

CS109 Lecture 20 Worksheet: ANSWER KEY

Logistic Regression

Summer 2026

Lecture 20 Worksheet: Answer Key

Instructor / TA copy. Problem-set solutions are hidden in the standard build; run `./convert-to-pdf.sh 20 solutions` for the full instructor/TA edition.

Problem 1 (Review): MLE Warm-Up

- (a) With $n = 5$ and $\sum x_i = 15$:

$$L(\lambda) = \prod_{i=1}^5 \frac{e^{-\lambda} \lambda^{x_i}}{x_i!}, \quad LL(\lambda) = -n\lambda + (\sum x_i) \ln \lambda - \underbrace{\sum \ln(x_i!)}_{\text{no } \lambda}.$$

- (b)

$$\frac{dLL}{d\lambda} = -n + \frac{\sum x_i}{\lambda} = 0 \implies \hat{\lambda} = \frac{\sum x_i}{n} = \frac{15}{5} = 3.$$

- (c) Those terms are constants with respect to λ , so their derivative is 0; they shift LL vertically but do not move its argmax.

Problem 2: Sigmoids and a Single Prediction

- (a) $z = -1 + 2(1) + 0.5(2) = 2$, so $\hat{y} = \sigma(2) = 0.8808$.
- (b) Predict $Y = 1$. Since σ is strictly increasing and $\sigma(0) = 0.5$, we have $\sigma(z) > 0.5 \iff z > 0$. So the 0.5 threshold on the probability is exactly the 0 threshold on the weighted sum.
- (c) Undecided means $\sigma(z) = 0.5$, i.e. $z = 0$:

$$-1 + 2x_1 + 0.5x_2 = 0 \implies x_2 = 2 - 4x_1.$$

This is a straight line in feature space — the **decision boundary**. (In m dimensions it is a hyperplane; this is why logistic regression is a *linear* classifier.)

- (d) $\theta_3 = 0$ means the feature has no influence on the prediction at all — x_3 drops out of the weighted sum. This is what we expect when the feature carries no information about Y . A large negative weight means the feature is strong *evidence against* $Y = 1$: increasing x_3 pushes z down and the predicted probability toward 0.

Problem 3: One Step of Gradient Ascent, by Hand

- (a) $\theta^T x = 0$ for both points, so $\hat{y}^{(1)} = \hat{y}^{(2)} = \sigma(0) = 0.5$. The model is maximally uncertain.

$$LL = \ln(0.5) + \ln(1 - 0.5) = 2 \ln 0.5 \approx -1.386.$$

- (b) Datapoint 1 ($y = 1$, $\hat{y} = 0.5$, so $y - \hat{y} = 0.5$) contributes $[0.5, 0.5, 0]$. Datapoint 2 ($y = 0$, $\hat{y} = 0.5$, so $y - \hat{y} = -0.5$) contributes $[-0.5, 0, -0.5]$. Summing:

$$\nabla LL = [0, 0.5, -0.5].$$

- (c) $\theta \leftarrow \theta + 1 \cdot \nabla LL$, so $\theta = [0, 0.5, -0.5]$.

- (d) Datapoint 1: $z = 0 + 0.5(1) - 0.5(0) = 0.5$, $\hat{y}^{(1)} = \sigma(0.5) = 0.6225$. Datapoint 2: $z = 0 + 0 - 0.5(1) = -0.5$, $\hat{y}^{(2)} = \sigma(-0.5) = 0.3775$, and since $y^{(2)} = 0$ the probability of the observed label is $1 - 0.3775 = 0.6225$.

$$LL = \ln(0.6225) + \ln(0.6225) \approx -0.948.$$

It rose from -1.386 to -0.948 , a gain of about 0.438 . Both labels became more likely — exactly what gradient *ascent* is supposed to do.

- (e) The two datapoints pull θ_0 in opposite directions with equal force: one has $y - \hat{y} = +0.5$ and the other -0.5 , and $x_0 = 1$ for both, so they cancel. The dataset is perfectly balanced (one label 1, one label 0), so there is no reason to shift the intercept.

Problem 4: Where the Gradient Comes From

- (a) For $y = 1$ the expression gives $\hat{y}$; for $y = 0$ it gives $1 - \hat{y}$. So $P(Y = y | X = x) = \hat{y}^y (1 - \hat{y})^{1-y}$, and taking logs turns exponents into coefficients:

$$LL(\theta) = y \ln \hat{y} + (1 - y) \ln(1 - \hat{y}).$$

- (b)

$$\frac{\partial LL}{\partial \hat{y}} = \frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}} = \frac{y(1-\hat{y}) - (1-y)\hat{y}}{\hat{y}(1-\hat{y})} = \frac{y - y\hat{y} - \hat{y} + y\hat{y}}{\hat{y}(1-\hat{y})} = \frac{y - \hat{y}}{\hat{y}(1-\hat{y})}.$$

