CS109 Lecture 21 Worksheet: ANSWER KEY

Comparing Classifiers

Summer 2026

Lecture 21 Worksheet: Answer Key

Instructor / TA copy. Problem-set solutions are hidden in the standard build; run `./convert-to-pdf.sh 21 solutions` for the full instructor/TA edition.

Problem 1 (Review): Logistic Regression Refresher

(a) $z = -1 + 3(1) - 1(1) = 1$, so $P(Y = 1 \mid X = x) = \sigma(1) = 0.7311$.

(b) $\frac{\partial LL}{\partial \theta_1} = x_1(y - \hat{y}) = 1(1 - 0.7311) = 0.2689$. It is positive, so the step $\theta_1 \mathrel{+}= \eta \cdot 0.2689$ *increases* θ_1 — the model was right but not confident enough, so it leans harder on the feature that was on.

(c) Feature 2 is evidence *against* $Y = 1$: turning it on lowers z and pulls the predicted probability down.

Problem 2: Brute Force Bayes and Why We Need an Assumption

(a) With $m = 2$ there are $2^2 = 4$ feature assignments, and for each we need a probability for $Y = 0$ and for $Y = 1$ — but those sum to 1, so 4 numbers suffice if we're careful. In the form presented in lecture (a conditional probability table $P(X = x \mid Y = y)$ for each class) we store $2 \times 2^2 = 8$ entries. Either bookkeeping is fine as long as students see the 2^m factor.

(b) $O(2^m)$ parameters — exponential in the number of features.

(c) Heart, $m = 22$: $2^{22} \approx 4.2$ million (times 2 for the two classes). Ancestry, $m = 100$: $2^{100} \approx 1.3 \times 10^{30}$. That is far more table entries than there are stars in the observable universe, though still short of its $\sim 10^{80}$ atoms — the joke in lecture is that we blow past every physical budget long before m gets interesting.

(d) Each of the 2^{100} cells needs its own estimate, and each estimate requires training examples that land in *that specific cell*. With 467 datapoints, at most 467 cells are nonempty — every other cell gets an estimate of 0/0. The model would have essentially no idea what to say about any new datapoint it hasn't seen verbatim. This is the curse of dimensionality: data, not memory, is the binding constraint.

(e) **Naive Bayes assumption:** the features are conditionally independent given the class label:

$$P(X_1 = x_1, \ldots, X_m = x_m \mid Y = y) = \prod_{j=1}^{m} P(X_j = x_j \mid Y = y).$$

Now we only need one probability per (feature, class) pair: $2m$ parameters plus the prior on Y, so $O(m)$ — linear instead of exponential. The assumption is usually false (features *are* correlated), which is why it is called "naive," but it makes estimation possible and the classifier often works well anyway.

Problem 3: Naive Bayes With a Prior

(problem from the 2022 Fall Final)

(a) Model $Y \sim \text{Bern}(p)$ with prior $p \sim \text{Beta}(3, 4)$. Observing 30 successes and 70 failures gives posterior

$$p \mid \text{data} \sim \text{Beta}(3 + 30,\ 4 + 70) = \text{Beta}(33, 74),$$

and the estimate is the posterior mean:

$$\hat{p} = \frac{33}{33 + 74} = \frac{33}{107} \approx 0.3084.$$

(b) A Beta(3, 4) prior acts like 2 imaginary successes and 3 imaginary failures already observed ($a - 1$ and $b - 1$ pseudo-counts). Those pseudo-observations have a success rate of $2/5 = 0.4$, above the data's 0.30, so blending them in pulls the estimate slightly *up*, from 0.3000 to 0.3084. With 100 real datapoints the prior only nudges the answer; with 5 datapoints it would dominate.

(c) The MLE would be $P(X_j = 1 \mid Y = 1) = 0/30 = 0$. Then *any* test datapoint with $X_j = 1$ gets $P(X = x \mid Y = 1) = 0$ from the product, so the model assigns probability 0 to the positive class no matter how much other evidence points that way — a single unseen feature value vetoes everything. A prior (Laplace: add 1 success and 1 failure) makes the estimate 1/32 instead of 0: small, but never a hard zero. This is exactly the "never assign probability 0 to something you haven't ruled out" lesson from the Beta lecture.

