CS 189/289A, Practice Midterm

Fall 2025

Name: _____

Email: _____@berkeley.edu

Student ID: _____

Name and SID of the person on your right: _____

Name and SID of the person on your left: _____

Instructions:

This exam consists of **X points** spread out over **X questions** and the Honor Code certification. The exam must be completed in **110 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.

For all Python questions, you may assume `Pandas` has been imported as `pd`, `NumPy` has been imported as `np`, and `Plotly Express` has been imported as `px`.

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: _____

1 Pandas Express [Pandas] *(Sp24 Data 100 Midterm)*

All code for each part must be written in Python. Assume that the `pandas` library has been imported as `pd`, the `numpy` library has been imported as `np`, and the Python RegEx library has been imported as `re`.

~~Milad~~ **Ozan** is a big fan of the ~~data-science~~ **machine-learning**-themed restaurant, Pandas Express. He tracks ~~Data 100~~ **CS189/289A** staff members' orders at the restaurant in a `DataFrame` called `orders`. Descriptions of its columns and its first five rows can be found below:

- `name`: Name of a Data 100 staff member (type = `str`).
- `item`: The food item they ordered (type = `str`).
- `size`: The portion size of the item - small, medium, or large (type = `str`).
- `veg`: 0 if the item is not vegetarian, 1 if the item is vegetarian (type = `numpy.int64`).
- `premium`: 0 if the item is not a premium item, 1 if the item is a premium item (type = `numpy.int64`).
- `price`: The price of the order in dollars (type = `numpy.float64`).

	name	item	size	veg	premium	price
0	Sean	Orange Chicken	large	0	0	11.4
1	Brandon	Chow Mein	medium	1	0	8.7
2	Brandon	Eggplant Tofu	small	1	0	5.4
3	Jessica	Honey Walnut Shrimp	medium	0	1	11.7
4	Sean	Broccoli Beef	small	0	0	5.4

(a) [2 Pts] Which of the following lines of code will output a `DataFrame` only containing rows with dishes that are both vegetarian **AND** have a `price` above $7.50? **Select all that apply.**

☐ `orders[(orders["veg"]==1) and (orders["price"]>7.5)]`

☐ `orders.loc[(orders["veg"]==1) & (orders["price"]>7.5)]`

☐ `orders[orders.isin(["veg" == 1, "price" > 7.5])]`

☐ `orders[(orders["veg"]==1) | (orders["price"]>7.5)]`

Solution: Option A is incorrect because you cannot use the word `and` to perform multi-condition filtering.

Option B is the correct way to perform a filter that satisfies multiple conditions.

Option C is an incorrect use of `.isin()` and would throw an error.

Option D is incorrect, as it only filters for rows that satisfy one of the specified conditions.

(b) [2 Pts] Fill in the blanks below to create `above_5`: a `DataFrame` with the same structure as `orders` but only containing rows that include an `item` that was ordered more than 5 times.

```
above_5 = orders.___A___(___B___).___C___(lambda sf: _____D_____)
```

(i) Fill in blank `A`:

Solution: `groupby`

(ii) Fill in blank `B`:

Solution: `"item"`

(iii) Fill in blank `C`:

Solution: `filter`

(iv) Fill in blank `D`:

Solution: `len(sf) > 5` or `sf.shape[0] > 5`

(c) [2 Pts] Fill in the blanks below to create `main_revenue`: a `Series` that displays the total money that each `main` generated across all orders. Assume that the `main` column was created correctly.

```
main_revenue = orders.____A____(___B___)[____C____].____D____
```

(i) Fill in blank `A`:

Solution: `groupby`

(ii) Fill in blank `B`:

Solution: `"main"`

(iii) Fill in blank `C`:

Solution: `"price"`

(iv) Fill in blank `D`:

Solution: `sum()`, `agg(sum)`, `agg("sum")`, or `agg(np.sum)`

(d) [2 Pts] Using `main_revenue` from the previous part, fill in the blanks below to find the `main` which generated the **median** amount of money across all orders. Assume that there are an odd number of entries in `main_revenue`.

Hint: `a // b` returns `a` divided by `b` rounded down to the nearest integer.

```
position = _________________A_________________ // 2

main_revenue._____B_____().______C_____[_____D_____]
```

(i) Fill in blank `A`:

Solution: `len(main_revenue)` or `main_revenue.shape[0]` or any code that gets the length of `main_revenue`.

(ii) Fill in blank `B`:

Solution: `sort_values`

(iii) Fill in blank `C`:

Solution: `iloc` or `index`
Note: the wording of this question caused some confusion as to whether students were supposed to get the actual median money amount, or its corresponding `main`. As a result, both options were accepted.

(iv) Fill in blank `D`:

Solution: `position`

(e) [2 Pts] Milad creates two new `DataFrame` objects: `normal` and `premium`, which contain normal and premium prices for each size, respectively.

	size	normal_price
0	small	5.4
1	medium	8.7
2	large	11.4

`normal`

	size	premium_price
0	large	11.0
1	medium	11.7
2	small	6.9

`premium`

Fill in the blanks to create a `DataFrame` called `prices` (shown below), which includes a new column `diff` displaying the `absolute` difference between each size.

	size	normal_price	premium_price	diff
0	small	5.4	6.9	1.5
1	medium	8.7	11.7	3.0
2	large	11.4	11.0	0.4

```
prices = normal._____A_____(_____________B_____________)
prices["diff"] = _____________________C_____________________
```

(i) Fill in blank `A`:

Solution: `merge`

(ii) Fill in blank `B`:

Solution: `right=premium, left_on="size", right_on="size"` or any combination of arguments that correctly achieves this merge.

(iii) Fill in blank `C`:

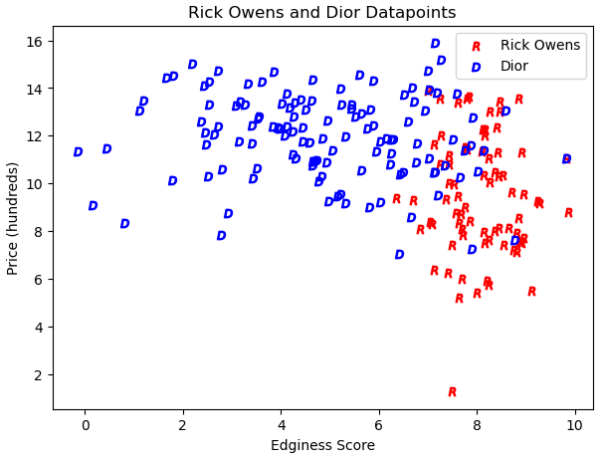
Solution:
```
abs(prices["premium_price"] - prices["normal_price"])
```

2 Idk what K means [K-means & GMM]

You work at a luxury consignment store called *GGRAILED*, where customers frequently bring in mysterious unlabeled jeans that need to be authenticated as either **Rick Owens** or **Dior**. Your boss, tired of eyeballing denim all day, decides to use machine learning to cluster jeans based on:

- **edginess score** (x_1): An objective measure of avant-garde-ness.
- **price** (x_2): The price in dollars.

You have a dataset $\mathcal{D} = \{\mathbf{x}_n, z_n\}_{n=1}^N$ consisting of N training examples, where each datapoint $\mathbf{x}_n = \begin{bmatrix} x_{n1} \\ x_{n2} \end{bmatrix}$, and each label $z_n = \begin{cases} 0 & \text{if the jeans corresponding to } \mathbf{x}_n \text{ are } \textbf{Rick Owens} \\ 1 & \text{if the jeans corresponding to } \mathbf{x}_n \text{ are } \textbf{Dior} \end{cases}$.



Your boss starts with the classic K-means algorithm, hoping to find two clusters corresponding to the two brands.

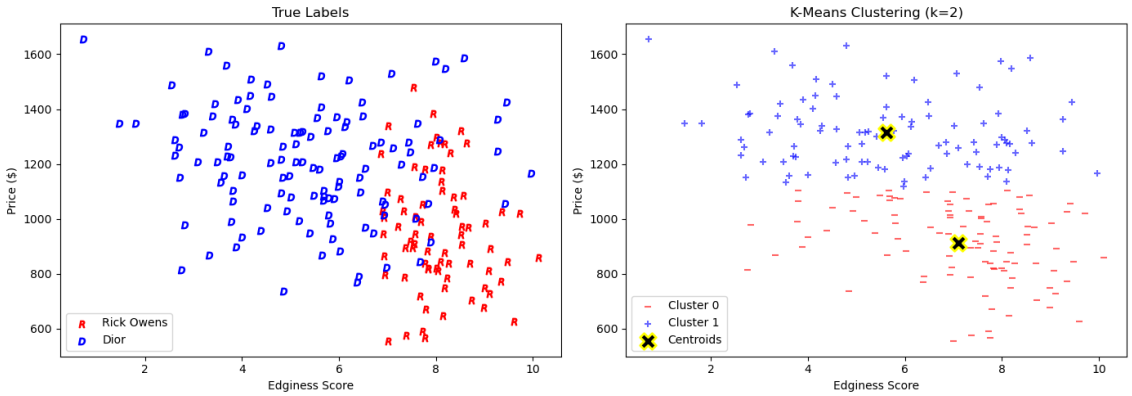
(a) [2 Pts] Select True or False for the following statements about K-Means algorithm.

