

CS 189/289A, Midterm

Fall 2026

Name: _____

Email: _____@berkeley.edu

Student ID: _____

Room where you're taking the exam: _____

Name and SID of the person on your right: _____

Name and SID of the person on your left: _____

Instructions:

This exam consists of **48 points** spread out over **7 questions**, plus **1 point** for the Honor Code certification. The exam must be completed in **80 minutes** unless you have accommodations supported by a DSP letter.

Note that you should **select one choice** for questions with **circular bubbles**. There is always at least one correct answer. Please **fully** shade in the circle to mark your answer. For all math questions, **please simplify your answer**. Please also **show your work** if a large box is provided. For all coding questions, you may use commas and/or one or more function calls in each blank.

You MUST write your Student ID number at the top of each page.

Honor Code [1 Pt]:

As a member of the UC Berkeley community, I act with honesty, integrity, and respect for others. I am the person whose name is on the exam, and I completed this exam in accordance with the Honor Code.

Signature: _____

This page has been intentionally left blank.

1 Lasso's Evil Twin [8 points]

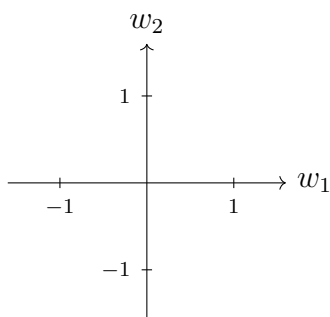
Throughout this question, $X \in \mathbb{R}^{N \times D}$ is a design matrix, $\mathbf{t} \in \mathbb{R}^N$ is a vector of targets, and $\lambda \geq 0$ is a regularization parameter. The *infinity norm* of a vector is its largest absolute entry,

$$\|\mathbf{w}\|_\infty = \max_i |w_i|.$$

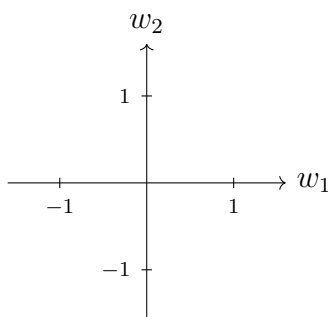
We study least squares with an infinity norm penalty,

$$\mathbf{w}_\lambda = \operatorname{argmin}_{\mathbf{w} \in \mathbb{R}^D} \frac{1}{2} \|X\mathbf{w} - \mathbf{t}\|_2^2 + \lambda \|\mathbf{w}\|_\infty. \quad (1)$$

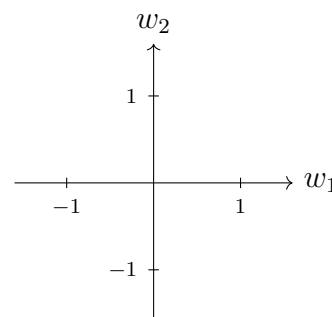
(a) [3 Pts] Sketch the boundary of each inequality in $\mathbb{R}^2$ on the axes provided.



(i) $\|\mathbf{w}\|_1 \leq 1$



(ii) $\|\mathbf{w}\|_2 \leq 1$



(iii) $\|\mathbf{w}\|_\infty \leq 1$

(b) The infinity-norm regularized model (1) was fit at several values of λ , and its mean squared error was measured on the training set and on a validation set.

λ	0	1	3	10	30
Training MSE	10.0	10.4	11.5	15.0	24.0
Validation MSE	14.8	13.9	12.6	13.5	20.1

(i) [1 Pt] Which λ should be deployed?

☐ 0

☐ 1

☐ 3

☐ 10

☐ 30

(ii) [1 Pt] Compared with the model at $\lambda = 3$, the model at $\lambda = 0$ has

☐ lower bias and higher variance.

☐ lower bias and lower variance.

☐ higher bias and lower variance.

☐ higher bias and higher variance.

(c) [1 Pt] Let $\mathbf{w}^* = (X^\top X)^{-1} X^\top \mathbf{t}$ be the ordinary least squares solution. Suppose the noisy ground-truth labels we observe are generated by a fixed true weight vector $\mathbf{w}_{\text{true}}$ plus zero-mean noise, $\mathbf{t} = X\mathbf{w}_{\text{true}} + \epsilon$. If ϵ 's variance is larger, which property of the least squares solution $\mathbf{w}^*$ changes?

