Regression gradients: linear, logistic and softmax

Ten problems on the gradients of linear, logistic and softmax regression in matrix form: the 2/N of the mean squared error, the normal equations and ridge, the logistic gradient $X^T(\sigma(Xw) - y)/N$, its Hessian and convexity, and the softmax-regression gradient $X^T(S - Y)/N$ with shapes, with worked solutions and the mistakes that lose a factor or a transpose.

BEFORE YOU START

Linear, logistic and softmax regression are one-layer networks, and their gradients are the first place the rules of the previous two pages meet a whole dataset at once. These ten problems derive each gradient in matrix form with the $1/N$ of a mean loss kept in view, solve the linear case in closed form with and without a ridge penalty, show why the logistic loss is convex, and then extend to many classes. The five mistakes at the end are the ones that survive a quick check: a constant dropped from the gradient, a closed form copied from a loss without the $1/N$, an extra sigmoid derivative, a Hessian over examples instead of weights, and a weight gradient in the other library's layout.

- 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 w, $\nabla_w L = (\partial u/\partial w)^\top \nabla_u L$.
- The data: N examples with d features each. Rows are examples: $X \in \mathbb{R}^{N\times d}$ has row n equal to $x^{(n)\top}$, so $x^{(n)} \in \mathbb{R}^d$ is example n as a column, and X_{nj} is feature j of example n.
- Linear regression: targets $y \in \mathbb{R}^N$, weights $w \in \mathbb{R}^d$, predictions Xw, and the mean squared error $L(w) = \frac{1}{N}\|Xw - y\|^2 = \frac{1}{N}\sum_n (x^{(n)\top}w - y_n)^2$. Ridge adds $\lambda\|w\|^2$ with $\lambda > 0$.
- $\sigma(t) = 1/(1 + e^{-t})$ is the logistic sigmoid; on a vector it acts entry by entry, and $\odot$ is the elementwise product.
- Logistic regression: labels $y_n \in \{0, 1\}$, probabilities $p = \sigma(Xw) \in \mathbb{R}^N$ with entries $p_n = \sigma(x^{(n)\top}w)$, per-example loss $\ell_n = -\big[y_n \log p_n + (1 - y_n)\log(1 - p_n)\big]$ (binary cross-entropy, natural log), and $L = \frac{1}{N}\sum_n \ell_n$. $D = \operatorname{diag}(p \odot (1 - p))$ is the $N \times N$ diagonal matrix of the weights $p_n(1 - p_n)$.
- Softmax regression with k classes: $W \in \mathbb{R}^{d\times k}$ has one column w_i per class, $b \in \mathbb{R}^k$, the logits are $Z = XW + \mathbf{1}b^\top \in \mathbb{R}^{N\times k}$ with $\mathbf{1}$ the all-ones vector, S is the softmax of Z taken row by row, Y holds the one-hot targets as rows, and $L = -\frac{1}{N}\sum_{n,i} Y_{ni} \log S_{ni}$.
- $\nabla_w^2 L$ is the Hessian, the $d \times d$ matrix of second derivatives $\partial^2 L/\partial w_i\, \partial w_j$; it is the Jacobian of $\nabla_w L$.

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

Problems

1. Let $L(w) = \frac{1}{N}\|Xw - y\|^2$. Compute $\nabla_w L$ and check its shape. difficulty ·

2. Set the gradient of Problem 1 to zero. Derive the normal equations, say when their solution is unique, and show it is a minimum. Does the $1/N$ change the answer? difficulty ··

3. Ridge: $L_\lambda(w) = \frac{1}{N}\|Xw - y\|^2 + \lambda\|w\|^2$ with $\lambda > 0$. Compute $\nabla_w L_\lambda$, solve $\nabla_w L_\lambda = 0$, and show the solution exists even when the columns of X are dependent. difficulty ··

4. Show that $\sigma'(t) = \sigma(t)(1 - \sigma(t))$ and that $1 - \sigma(t) = \sigma(-t)$. difficulty ·

5. One example: $x \in \mathbb{R}^d$, $y \in \{0, 1\}$, $p = \sigma(x^\top w)$ and $\ell = -\big[y \log p + (1 - y)\log(1 - p)\big]$. Compute $\nabla_w \ell$. difficulty ··

