CS109 Lecture 25 Worksheet: ANSWER KEY

Regression and Reinforcement Learning

Summer 2026

Lecture 25 Worksheet: Answer Key

Instructor / TA copy.

Problem 1 (Review): One Step of Backpropagation

(a) $z^{(2)} = 0(1) + 1(0.6) + (-1)(0.2) = 0.4$, so $\hat{y} = \sigma(0.4) = 0.5987$.

(b) Since $Y \mid X = x \sim \text{Bern}(\hat{y})$, the log-likelihood of a single datapoint is

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

The true label is $y = 1$ — this is given in the problem, *not* obtained by thresholding $\hat{y}$ at 0.5 — so the second term vanishes and

$$LL = \log(0.5987) \approx \boxed{-0.5130}.$$

Teaching note: students frequently assume y was read off from $\hat{y} > 0.5$. It was not. $\hat{y}$ is what the model believes; y is what actually happened, and it comes from the dataset. Had the true label been $y = 0$, the *same* forward pass would give $LL = \log(1 - 0.5987) = \log(0.4013) \approx -0.9130$. Thresholding at 0.5 is a *decision* step from Lecture 21 (accuracy, confusion matrices) and is never performed during training — if it were, the error $y - \hat{y}$ would be 0 here and nothing would ever learn. Worth noting too that this datapoint would be classified *correctly* under a 0.5 threshold, yet $LL = -0.513$ is far from 0: the model got the answer right while being barely more confident than a coin flip.

(c) **Where the formula comes from.** The gradient we want is of the *log-likelihood*, and it reaches $\theta_j^{(2)}$ through two links: $\theta_j^{(2)} \to \hat{y} \to LL$. By the chain rule,

$$\frac{\partial LL}{\partial \theta_j^{(2)}} = \underbrace{\frac{\partial LL}{\partial \hat{y}}}_{\frac{y-\hat{y}}{\hat{y}(1-\hat{y})}} \cdot \underbrace{\frac{\partial \hat{y}}{\partial \theta_j^{(2)}}}_{\hat{y}(1-\hat{y})\, h_j} = \frac{y - \hat{y}}{\hat{y}(1-\hat{y})} \cdot \hat{y}(1-\hat{y})\, h_j = h_j \left[y - \hat{y}\right].$$

The $\hat{y}(1-\hat{y})$ **cancels completely** — the sigmoid derivative from the second factor is exactly annihilated by the denominator that the log produced in the first. This is the same cancellation as Problem 5(a) of the Lecture 22 worksheet, and it is why a formula that ought to be messy comes out as (error) $\times$ (input).

Common error worth watching for. Students often write $\frac{\partial \hat{y}}{\partial \theta_j^{(2)}} = \hat{y}(1 - \hat{y}) h_j$ and stop there, reporting $0.2403 \times 0.6 = 0.1442$ for $j = 1$ instead of 0.2408. That expression is *correct*, but it answers a different question: it is how fast the **prediction** moves, not how fast the **log-likelihood** moves. Only the latter

is the objective we are maximizing. The missing factor is $\frac{\partial LL}{\partial \hat{y}} = 0.4013/0.2403 = 1.6703$, and indeed $1.6703 \times 0.1442 = 0.2408$. It is not a sign slip or an arithmetic mistake — it is differentiating the wrong function, so it is worth calling out explicitly rather than marking silently.

The calculation. Since the same error term is shared by all three parameters and only the multiplier h_j changes, compute the error once:

$$y - \hat{y} = 1 - 0.5987 = 0.4013.$$

Then multiply it by each component of $h = [h_0, h_1, h_2] = [1,\ 0.6,\ 0.2]$:

$$\frac{\partial LL}{\partial \theta_0^{(2)}} = h_0\,[y - \hat{y}] = (1)(0.4013) = \boxed{0.4013},$$

$$\frac{\partial LL}{\partial \theta_1^{(2)}} = h_1\,[y - \hat{y}] = (0.6)(0.4013) = \boxed{0.2408},$$

$$\frac{\partial LL}{\partial \theta_2^{(2)}} = h_2\,[y - \hat{y}] = (0.2)(0.4013) = \boxed{0.0803}.$$

Three observations worth making at the board. First, all three gradients are **positive**, because $\hat{y}$ undershot the true label $y = 1$; gradient ascent will therefore increase all three weights, pushing $\hat{y}$ up. Second, the gradients are in the ratio $1 : 0.6 : 0.2$, exactly the ratio of the activations — **the more active a hidden neuron was, the more credit or blame its weight receives.** A neuron with $h_j = 0$ would receive no update at all, since it contributed nothing to this prediction. Third, the bias gradient is the raw error 0.4013 itself, because $h_0 = 1$ always; the bias absorbs the error uniformly regardless of what any neuron did.