☐ its expected value

☐ both

☐ its variance

☐ neither

(d) [1 Pt] Which penalty is not guaranteed to increase when the magnitude of an individual weight (an entry of $\mathbf{w}$) increases?

☐ ridge (ℓ_2)

☐ infinity norm (ℓ_∞)

☐ lasso (ℓ_1)

☐ none

(e) [1 Pt] Which of the following are linear regression models? **Select all that apply.**

☐ $y = w_0 + w_1x + w_2x^2$

☐ $y = w_0 + w_1 \exp(-(x-3)^2)$

☐ $y = \sigma(w_0 + w_1x)$, with σ the logistic sigmoid

☐ $y = w_0 + w_1 \sin(w_2x)$

2 Baby Steps [8 points]

For a small regression problem, the training loss written out as a function of the weights $\mathbf{w}$ is

$$L(\mathbf{w}) = w_1^2 - 4w_1 + 4w_2^2 + 8w_2 + 8, \quad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}.$$

We minimize L using gradient descent:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla L(\mathbf{w}^{(\tau)}).$$

Gradient descent starts from the initial weights and uses the learning rate

$$\mathbf{w}^{(0)} = \begin{bmatrix} 7 \\ 4 \end{bmatrix}, \quad \eta = \frac{1}{10}.$$

- (a) [2 Pts] Compute $\frac{\partial L}{\partial w_1}$ and $\frac{\partial L}{\partial w_2}$. Then write $\nabla L(\mathbf{w})$ as a column vector in terms of w_1 and w_2 . Do **not** plug in $\mathbf{w}^{(0)}$. Show your work in the box, then write your answer in the answer box.

$$\nabla L(\mathbf{w}) =$$

- (b) [2 Pts] Regardless of your answer to (a), suppose that the gradient at the initial weights is

$$\nabla L(\mathbf{w}^{(0)}) = \begin{bmatrix} 10 \\ 40 \end{bmatrix}.$$

Compute $\mathbf{w}^{(1)}$, the weights after one step of gradient descent with $\eta = \frac{1}{10}$. Show your work in the box, then write each coordinate in its answer box.

$$w_1^{(1)} =$$

$$w_2^{(1)} =$$

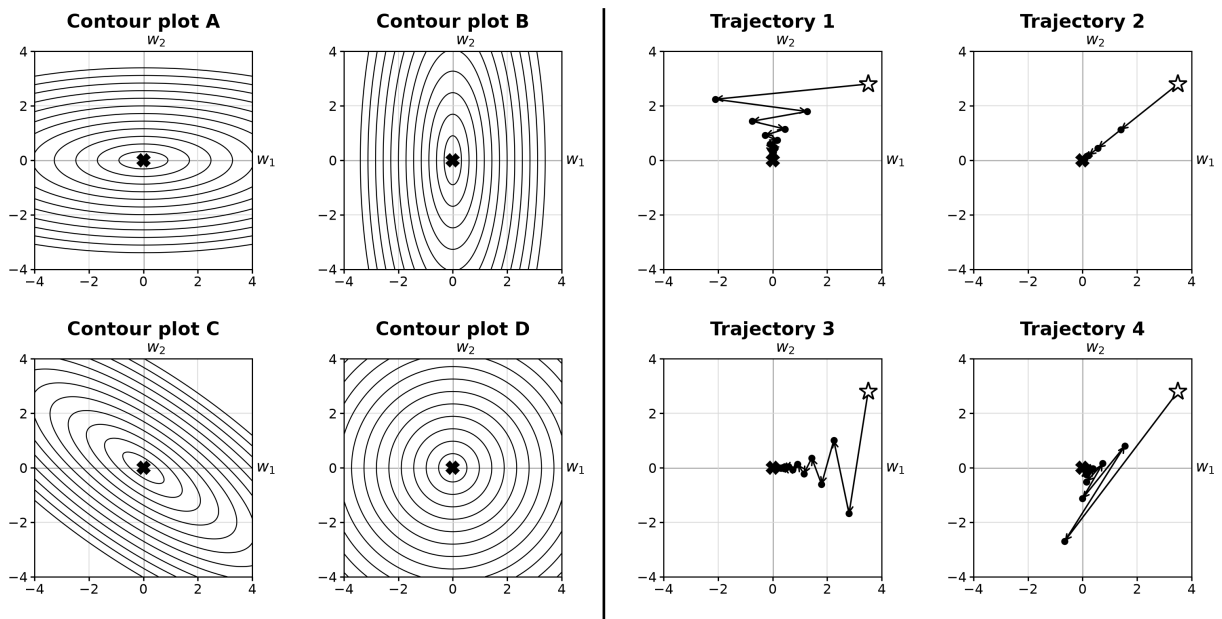
(c) [2 Pts] Contour plots A–D each show the isocontours of a quadratic loss of the form