6. Batch: $L = \frac{1}{N}\sum_n \ell_n$ with ℓ_n as in Problem 5 for example n. Write $\nabla_w L$ in matrix form. difficulty ··

7. Compute the Hessian $\nabla_w^2 L$ of the logistic loss of Problem 6. Show that it is positive semidefinite, so that L is convex. difficulty ···

8. Softmax regression: $Z = XW + \mathbf{1}b^\top$, S the row-wise softmax of Z, Y one-hot rows, $L = -\frac{1}{N}\sum_{n,i} Y_{ni} \log S_{ni}$. Compute $\nabla_Z L$, $\nabla_W L$ and $\nabla_b L$, with shapes. difficulty ···

9. Take $k = 2$ and no bias, so $Z = XW$ with columns w_1, w_2. Show that $S_{n1} = \sigma\big(x^{(n)\top}(w_1 - w_2)\big)$ and that the loss is the logistic loss of Problem 6 with $w = w_1 - w_2$ and $y_n = Y_{n1}$. What does that say about W? difficulty ···

10. Newton's method updates $w_{\text{new}} = w - (\nabla_w^2 L)^{-1}\nabla_w L$. Write the step for logistic regression and show that the $1/N$ cancels. difficulty ···

Answers

1. $\nabla_w L = \frac{2}{N} X^\top (Xw - y)$
2. $X^\top X\, w = X^\top y$; when the columns of X are independent, $w^* = (X^\top X)^{-1} X^\top y$, the unique minimizer, because the Hessian $\frac{2}{N} X^\top X$ is positive definite
3. $\nabla_w L_\lambda = \frac{2}{N} X^\top (Xw - y) + 2\lambda w$; $w^* = (X^\top X + N\lambda I)^{-1} X^\top y$
4. $\sigma'(t) = \sigma(t)(1 - \sigma(t))$; $1 - \sigma(t) = \sigma(-t)$
5. $\nabla_w \ell = (\sigma(x^\top w) - y)\, x$
6. $\nabla_w L = \frac{1}{N} X^\top (\sigma(Xw) - y)$
7. $\nabla_w^2 L = \frac{1}{N} X^\top D X$ with $D = \operatorname{diag}(p \odot (1 - p))$, $p = \sigma(Xw)$; it is positive semidefinite, so L is convex
8. $\nabla_W L = \frac{1}{N} X^\top (S - Y)$ $(d \times k)$; $\nabla_b L = \frac{1}{N} (S - Y)^\top \mathbf{1}$ (k)
9. $S_{n1} = \sigma\big(x^{(n)\top}(w_1 - w_2)\big)$, so two-class softmax regression is logistic regression with $w = w_1 - w_2$
10. $w_{\text{new}} = w - (X^\top D X)^{-1} X^\top (p - y)$

Worked solutions

Problem 1

> Let $L(w) = \frac{1}{N}\|Xw - y\|^2$. Compute $\nabla_w L$ and check its shape.

Let $r = Xw - y \in \mathbb{R}^N$, the residuals, so $L = \frac{1}{N} r^\top r = \frac{1}{N}\sum_n r_n^2$ with $r_n = x^{(n)\top} w - y_n$.	Naming the residual separates the square from the affine map inside it, so each can be differentiated on its own.
$\nabla_r L = \frac{2}{N} r$, $N \times 1$.	Only the n-th term of the sum contains r_n, and r_n^2 differentiates to $2r_n$.
$\partial r / \partial w = X$, $N \times d$.	r is X times w minus a constant, and the Jacobian of Ax is A (the matrix-calculus page, Problem 2).
$\nabla_w L = (\partial r / \partial w)^\top \nabla_r L = X^\top \frac{2}{N} r$.	L depends on w only through r, so the chain rule for gradients applies.
$\nabla_w L = \frac{2}{N} X^\top (Xw - y)$	$(d \times N)(N \times 1) = d \times 1$, the shape of w. Entry j is $\frac{2}{N}\sum_n X_{nj} r_n$: every example's residual, weighted by its feature j, averaged over the data.

Problem 2

> Set the gradient of Problem 1 to zero. Derive the normal equations, say when their solution is unique, and show it is a minimum. Does the $1/N$ change the answer?