- (c) Let $z = \theta^T x = \sum_k \theta_k x_k$. Then $\frac{\partial z}{\partial \theta_j} = x_j$, and

$$\frac{\partial \hat{y}}{\partial \theta_j} = \sigma'(z) \frac{\partial z}{\partial \theta_j} = \sigma(z)[1 - \sigma(z)] x_j = \hat{y}(1 - \hat{y}) x_j.$$

- (d)

$$\frac{\partial LL}{\partial \theta_j} = \frac{y - \hat{y}}{\hat{y}(1 - \hat{y})} \cdot \hat{y}(1 - \hat{y}) x_j = \boxed{x_j [y - \hat{y}]}$$

The $\hat{y}(1 - \hat{y})$ from the log-likelihood derivative cancels the identical factor from the sigmoid derivative. Over the whole dataset, $\frac{\partial LL}{\partial \theta_j} = \sum_{i=1}^n x_j^{(i)} [y^{(i)} - \sigma(\theta^T x^{(i)})]$.

- (e) $y - \hat{y}$ is the model's *error* on this example (signed). The update nudges θ_j in proportion to that error, scaled by how much feature j was “on” for this datapoint — features that were 0 get no credit or blame, and a confident correct prediction ($y - \hat{y} \approx 0$) produces almost no change.

Problem 5: Logistic Regression Code (pset7: logistic_regression)

(Problem set problem)

Problem 6: Conceptual Rapid Fire

- (a) The likelihood is a product of n numbers in $(0, 1)$, which underflows to 0 numerically and is painful to differentiate. The log turns it into a sum, so the derivative is a sum of simple terms. Because log is strictly increasing, the argmax is unchanged.
- (b) Linear regression can output values outside $[0, 1]$, which cannot be probabilities, and it implicitly assumes Gaussian noise around a real-valued target. Logistic regression models $Y | X$ as a Bernoulli whose parameter is squashed into $[0, 1]$ by the sigmoid — the right generative story for a binary label — and its log-likelihood is concave, so gradient ascent finds the global optimum.
- (c) We expect $\theta_i \approx 0$. If X_i carries no information about Y , then any nonzero weight only adds noise to z and lowers the likelihood on average, so the MLE drives it toward 0. (With finite data it will be small but not exactly 0.)
- (d) (*2022 Fall Final*) It has no effect on the predictions. That feature is 0 for every datapoint, so $\theta_{\text{new}} \cdot 0 = 0$ always: it never enters the weighted sum. Its gradient is $x_j(y - \hat{y}) = 0 \cdot (y - \hat{y}) = 0$, so the parameter never moves from its initial value. The model trains exactly as before, just with one useless parameter along for the ride.
- (e) (*2022 Fall Final*) Almost nothing changes in the training loop — you just do feature engineering first. Before training, append a new column to every datapoint whose value is the product $x_1 x_2$, so each x becomes $[1, x_1, x_2, x_1 x_2]$. The gradient formula $x_j(y - \hat{y})$ is unchanged; θ_3 is learned the same way as every other weight. (This is also the standard trick for making a linear classifier handle data that isn't linearly separable in the original features.)

Challenge Problem: Logistic Regression With a Prior

(problem from the 2023 Winter Final)

- (a) For a standard normal, $f(\theta) \propto e^{-\theta^2/2}$, so

$$\frac{f(0)}{f(2)} = \frac{e^0}{e^{-2}} = e^2 \approx 7.389 \text{ times larger.}$$

- (b) By Bayes:

$$f(\theta | y_1, \dots, y_n) = \frac{f(\theta) P(y_1, \dots, y_n | \theta)}{P(y_1, \dots, y_n)}.$$

The denominator does not depend on θ , so it cannot affect the argmax and can be dropped. By the i.i.d. assumption the likelihood factors: $P(y_1, \dots, y_n | \theta) = \prod_i f(y_i | \theta)$. Taking log (monotone, so argmax preserved) turns the product into a sum:

$$\theta_{\text{MAP}} = \arg \max_{\theta} \left[\log f(\theta) + \sum_{i=1}^n \log f(y_i | \theta) \right].$$

- (c) The prior term is $\log f(\theta) = -\frac{\theta^2}{2} + \text{const}$, whose derivative is $-\theta$. The likelihood term is the usual logistic regression log-likelihood with the single-feature gradient $x_i(y_i - \sigma(\theta x_i))$. So

$$\frac{\partial}{\partial \theta} [\cdot] = -\theta + \sum_{i=1}^n x_i [y_i - \sigma(\theta x_i)],$$

and gradient ascent is $\theta \leftarrow \theta + \eta(-\theta + \sum_i x_i [y_i - \sigma(\theta x_i)])$.

The extra $-\theta$ term pulls the parameter back toward 0 on every step — it penalizes large weights. This is exactly L_2 regularization (weight decay), which machine learning derives from scratch and we get for free by

putting a Gaussian prior on θ . It matters most when n is small, and washes out as the data term grows with n .