True	False	Statement
		The K-means algorithm will always converge to a solution.
		The K-means algorithm will only find the global optimum.
		The K-means algorithm will only find a single unique solution.
		The K-means algorithm will always find cluster assignments that makes the objective function zero.

Solution:

True	False	Statement
X		The K-means algorithm will always converge to a solution.
	X	The K-means algorithm will only find the global optimum.
	X	The K-means algorithm will only find a single unique solution.
	X	The K-means algorithm will always find cluster assignments that makes the objective function zero.

(b) Your boss tried to run K-means with $K = 2$ but got questionable results:



(i) [2 Pts] Briefly describe what most likely caused this problem. Hint: Recall the objective function that K-means aims to minimize.

Solution:
The scales of the features are wildly different, so the distance between points is dominated by the price dimension, and the edginess score is essentially ignored.

(ii) [2 Pts] Briefly describe how this problem can be fixed.

Solution:
We can normalize the features to have the same scale.

After scrolling on TokTik for a bit, you have an epiphany: there might actually be a third class of jeans, **Hedi Slimane-era Dior**, that you must consider as a separate class. With this new insight, you decide to set $K = 3$ and re-fit the K-means model to your dataset.

(c) [2 Pts] Recall the objective function for K-means clustering:

$$J = \sum_{k=1}^{K} \sum_{n=1}^{N} r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

(i) How will J most likely change when increasing K from 2 to 3?

◯ Increase
◯ Stay the same
◯ **Decrease**

(ii) Justify your answer to part (i). Points will not be awarded for justifying an incorrect answer to (i).

Solution: Increasing K from 2 to 3 gives the K-means algorithm more clusters, which means it can assign each point to a closer centroid and reduce the distance between points and their assigned centroids.

(d) [2 Pts] Recall the definition of Bias^2 for a model f:

$$\text{Bias}^2 = \mathbb{E}[f(\mathbf{x})] - h(\mathbf{x})$$

(i) What will most likely happen to the K-means clustering model's bias when increasing K from 2 to 3?

◯ Increase
◯ Stay the same
◯ **Decrease**

(ii) Justify your answer to part (i).

Solution: Increasing the number of clusters makes the K-means model more flexible and better able to capture the underlying structure of the data. This reduces model bias, since the model is less likely to underfit by forcing points from different true clusters into the same group.

(e) [2 Pts] Recall the definition of Variance for a model f:

$$\text{Variance} = \mathbb{E}\left[(f_{\mathbf{w}}(\mathbf{x}) - \mathbb{E}[f_{\mathbf{w}}(\mathbf{x})])^2\right]$$

(i) What will most likely happen to the K-means clustering model's variance when increasing K from 2 to 3?

◯ **Increase**
◯ Stay the same
◯ Decrease

(ii) Justify your answer to part (i).

Solution: Increasing K from 2 to 3 makes the K-means model more flexible and allows it to fit the data more closely. However, this also means the model's output will be more sensitive to small changes in the data, increasing the spread of possible clusterings it can produce on different datasets.

Your boss talks to your boss's boss (a machine-learning expert) who says that you should use GMM instead of K-means, and that you should only use two clusters.

(f) Compared to K-means "hard" assignments $r_{nk} \in \{0, 1\}$, Expectation-Maximization for Gaussian Mixture Models uses responsibilities γ_{nk} that:

True	False	Statement
		Are in $\{0, 1\}$.
		Lie in $[0, 1]$.
		Sum to 1 across all components K.
		Sum to K across all data points K.

Solution:

True	False	Statement
	X	Are in $\{0, 1\}$.
X		Lie in $[0, 1]$.
X		Sum to 1 across all components K.
	X	Sum to K across all data points K.

(g) If Dior jeans are far more common than Rick Owens in the training set, which statement is most likely to be true?

◯ $\pi_R > \pi_D$

◯ $\pi_R < \pi_D$

◯ $\pi_R = \pi_D$

Suppose we fit a GMM model with $K = 2$ components to our dataset. We obtain the following parameters:

- **Component 1 (Rick Owens)**: $\boldsymbol{\mu}_R$, $\boldsymbol{\Sigma}_R$, π_R
- **Component 2 (Dior)**: $\boldsymbol{\mu}_D$, $\boldsymbol{\Sigma}_D$, π_D

(h) [2 Pts] Write the probability of a datapoint $p(\mathbf{x})$ under our Gaussian Mixture Model.

Solution:

$$\begin{aligned} p(\mathbf{x}) &= \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k) \\ &= \pi_R \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_R, \boldsymbol{\Sigma}_R) + \pi_D \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_D, \boldsymbol{\Sigma}_D) \end{aligned}$$

(i) [2 Pts] Given an unlabeled datapoint $\mathbf{x}$, what is the probability our model assigns to $\mathbf{x}$ belonging to the **Rick Owens** component? Hint: what do γ_{nk} represent?

Solution:

$$\begin{aligned} p(\mathbf{x} \text{ is Rick Owens}) &= \frac{p(\mathbf{x} \mid \mathbf{x} \text{ is } \textbf{Rick Owens})}{p(\mathbf{x} \mid \mathbf{x} \text{ is } \textbf{Rick Owens}) + p(\mathbf{x} \mid \mathbf{x} \text{ is } \textbf{Dior})} \\ &= \frac{\pi_R \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_R, \boldsymbol{\Sigma}_R)}{\pi_R \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_R, \boldsymbol{\Sigma}_R) + \pi_D \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_D, \boldsymbol{\Sigma}_D)} \end{aligned}$$

3 Let's get lifted [Linear Regression, Coding]

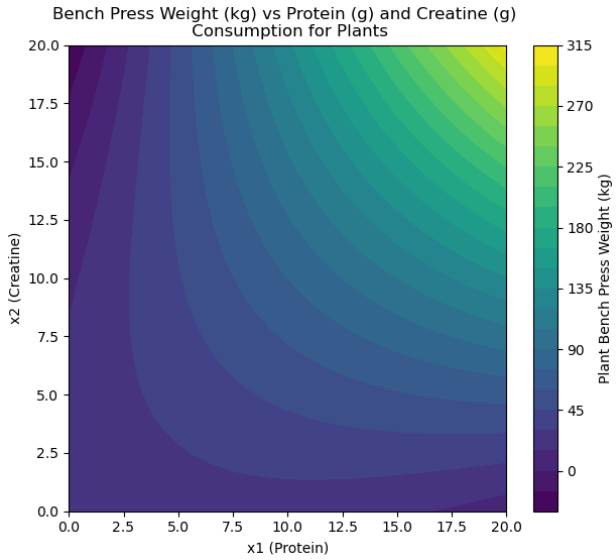
You've spent too much time mirin' Zyzz on Twitter (formerly X) and now you've decided to test whether plants can, too, get *ripped* by feeding them protein powder and creatine. As one does, you conduct a series of experiments where you administer varying quantities of protein and creatine to plants and measure how much they can bench press and whether they became *ripped*.

$$\mathbf{x} = \begin{bmatrix} x_1 & \text{(grams of protein mixed into the soil)} \\ x_2 & \text{(grams of creatine added to the water)} \end{bmatrix}$$

$$t_b \in \mathbb{R} \quad \text{the bench press weight of plant in kg}$$

$$t_s = \begin{cases} 1 & \text{if plant became } \textit{ripped} \\ 0 & \text{otherwise} \end{cases}$$

(a) [2 Pts] Suppose the true relationship between $\mathbf{t}_b$ and $\mathbf{x}$ is given by a degree-2 polynomial. Give a suitable basis function $\boldsymbol{\phi}(\mathbf{x})$ for this setting..



Solution: $\boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} 1 & x_1 & x_2 & x_1^2 & x_2^2 & x_1 x_2 \end{bmatrix}^\top$

Note: as long as the answer contains a second-degree term, the basis function could be considered "suitable", but most complete answer would have all the fields provided above.