$\frac{2}{N} X^\top (Xw - y) = 0 \iff X^\top X\, w = X^\top y$.	Multiply both sides by $N/2 \neq 0$. Scaling a loss by a positive constant does not move its minimizer, so the $1/N$ changes the size of gradient steps, never where they lead.
$\nabla_w^2 L = \frac{2}{N} X^\top X$, $d \times d$.	The gradient $\frac{2}{N} X^\top X w - \frac{2}{N} X^\top y$ is affine in w, so its Jacobian is the matrix that multiplies w.

$v^\top X^\top X v = (Xv)^\top (Xv) = \|Xv\|^2 \geq 0$ for every $v \in \mathbb{R}^d$.	Associativity groups the product into a vector with itself, and a squared norm cannot be negative.
If the columns of X are independent, $Xv = 0$ only for $v = 0$, so $v^\top X^\top X v > 0$ for $v \neq 0$ and $X^\top X$ is invertible.	Xv is the combination of the columns of X with coefficients v; independence means only the zero combination vanishes. A positive definite matrix has no null space.
$X^\top X\, w = X^\top y$; **when the columns of X are independent,** $w^* = (X^\top X)^{-1} X^\top y$**, the unique minimizer, because the Hessian $\frac{2}{N} X^\top X$ is positive definite**	A function whose Hessian is positive definite everywhere is strictly convex, so its one stationary point is the global minimum. With dependent columns (fewer examples than features, or a repeated feature) $X^\top X$ is singular and the minimizers form a whole affine set; Problem 3 removes that.

Problem 3

> Ridge: $L_\lambda(w) = \frac{1}{N}\|Xw - y\|^2 + \lambda\|w\|^2$ with $\lambda > 0$. Compute $\nabla_w L_\lambda$, solve $\nabla_w L_\lambda = 0$, and show the solution exists even when the columns of X are dependent.

$\nabla_w \big(\lambda \|w\|^2\big) = 2\lambda w.$	$\lambda\|w\|^2 = \lambda \sum_j w_j^2$, and only the j-th term contains w_j.
$\nabla_w L_\lambda = \frac{2}{N} X^\top (Xw - y) + 2\lambda w.$	The gradient of a sum is the sum of the gradients; the first term is Problem 1.
Setting it to zero and multiplying by $N/2$: $X^\top X\, w - X^\top y + N\lambda\, w = 0$, so $(X^\top X + N\lambda I)\, w = X^\top y.$	The data term carries the $1/N$ of the mean and the penalty does not, so clearing the $1/N$ leaves an N on the penalty.
$v^\top (X^\top X + N\lambda I)\, v = \|Xv\|^2 + N\lambda\|v\|^2 > 0$ for every $v \neq 0$.	The first term is ≥ 0 (Problem 2, step 3) and the second is > 0 because $N\lambda > 0$, whatever X is; so the matrix is positive definite and invertible.

$\nabla_w L_\lambda = \frac{2}{N} X^\top (Xw - y) + 2\lambda w$; $w^* = (X^\top X + N\lambda I)^{-1} X^\top y$	Step 4 holds for every X, including $N < d$, so the solution always exists and is unique. For the summed loss $\|Xw - y\|^2 + \lambda' \|w\|^2$ the same steps give the familiar $(X^\top X + \lambda' I)^{-1} X^\top y$; the two agree when $\lambda' = N\lambda$.

Problem 4

> Show that $\sigma'(t) = \sigma(t)(1 - \sigma(t))$ and that $1 - \sigma(t) = \sigma(-t)$.