$$L(\mathbf{w}) = \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^\top H(\mathbf{w} - \mathbf{w}^*)$$

for a different symmetric positive definite matrix H , where the cross marks the minimizer $\mathbf{w}^*$. Trajectories 1–4 show gradient descent run on these four losses, one trajectory per loss. All four runs use the same initial point (the star), minimizer (the cross), and learning rate:

$$\mathbf{w}^{(0)} = \begin{bmatrix} 3.5 \\ 2.8 \end{bmatrix}, \quad \mathbf{w}^* = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \eta = 0.2.$$

Match each contour plot to the trajectory that gradient descent produces on its loss by writing the trajectory number (1, 2, 3, or 4) in each answer box. Each trajectory is used exactly once.



Contour plot A:

Contour plot B:

Contour plot C:

Contour plot D:

(d) In practice, plain gradient descent is replaced by one of its variants. For each description, fill in the bubble of the optimizer it describes.

- (i) [1 Pt] This optimizer keeps a single learning rate η for every coordinate, but replaces the gradient in the update by an exponentially weighted sum of the past gradients.

☐ momentum ☐ RMSProp ☐ Adam ☐ AdamW

- (ii) [1 Pt] This optimizer applies weight decay by shrinking the parameters directly, $\mathbf{w} \leftarrow \mathbf{w} - \eta\lambda\mathbf{w}$, as a step separate from its gradient-based update, rather than adding $\lambda\mathbf{w}$ to the gradient.

☐ momentum ☐ RMSProp ☐ Adam ☐ AdamW

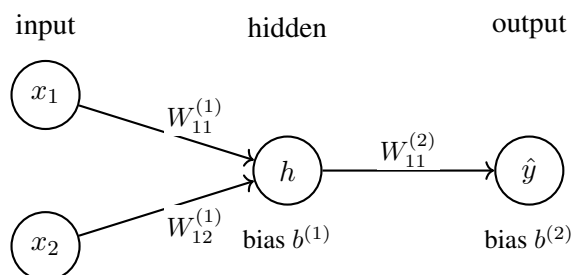
3 GooberNet: A 9-Parameter Model [8 points]

GooberNet is a small fully-connected neural network with one hidden layer. For an input $\mathbf{x} = [x_1 \ x_2]^\top$ it computes

$$\mathbf{z} = W^{(1)}\mathbf{x} + \mathbf{b}^{(1)}, \quad \mathbf{h} = \phi(\mathbf{z}), \quad u = W^{(2)}\mathbf{h} + b^{(2)}, \quad \hat{y} = \sigma(u),$$

where the hidden activation ϕ is applied elementwise and is specified in each part, and $\sigma(t) = \frac{1}{1+e^{-t}}$ is the logistic sigmoid.

- (a) [2 Pts] Suppose the hidden layer has a single unit with the identity activation $\phi(t) = t$, so $W^{(1)} \in \mathbb{R}^{1 \times 2}$, $b^{(1)} \in \mathbb{R}$, $W^{(2)} \in \mathbb{R}^{1 \times 1}$, and $b^{(2)} \in \mathbb{R}$.