You collected all your data into a `pd.DataFrame` that you affectionately named `df`,

	Protein (g)	Creatine (g)	Bench Press (kg)	Ripped
0	7.474747	5.252525	43.032064	False
1	3.838384	0.404040	21.945236	False
2	15.151515	6.666667	73.779920	False
3	16.969697	14.343434	170.846651	True
4	17.373737	1.616162	28.285571	False

you've also chosen a a suitable basis function $\phi(\mathbf{x})$, and implemented it in Python.

```
def basis_function(row):
    ...implementation omitted...
```

You now want to perform linear regression to fit a model of the form $y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^\top \phi(\mathbf{x})$.

(b) [7 Pts] The code below attempts to fit a linear model, but some lines are buggy. Identify every buggy line and write the corrected version. Assume `df` and `basis_function` are defined.

```
  X = df[['Creatine (g)', 'Protein (g)']]
  y = df['Bench Press (kg)']
1 X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=67)
2 phi_X_train = X_train.apply(basis_function, axis=1)
3 phi_X_test = X_test.apply(basis_function, axis=1)
4 model = LogisticRegression()
5 model.fit(X_train, y_train)
6 y_test_pred = model.predict(phi_X_test)
7 test_mse = np.mean((y_test_pred - y)**2)
```

Line	Buggy? (Yes/No)	Corrected line
1		
2		
3		
4		
5		
6		
7		

Solution:

Line	Buggy? (Yes/No)	Corrected line
1	No	
2	Yes	`phi_X_train = X_train.apply(basis_function)`
3	No	
4	Yes	`model = LinearRegression()`
5	Yes	`model.fit(phi_X_train, y_train)`
6	No	
7	Yes	`test_mse = np.mean((y_test_pred - y_test)**2)`

(c) *(Taken from a previous CS189 Exam)* Let $\mathbf{w}^\star$ be the solution you obtain in standard least-squares linear regression on inputs $\mathbf{\Phi}$ and labels $\mathbf{t}_b$.

If you scale all input features (but not the labels) by a factor of $\alpha > 0$ before doing the regression, (i.e., use $\tilde{\mathbf{\Phi}} = \alpha\,\mathbf{\Phi}$ and perform least squares to find $\tilde{\mathbf{w}}$), what solution $\tilde{\mathbf{w}}$ do you obtain in terms of $\mathbf{w}^\star$ and α? Assume $(\mathbf{\Phi}^\top\mathbf{\Phi})$ is invertible.

Solution: Plugging in $\alpha\mathbf{\Phi}$ into the normal equations, we get:

$$\begin{aligned}(\alpha\mathbf{\Phi})^\top(\alpha\mathbf{\Phi})\tilde{\mathbf{w}} &= (\alpha\mathbf{\Phi})^\top\mathbf{t}_b\\ \alpha^2\mathbf{\Phi}^\top\mathbf{\Phi}\tilde{\mathbf{w}} &= \mathbf{\Phi}^\top\mathbf{t}_b\\ \tilde{\mathbf{w}} &= \frac{1}{\alpha}\left(\mathbf{\Phi}^\top\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^\top\mathbf{t}_b\\ \tilde{\mathbf{w}} &= \frac{1}{\alpha}\mathbf{w}^\star\end{aligned}$$

A fun problem that might seem familiar...

Some measurements are more reliable than others (e.g., differing noise levels). We will derive weighted least squares to account for this.

Suppose we have nonnegative weights $\{\alpha_n\}_{n=1}^N$ for each observation and our goal is to

minimize the weighted least squares objective:

$$L(\mathbf{w}) \;=\; \sum_{n=1}^{N} \alpha_n (t_n - \mathbf{x}_n^\top \mathbf{w})^2.$$

(d) Write the weighted least-squares loss function $L(w)$ in matrix form using the diagonal matrix:

$$A = \mathrm{diag}(\alpha_1, \ldots, \alpha_N) \in \mathbb{R}^{N \times N}.$$

Your answer should not contain any summations.

Solution:

$$L(\mathbf{w}) \;=\; \sum_{n=1}^{N} \alpha_n (t_n - \mathbf{x}_n^\top \mathbf{w})^2 \;=\; \alpha_1 (t_1 - \mathbf{x}_1^\top \mathbf{w})^2 + \cdots + \alpha_N (t_N - \mathbf{x}_N^\top w)^2.$$

Let $A = \mathrm{diag}(\alpha_1, \ldots, \alpha_N)$. Then

$$L(w) \;=\; (t - \mathbf{\Phi} w)^\top A \, (t - \mathbf{\Phi} w).$$

(e) Compute the gradient of the loss and express your answer in matrix form (no summations). You can do this by working with the summation form and taking partial derivatives or by using the following gradient identities:

$$\nabla_w \left(u^\top w \right) = u$$
$$\nabla_w \left(w^\top A w \right) = (A + A^\top) w$$

Solution: s

$$L(w) = t^\top A \, t - 2 w^\top \mathbf{\Phi}^\top A \, t + w^\top \mathbf{\Phi}^\top A \, \mathbf{\Phi} \, w.$$

Taking the gradient with respect to w and using $A = A^\top$:

$$\nabla_w L(w) = -2 \, \mathbf{\Phi}^\top A \, t + 2 \, \mathbf{\Phi}^\top A \, \mathbf{\Phi} \, w.$$

(f) [3 Pts] Set the gradient equal to zero, and solve for the closed-form solution for w^* that minimizes the weighted least squares objective. in terms of $\mathbf{\Phi}$, t, and A, assuming $\mathbf{\Phi}^\top A \, \mathbf{\Phi}$ is invertible.

Solution:

$$\mathbf{\Phi}^\top A \, \mathbf{\Phi} \, w^* = \mathbf{\Phi}^\top A \, t, \quad w^* = (\mathbf{\Phi}^\top A \, \mathbf{\Phi})^{-1} \mathbf{\Phi}^\top A \, t.$$

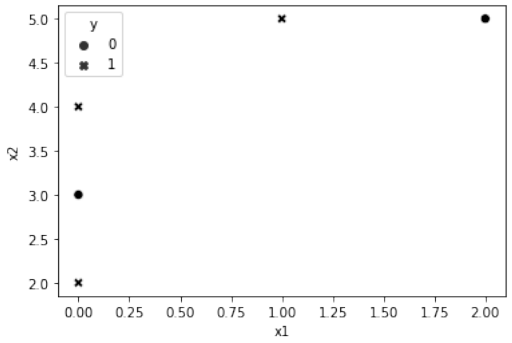
4 Catching Telce [Logistic Regression] *(Sp24 Data 100 Final)*

Shiny notices that professional football player Kravis Telce plays much better when superstar singer Saylor Twift is at his game. She collects data from a handful of games this past season, assigning each game a target of $t_n = 1$ if Saylor Twift was in attendance and $t_n = 0$ if not.

$\mathbf{X}_{:,1}$	$\mathbf{X}_{:,2}$	t
0	3	0
0	4	1
1	5	1
2	5	0
0	2	1

(a) [2 Pts] What is the maximum accuracy a logistic regression model can achieve for this dataset? Accuracy = $\boxed{}$

Solution: 0.8. A plot of the data can be seen below:



As you can see, this is not a linearly separable dataset, so perfect accuracy cannot be achieved. However, we can only get one incorrect classification (the lower left datapoint) with a linear boundary, giving us an accuracy of 4/5.

(b) [4 Pts] Shiny trains a logistic regression model with an intercept term and finds the optimal model parameters to be $\mathbf{w} = \begin{bmatrix}-3 & 1 & \frac{1}{2}\end{bmatrix}^\top$. The next week, Shiny records the data point $\mathbf{x}_{\text{new}} = \begin{bmatrix}x_{\text{new},1} & x_{\text{new},2}\end{bmatrix}^\top = \begin{bmatrix}1 & 4\end{bmatrix}^\top$ and the augmented input $\tilde{\mathbf{x}}_{\text{new}} = \begin{bmatrix}1 & \mathbf{x}_{\text{new}}\end{bmatrix}^\top = \begin{bmatrix}1 & 1 & 4\end{bmatrix}^\top$.

(i) What is the probability that Saylor Twift was in attendance at the game corresponding to $\mathbf{x}_{\text{new}}$?

Probability = _____

Solution:

$$y(\mathbf{x}_{\text{new}}) = p(t = 1 \mid \mathbf{x}_{\text{new}}) = \sigma\big(\mathbf{w}^T \mathbf{x}_{\text{new}}\big) = \sigma(0) = \frac{1}{1 + e^0} = \tfrac{1}{2}$$

(ii) If Saylor Twift attended the corresponding game, what is the cross-entropy loss incurred by the model on $\mathbf{x}_{\text{new}}$?

