \Sigma' - \log \Sigma)\,\mathbf{1}$, where $\Sigma = \sum_k e^{z_k}$ and $\Sigma' = \sum_k e^{z'_k}$.	Taking logs, $\log s_i = z_i - \log \Sigma = z'_i - \log \Sigma'$, and the correction is the same for every i. Step 4 is about first-order changes; this covers finite ones.

rank $= n - 1$; **null space** span$\{\mathbf{1}\}$: **logits are determined by the probabilities only up to a common shift, which is Problem 1 again**	Only the differences $z_i - z_j = \log(s_i/s_j)$ are fixed by s. Because J is symmetric, its column space is the orthogonal complement of its null space: every first-order change $J\,dz$ in s has entries summing to zero.

Problem 5

> Let $L = -\sum_i y_i \log s_i$ with y one-hot. Compute $\nabla_z L$.

Let c be the index with $y_c = 1$. Then $L = -\log s_c$.	Every other term of the sum has $y_i = 0$.
With $\Sigma = \sum_k e^{z_k}$, $\log s_c = z_c - \log \Sigma$, so $L = -z_c + \log \Sigma$.	The log of a quotient is a difference, and $\log e^{z_c} = z_c$. This form never divides by s_c.
$\partial L/\partial z_j = -\delta_{jc} + e^{z_j}/\Sigma = -\delta_{jc} + s_j$.	z_c contains z_j only when $j = c$; the derivative of $\log \Sigma$ is $(1/\Sigma)\,\partial\Sigma/\partial z_j$, and only the j-th term of Σ depends on z_j.
$\delta_{jc} = y_j$.	y is 1 at c and 0 elsewhere, which is the definition of δ_{jc} as a function of j.
$\nabla_z L = s - y$	Entry j of the gradient is $\partial L/\partial z_j = s_j - y_j$, and the result is $n \times 1$, the shape of z. Sanity check: its entries sum to $1 - 1 = 0$, as they must, because L depends on z only through s, which ignores a shift along $\mathbf{1}$.

Problem 6

> Let $\ell = \log \mathrm{softmax}(z)$, i.e. $\ell_i = z_i - \log \sum_k e^{z_k}$. Compute $\partial \ell/\partial z$. Use it to redo Problem 5 in one line.

With $\Sigma = \sum_k e^{z_k}$, $\partial \ell_i/\partial z_j = \delta_{ij} - e^{z_j}/\Sigma = \delta_{ij} - s_j$.	$\partial z_i/\partial z_j = \delta_{ij}$, and $\partial \log \Sigma/\partial z_j = s_j$ as in Problem 5, step 3; the second term is the same for every i.

δ_{ij} is entry (i, j) of I, and s_j is entry (i, j) of $\mathbf{1}s^\top$.	$(\mathbf{1}s^\top)_{ij} = 1 \cdot s_j$: every row of $\mathbf{1}s^\top$ is $s^\top$, matching a term that does not depend on i.
$\partial \ell / \partial z = I - \mathbf{1}s^\top$, $n \times n$.	Step 1 entry by entry. It is not symmetric: entry (i, j) subtracts s_j and entry (j, i) subtracts s_i. Multiplying by $\operatorname{diag}(s)$ on the left gives $\operatorname{diag}(s) - ss^\top$, Problem 3, because $s = e^\ell$ elementwise has Jacobian $\operatorname{diag}(s)$ with respect to ℓ.
$L = -y^\top \ell$, so $\nabla_z L = (\partial \ell / \partial z)^\top \nabla_\ell L = -(\partial \ell / \partial z)^\top y$.	$L = -\sum_i y_i \ell_i$ is linear in ℓ with gradient $-y$; then the chain rule for gradients.
$(I - \mathbf{1}s^\top)^\top y = y - s\,\mathbf{1}^\top y$.	$(\mathbf{1}s^\top)^\top = s\,\mathbf{1}^\top$, and associativity lets $\mathbf{1}^\top y$ be computed first.
$\partial \ell / \partial z = I - \mathbf{1}s^\top$; **then** $\nabla_z L = -(\partial \ell / \partial z)^\top y = -(y - s\,\mathbf{1}^\top y) = s - y$	$\mathbf{1}^\top y = \sum_i y_i = 1$ for a one-hot y. This is why cross-entropy is usually computed through log-softmax: it is evaluated from z directly, as $z - \log \Sigma\, \mathbf{1}$, so $\log s_i$ never has to be taken of an s_i that has underflowed to 0.