Problem 4: Train, Test, and Overfitting

(a) "Is that train accuracy or test accuracy?" A model can memorize its training set and score near-perfectly there while being useless on new data. Only performance on data the model never saw during training tells you anything about deployment.

(b) Random Forest: train 0.8726 vs. test 0.8500, a gap of 0.0226 — the largest in the table. Decision Tree is close behind (0.8514 vs. 0.8307). By contrast Naive Bayes and Logistic Regression actually score *higher* on test than train, which is a sign they are not overfitting at all (and, with a test set this size, a reminder that these numbers have real error bars).

(c) Without a baseline you cannot tell whether a number is good. On a dataset that is 90% class 1, a model reporting 89% accuracy is *worse than doing nothing*. The baseline calibrates your sense of what the accuracy scale means for this particular dataset.

(d) How big is the test set, and what are the error bars on each accuracy? A 2-point gap on a few hundred test points is well within sampling noise. Use bootstrapping (resample the test set with replacement, recompute the accuracy difference many times) to get a confidence interval on the *difference*, or a p-value for the hypothesis that the two models are equally good. This is exactly the machinery from the bootstrapping lecture.

(e) Any dataset whose classes interleave: the classic is XOR, with $y = 1$ at $(0, 0)$ and $(1, 1)$ and $y = 0$ at $(0, 1)$ and $(1, 0)$; concentric rings work too. No single line separates them. Both models still often work well because (i) real high-dimensional data is frequently *close* to linearly separable even when not perfectly so, (ii) a linear boundary that gets most points right is still a useful classifier, and (iii) you can add interaction or polynomial features (as in the Lecture 20 conceptual problem) to make the data linearly separable in an expanded feature space.

Problem 5: Is My Model Calibrated?

(a) Observed fractions of $Y = 1$:

Group	Stated	Observed	Verdict
Very low	0.1	$8/100 = 0.08$	close
Low	0.3	$30/100 = 0.30$	calibrated
Borderline	0.5	$52/100 = 0.52$	close
High	0.7	$75/100 = 0.75$	close
Very high	0.9	$70/100 = 0.70$	**badly off**

The model is well calibrated everywhere except at the top of its range.

(b) Every patient in that group has output probability > 0.5, so all 100 are predicted $Y = 1$. Accuracy $= 70/100 = 0.70$.

(c) $X \sim \text{Bin}(100, 0.9)$, so

$$P(X = 70) = \binom{100}{70}(0.9)^{70}(0.1)^{30} \approx 1.8 \times 10^{-8}.$$

Very surprised. The expected count is 90 with standard deviation $\sqrt{100(0.9)(0.1)} = 3$, so 70 is about 6.7 standard deviations below the mean. This is not sampling noise — the model really is miscalibrated in this bucket.

(d) **Overconfident.** It claims 90% certainty but is right only 70% of the time. In a medical setting that is the dangerous direction of error: a doctor reading "0.9" would act far more decisively than the evidence supports.

(e) A classifier that ignores its input and always outputs the base rate. If 30% of patients have the disease, a model that says "0.3" for everyone is perfectly calibrated — among all patients it labels 0.3, exactly 30% do have the disease — yet it has zero discriminative power and, thresholded at 0.5, never predicts a positive case at all. Calibration measures whether the *numbers mean what they say*; accuracy measures whether the *decisions are right*. You want both.

Problem 6: When Accuracy Lies — Precision and Recall

(a) $TP = 40$, $FP = 10$, $FN = 20$, $TN = 130$.

$$\text{accuracy} = \frac{40 + 130}{200} = 0.85, \quad \text{precision} = \frac{40}{50} = 0.80, \quad \text{recall} = \frac{40}{60} \approx 0.667.$$

(b) "Always predict 0" gets the 140 true negatives right and the 60 positives wrong: accuracy $= 140/200 = 0.70$. Recall $= 0/60 = 0$. Precision is $0/0$ — undefined, since it never makes a positive prediction. So a model that catches *no fraud at all* still scores 70% accuracy; this is the whole reason precision and recall exist.

(c) Lowering the threshold makes the model predict 1 more often. Recall goes **up** (you catch more of the true positives; FN can only shrink). Precision typically goes **down** (the newly flagged cases are the ones the model was least sure about, so more of them are false positives). This tradeoff traced out over all thresholds is the precision-recall curve.