Loss = _____

Solution: We plug our answer from part (i) into the cross-entropy loss formula:

$$-\big[t_{\text{new}} \ln y(\mathbf{x}_{\text{new}}) + (1 - t_{\text{new}}) \ln\big(1 - y(\mathbf{x}_{\text{new}})\big)\big]$$

$$= -\left[(1) \ln\left(\tfrac{1}{2}\right) + (0) \ln\left(1 - \tfrac{1}{2}\right)\right]$$

$$= -\ln\left(\tfrac{1}{2}\right)$$

(c) [5 Pts] The table below shows a sample of validation data and predictions.

(i) What is the maximum possible recall attainable while maintaining an accuracy above 0.75? What is the range of classification thresholds that achieves this?

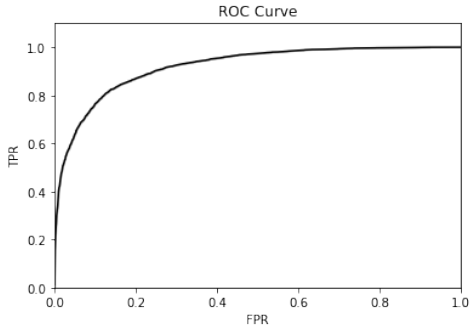
t_n	$p(t = 1|\mathbf{x}_{\text{new}})$
1	0.3
0	0.2
0	0.7
0	0.4
1	0.8

Recall = _____; Threshold Range = (_____, _____]

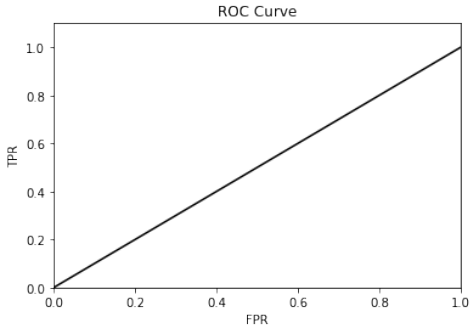
Solution: Recall is high if there are few false negatives and more true positives. A maximum recall of 1 can be achieved with a threshold under 0.3 (due to the 1-labeled point with a probability of 0.3), but this falls short of the 0.75 accuracy requirement (it gets 0.6). As such, **the maximum possible recall attainable is 0.5**, because we must have one False Negative (at 0.3) and one True Positive (at 0.8). This can be achieved with a threshold in the range (0.3, 0.8], but to satisfy the 0.75 accuracy requirement, we require **the range to be (0.7, 0.8].**

(ii) Suppose we have 4 classification threshold values: 0, 0.25, 0.5, and 0.75. Which ROC curve was generated by these thresholds on the above validation set?

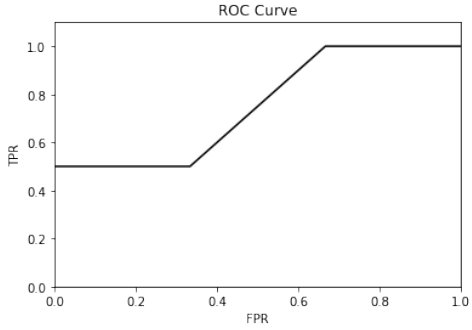
A.



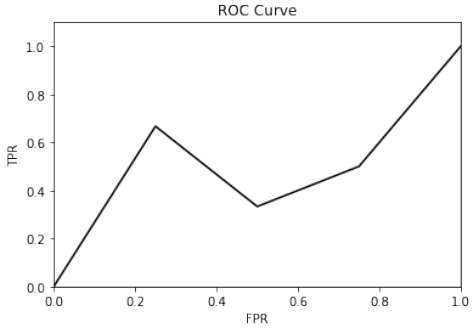
B.



C.



D.



◯ A
◯ **C**

◯ B
◯ D

Solution: These thresholds yield (FPR, TPR) combinations of $(1, 1)$, $(\frac{2}{3}, 1)$, $(\frac{1}{3}, \frac{1}{2})$, and $(0, \frac{1}{2})$.

5 Decisions, Decisions [Logistic Regression] *(Sp23 Data 100 Final)*

Consider the following binary classification problem with 4 data points. Suppose that we use a logistic regression model to predict the probability that $g = 1$ given $\mathbf{x}$:

$$y(\mathbf{x}; \mathbf{w}) = p_{\mathbf{w}}(t = 1|\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x})$$

x_1	x_2	t
0	1	1
1	0	0
-1	0	1
0	-1	0

(a) [1 Pt] After fitting a logistic regression model (without regularization) on features x_1, x_2 without an intercept and class labels y, we find that the predicted probabilities returned by sklearn are given below.

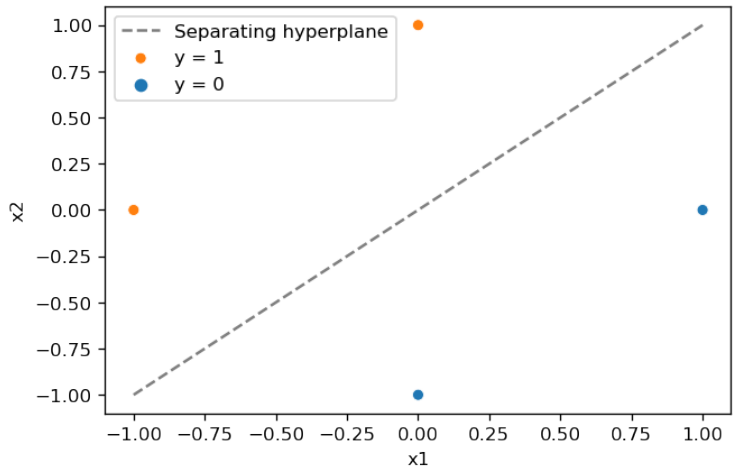
```
array([[2.69411751e-05, 9.99973059e-01],
       [9.99973059e-01, 2.69411751e-05],
       [2.69411751e-05, 9.99973059e-01],
       [9.99973059e-01, 2.69411751e-05]])
```

Is this data linearly separable?

◯ Yes

◯ No

Solution: A dataset is linearly separable if a line can be drawn in terms of the features x_i that perfectly separates the two classes y. A sketch of the data shows that this dataset is linearly separable:



(b) [2 Pts] Given your answer to part (a), what are weights $\mathbf{w} = \begin{bmatrix} w_1 & w_2 \end{bmatrix}^\top$ that minimize mean cross entropy loss? Assume no regularization.

$w_1 =$ [ ]

$w_2 =$ [ ]

Solution: When a dataset is linearly separable, the weights of an unregularized logistic regression model diverge to infinity. To determine the appropriate signs, consider the provided datapoints.

When $x_1 = 1, x_2 = 0$, $y(\mathbf{x}; \mathbf{w}) = \frac{1}{1+e^{-w_1(1)-w_2(0)}} = \frac{1}{1+e^{-w_1}}$ should approach $y = 0$.

This occurs as $w_1 \to -\infty$

When $x_1 = 0, x_2 = 1$, $y(\mathbf{x}; \mathbf{w}) = \frac{1}{1+e^{-w_1(0)-w_2(1)}} = \frac{1}{1+e^{-w_2}}$ should approach $y = 1$.

This occurs as $w_2 \to \infty$.

(c) [2 Pts] We train a new logistic regression model (without an intercept) with regularization and find the optimal model parameters to be $\mathbf{w} = \begin{bmatrix} -\frac{2}{3} & \frac{2}{3} \end{bmatrix}^\top$.

Suppose that we observe a new data point $\mathbf{x}_{\text{new}} = \begin{bmatrix} x_{\text{new},1} & x_{\text{new},2} \end{bmatrix}^\top = \begin{bmatrix} 2 & 1 \end{bmatrix}^\top$. Calculate the probability that our model believes $\mathbf{x}_{\text{new}}$ belongs to class 0.

Note: You may leave your final answer as an expression in terms of e.

Answer: _____

Solution:

$$\begin{aligned} y(\mathbf{x}; \mathbf{w}) &= P_{\mathbf{w}}(t_{\text{new}} = 1 | \mathbf{x}_{\text{new}}) \\ &= \sigma\Big(\begin{bmatrix} 2 & 1 \end{bmatrix} \cdot \begin{bmatrix} -\frac{2}{3} \\ \frac{2}{3} \end{bmatrix} \Big) \\ &= \sigma\Big(\frac{2}{3}\Big) \\ &= \frac{1}{1 + e^{\frac{2}{3}}} \end{aligned}$$

$$P_{\mathbf{w}}\left(t_{\text{new}} = 0 | \mathbf{w}_{\text{new}}\right) = 1 - \frac{1}{1 + e^{\frac{2}{3}}}$$

(d) [2 Pts] We decide to use a new dataset of 6 training points. For the remainder of this question, we will **only consider these 6 new training points**. We train a new logistic regression model to find the model's predicted probabilities, displayed in the table below.