Problem 7

> Batched: $Z \in \mathbb{R}^{N \times n}$, $S = \operatorname{softmax}(Z)$ row-wise, Y one-hot rows, $L = -\frac{1}{N} \sum_{k,i} Y_{ki} \log S_{ki}$. Compute $\nabla_Z L$ and state its shape.

$L = \frac{1}{N} \sum_k L_k$ with $L_k = -\sum_i Y_{ki} \log S_{ki}$.	Group the double sum by rows; L_k is the loss of example k.
L_k depends on Z only through row k.	The softmax is taken row by row, so row k of S is computed from row k of Z alone, and L_k uses only row k of S and Y.
$\partial L / \partial Z_{kj} = \frac{1}{N} \partial L_k / \partial Z_{kj} = \frac{1}{N}(S_{kj} - Y_{kj})$.	By step 2, only the k-th term of the mean contains Z_{kj}; the value is Problem 5 applied to row k, whose softmax is row k of S.

$\nabla_Z L = \frac{1}{N}(S - Y)$, $N \times n$	Entry (k, j) of a matrix gradient is $\partial L / \partial Z_{kj}$, and step 3 fills every entry. The shape is that of Z.

Problem 8

> Repeat Problem 5 for soft labels y with $y_i \geq 0$, $\sum_i y_i = 1$. What changes if $\sum_i y_i \neq 1$?

With $\Sigma = \sum_k e^{z_k}$, $L = -\sum_i y_i (z_i - \log \Sigma) = -\sum_i y_i z_i + \left(\sum_i y_i\right) \log \Sigma$.	$\log s_i = z_i - \log \Sigma$, and $\log \Sigma$ does not depend on i, so it leaves the sum with coefficient $\sum_i y_i$. Nothing about y has been assumed yet.
$\partial L / \partial z_j = -y_j + \left(\sum_i y_i\right) s_j$.	Only the j-th term of $\sum_i y_i z_i$ contains z_j, and $\partial \log \Sigma / \partial z_j = s_j$ (Problem 5, step 3).
$\nabla_z L = \left(\sum_i y_i\right) s - y$, **which is $s - y$ when the labels sum to** 1	Step 2 for every j. One-hot labels are the case $\sum_i y_i = 1$, so Problem 5 is a special case. If the labels sum to $a \neq 1$, $a > 0$, the gradient is $as - y$, which is zero only at $s = y/a$: the loss pulls s toward the normalized labels, and $s - y$ is not its gradient.

Problem 9

> With temperature, $s = \text{softmax}(z/T)$. Compute $\partial s / \partial z$. What happens as $T \to \infty$ and as $T \to 0^+$?

Let $u = z/T$, so $s = \text{softmax}(u)$ and $\partial u / \partial z = \frac{1}{T} I$.	Each $u_i = z_i / T$ depends only on z_i; what remains is the softmax of Problem 3.
$\partial s / \partial z = \left(\text{diag}(s) - ss^\top\right) \frac{1}{T} I = \frac{1}{T}\left(\text{diag}(s) - ss^\top\right)$, with $s = \text{softmax}(u)$.	Chain rule, outer Jacobian (Problem 3 at u) on the left.
$T \to \infty$: $s \to \mathbf{1}/n$, and every entry of $\partial s / \partial z$ is at most $1/(4T)$ in size, so the Jacobian $\to 0$.	$u \to 0$ and the softmax is continuous. The bracket's entries are $s_i(1 - s_i) \leq \frac{1}{4}$ and $-s_i s_j$ with $s_i + s_j \leq 1$, so $s_i s_j \leq \frac{1}{4}$.