$\sigma(t) = (1 + e^{-t})^{-1}$, so $\sigma'(t) = \dfrac{e^{-t}}{(1 + e^{-t})^2}$.	Chain rule: u^{-1} differentiates to $-u^{-2}$, and $1 + e^{-t}$ to $-e^{-t}$; the two minus signs cancel.
$1 - \sigma(t) = \dfrac{1 + e^{-t} - 1}{1 + e^{-t}} = \dfrac{e^{-t}}{1 + e^{-t}}$.	Put 1 over the common denominator.
$\sigma'(t) = \dfrac{1}{1 + e^{-t}} \cdot \dfrac{e^{-t}}{1 + e^{-t}} = \sigma(t)(1 - \sigma(t))$.	Split the square in step 1's denominator; the second factor is step 2.
$\dfrac{e^{-t}}{1 + e^{-t}} = \dfrac{1}{e^{t} + 1} = \sigma(-t)$.	Multiply the numerator and denominator by e^t; the result is the definition of σ at $-t$.
$\sigma'(t) = \sigma(t)(1 - \sigma(t))$; $1 - \sigma(t) = \sigma(-t)$	Both factors lie in $(0, 1)$ and sum to 1, so $\sigma'(t) \le \frac{1}{4}$, with equality at $t = 0$. The second identity says the probability of class 0 is the sigmoid of the negated logit, so the two classes are treated symmetrically.

Problem 5

> One example: $x \in \mathbb{R}^d$, $y \in \{0, 1\}$, $p = \sigma(x^\top w)$ and $\ell = -\big[y \log p + (1 - y) \log(1 - p)\big]$. Compute $\nabla_w \ell$.

Let $t = x^\top w$, the logit, so $p = \sigma(t)$.	ℓ depends on w only through the scalar t, so the chain rule has one link per variable.

$\partial \ell / \partial p = -\dfrac{y}{p} + \dfrac{1-y}{1-p} = \dfrac{p-y}{p(1-p)}.$	$\log u$ differentiates to $1/u$, and $\log(1-p)$ to $-1/(1-p)$; over the common denominator the numerator is $-y(1-p) + (1-y)p = p - y$.
$\partial p / \partial t = p(1-p).$	Problem 4 at t.
$\partial \ell / \partial t = p - y.$	Chain rule, steps 2 and 3. The $p(1-p)$ cancels, which is allowed because $0 < p < 1$ for every finite t.
$\nabla_w t = x.$	$t = x^\top w$ is linear in w, and $\nabla_w (a^\top w) = a$ (the matrix-calculus page, Problem 3).
$\nabla_w \ell = (\sigma(x^\top w) - y)\, x$	$\nabla_w \ell = (\partial \ell / \partial t)\, \nabla_w t$, because t is a scalar. The shape is $d \times 1$. It is the prediction error times the input, the same form as linear regression's residual times input.

Problem 6

> Batch: $L = \frac{1}{N} \sum_n \ell_n$ with ℓ_n as in Problem 5 for example n. Write $\nabla_w L$ in matrix form.

$\nabla_w L = \frac{1}{N} \sum_n (p_n - y_n)\, x^{(n)}$ with $p_n = \sigma(x^{(n)\top} w)$.	The gradient of a mean is the mean of the gradients; each term is Problem 5 for example n.
For any $c \in \mathbb{R}^N$, $\sum_n c_n\, x^{(n)} = X^\top c$.	The columns of $X^\top$ are the $x^{(n)}$, and a matrix times a vector combines its columns with the vector's entries as weights.
$p = \sigma(Xw)$ has entries p_n.	Entry n of Xw is row n of X times w, that is $x^{(n)\top} w$, and σ acts entry by entry.

$\nabla_w L = \frac{1}{N} X^\top (\sigma(Xw) - y)$	Step 2 with $c = p - y$. The shape is $(d \times N)(N \times 1) = d \times 1$. It is Problem 1's gradient with the prediction Xw replaced by $\sigma(Xw)$ and without the 2, which came from the square. Because σ is nonlinear, setting it to zero has no closed-form solution, so the weights are found iteratively (Problem 10).

Problem 7

> Compute the Hessian $\nabla_w^2 L$ of the logistic loss of Problem 6. Show that it is positive semidefinite, so that L is convex.