Data Point #	x_1	x_2	t	$y(\mathbf{x}; \mathbf{w})$
1	1	1	1	0.65
2	0.5	2.5	1	0.90
3	0.75	0.5	1	0.75
4	0	0	1	0.85
5	0.5	1.5	0	0.70
6	1	0	0	0.45

Which data point incurs the highest cross-entropy loss?

Hint: You may find that computing cross-entropy loss is not required to answer this question correctly.

◯ Point 1 ◯ Point 3 ◯ **Point 5**

◯ Point 2 ◯ Point 4 ◯ Point 6

Solution: The prediction $\hat{y} = 0.70$ for Point 5 is furthest from the corresponding observed value $y = 0$.

(e) [2 Pts] What are the range(s) of classification thresholds T that maximize(s) accuracy? **Select all that apply**.

Note: Interval notation **(**a**,** b**]** refers to the range of integers from (but not including) a up to and including b.

☐ [0, .45] ☐ (.65, .70] ☐ (.75, .85]

☐ **(.45, .65]** ☐ **(.70, .75]** ☐ (.85, .90]

Solution: Any threshold between .45 and .65 results in 4 true positives and 1 true negative. Any threshold between .70 and .75 results in 3 true positives and 2 true negatives. In either case, the accuracy of the model is $\frac{TP+TN}{TP+TN+FP+FN} = \frac{5}{6}$

(f) [2 Pts] Suppose we are interested in evaluating our model's performance with an ROC curve.

To construct an ROC curve, we first need to find our model's TPR and FPR at various classification thresholds T. The TPR and FPR for the majority of thresholds have been computed for you in the table below; please select the correct values for the remaining entries.

Note: Multiple letters may correspond to the same value.

T	TPR	FPR
0.95	0	0
0.9	1/4	0
0.85	1/2	0
0.75	A	B
0.7	C	D
0.65	1	0.5
0.45	1	1

For your convenience, the data table from **part (d)** is repeated below.

Data Point #	x_1	x_2	t	$y(\mathbf{x};\mathbf{w})$
1	1	1	1	0.65
2	0.5	2.5	1	0.90
3	0.75	0.5	1	0.75
4	0	0	1	0.85
5	0.5	1.5	0	0.70
6	1	0	0	0.45

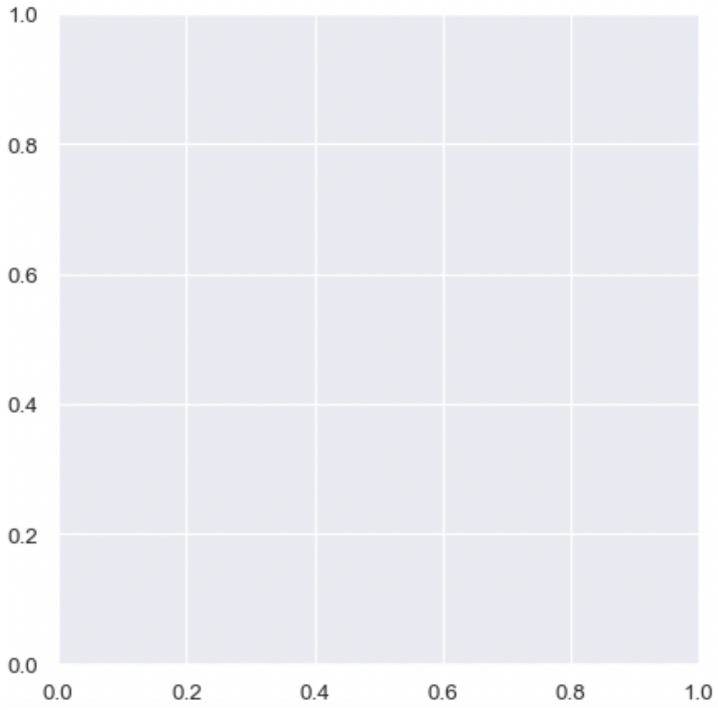
	0	1/4	2/4	3/4
(i) A	○	○	○	●
(ii) B	●	○	○	○
(iii) C	○	○	○	●
(iii) D	○	○	●	○

Solution: When the threshold is 0.75, there are 3 true positives, 2 true negatives, and 1 false negative. The resulting metrics are $TPR = \frac{3}{3+1} = \frac{3}{4}$ and $FPR = \frac{0}{0+2}$

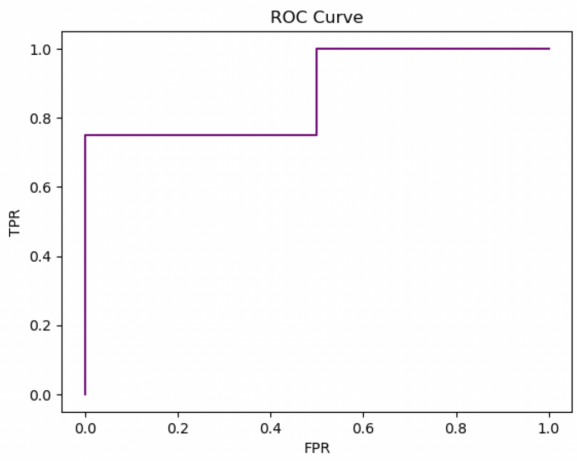
When the threshold is 0.7, there are 3 true positives, 1 true negative, 1 false positive, and 1 false negative. The resulting metrics are $TPR = \frac{3}{3+1} = \frac{3}{4}$ and $FPR = \frac{1}{1+1} = \frac{1}{2} = \frac{2}{4}$

(g) [3 Pts] Let's construct our ROC curve! For various threshold values, the x-axis of an ROC curve displays the FPR, while the y-axis shows the TPR.

Given your answers from the last subpart, construct the ROC curve on the given plot **and** compute the AUC (Area under the ROC curve). You can write the AUC as a simplified fraction or decimal.



AUC: _____



Solution:

The AUC under this curve can be found by taking areas of rectangles. We have $(3/4 * 1/2) + (1 * 1/2) = 7/8$

6 Irregular Regularization [Regularization] *(Sp23 Data 100 Final)*

Congratulations! You just landed a new job as a data scientist working at Corn Hole Inc., which has a training program for corn hole, a competitive party game with bean bags. Novices enrolled in the program practice corn hole for a given number of hours then play in a scrimmage game totaling 100 points. You decide to model a linear relationship between a novice's practice time (in hours), x, and their scrimmage score, y. Your model takes on the form $\hat{y} = w^T x$, where w is a scalar.

(a) [2 Pts] Rather than using L1 or L2 regularization, you decide to create a new regularization term to penalize models with large magnitudes of w. Which of the following options is the most appropriate choice of regularization term?

- ◯ e^{-w^2}
- ◯ $5^{|w|}$
- ◯ $\frac{1}{w}$
- ◯ w^3

Solution: An appropriate regularization term should penalize complex models. This means that the regularization term should increase the loss function by a large, positive value when the parameter w has large magnitude. Of the possible choices, only $5^{|w|}$ is large and positive for both very large positive and negative w.

(b) [5 Pts] Your colleague asks you to use a different regularization term called "Irregular Regularization." The Irregular Regularization objective function takes the form on the right, where λ is the regularization penalty hyperparameter.

$$\left(\frac{1}{n}\sum_{i=1}^{n}(t_i - wx_i)^2\right) + \lambda(e^{2+\lambda})w^2$$

Derive the optimal solution for $\hat{w}$ using the Irregular Regularization objective function in terms of x_i, t_i, λ, **and** n. Please **write your final answer on the provided line**.