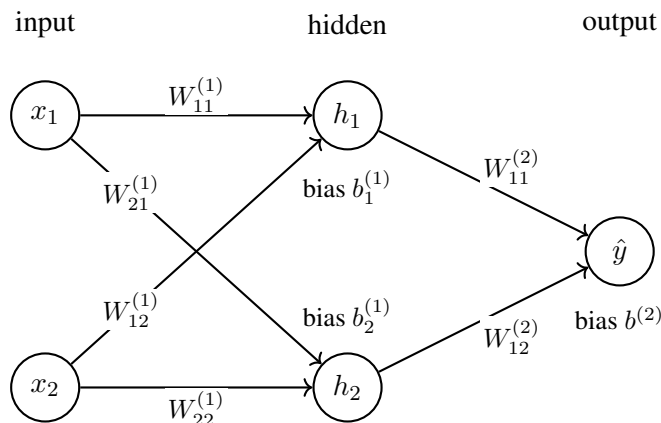


Let the inputs be bits, $x_1, x_2 \in \{0, 1\}$. Which parameter setting makes the network compute OR, i.e. $\hat{y} \geq \frac{1}{2}$ when at least one of x_1, x_2 is 1, and $\hat{y} < \frac{1}{2}$ when both are 0?

	$W_{11}^{(1)}$	$W_{12}^{(1)}$	$b^{(1)}$	$W_{11}^{(2)}$	$b^{(2)}$
☐	1	1	0	1	$-\frac{3}{2}$
☐	2	2	-1	1	0
☐	1	1	0	-1	$\frac{1}{2}$
☐	1	-1	0	1	$-\frac{1}{2}$

- (b) Now suppose the hidden layer has two units with the ReLU activation, $\phi(t) = \text{ReLU}(t) = \max(0, t)$, so the network computes

$$\mathbf{z} = W^{(1)}\mathbf{x} + \mathbf{b}^{(1)}, \quad \mathbf{h} = \text{ReLU}(\mathbf{z}), \quad u = W^{(2)}\mathbf{h} + b^{(2)}, \quad \hat{y} = \sigma(u).$$



We train with the binary cross-entropy loss

$$L = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})], \quad \text{for which} \quad \frac{\partial L}{\partial u} = \hat{y} - y.$$

- (i) [1 Pt] Let $W_{11}^{(1)}$ be the weight from x_1 to h_1 . By the chain rule, $\frac{\partial L}{\partial W_{11}^{(1)}}$ equals which of the following?

- ☐ $(\hat{y} - y) \cdot W_{11}^{(2)} \cdot \text{ReLU}'(z_1) \cdot x_2$
☐ $(\hat{y} - y) \cdot W_{11}^{(2)} \cdot x_1$
☐ $(\hat{y} - y) \cdot W_{11}^{(2)} \cdot \text{ReLU}'(z_1) \cdot x_1$
☐ $(\hat{y} - y) \cdot \sigma'(u) \cdot W_{11}^{(2)} \cdot \text{ReLU}'(z_1) \cdot x_1$
☐ $(\hat{y} - y) \cdot h_1$

- (ii) [2 Pts] This part asks about a different weight from part (i): $W_{11}^{(2)}$, the weight from h_1 to the output $\hat{y}$. On one training example with target $y = 1$, the forward pass gives $\mathbf{h} = [1 \ 0]^\top$ and $\hat{y} = \frac{1}{2}$. Compute $\frac{\partial L}{\partial W_{11}^{(2)}}$ on this example. Show your work in the box, then write your answer in the answer box.

$$\frac{\partial L}{\partial W_{11}^{(2)}} =$$

(c) Fill in exactly one bubble per item.

- (i) [1 Pt] The universal approximation theorem guarantees that if a one-hidden-layer network is sufficiently wide, gradient descent on the training loss will find parameters that approximate any continuous target function to any desired accuracy.
- ☐ True ☐ False
- (ii) [1 Pt] For a hidden unit with sigmoid activation $h = \sigma(z)$, if its pre-activation z is a large positive number, the gradient of the loss with respect to that unit's incoming weights is close to zero even when the loss itself is large.
- ☐ True ☐ False
- (iii) [1 Pt] GooberNet is repurposed for regression: the target y is now a real number rather than a bit. The network currently uses the output layer $\hat{y} = \sigma(u)$ with the binary cross-entropy loss. Which pair of (output layer, loss) should replace it?
- ☐ output layer: $\hat{y} = u$; loss: $(\hat{y} - y)^2$ (squared error)
- ☐ output layer: $\hat{y} = \sigma(u)$; loss: $(\hat{y} - y)^2$ (squared error)
- ☐ output layer: $\hat{y} = \text{ReLU}(u)$; loss: binary cross-entropy
- ☐ output layer: $\hat{y} = \sigma(u)$; loss: binary cross-entropy (no change needed)

4 Triple T [4 points]

You're flipping a coin with your good friend Tung. For each tail, Tung gets a point, and for each head, you get a point. However, you observe that the first three flips are TTT — slightly suspicious!