$\nabla_w L = \frac{1}{N} \sum_n (p_n - y_n)\, x^{(n)}$.	Problem 6, step 1.
$\nabla_w p_n = p_n(1 - p_n)\, x^{(n)}$.	$p_n = \sigma(t_n)$ with $t_n = x^{(n)\top} w$: Problem 4 for $\partial p_n / \partial t_n$ and Problem 5, step 5 for $\nabla_w t_n$.
$\frac{\partial}{\partial w}\left[(p_n - y_n)\, x^{(n)}\right] = x^{(n)}\, (\nabla_w p_n)^\top = p_n(1 - p_n)\, x^{(n)} x^{(n)\top}$, $d \times d$.	$x^{(n)}$ and y_n do not depend on w; for a scalar $c(w)$ times a constant vector a, entry (i, j) of the Jacobian is $a_i\, \partial c / \partial w_j$, which is $a (\nabla_w c)^\top$.
$\sum_n D_{nn}\, x^{(n)} x^{(n)\top} = X^\top D X$, with $D_{nn} = p_n(1 - p_n)$.	$(X^\top D X)_{ij} = \sum_n X_{ni}\, D_{nn}\, X_{nj}$, which is entry (i, j) of the weighted sum of outer products, because $X_{ni} = x_i^{(n)}$.
$v^\top X^\top D X\, v = \sum_n p_n(1 - p_n)\, (x^{(n)\top} v)^2 \geq 0$ for every $v \in \mathbb{R}^d$.	Entry n of Xv is $x^{(n)\top} v$, D weights its square by $p_n(1 - p_n)$, and every weight is positive because $0 < p_n < 1$.

$\nabla_w^2 L = \frac{1}{N} X^\top D X$ **with** $D = \operatorname{diag}(p \odot (1-p))$, $p = \sigma(Xw)$; **it is positive semidefinite, so** L **is convex**	A function whose Hessian is positive semidefinite everywhere is convex, so every stationary point is a global minimum and gradient descent cannot be trapped in a worse one. With independent columns the inequality in step 5 is strict for $v \neq 0$ and L is strictly convex. Unlike Problem 2's Hessian, this one changes with w through D.

Problem 8

> Softmax regression: $Z = XW + \mathbf{1}b^\top$, S the row-wise softmax of Z, Y one-hot rows, $L = -\frac{1}{N}\sum_{n,i} Y_{ni} \log S_{ni}$. Compute $\nabla_Z L$, $\nabla_W L$ and $\nabla_b L$, with shapes.

For one example with logits $z \in \mathbb{R}^k$, softmax s and one-hot target y at class c: $\ell = -z_c + \log \sum_i e^{z_i}$, so $\nabla_z \ell = s - y$.	$\log s_c = z_c - \log \sum_i e^{z_i}$; the log-sum-exp differentiates to $e^{z_j} / \sum_i e^{z_i} = s_j$, and z_c to 1 exactly at $j = c$, which is y_j. The next page derives this result three ways.
$\nabla_Z L = \frac{1}{N}(S - Y)$, $N \times k$.	Row n of Z feeds only ℓ_n, through its own row of the softmax, and the mean puts $\frac{1}{N}$ on each ℓ_n; row n is step 1 for example n.
$Z_{ni} = \sum_j X_{nj} W_{ji} + b_i$, so $\dfrac{\partial L}{\partial W_{ji}} = \sum_n \dfrac{\partial L}{\partial Z_{ni}} X_{nj}$.	W_{ji} sits in column i of Z, in every row n, with coefficient X_{nj}: the loss reaches it through all N examples, so their contributions add.
$\sum_n X_{nj} (\nabla_Z L)_{ni} = (X^\top \nabla_Z L)_{ji}$.	$(X^\top)_{jn} = X_{nj}$, and the sum over n is the definition of the matrix product.
$\partial L / \partial b_i = \sum_n (\nabla_Z L)_{ni}$, so $\nabla_b L = (\nabla_Z L)^\top \mathbf{1}$.	$\mathbf{1}b^\top$ adds b_i to every row of column i with coefficient 1, and $(\nabla_Z L)^\top \mathbf{1}$ sums each column of $\nabla_Z L$.

$\nabla_W L = \frac{1}{N} X^\top (S - Y)$ **(**$d \times k$**)**; $\nabla_b L = \frac{1}{N} (S - Y)^\top \mathbf{1}$ **(**k**)**	$(d \times N)(N \times k) = d \times k$ and $(k \times N)(N \times 1) = k \times 1$, the shapes of W and b. Column i of $\nabla_W L$ is $\frac{1}{N} X^\top (S_{:,i} - Y_{:,i})$, Problem 6's form once per class, with the softmax column in place of $\sigma(Xw)$.