(d) **Fraud:** favor recall. A missed fraud is a direct financial loss and a customer harmed, while a false positive costs one declined transaction and a phone call — annoying but cheap and reversible. **Invasive screening:** favor precision. A false positive sends a healthy patient into a procedure with real risk, cost, and anxiety, so you want high confidence before flagging. (Reasonable students will argue the opposite for screening if the disease is deadly and the follow-up is safe — accept it if they justify the tradeoff explicitly. The point is that the metric follows from the cost of each error type, not from the math.)

Problem 7: Is the Model Fair?

(a) $P(G = 1 \mid D = 1) = 60/200 = 0.30$; $P(G = 1 \mid D = 0) = 100/200 = 0.50$. Parity requires these be equal; they are not. The ratio is $0.30/0.50 = 0.60$, well below the 0.8 threshold of the disparate-impact "80% rule," so the model would be legally suspect.

(b) $P(T = 1 \mid G = 1, D = 1) = 48/60 = 0.80$; $P(T = 1 \mid G = 1, D = 0) = 70/100 = 0.70$. The gap is $0.10 \leq \varepsilon = 0.2$, so the model **does** satisfy the relaxed calibration criterion: when it says "approve," it is about equally right for both groups.

(c) "Fair" is not a single well-defined property — it is a family of incompatible criteria, and you have to choose which one your application requires. (There are formal impossibility results: except in degenerate cases, no classifier can simultaneously satisfy parity and calibration when the base rates differ between groups.) The COMPAS debate is exactly this: its defenders pointed to calibration, its critics pointed to error-rate disparities, and both were reading the numbers correctly.

(d) Sensitive attributes are *redundantly encoded* in the other features. Zip code, purchase history, which pages you follow, and dozens of other innocuous-looking variables are individually correlated with the protected attribute, so a model with enough features can reconstruct it even when it was never given it. Sapiezynski et al. (2019) showed that Facebook's "Special Ad Audiences," built explicitly to exclude gender, were just as gender-biased as the ordinary lookalike audiences, and nearly as age-biased. Deleting the column removes your ability to *measure* the bias without removing the bias itself — which is the argument for fairness through *awareness*.

Problem 8: Decision Tree Entropy (pset7: decision_tree_entropy)

(Problem set problem)

Challenge Problem: Platt Recalibration

(problem from the 2024 Winter Final)

(a) $z = 2(0.9) - 0.5 = 1.3$, so $P(Y = 1 \mid \hat{p}) = \sigma(1.3) \approx 0.7858$. (Note it pulled 0.9 down toward the observed 0.7 — which is the point.)

(b) This is logistic regression with a single parameter, where the "feature" is the uncalibrated output $\hat{p}$. Let $q^{(i)} = \sigma(a\hat{p}^{(i)} - 0.5)$. Assume $Y^{(i)} \mid \hat{p}^{(i)} \sim \text{Bern}(q^{(i)})$ and that the datapoints are i.i.d. Then

$$L(a) = \prod_{i=1}^{n} \left(q^{(i)}\right)^{y^{(i)}} \left(1 - q^{(i)}\right)^{1-y^{(i)}},$$

$$LL(a) = \sum_{i=1}^{n} y^{(i)} \log q^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - q^{(i)}\right).$$

Differentiate by the chain rule, using $\frac{\partial LL}{\partial q} = \frac{y-q}{q(1-q)}$ and the given $\frac{\partial q}{\partial a} = q(1-q)\hat{p}$. The $q(1-q)$ cancels, exactly as in the Lecture 20 derivation:

$$\frac{\partial LL}{\partial a} = \sum_{i=1}^{n} \hat{p}^{(i)} \left[y^{(i)} - \sigma\left(a\hat{p}^{(i)} - 0.5\right)\right].$$

There is no closed form, so run gradient ascent: initialize a (say $a = 1$), then repeat $a \mathrel{+}= \eta \sum_i \hat{p}^{(i)}[y^{(i)} - q^{(i)}]$ until convergence. LL is concave in a, so this reaches the global maximum.

(c) The original classifier's probabilities on its own training data are optimistically distorted — it has already fit that data, so its outputs there look better calibrated than they really are. Fitting a on those numbers would learn "no correction needed" and leave the deployed miscalibration in place. The

recalibration model must see the classifier's behavior on data it has never trained on, which in practice means a held-out validation split (and then a *third* split to honestly report final performance).