(d) $h_1(1 - h_1) = 0.6(0.4) = 0.24$, $\theta_1^{(2)} = 1$, $x_1 = 2$:

$$\frac{\partial LL}{\partial \theta_{1,1}^{(1)}} = 0.4013 \times 1 \times 0.24 \times 2 \approx \boxed{0.1926}.$$

(e) With no output sigmoid the prediction is an unbounded real number, so a Bernoulli is impossible — a Bernoulli's support is $\{0, 1\}$ and its parameter must lie in $[0, 1]$, but $z^{(2)}$ can be -37 or 412. The natural replacement is

$$Y \mid X = x \sim N(\theta^T x,\ \sigma^2),$$

a Normal centered on the prediction. **That single substitution is the entire content of today's first half.** Everything else — write the log-likelihood, differentiate, gradient ascend — is unchanged. It is worth pausing here at the board: students often think regression is a different subject from classification. It is the same subject with a different distribution plugged into step 1.

Problem 2: The Landscape of Machine Learning Tasks

(a) **Binary classification.** $Y \sim \text{Bern}(p)$, support $\{0, 1\}$.

(b) **Regression.** Y is a real-valued (continuous) random variable; in principle support $\mathbb{R}$, though points are non-negative in practice. This is today's model.

(c) **Multi-class classification.** $Y \sim$ Categorical over 1000 classes, support $\{1, \ldots, 1000\}$; predicted with a softmax.

(d) **Multi-class classification** (which, applied repeatedly, is generation). $Y \sim$ Categorical over the vocabulary.

(e) **Reinforcement learning.** There is no Y label at all — only a scalar reward signal, and it arrives late.

(f) An LLM is a multi-class classifier over the vocabulary. "Generation" is what you get when you *sample* from that Categorical instead of taking its argmax, feed the sampled token back in as input, and repeat. The model is a classifier; generation is a loop around it. (This is also why LLM outputs are non-deterministic and why "temperature" is a knob on a Categorical.)

Problem 3: The Linear Regression Model

(a) $\theta^T x^{(i)}$ is a *constant* once $x^{(i)}$ and θ are fixed. Two Lecture 9 facts: (i) adding a constant c to a Normal gives a Normal with the mean shifted by c and the variance unchanged; (ii) $Z \sim N(0, \sigma^2)$, so $\theta^T x^{(i)} + Z \sim N(\theta^T x^{(i)} + 0,\ \sigma^2)$. Hence $Y^{(i)} \mid X = x^{(i)} \sim N(\theta^T x^{(i)}, \sigma^2)$.

(b) Three assumptions about the errors: they are **Normally distributed**; they are **independent** of each other (and of x); and they are **identically distributed**, in particular with the *same* variance σ^2 for every datapoint (homoscedasticity). A fourth, hiding in plain sight, is that the mean of Y is a **linear** function of x.

(c) Mean-zero noise is what makes the model *unbiased*: $E[Y \mid X = x] = \theta^T x$, so the line we fit is the conditional expectation. If the noise truly had mean 5, the model could not tell that apart from an intercept that is 5 too small — the fitted θ_0 would simply absorb it, landing 5 above the "true" intercept. Constant bias and the intercept are not identifiable separately, which is exactly why we can assume mean 0 without loss of generality.

(d) Any dataset where the spread grows with the magnitude of the prediction. Examples: household income vs. spending (rich households vary far more in dollars); Caltrain ridership at 3am vs. rush hour; house price vs. square footage. Predicting a quantity that is bounded below by 0 almost always produces heteroscedasticity near the boundary.

(e)

	Logistic Regression	Linear Regression
Distribution of $Y \mid X = x$	$\text{Bern}(\sigma(\theta^T x))$	$N(\theta^T x, \sigma^2)$
Range of the prediction	$(0, 1)$ — a probability	$(-\infty, \infty)$ — a real number
Function applied to $\theta^T x$	sigmoid	identity (none)
Log-likelihood of one datapoint	$y \log \hat{y} + (1-y)\log(1-\hat{y})$	$-\log(\sigma\sqrt{2\pi}) - \frac{(y-\hat{y})^2}{2\sigma^2}$

Teaching note: the row that matters is the last one. Both are "the log of the PMF/PDF of the assumed distribution, evaluated at the observed y." Nothing about the recipe changed.