- (a) [1 Pt] You begin by assuming that the outcome of each coin flip is independent, and follows a Bernoulli distribution with parameter $\theta \in [0, 1]$, where θ is the probability of a head.

Recall the distribution of a Bernoulli random variable, where $x = 1$ denotes a head and $x = 0$ a tail:

$$\text{Bern}(x \mid \theta) = \theta^x (1 - \theta)^{1-x}.$$

Which of the following optimization problems correctly formulates the search for the Maximum Likelihood Estimate $\hat{\theta}_{\text{MLE}}$ for the observed sequence TTT ?

- ☐ $\arg \max_{\theta \in [0,1]} \left(3 \log(\theta) \right)$ ☐ $\arg \max_{\theta \in [0,1]} \left(3 \log(1 - \theta) + \log p(\theta) \right)$
- ☐ $\arg \max_{\theta \in [0,1]} \left(3 \log(1 - \theta) \right)$ ☐ $\arg \max_{\theta \in [0,1]} \left(3 \log(\theta(1 - \theta)) \right)$

- (b) [3 Pts] You begin to suspect that Tung is stalling on purpose to throw you off your rhythm. You instead focus on how long Tung takes to flip the coin each round. You model his waiting time T (in seconds) as an Exponential random variable $T \sim \text{Exponential}(\lambda)$ with rate parameter $\lambda > 0$ and PDF:

$$f(t \mid \lambda) = \lambda e^{-\lambda t}, \quad t \geq 0$$

You record M independent waiting times $\mathcal{D} = \{t_1, t_2, \dots, t_M\}$.

Write down the log-likelihood function $\ell(\lambda) = \log p(\mathcal{D} \mid \lambda)$ in terms of M , λ , and the t_i . Your final answer should **not** contain $\prod$, e , or $\exp$. Show your work in the box, then write your answer in the answer box.

$\ell(\lambda) =$

5 Set Your Boundaries [4 points]

Throughout this question, we consider binary logistic regression with classes C_0 and C_1 . For an input $\mathbf{x} \in \mathbb{R}^D$, the model predicts

$$P(C_1 | \mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w}^\top \mathbf{x} + w_0), \quad \sigma(a) = \frac{1}{1 + e^{-a}}.$$

We predict C_1 whenever

$$P(C_1 | \mathbf{x}, \mathbf{w}) \geq T,$$

where $T \in (0, 1)$ is a classification threshold.

- (a) [2 Pts] Suppose C_0 denotes a *legitimate credit card transaction* and C_1 a *fraudulent one*, so fraudulent is the positive class. A false positive incurs a loss of 1 and a false negative a loss of 100. Let $p = P(C_1 | \mathbf{x}, \mathbf{w})$ be the model's probability that $\mathbf{x}$ is fraudulent.

Under the model's distribution, write the expected loss at $\mathbf{x}$ if we predict C_0 and if we predict C_1 , in terms of p . Show your work in the box, then write your answers in the answer boxes.

Predict C_0 :

Predict C_1 :

- (b) [1 Pt] Suppose we increase the probability threshold T used to convert the outputs of a fixed logistic regression model into 0/1 predictions. We do not change the fitted model or the test data. As the threshold increases, recall

- | | |
|---|---|
| ☐ can only decrease. | ☐ can only increase or stay the same. |
| ☐ can only decrease or stay the same. | ☐ can move in either direction. |

- (c) [1 Pt] Which of the following metrics is not tied to one classification threshold?

- | | |
|---------------------------------|---|
| ☐ accuracy | ☐ recall |
| ☐ precision | ☐ Area under the curve of ROC |

6 K-Nearest Players [10.5 points]

The 2026–27 NBA season tips off today! A pandas DataFrame `players` holds one row per NBA player with the columns `player`, `position` (one of "G", "F", "C" for guard, forward, center), `height_in` (inches), and `weight_lb` (pounds). Throughout this question, we try to predict a player's position from height and weight alone. Assume `numpy` is imported as `np`, `pandas` as `pd`, and that `train_test_split`, `StandardScaler`, and `KNeighborsClassifier` are imported from `scikit-learn`.