$\hat{w} =$ _____

Solution: Take the derivative of the objective function with respect to w, set it equal to 0, and solve for w.

$$\frac{\partial L}{\partial w} = \frac{1}{n}\sum_{i=1}^{n} 2(t_i - wx_i)(-x_i) + \lambda(e^{2+\lambda})(2w)$$

$$0 = \frac{-2}{n}\sum_{i=1}^{n}(x_i t_i - wx_i^2) + \lambda(e^{2+\lambda})(2w)$$

$$0 = \frac{-1}{n}\sum_{i=1}^{n} x_i t_i + \frac{w}{n}\sum_{i=1}^{n} x_i^2 + \lambda(e^{2+\lambda})w$$

$$0 = \frac{-1}{n}\sum_{i=1}^{n} x_i t_i + w\left(\frac{1}{n}\sum_{i=1}^{n}(x_i^2) + \lambda(e^{2+\lambda})\right)$$

$$\frac{1}{n}\sum_{i=1}^{n} x_i t_i = w\left(\frac{1}{n}\sum_{i=1}^{n}(x_i^2) + \lambda(e^{2+\lambda})\right)$$

$$\hat{w} = \frac{\frac{1}{n}\sum_{i=1}^{n} x_i t_i}{\frac{1}{n}\sum_{i=1}^{n}(x_i^2) + \lambda(e^{2+\lambda})}$$

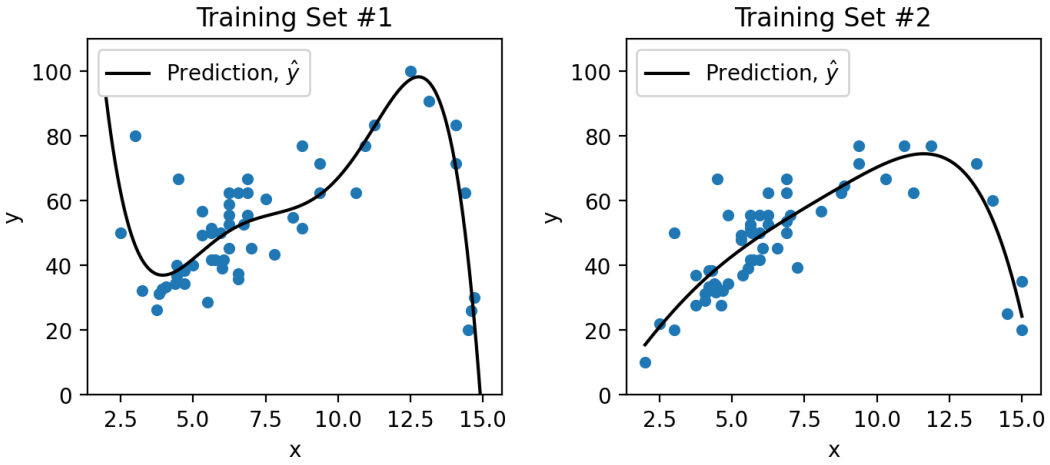
(c) [2 Pts] You now wish to evaluate your model's performance by computing its MSE on the training set. Assume that the optimal choice of the model parameter under Irregular Regularization is represented by $\hat{w}$. Which of the following options should be used to compute the MSE of the regularized model on the training data?

- ◯ $\frac{1}{N}\sum_{n=1}^{N}(t_n - \hat{w}x_n)^2$
- ◯ $\frac{1}{N}\sum_{n=1}^{N}(y_n - \hat{w}x_n)^2 + \lambda|\hat{w}|$
- ◯ $\frac{1}{N}\sum_{n=1}^{N}(y_n - \hat{w}x_n)^2 + \lambda\hat{w}^2$

◯ $\frac{1}{N}\sum_{n=1}^{N}(y_n - \hat{w}x_n)^2 + \lambda(e^{2+\lambda})\hat{w}^2$

(d) [3 Pts] Next, you consider models other than the linear model you constructed above. You learn that your new company has a strange policy where all models are fitted to **two training sets**, Training Set #1 and Training Set #2. You next decide to evaluate a new model fit to each of these two training sets.

Consider a quintic (degree 5) model $\hat{y} = \hat{q}_0 + \hat{q}_1 x + ... + \hat{q}_5 x^5$, where $\hat{q}_j$ is the MSE parameter estimate for feature j of this quintic model. You first fit this model to Training Set #1; then, you fit this model again to Training Set #2.



(i) The primary contributor to this model's risk is:

◯ High bias
◯ **High variance**

7 Tikhonov Regularization [Regularization] *(Borrowed from previous CS189 offerings.)*

(Note from the curator: this isn't in scope for our exam, but it gives good practice with algebraic manipulations for use in derivations, and also provides a glimpse into the type of problems you might see in early 182 homeworks. To the mathematically-inclined student: try it out!)

Tikhonov-regularized regression is a generalization of ridge regression specified by the optimization problem

$$\arg\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{t} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{\Gamma}\mathbf{w}\|_2^2\,,$$

For some full rank matrix $\mathbf{\Gamma} \in \mathbb{R}^{D\times D}$.

In this problem, we look at Tikhonov regularization from a probabilistic standpoint and how it relates to the MAP estimator for a certain choice of prior on the parameters $\mathbf{w}$.

Let $\mathbf{x} \in \mathbb{R}^D$ be a D-dimensional vector and $T \in \mathbb{R}$ be a one-dimensional random variable. Assume a linear-Gaussian model: $T|\mathbf{x}, \mathbf{w} \sim \mathcal{N}(\mathbf{x}^\top\mathbf{w}, 1)$. Suppose that $\mathbf{w} \in \mathbb{R}^D$ is a D-dimensional Gaussian random vector $\mathbf{w} \sim \mathcal{N}(0, \mathbf{\Sigma})$, where $\mathbf{\Sigma}$ is a known symmetric positive-definite covariance matrix.

(Questions begin on next page)

(a) [2 Pts] Let us assume that we are given N training data points $\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_N, t_N)\}$. Derive the posterior distribution of $\mathbf{w}$ given the training data. What is the MAP estimate of $\mathbf{w}$? Compare this result to the solution you achieve in your homework. Comment on how Tikhonov regularization is a generalization of ridge regression from a probabilistic perspective.

[Hint: You may find the following lemma useful. If the probability density function of a random variable is of the form

$$f(\mathbf{v}) = C \cdot \exp\left\{-\frac{1}{2}\mathbf{v}^\top \mathbf{A}\mathbf{v} + \mathbf{b}^\top \mathbf{v}\right\}$$

where C is some constant to make $f(\mathbf{v})$ integrate to 1 and $\mathbf{A}$ is a symmetric positive definite matrix, then $\mathbf{v}$ is distributed as $N(\mathbf{A}^{-1}\mathbf{b}, \mathbf{A}^{-1})$.

Solution: We'll start the derivation by applying Bayes' Theorem and omitting the constant terms in the denominator which don't depend on $\mathbf{w}$.

$$f(\mathbf{w}|\mathbf{x}_1, \cdots, \mathbf{x}_n, t_1, \cdots, t_n) \propto \left[\prod_{i=1}^{n} f(t_n|\mathbf{x}_n, \mathbf{w})\right] f(\mathbf{w})$$

Then, we write out the pdf directly.

$$\left[\prod_{i=1}^{n} f(t_n|\mathbf{x}_n, \mathbf{w})\right] f(\mathbf{w}) \propto \left[\prod_{i=1}^{n} \exp\left\{-\frac{1}{2}(t_n - \mathbf{x}_n^\top \mathbf{w})^2\right\}\right] \exp\left\{-\frac{\mathbf{w}^\top \boldsymbol{\Sigma}^{-1}\mathbf{w}}{2}\right\}$$

We from expand out the squared term and combine it with the exponent in the prior, grouping terms with shared factors.

$$\begin{aligned}
&\left[\prod_{i=1}^{n} \exp\left\{-\frac{1}{2}(t_n - \mathbf{x}_n^\top \mathbf{w})^2\right\}\right] \exp\left\{-\frac{\mathbf{w}^\top \boldsymbol{\Sigma}^{-1}\mathbf{w}}{2}\right\} \\
&= \left[\exp\left\{-\frac{1}{2}\sum_{i=1}^{n}(t_n - \mathbf{x}_n^\top \mathbf{w})^2\right\}\right] \exp\left\{-\frac{\mathbf{w}^\top \boldsymbol{\Sigma}^{-1}\mathbf{w}}{2}\right\} \\
&\propto \exp\left\{-\frac{1}{2}[\mathbf{w}^T(\mathbf{X}^\top\mathbf{X} + \boldsymbol{\Sigma}^{-1})\mathbf{w} - 2(\mathbf{X}^\top \mathbf{t})^\top \mathbf{w}]\right\},
\end{aligned}$$

Now, we have $\mathbf{t} \in \mathbb{R}^n$ with nth entry t_n. Based on the lemma we provided above, we can therefore determine the posterior of $\mathbf{w}|\mathbf{x}_1, \cdots, \mathbf{x}_n, t_1, \cdots, t_n$ is $\mathcal{N}\big((\mathbf{X}^\top\mathbf{X} + \boldsymbol{\Sigma}^{-1})^{-1}\mathbf{X}^\top\mathbf{t}, (\mathbf{X}^\top\mathbf{X} + \boldsymbol{\Sigma}^{-1})^{-1}\big)$.

Solution: We provide a proof of the Lemma below.

Lemma: If the probability density function of a random variable is of the form

$$f(\mathbf{v}) = C \cdot \exp\left\{-\frac{1}{2}\mathbf{v}^\top \mathbf{A}\mathbf{v} + \mathbf{b}^\top \mathbf{v}\right\},$$

where C is some constant to make $f(\mathbf{v})$ integrate to 1, and $\mathbf{A}$ is a symmetric positive definite matrix, then $\mathbf{v}$ is distributed as $N(\mathbf{A}^{-1}\mathbf{b}, \mathbf{A}^{-1})$.