Problem 4: Deriving Least Squares from Maximum Likelihood

(a) By independence,

$$L(\theta) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{y^{(i)} - \theta^T x^{(i)}}{\sigma}\right)^2\right].$$

(b) Taking logs turns the product into a sum, and $\log e^u = u$:

$$LL(\theta) = \sum_{i=1}^{n} \log\left[\frac{1}{\sigma\sqrt{2\pi}}\right] - \frac{1}{2}\left(\frac{y^{(i)} - \theta^T x^{(i)}}{\sigma}\right)^2.$$

(c) The first term is $n \log \frac{1}{\sigma\sqrt{2\pi}}$, a constant with respect to θ, so it cannot change the argmax. The remaining term is $-\frac{1}{2\sigma^2}\sum_i (y^{(i)} - \theta^T x^{(i)})^2$, and $\frac{1}{2\sigma^2} > 0$ is also a θ-free constant; scaling by a positive

constant does not move the argmax either. Dropping both and flipping the sign turns the maximization into a minimization:

$$\underset{\theta}{\operatorname{argmax}}\, LL(\theta) = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y^{(i)} - \theta^T x^{(i)}\right)^2.$$

Say it out loud: *minimizing the sum of squared errors is maximum likelihood estimation under Gaussian noise.* Least squares is not a convention, an aesthetic preference, or a computational convenience. It is what MLE reduces to when you assume the errors are Normal — and it would be a *different* objective under a different noise assumption (see the Challenge Problem). Every "just minimize squared error" you have ever seen in a physics lab, an econometrics paper, or a machine learning library was silently assuming a Gaussian.

(d) Differentiate term by term. The chain rule is applied to the outer square: let $u_i = y^{(i)} - \theta^T x^{(i)}$, so $\frac{\partial u_i}{\partial \theta_j} = -x_j^{(i)}$ (the minus sign comes from θ appearing with a minus inside u_i). Then

$$\frac{\partial}{\partial \theta_j}\left[-\sum_i u_i^2\right] = -\sum_i 2u_i \frac{\partial u_i}{\partial \theta_j} = -\sum_i 2u_i(-x_j^{(i)}) = \sum_{i=1}^{n} 2\left(y^{(i)} - \theta^T x^{(i)}\right) x_j^{(i)}.$$

The two minus signs — one from maximizing the *negative* SSE, one from the chain rule — cancel, which is why the final expression is pleasingly positive.

(e) The pattern is

$$\frac{\partial LL}{\partial \theta_j} \;\propto\; \sum_i \underbrace{\left[y^{(i)} - \hat{y}^{(i)}\right]}_{\text{error}} \cdot \underbrace{x_j^{(i)}}_{\text{input}},$$

identical in all three cases up to a constant factor (which is absorbed into the learning rate). Gradient ascent:

$$\theta_j \;\leftarrow\; \theta_j + \eta \sum_{i=1}^{n} \left[y^{(i)} - \hat{y}^{(i)}\right] x_j^{(i)}.$$

Worth emphasizing: this is the same line of code for logistic regression, linear regression, and the output layer of a neural network. Only the function computing $\hat{y}$ differs. As noted in Problem 3(d) of Lecture 22, this is not luck — sigmoid and the identity are the canonical links for the Bernoulli and the Normal, and for canonical links the score is always (observed − expected) times input.

Problem 5: Fitting a Line by Hand

(a) With $\theta = [0, 2]$: $\hat{y} = [2, 4, 6, 8]$, errors $y - \hat{y} = [1, 1, 0, 1]$, and

$$\text{SSE} = 1 + 1 + 0 + 1 = \boxed{3}.$$

(b) With $x_0 = 1$ everywhere:

$$\frac{\partial}{\partial \theta_0} = 2\sum_i (y^{(i)} - \hat{y}^{(i)})(1) = 2(1 + 1 + 0 + 1) = \boxed{6},$$

$$\frac{\partial}{\partial \theta_1} = 2\sum_i (y^{(i)} - \hat{y}^{(i)}) x_1^{(i)} = 2\left[1(1) + 1(2) + 0(3) + 1(4)\right] = 2(7) = \boxed{14}.$$

(c) $\theta \leftarrow [0 + 0.01(6),\ 2 + 0.01(14)] = [0.06,\ 2.14]$. New predictions:

$$\hat{y} = [2.20,\ 4.34,\ 6.48,\ 8.62], \qquad y - \hat{y} = [0.80,\ 0.66,\ -0.48,\ 0.38].$$

$$\text{SSE} = 0.64 + 0.4356 + 0.2304 + 0.1444 = \boxed{1.4504}.$$

It improved substantially, from 3 to 1.45. Note that the third datapoint has *overshot* into a negative error — gradient ascent moves all parameters at once, so individual datapoints can get worse while the total gets better.