- (a) A KNN classifier has predicted the position of every player in `players`, and the predictions are stored in a new column of `players` called `predicted_position`. A few rows of `players` are shown below.

<code>player</code>	<code>height_in</code>	<code>weight_lb</code>	<code>position</code>	<code>predicted_position</code>
Stephen Curry	74	185	G	G
Brandin Podziemski	76	205	G	G
Moses Moody	77	211	G	G
Jimmy Butler III	78	230	F	F
Draymond Green	78	230	F	F
Al Horford	80	240	C	F
...	...	...	...	...

- (i) [2 Pts] Write a single Python expression that computes the accuracy of the predictions in `predicted_position` over all rows of `players`, using only `players` and `pandas` and/or `NumPy` operations. Do **not** use `len`. Write your answer in the box.

- (ii) [1 Pt] The classifier is a `KNeighborsClassifier` with $k = 50$, and `knn.classes_` is `["C", "F", "G"]` (center, forward, guard). For Al Horford's row, `knn.predict_proba` returns `[0.20, 0.74, 0.06]`. Write a single number in each answer box.

The classifier's confidence in this prediction

The number of Horford's 50 nearest training neighbors that are **guards** ("G")

- (b) [3 Pts] Andrew wrote the pipeline below to choose k for a KNN position classifier and report its test accuracy. It runs without error and prints best $k = 1$, test accuracy = 0.786, and he is about to report that number. There are **three** major bugs. None of them is a syntax error, and each is fixed by rewriting a single line. For each bug, give the line number and the corrected version of that line.

```

1 # split into train, validation, and test
2 X = players[["height_in", "weight_lb"]].to_numpy()
3 y = players["position"].to_numpy()

4 split = train_test_split(X, y, test_size=0.2)
5 X_train, X_test, y_train, y_test = split

6 split = train_test_split(X, y, test_size=0.25)
7 X_train, X_val, y_train, y_val = split

8 # standardize
9 scaler = StandardScaler().fit(X)
10 X_train_sc = scaler.transform(X_train)
11 X_val_sc = scaler.transform(X_val)
12 X_test_sc = scaler.transform(X_test)

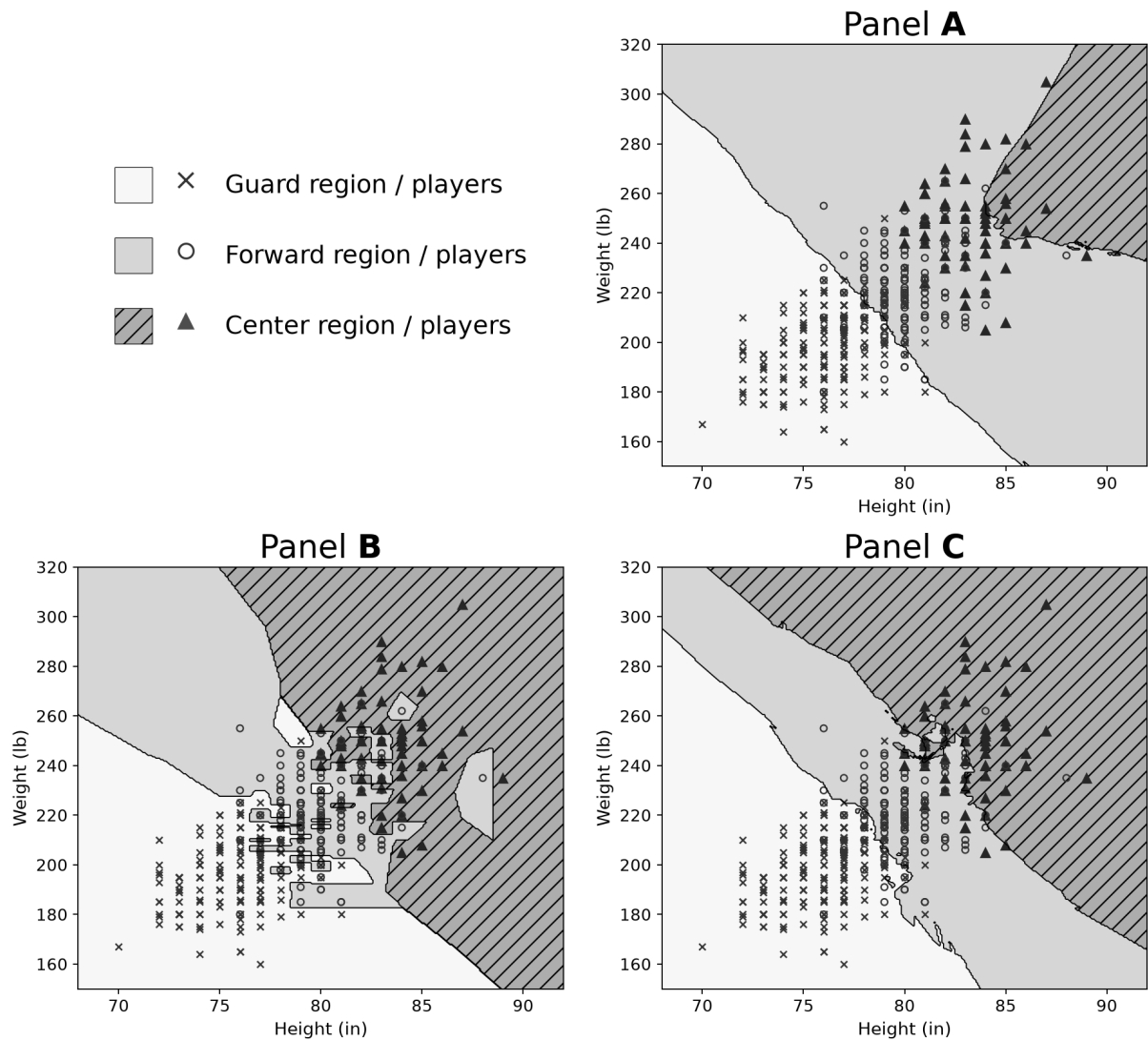
13 # choose k
14 best_k, best_acc = None, 0.0
15 for k in range(1, 101):
16     knn = KNeighborsClassifier(n_neighbors=k).fit(X_train_sc, y_train)
17     acc = knn.score(X_train_sc, y_train)
18     if acc > best_acc:
19         best_k, best_acc = k, acc

20 # final model
21 knn = KNeighborsClassifier(n_neighbors=best_k).fit(X_train_sc, y_train)
22 test_acc = knn.score(X_test_sc, y_test)
23 print(f"best k = {best_k}, test accuracy = {test_acc:.3f}")