The proof of the lemma is as follows:

$$\begin{aligned} f(\mathbf{v}) &= C \cdot \exp\Big\{ -\frac{1}{2}\mathbf{v}^\top \mathbf{A}\mathbf{v} + \mathbf{b}^\top \mathbf{v}\Big\}, \\ &= C_1 \cdot \exp\Big\{ -\frac{1}{2}\left(\mathbf{v} - (\mathbf{A}^{-1}\mathbf{b})\right)^\top \mathbf{A}\left(\mathbf{v} - (\mathbf{A}^{-1}\mathbf{b})\right) - \frac{1}{2}\mathbf{b}^\top \mathbf{A}^{-1}\mathbf{b}\Big\}, \\ &= C_2 \cdot \exp\Big\{ -\frac{1}{2}\left(\mathbf{v} - (\mathbf{A}^{-1}\mathbf{b})\right)^\top \mathbf{A}\left(\mathbf{v} - (\mathbf{A}^{-1}\mathbf{b})\right) \Big\}, \end{aligned}$$

which is the density of a Gaussian random variable with mean $\mathbf{A}^{-1}\mathrm{b}$ and covariance matrix $\mathbf{A}^{-1}$.

Now that we have the posterior distribution of w, it is simple to find the MAP estimate. The MAP estimate is simply a maximization over the posterior probability. Since we know the posterior of w is a Gaussian, we have that the MAP estimate is the mean of that Gaussian. So we have that MAP estimate of w is $(\mathbf{X}^\top\mathbf{X} + \mathbf{\Sigma}^{-1})^{-1}\mathbf{X}^\top\mathrm{t}$.

In ridge regression, we also assume a Gaussian prior on w. However, we constrain the covariance of the prior to be a simple identity scaled by some constant. Tikhonov regularization generalizes this and allows us to select any covariance matrix for our prior on w. Note in particular that this means our prior can involve non-zero covariance terms, or scale variances in different directions differently (e.g., the first entry of w may have a smaller variance than the second entry; Tikhonov regularization allows us to encode this knowledge in the problem setup. Being able to give priority to certain directions in feature space can be important if we want to be able to take advantage of the regularizing effect of lots of features.

(b) [2 Pts] Let us extend this result from the previous part to the case where we introduce observation noise variables Z_n that are not independent across samples, i.e. $\mathbf{Z}$ is not $N(\mathbf{0}, \mathbb{I}_n)$ but instead distributed as $N(\mu_z, \mathbf{\Sigma}_z)$ for some mean μ_z and some covariance $\mathbf{\Sigma}_z$ (still independent of the parameter $\mathbf{w}$). We make the reasonable assumption that the $\mathbf{\Sigma}_z$ is invertible. Derive the posterior distribution of $\mathbf{w}$ by appropriately changing coordinates.

(Hint: Write $\mathbf{Z}$ as a function of a standard normal Gaussian vector $\mathbf{V} \sim N(\mathbf{0}, \mathbb{I}_n)$ and use the result in (a) for an equivalent model of the form $\widetilde{\mathbf{t}} = \widetilde{\mathbf{X}}\mathbf{w} + \mathbf{V}$.)

Solution: By changing variables, our goal is to reduce to the previous case — with white observation noises.

We want to define a matrix $\mathbf{\Sigma}_z^{1/2}$ such that $\mathbf{\Sigma}_z^{1/2}(\mathbf{\Sigma}_z^{1/2})^\top = \mathbf{\Sigma}_z$. According to eigenvalue decomposition $\mathbf{\Sigma}_z = \mathbf{U}\Delta\mathbf{U}^\top$. Exploiting the spectral theorem's guarantee of orthonormal eigenvalues and all positive eigenvalues to form the diagonal matrix Δ, we get that we can choose $\mathbf{\Sigma}_z^{\frac{1}{2}} = \mathbf{U}\Delta^{\frac{1}{2}}$.

Now, we define $\mathbf{V}$ such that $\mathbf{Z} = \mu_z + \mathbf{\Sigma}_z^{\frac{1}{2}}\mathbf{V}$. Then $\mathbf{V} \sim N(\mathbf{0}, \mathbb{I})$ and $\mathbf{V} = \mathbf{\Sigma}_z^{-\frac{1}{2}}(\mathbf{Z} - \mu_z)$.

$$
\begin{aligned}
\mathbf{t} &= \mathbf{X}\mathbf{w} + \mathbf{Z} \\
\mathbf{t} - \mu_z &= \mathbf{X}\mathbf{w} + \mathbf{Z} - \mu_z \\
\mathbf{\Sigma}_z^{-\frac{1}{2}}(\mathbf{t} - \mu_z) &= \mathbf{\Sigma}_z^{-\frac{1}{2}}\mathbf{X}\mathbf{w} + \mathbf{\Sigma}_z^{-\frac{1}{2}}(\mathbf{Z} - \mu_z) \\
\mathbf{\Sigma}_z^{-\frac{1}{2}}(\mathbf{t} - \mu_z) &= \mathbf{\Sigma}_z^{-\frac{1}{2}}\mathbf{X}\mathbf{w} + \mathbf{V}
\end{aligned}
$$

So the problem reduces to the previous part by denoting $\mathbf{\Sigma}_z^{-\frac{1}{2}}\mathbf{X}$ as $\widetilde{\mathbf{X}}$ and denoting $\widetilde{\mathbf{y}}$ as $\mathbf{\Sigma}_z^{-\frac{1}{2}}(\mathbf{t} - \mu_z)$, which yields the posterior of $\mathbf{w}|\mathbf{x}_1, \cdots, \mathbf{x}_n, t_1, \cdots, t_n$ as a multivariate Gaussian

$$N\left((\widetilde{\mathbf{X}}^\top\widetilde{\mathbf{X}} + \mathbf{\Sigma}^{-1})^{-1}\widetilde{\mathbf{X}}^\top\widetilde{\mathbf{y}}, (\widetilde{\mathbf{X}}^\top\widetilde{\mathbf{X}} + \mathbf{\Sigma}^{-1})^{-1}\right)$$

or

$$N\left((\mathbf{X}^\top\mathbf{\Sigma}_z^{-1}\mathbf{X} + \mathbf{\Sigma}^{-1})^{-1}\mathbf{X}^\top\mathbf{\Sigma}_z^{-1}(\mathbf{t} - \mu_z), (\mathbf{X}^\top\mathbf{\Sigma}_z^{-1}\mathbf{X} + \mathbf{\Sigma}^{-1})^{-1}\right).$$

Notice here how the $\mathbf{\Sigma}_z^{-1}$ matrix is now sitting in between the $\mathbf{X}^\top\mathbf{\Sigma}_z^{-1}\mathbf{X}$ terms.

8 Graduate Descent [Gradient Descent] *(Sp24 Data 100 Final Q2)*

Mir wants to build a model to predict the probability that he will get into each graduate school he applies to. He comes up with the following loss function with the corresponding gradient vector:

$$L(w_0, w_1) = \frac{1}{N}\sum_{n=1}^{N}\Big(t_n - \frac{w_0}{x_n} - \ln(w_1)\, x_n + w_0 w_1 (x_n^2 + 1)\Big)$$

$$\nabla_{\mathbf{w}} L = \begin{bmatrix} \frac{1}{N}\sum_{n=1}^{N}\Big(- \frac{1}{x_n} + w_1 x_n^2 + w_1 \Big) \\ \frac{1}{N}\sum_{n=1}^{N}\Big(- \frac{x_n}{w_1} + w_0 x_n^2 + w_0 \Big) \end{bmatrix}$$

(a) [4 Pts] Mir runs a stochastic gradient descent algorithm. He initializes the model's weights as $w_0^{(0)} = 4$ and $w_1^{(0)} = 1$, then randomly selects the data point $x_n = 2$ to perform one stochastic gradient descent update. After running one iteration, Mir updates w_1 to be $w_1^{(1)} = -5$. What was Mir's learning rate (η) for this update?