$T \to 0^+$, unique largest logit z_m, gap $g = z_m - \max_{k \neq m} z_k > 0$: for $i \neq m$, $s_i \leq e^{-g/T}$, and $1 - s_m \leq (n-1)e^{-g/T}$.	The denominator of s_i contains $e^{z_m/T}$, so $s_i \leq e^{(z_i - z_m)/T}$; then $1 - s_m = \sum_{k \neq m} s_k$.
So $s \to$ the one-hot vector at m, a hard argmax, and every entry of $\partial s / \partial z$ is at most $(n-1)e^{-g/T}/T$ in size, which $\to 0$.	Entry $s_i(\delta_{ij} - s_j)$ is bounded by s_i, by s_j, or (if $i = j = m$) by $1 - s_m$; step 4 bounds the one whose index is not m.
$T \to 0^+$, $r \geq 2$ logits tied for the largest: each tied $s_i \to 1/r$, so the diagonal entry $\frac{1}{T} s_i(1 - s_i) \to \infty$.	For tied i, $s_i = 1/\big(r + \sum_k e^{(z_k - z_i)/T}\big)$, summed over untied k, whose terms $\to 0$; so $s_i(1 - s_i) \to \frac{1}{r}(1 - \frac{1}{r}) > 0$.
$\partial s / \partial z = \frac{1}{T}\big(\operatorname{diag}(s) - ss^\top\big)$ **with** $s = \operatorname{softmax}(z/T)$; **as** $T \to \infty$, $s \to \mathbf{1}/n$ **and the Jacobian** $\to 0$; **as** $T \to 0^+$, **when the largest logit is unique the softmax approaches a hard argmax and the Jacobian** $\to 0$, **while at a tie for the largest logit entries of the Jacobian grow without bound**	Steps 2, 3, 5 and 6.

Problem 10

> Compute $\nabla_s L$ for $L = -\sum_i y_i \log s_i$, then apply the chain rule through the Jacobian of Problem 3 and show you recover the answer to Problem 5.

$\partial L / \partial s_i = -y_i / s_i$, so $\nabla_s L = -y/s$, $n \times 1$.	Treat L as a function of s: only the i-th term contains s_i, and the derivative of $\log t$ is $1/t$. Every $s_i > 0$, so the division is defined.
$\nabla_z L = J^\top \nabla_s L$ with $J = \partial s / \partial z = \operatorname{diag}(s) - ss^\top$.	The chain rule for gradients: the gradient with respect to the input is the transposed Jacobian times the gradient with respect to the output.

$J^\top = J.$	Problem 3.
$\text{diag}(s)\,(-y/s) = -y.$	A diagonal matrix acts as an elementwise product: $s_i \cdot (-y_i/s_i) = -y_i$.
$-ss^\top(-y/s) = s\left(s^\top(y/s)\right) = s\left(\sum_i y_i\right) = s\,(\mathbf{1}^\top y).$	Associativity lets the scalar $s^\top(y/s) = \sum_i s_i\, y_i/s_i$ be computed first, and each s_i cancels.
$\nabla_z L = (\text{diag}(s) - ss^\top)(-y/s) = -y + s\,(\mathbf{1}^\top y).$	Steps 4 and 5 are the two terms of $J(-y/s)$.
$\nabla_z L = J^\top \nabla_s L = s - y$, **the same as Problem 5**	$\mathbf{1}^\top y = 1$ for a one-hot y. The $1/s_i$ in $\nabla_s L$ cancels against the s_i in J, so the large entry $-1/s_c$ of $\nabla_s L$, at the target index c when the target's probability s_c is small, becomes a bounded $\nabla_z L$: every entry of $s - y$ lies in $[-1, 1]$.

Where this goes wrong

1. Using the i = j case for the whole Jacobian

The diagonal entry $s_i(1 - s_i)$ has exactly the form of the sigmoid's derivative $\sigma(1 - \sigma)$, and the Jacobians page showed that the sigmoid's Jacobian is diagonal.

$\partial s_i/\partial z_i = s_i(1 - s_i)$	Right so far: this is the $i = j$ case of Problem 2, and it is the correct diagonal.
"This is $\sigma(1 - \sigma)$ again, and the sigmoid's Jacobian is $\text{diag}(\sigma'(z))$, so the softmax's is diagonal too."	The analogy that causes the mistake: the sigmoid is elementwise, but every s_i contains every z_j through the shared denominator.