Problem 9

> Take $k = 2$ and no bias, so $Z = XW$ with columns w_1, w_2. Show that $S_{n1} = \sigma\big(x^{(n)\top}(w_1 - w_2)\big)$ and that the loss is the logistic loss of Problem 6 with $w = w_1 - w_2$ and $y_n = Y_{n1}$. What does that say about W?

For $z \in \mathbb{R}^2$, $\operatorname{softmax}(z)_1 = \dfrac{e^{z_1}}{e^{z_1} + e^{z_2}} = \dfrac{1}{1 + e^{-(z_1 - z_2)}} = \sigma(z_1 - z_2)$.	Divide the numerator and the denominator by e^{z_1}.
$\operatorname{softmax}(z)_2 = 1 - \sigma(z_1 - z_2)$.	The two entries of a softmax sum to 1.
Row n of XW is $\big(x^{(n)\top} w_1,\ x^{(n)\top} w_2\big)$, so $z_1 - z_2 = x^{(n)\top}(w_1 - w_2)$.	Column i of XW is Xw_i, and $x^{(n)\top}$ distributes over the difference.
With $y_n = Y_{n1}$ and $Y_{n2} = 1 - y_n$, $-\sum_i Y_{ni} \log S_{ni} = -\big[y_n \log p_n + (1 - y_n) \log(1 - p_n)\big]$ with $p_n = \sigma(x^{(n)\top} w)$, $w = w_1 - w_2$.	A one-hot row with two classes is $(y_n, 1 - y_n)$; steps 1 to 3 give $S_{n1} = p_n$ and $S_{n2} = 1 - p_n$.
$S_{n1} = \sigma\big(x^{(n)\top}(w_1 - w_2)\big)$, **so two-class softmax regression is logistic regression with** $w = w_1 - w_2$	Only the difference $w_1 - w_2$ is determined by the data: adding the same vector to both columns leaves every probability, and so the loss, unchanged. Two-class softmax regression therefore has a whole set of minimizers where logistic regression may have one; a ridge penalty on W picks the one with $w_2 = -w_1$. The next page meets the same shift invariance in the logits.

Problem 10

> Newton's method updates $w_{\text{new}} = w - (\nabla_w^2 L)^{-1} \nabla_w L$. Write the step for logistic regression and show that the $1/N$ cancels.

$\nabla_w L = \frac{1}{N} X^\top (p - y)$ and $\nabla_w^2 L = \frac{1}{N} X^\top D X$, with $p = \sigma(Xw)$.	Problems 6 and 7.
$(\nabla_w^2 L)^{-1} \nabla_w L = N\,(X^\top D X)^{-1}\, \frac{1}{N} X^\top (p - y)$.	$(cA)^{-1} = c^{-1} A^{-1}$ for a scalar $c \neq 0$. The inverse exists when the columns of X are independent, because then $X^\top D X$ is positive definite (Problem 7).
$w_{\text{new}} = w - (X^\top D X)^{-1} X^\top (p - y)$	The N and the $\frac{1}{N}$ cancel, so Newton's step is the same for the mean loss and the summed loss, while a gradient step changes by a factor of N between them. Each step solves a weighted least-squares problem with weights D, recomputed from the new p: this is why the method is called iteratively reweighted least squares.

Where this goes wrong

1. Dropping the 2/N from the mean-squared-error gradient

Many texts write the least-squares loss as $\frac{1}{2}\|Xw - y\|^2$, whose gradient carries no constant, and the clean form is the one that gets remembered.

$L(w) = \frac{1}{N}\|Xw - y\|^2$	Right so far: the mean squared error as defined on this page.
"The gradient of a least-squares loss is $X^\top (Xw - y)$."	The shortcut that causes the mistake: the gradient of $\frac{1}{2}\|Xw - y\|^2$, recalled without the loss it belongs to.

$\nabla_w L = X^\top (Xw - y)$ ✗	The gradient of the mean squared error is $\frac{2}{N} X^\top (Xw - y)$ (Problem 1). Setting either to zero gives the same normal equations, which hides the slip; but a gradient step with this formula is $N/2$ times too large, so the learning rate has to absorb the factor, and it changes whenever the batch size does.

2. Ridge closed form without the N

The ridge formula $(X^\top X + \lambda I)^{-1} X^\top y$ is printed in every textbook, and a loss written as a mean looks like the same loss.