(d) At $\theta = [1.0, 1.9]$: $\hat{y} = [2.9,\ 4.8,\ 6.7,\ 8.6]$, so the residuals are

$$[\,0.1,\ 0.2,\ -0.7,\ 0.4\,], \qquad \text{sum} = 0.$$

They must sum to zero because at the optimum every partial derivative is zero, and the $j = 0$ component is $2\sum_i (y^{(i)} - \hat{y}^{(i)}) x_0^{(i)} = 2\sum_i (y^{(i)} - \hat{y}^{(i)})$, since $x_0 = 1$. Setting that to zero says precisely that the residuals sum to zero. **The intercept's only job is to make the errors balance.** (Corollary: if you drop the bias term, this guarantee disappears.)

(e) $\text{SSE} = 0.01 + 0.04 + 0.49 + 0.16 = \boxed{0.70}$, versus 3 at the start and 1.45 after one step. No choice of θ can do better on this dataset.

Problem 6: The Noise Variance Is a Parameter Too

(a) $\hat{\sigma}^2 = \frac{1}{4}(0.01 + 0.04 + 0.49 + 0.16) = \frac{0.70}{4} = \boxed{0.175}$, so $\hat{\sigma} \approx 0.418$.

Note the identity worth pointing out at the board: $\hat{\sigma}^2 = \text{SSE}/n$. The variance MLE *is* the minimized objective, divided by n.

(b) $\hat{\sigma}^2 = \frac{1}{5}(9 + 25 + 1 + 16 + 49) = \frac{100}{5} = \boxed{20}$, so $\hat{\sigma} = \sqrt{20} \approx 4.472$.

(c) $Z \sim N(0, 20)$, and

$$P(|Z| > 10) = 2\left[1 - \Phi\left(\frac{10}{4.472}\right)\right] = 2\left[1 - \Phi(2.236)\right] \approx 2(0.0127) = \boxed{0.0253}.$$

About a 2.5% chance of being off by more than 10 riders.

(d) On the training set the residuals are the ones the fit was chosen to make small, so $\hat{\sigma}^2$ would be optimistically low — you would systematically *understate* your uncertainty, and the more parameters the model has, the worse the understatement. (In the extreme, a model with n parameters fits n points exactly and reports $\hat{\sigma}^2 = 0$, claiming perfect certainty.) This is the same train/test discipline from Lecture 21, now applied to the error bars rather than to accuracy.

(e) $\pm 1.96\hat{\sigma} = \pm 1.96(4.472) \approx \pm 8.77$, giving roughly

$$(111.2,\ 128.8).$$

This is the real prize of the probabilistic framing: a Normal distribution over outcomes, not just a number. A point prediction of 120 tells a dispatcher nothing about whether to hold a spare bus.

Problem 7: Reinforcement Learning

(a) An infinite geometric series (Lecture 5 / the geometric random variable):

$$G = \sum_{t=0}^{\infty} \gamma^t = \frac{1}{1-\gamma} = \frac{1}{0.1} = \boxed{10}.$$

(b) By linearity of expectation, extended to an infinite sum (valid here because the series converges absolutely — rewards are bounded and $\gamma < 1$):

$$E[G] = \sum_{t=0}^{\infty} \gamma^t E[R_t] = 0.5 \sum_{t=0}^{\infty} 0.8^t = \frac{0.5}{1 - 0.8} = \boxed{2.5}.$$

Linearity does not require independence, which is a good thing — in a real environment R_t and R_{t+1} are heavily dependent.

(c) With $\gamma = 1$ the sum need not converge: an immortal agent collecting $+1$ forever has infinite return, and so does one collecting $+1$ every other step, so the objective cannot distinguish between them. Discounting both guarantees a finite expectation and encodes a preference for reward *sooner*. (It is also a reasonable model of a nonzero per-step probability of the episode ending.)

(d) Beta is conjugate to Bernoulli (Lecture 14): posterior $=$ Beta($\alpha +$ successes, $\beta +$ failures).