$\partial s/\partial z = \operatorname{diag}\big(s \odot (1 - s)\big)$ ✗	The off-diagonal $-s_i s_j$ terms are dropped. The true Jacobian $\operatorname{diag}(s) - ss^\top$ has columns that sum to zero, since raising one logit takes probability from the others; this one's column j sums to $s_j(1 - s_j) > 0$, as if raising z_j could raise s_j without lowering anything, pushing the total above 1.

2. Stopping at the gradient with respect to s

The loss is written in terms of s, so differentiating the formula as written feels like the whole job.

$L = -\sum_i y_i \log s_i$, so $\partial L/\partial s_i = -y_i/s_i$	Right so far: this is $\nabla_s L$, step 1 of Problem 10.
"L is a function of the probabilities, and the probabilities are the network's output, so this is the gradient to send back."	The shortcut that causes the mistake: the formula shows s, but the parameters reach L through z, and the softmax sits between them.
$\nabla_z L = -y/s$ ✗	That is $\nabla_s L$; one more chain-rule step through the softmax, multiplying by $(\operatorname{diag}(s) - ss^\top)^\top$, turns it into $s - y$ (Problem 10). The symptoms: $-y/s$ is zero at every non-target logit, so only the target is ever pushed, and its size $1/s_c$, with c the target index, grows without bound as the prediction gets worse, where every entry of $s - y$ stays in $[-1, 1]$.

3. Losing the 1/N in the batch

Each example contributes $s - y$ for its own row, and stacking those rows is the natural way to write the batch.

$L = \frac{1}{N}\sum_k L_k$, and the gradient of L_k with respect to row k of Z is row k of $S - Y$	Right so far: this is Problem 5 for one example, and the loss is the mean.

“Stack the per-example gradients into a matrix.”	The shortcut that causes the mistake: stacking gives the gradient of the sum $\sum_k L_k$, and the mean is a different function.
$\nabla_Z L = S - Y$ ✗	For the mean loss the correct answer is $\frac{1}{N}(S - Y)$ (Problem 7): the mean divides by N, and so does its gradient. With a sum loss $S - Y$ is right. The two differ by a factor of N, which you would then have to absorb into the learning rate, and which changes whenever the batch size does.

4. Applying s − y to labels that do not sum to one

Soft labels come from label smoothing, from a teacher model, or from annotators' votes, and a vector built by hand can easily fail to sum to one; here the entries sum to 0.9.

$y_i \geq 0$, $\sum_i y_i = 0.9$, $L = -\sum_i y_i \log s_i$	Right so far: the loss is well defined for any label vector, and nothing has been differentiated yet.
“The gradient of softmax plus cross-entropy is $s - y$.”	The shortcut that causes the mistake: remembering the result without the hypothesis it used, $\mathbf{1}^\top y = 1$ (Problems 5, 6 and 10 each use it in their last step).
$\nabla_z L = s - y$ with $\sum_i y_i = 0.9$ ✗	Problem 8 gives $(\sum_i y_i)\, s - y = 0.9\, s - y$, so the error is $0.1\, s$. It never goes away: the entries of $s - y$ sum to 0.1, but those of the true gradient sum to 0, because the loss does not change when every logit is shifted by the same amount.

5. Treating the logits as identifiable

Training drives $\text{softmax}(z)$ toward the target probabilities, and it is tempting to solve for the logits that get there.

At the optimum, $\text{softmax}(z) = p$ with every $p_i > 0$	Right so far: an equation in z that has solutions, for example $z_i = \log p_i$.
"n equations in n unknowns, so the solution is unique."	The shortcut that causes the mistake: counting equations. Both sides sum to 1, so only $n - 1$ of the equations are independent.
"The model has learned $z = \log p$." ✗	Any $z + c\mathbf{1}$ gives the same s (Problem 1), and the null space of the softmax Jacobian is $\text{span}\{\mathbf{1}\}$ (Problem 4): only differences of logits mean anything, $z_i - z_j = \log(p_i/p_j)$. A single raw logit, or a comparison of raw logits across examples or models, carries an arbitrary offset.

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 5 of *Deep Learning Math by Hand*, a 250-problem workbook with worked solutions: eulerpad.app/workbooks

eulerpad.app/practice/softmax-jacobian