$L_\lambda(w) = \frac{1}{N}\|Xw - y\|^2 + \lambda \|w\|^2$	Right so far: a mean data term plus a penalty, the form most libraries document.
"The ridge solution is $(X^\top X + \lambda I)^{-1} X^\top y$."	The shortcut that causes the mistake: the textbook formula belongs to the summed loss $\|Xw - y\|^2 + \lambda \|w\|^2$, which has no $1/N$ on the data term.
$w^* = (X^\top X + \lambda I)^{-1} X^\top y$ ✗	For the mean loss the minimizer is $(X^\top X + N\lambda I)^{-1} X^\top y$ (Problem 3): clearing the $1/N$ multiplies the penalty by N. With the formula above the penalty is N times weaker than the one written down, and the effective regularization shrinks as the dataset grows.

3. Logistic gradient with an extra p(1 − p)

A chain rule through a sigmoid output always produces a σ' factor, and with a squared-error loss that factor really does stay in the gradient.

$\partial p_n / \partial t_n = p_n (1 - p_n)$	Right so far: Problem 4, at the logit $t_n = x^{(n)\top} w$.

"The derivative of the loss with respect to the prediction is the error $p_n - y_n$; multiply by σ'."	The analogy that causes the mistake: $p - y$ is the derivative of the squared error $\frac{1}{2}(p-y)^2$. For cross-entropy the derivative is $(p-y)/\big(p(1-p)\big)$ (Problem 5, step 2), and its denominator cancels the σ'.
$\nabla_w L = \frac{1}{N} X^\top \big[(p-y) \odot p \odot (1-p)\big]$ ✗	This is the gradient of the mean of $\frac{1}{2}(\sigma(x^{(n)\top} w) - y_n)^2$, a different loss and not a convex one. The correct gradient is $\frac{1}{N} X^\top (p-y)$ (Problem 6). The symptom: an example that is confidently wrong, with p_n near 1 when $y_n = 0$, gets a factor $p_n(1-p_n)$ near 0, so learning stalls exactly where the error is largest.

4. Hessian built as X D X^T

Each example contributes $p_n(1-p_n)\, x^{(n)} x^{(n)\top}$, and an outer product $xx^\top$ looks as if it should become $XX^\top$ for the whole batch.

$\nabla_w^2 L = \frac{1}{N} \sum_n p_n(1-p_n)\, x^{(n)} x^{(n)\top}$	Right so far: Problem 7, step 3, summed over the examples.
"Stack the $x^{(n)}$ into X, and the sum of outer products becomes $XDX^\top$."	The analogy that causes the mistake: treating $x^{(n)}$ as a column of X, when with rows as examples $x^{(n)\top}$ is a row of X.
$\nabla_w^2 L = \frac{1}{N} X D X^\top$ ✗	It is $N \times N$, a matrix over pairs of examples, where the Hessian is $d \times d$, over pairs of weights. A sum of outer products of the rows of X is $X^\top D X$ (Problem 7, step 4). When $N = d$ the shapes agree, and Newton's step (Problem 10) runs with the wrong matrix.

5. Softmax-regression gradient in the k × d layout

PyTorch's `nn.Linear` stores its weight as output features by input features and computes $XW^\top + \mathbf{1}b^\top$. In that layout the weight gradient is $(\nabla_Z L)^\top X$, and the formula travels with the reader to code where W is $d \times k$.

$\nabla_Z L = \frac{1}{N}(S - Y)$, $N \times k$	Right so far: Problem 8, step 2.
"The weight gradient is the upstream gradient, transposed, times the input."	The habit that causes the mistake: the rule for $Z = XW^\top + \mathbf{1}b^\top$, where W is $k \times d$, applied to $Z = XW + \mathbf{1}b^\top$.
$\nabla_W L = \frac{1}{N}(S - Y)^\top X$ ✗	It is $k \times d$, the transpose of W's $d \times k$. In $Z = XW$, W_{ji} multiplies X_{nj} into Z_{ni}, so the gradient is $\frac{1}{N}X^\top(S - Y)$ (Problem 8). The entries are the right numbers transposed; when $d = k$ the update runs and silently applies the transpose.

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

eulerpad.app/practice/regression-gradients