```

Line #	Corrected line

- (c) [1.5 Pts] The three panels below show the decision regions of KNN classifiers trained on standardized height and weight for 446 players, with $k \in \{1, 15, 130\}$ in some order. Each marker ($\times$, $\circ$, $\blacktriangle$) is one of these players; the legend in the top-left corner shows each position's marker and region shading.



Match each panel to its k by writing the panel letter (A, B, or C) in each answer box. Each letter is used exactly once.

 $k = 1:$

 $k = 15:$

 $k = 130:$

- (d) [1 Pt] A standardized KNN classifier was fit on 446 players and evaluated once on the 112 held-out test players. Rows are true positions, columns are predicted positions.

True position	G	F	C
Predicted position	48	4	0
	3	38	2
	0	5	12

Write the precision and the recall of the classifier for the forward class (F), each as a fraction of two integers. You do **not** need to simplify the fractions.

Precision for F:

Recall for F:

- (e) Tall players tend to be heavy, so height and weight carry overlapping information. PCA on the standardized training data (z_{height} , z_{weight}) gives the principal components below.

	eigenvalue	unit eigenvector
first principal component	$\lambda_1 = 1.6$	$\mathbf{v}_1 = \frac{1}{\sqrt{2}}(1, 1)$
second principal component	$\lambda_2 = 0.4$	$\mathbf{v}_2 = \frac{1}{\sqrt{2}}(1, -1)$

- (i) [1 Pt] Two more players are listed below with their raw and standardized features. A player's coordinate along a principal component is the projection of their standardized features onto it. Which statement is correct?