- Red: Beta$(1+7,\ 1+3) =$ Beta$(8,4)$. Mean $= \frac{8}{12} = \boxed{0.667}$, variance $= \frac{8\cdot 4}{12^2\cdot 13} = 0.0171$, sd ≈ 0.131.
- Green: Beta$(1+2,\ 1+0) =$ Beta$(3,1)$. Mean $= \frac{3}{4} = \boxed{0.750}$, variance $= \frac{3\cdot 1}{4^2\cdot 5} = 0.0375$, sd ≈ 0.194.

(e) The greedy critter eats **green** ($0.750 > 0.667$). But green's estimate rests on two observations and has half again the standard deviation of red's: two lucky berries out of two is entirely consistent with a true θ of 0.4. If the critter commits to green forever it will never gather the evidence that would reveal the mistake, and it also never learns whether red is secretly better than 0.667. This is the **exploration/exploitation tradeoff**. One principled fix, which students already have all the machinery for: sample a θ from each posterior and eat whichever sample is larger — that is Thompson sampling, and it is a CS109 Beta distribution doing reinforcement learning.

(f) Structural differences (any two):

1. **No labels.** Supervised learning is told the right answer for each input; RL is told only how good the outcome was, never what it should have done instead.
2. **Delayed and credit-assigned reward.** A reward at step 500 may be due to an action at step 3. Supervised learning never has to figure out *which* decision earned the feedback.
3. **The agent generates its own data.** The distribution of training data depends on the policy, so the i.i.d. assumption we have relied on all quarter fails outright — taking an action changes what you will see next.
4. **Exploration is required.** A supervised learner cannot improve its dataset by acting; an RL agent must deliberately try things it currently believes are suboptimal.

Challenge Problem: What If the Noise Isn't Gaussian?

(a) With $y^{(i)} = \theta^T x^{(i)} + Z^{(i)}$ and $Z \sim \text{Laplace}(0, b)$:

$$LL(\theta) = \sum_{i=1}^{n} \log \frac{1}{2b} - \frac{\left|y^{(i)} - \theta^T x^{(i)}\right|}{b}.$$

The first term and the $\frac{1}{b}$ are θ-free, so

$$\operatorname*{argmax}_{\theta} LL(\theta) = \operatorname*{argmin}_{\theta} \sum_{i=1}^{n} \left|y^{(i)} - \theta^T x^{(i)}\right|,$$

least *absolute* deviations. The objective changed only because the assumption changed; the method did not.

(b) Squared error is far more disturbed. An outlier at distance d contributes d^2 to the squared objective but only d to the absolute one, so with $d \approx 10{,}000$ the single point contributes on the order of 10^8 and effectively dictates the fit. The deep reason is in the PDFs: the Gaussian's tail decays like e^{-z^2}, so a residual of 10,000 is assigned a likelihood so astronomically small that the model will contort itself to explain it away; the Laplace tail decays only like $e^{-|z|}$, so a large residual is improbable but not unthinkable. **How fast your assumed distribution's tail decays is exactly how much a single weird datapoint can bully your model.** Least absolute deviations is "robust regression" for precisely this reason.

(c) With $\theta_j \sim N(0, \tau^2)$ i.i.d.,

$$\log f(\theta) = \sum_j \left[\log \frac{1}{\tau\sqrt{2\pi}} - \frac{\theta_j^2}{2\tau^2}\right], \qquad \sum_i \log f(y^{(i)} \mid x^{(i)}, \theta) = C - \frac{1}{2\sigma^2} \sum_i \left(y^{(i)} - \theta^T x^{(i)}\right)^2.$$

Dropping constants and multiplying the whole objective by $-2\sigma^2$ (which flips argmax to argmin):

$$\theta_{\text{MAP}} = \operatorname*{argmin}_{\theta} \sum_{i=1}^{n} \left(y^{(i)} - \theta^T x^{(i)} \right)^2 + \frac{\sigma^2}{\tau^2} \sum_j \theta_j^2, \qquad \boxed{\lambda = \frac{\sigma^2}{\tau^2}}.$$

L2 regularization is a Gaussian prior on the weights. (And an L1 penalty — lasso — is a Laplace prior. Same trick as part (a), on the parameters instead of the noise.)

(d) As $\tau^2 \to \infty$, $\lambda \to 0$ and MAP collapses to MLE. A prior with infinite variance is the statement "I believe nothing in particular about θ before seeing data," so the prior stops pulling and ordinary least squares returns. Conversely, small τ^2 means a confident prior that the weights are near zero, which shrinks the fit hard toward the flat line — the same MLE/MAP dial from Lecture 14, now with regression on the other end of it.