η = _____

Solution: Based on our gradient from above, we can set up our gradient descent update rule for w_1 as shown below. Note that we do not need to worry about the summation because we only use one data point in stochastic gradient descent.

$$w_1^{(t)} = w_1^{(t-1)} - \eta\Big(- \frac{x_n}{w_1^{(t-1)}} + w_0^{(t-1)} x_n^2 + w_0^{(t-1)}\Big)$$

Plugging in the values we're given:

$$-5 = 1 - \eta\Big(- \frac{2}{1} + 4(2^2) + 4\Big)$$

This simplifies to

$$-5 = 1 - \eta(18)$$

From here, we can use some algebra to solve for $\eta = \frac{1}{3}$

(b) [2 Pts] Which of the following statements are true? **Select all that apply.**

- ☐ **A. The initial weight(s) chosen for gradient descent can impact whether it converges to the true optimal weights.**
- ☐ **B. The learning rate η can impact whether gradient descent converges to the true optimal weights.**
- ☐ **C. The choice of loss function can impact whether gradient descent converges to the true optimal weights.**
- ☐ D. Both batch and stochastic gradient descent compute the true gradient of the loss surface during each iteration.

Solution: Option A is correct. If the loss function has local minima other than the global minima, gradient descent may converge to the local minima depending on where the weights are initialized.
Option B is correct. An inappropriate choice of learning rate can cause gradient descent never to converge or potentially even diverge.
Option C is correct. Like Option A, a non-convex loss function may lead to gradient descent converging to the wrong place.
Option D is incorrect. Batch gradient descent computes the true gradient, but stochastic only uses a small sample to compute this gradient.

9 Gradient Grab Bag [Gradient Descent] *(Sp23 Data 100 Final)*

(a) [2 Pts] Consider the constant model $y(x, w) = w$ used to model a target t from a dataset $\{t_1, \ldots, t_N\}$. For which of the following objective functions is gradient descent **guaranteed** to find the optimal model parameter w, assuming an appropriate choice of step size and sufficient time for convergence? Select all that apply.

☐ $L(w) = \frac{1}{N}\sum_{n=1}^{N} w^3$

☐ $L(w) = \frac{1}{N}\sum_{n=1}^{N}(t_n - w)^6$

☐ $L(w) = \frac{1}{N}\sum_{n=1}^{N}(t_n - w)$

☐ $L(w) = \frac{1}{N}\sum_{n=1}^{N}(t_n - w)^2 + \lambda w^2$

Solution: Gradient descent can optimize a **convex** function. Convexity can be tested by computing the second derivative of each objective function with respect to w. Options 2 and 4 have second derivatives that are positive for all values of w, and are hence convex.

(b) [5 Pts] Now, consider a more complex model: $y(x, \mathbf{w}) = w_0 + w_1^2 x^2 + e^x$. Suppose that MSE is used as the objective function, L, to select the optimal choice of w_0 and w_1 in the model above. Assume $\mathbf{w}$ represents the 2×1 column vector $[w_0, w_1]^T$.

Which of the following expressions represents the **gradient vector** $\nabla_{\mathbf{w}} L$? To receive full credit, **show your work** in the box below.

◯ $\nabla_{\mathbf{w}} L = \begin{bmatrix} \frac{2}{N}\sum_{n=1}^{N}(t_n - w_0 - w_1^2 x_n^2 - e^{x_n})(-2w_1^2 x_n - e^{x_n}) \\ \frac{2}{N}\sum_{n=1}^{N}(t_n - w_0 - w_1^2 x_n^2 - e^{x_n}) \end{bmatrix}$

◯ $\nabla_{\mathbf{w}} L = \begin{bmatrix} \frac{2}{N}\sum_{n=1}^{N}(t_n - w_0 - w_1^2 x_n^2 - e^{x_n})(-1) \\ \frac{2}{N}\sum_{n=1}^{N}(t_n - w_0 - w_1^2 x_n^2 - e^{x_n})(+1) \end{bmatrix}$

◯ $\nabla_{\mathbf{w}} L = \begin{bmatrix} \frac{2}{N}\sum_{n=1}^{N}(t_n - w_0 - w_1^2 x_n^2 - e^{x_n})(-1) \\ \frac{2}{N}\sum_{n=1}^{N}(t_n - w_0 - w_1^2 x_n^2 - e^{x_n})(-2w_1 x_n^2) \end{bmatrix}$

◯ $\nabla_{\mathbf{w}} L$ is undefined for some $\mathbf{w}$

◯ None of the above. If you select this choice, circle your final answer for the gradient vector in the box below.

Solution: For the given model, the MSE objective function has the form:

$$L(w_0, w_1) = \frac{1}{N}\sum_{n=1}^{N}(t_n - w_0 - w_1^2 x_n^2 - e^{x_n})^2$$

To find the gradient vector, we take the derivative of L with respect to w_0 and w_1, then stack these results in a column vector.

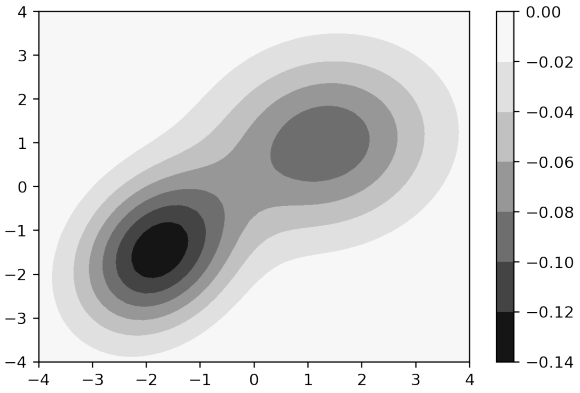
$$\frac{\partial L}{\partial w_0} = \frac{2}{N}\sum_{n=1}^{N}(t_n - w_0 - w_1^2 x_n^2 - e^{x_n})(-1)$$

$$\frac{\partial L}{\partial w_1} = \frac{2}{N}\sum_{n=1}^{N}(t_n - w_0 - w_1^2 x_n^2 - e^{x_n})(-2w_1 x_n^2)$$

Hence, Option 3 is correct.

(c) [2 Pts] Consider the plot shown to the right. This visualization is a contour plot of the loss surface for a model with two parameters, w_0 and w_1. The loss is computed on a dataset with observations of the form (x_1, x_2, t).

What values are plotted on the horizontal and vertical axes, respectively, of this contour plot?



◯ x_1 and x_2 ◯ The magnitude of loss and t

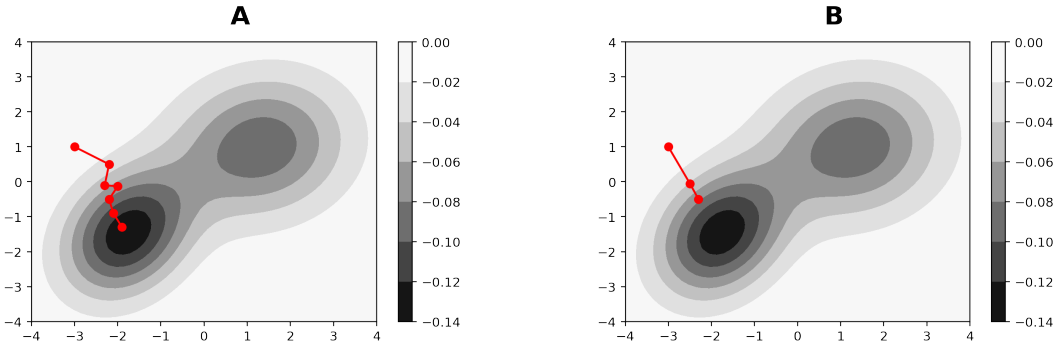
◯ w_0 **and** w_1 ◯ The magnitude of loss and x_1

(d) [2 Pts] Which of the following Gradient Descent (GD) techniques are **guaranteed** to converge to the optimal choice of model parameters for the loss surface in part (c)?

☐ Batch GD
☐ Mini-batch GD
☐ Stochastic GD
☐ **None of the above**

Solution: Gradient descent is guaranteed to converge on a **convex** function with an appropriate choice of step size and sufficient time for converge. The contour plot above contains two local minima, so the loss surface it represents cannot be convex. It is possible that the three gradient descent techniques listed above may identify the incorrect local minimum as the optimal solution for $\mathbf{w}$.

(e) [2 Pts] The plots below show the same loss surface as in part (c), now annotated with updates for **two full gradient descent epochs** after initialization at the same starting position. Each plot displays the update iterations for a different gradient descent technique: batch gradient descent or mini-batch gradient descent.



In the boxes below, indicate which plot corresponds to batch gradient descent and which plot corresponds to mini-batch gradient descent. Fill each box with “A" or “B".

Batch gradient descent

Mini-batch gradient descent

Solution:

Batch gradient descent: B

Mini-batch gradient descent: A

- Mini-batch gradient descent only uses a subset of the total training set to compute the gradient vector at each update, so it is not guaranteed to always find the direction of greatest descent towards the global minimum. This causes the mini-batch gradient descent updates to zig-zag away from the path to the true minimum
- Additionally, mini-batch gradient descent includes multiple updates in each epoch. This is the reason for the extra steps taken in the mini-batch plot