	raw		standardized	
	height_in	weight_lb	z_{height}	z_{weight}
Nikola Jokić	83	284	1.3	2.7
Chet Holmgren	85	208	2.0	-0.3

- ☐ They have about the same coordinate along $\mathbf{v}_1$, since they are about the same height.
☐ Jokić's coordinate along $\mathbf{v}_2$ is larger than Holmgren's, since he is much heavier.
☐ Holmgren's coordinate along $\mathbf{v}_2$ is larger than Jokić's, since he is taller and lighter.
☐ Holmgren's coordinate along $\mathbf{v}_2$ is close to zero, since his standardized weight is close to zero.
- (ii) [1 Pt] What fraction of the total variance is kept by projecting onto the first principal component $\mathbf{v}_1$? Give your answer as a fraction or decimal. You do **not** need to simplify it.

Fraction of the total variance kept:

7 I Gauss We Are Parking Here? [5.5 points]

A wheeled robot with a fixed arm has to open an oven door. The arm only succeeds from certain base positions, so before reaching, the robot drives along a straight rail that runs parallel to the oven door and stops there (it *parks*). Erica models the good parking spots as a two-component Gaussian mixture over the position $x \in \mathbb{R}$ along the rail, in meters, with $x = 0$ directly in front of the oven (negative on one side, positive on the other).

Let $z \in \{1, 2\}$ be the latent (unobserved) mixture component, i.e. the cluster, that a parking spot comes from, and let $\theta = \{\pi_1, \pi_2, \mu_1, \mu_2\}$ be the mixture's parameters. Both components share a known variance σ^2 :

$$p(z = k) = \pi_k, \quad p(x \mid z = k) = \mathcal{N}(x; \mu_k, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x - \mu_k)^2}{2\sigma^2}\right).$$

- (a) [2 Pts] Find the posterior probability that a parking spot at position x came from component k , $p(z = k \mid x, \theta)$. You may leave Gaussian densities in $\mathcal{N}(\cdot; \cdot, \cdot)$ notation. You do **not** need to write out or simplify the density formula. Show your work in the box.

(b) [2.5 Pts] Erica fits θ from four logged positions at which the arm succeeded,

$$x_1 = -2, \quad x_2 = -1, \quad x_3 = 1, \quad x_4 = 2,$$

and runs EM with $\sigma^2 = 1$ held fixed, initialized at $\pi_1 = \pi_2 = \frac{1}{2}$, $\mu_1 = -0.5$, $\mu_2 = 0.5$.

Rounded to one decimal place, the first E-step gives the responsibilities

$$\gamma_{nk} = p(z_n = k \mid x_n, \theta)$$

$$\gamma_{11} = 0.9, \quad \gamma_{21} = 0.7, \quad \gamma_{31} = 0.3, \quad \gamma_{41} = 0.1$$

for component 1 (and $\gamma_{n2} = 1 - \gamma_{n1}$ for component 2). Use these rounded values. The M-step sets $N_k = \sum_n \gamma_{nk}$, $\pi_k = N_k/4$, and $\mu_k = \frac{1}{N_k} \sum_n \gamma_{nk} x_n$. Compute π_1 and μ_1 after this M-step. Show your work in the box, then write your answers in the answer boxes.

$\pi_1 =$

$\mu_1 =$

(c) [1 Pt] Erica considers using K-means (Lloyd's algorithm) instead of EM. Which of the following is true of K-means on a general dataset (not just the four positions above)?

- ☐ Each iteration minimizes the objective $\sum_n \|x_n - \mu_{z_n}\|^2$ over one set of variables with the other held fixed: first the assignments z with the centers fixed, then the centers μ with the assignments fixed. The objective therefore never increases.
- ☐ Like the E-step of EM, the assignment step gives each point a probability of belonging to each cluster, and the center update weights the points by those probabilities.
- ☐ Because the objective never increases, Lloyd's algorithm converges to the global minimum of the objective from any initialization.
- ☐ The number of clusters can be chosen by running K-means for several values of K and keeping the K with the smallest objective.

You are done with the midterm! Congratulations!

Use this page to draw your favorite CS 189/289A moment!

A large, empty rectangular box with a thin black border, intended for a student to draw their favorite CS 189/289A moment.