USC CSCI 467 Intro to Machine Learning	**Final Exam** May 13, 2025, 2:00-4:00pm	Spring 2025 Instructor: Robin Jia

Name: _____

USC e-mail: _____@usc.edu

Answer the questions in the spaces provided. **If you run out of space, continue your work on the last two pages, and indicate that your answer is there**. You may use the backs of pages for scratch work only. **Please use pen for ease of grading**. This exam has 7 questions, for a total of 150 points.

This content is protected and may not be shared, uploaded, or distributed.

Question 1: Predicting College Admissions, Part II (32 points)

Recall from the midterm exam that you developed a classifier to predict whether an applicant would be admitted to USC. Word gets around about your classifier and you are soon contacted by another university, the Unregulated College of Learning Algorithms (UCLA for short). Inspired by your work, UCLA has developed their own classifier and wants to use it to automate their college admissions process. They want you to help them evaluate their model.

(a) (3 points) Of the three types of fairness-related harms we discussed in class, which is most relevant for this problem? Circle the best answer.

A. Allocative harms

B. Representational harms

C. Unequal accuracy

Solution: Allocative harms are most relevant here, since we are classifying individuals and deciding whether to give them opportunities.

(b) (5 points) You learn that UCLA's classifier was trained to predict whether a student would graduate from UCLA with a high GPA, based on their application materials. Their justification is that they want to admit students who they believe will succeed in their classes. You also know that some degree programs at UCLA are much harsher in grading than other degree programs. Based on this, explain how UCLA's classifier could be biased against some applicants.

Solution: Students in harsher degree programs will generally have a lower GPA than similar students in easier degree programs. Thus the classifier will be biased against students interested in those harshly graded subjects, and biased towards students interested in other subjects.

(c) You hear rumors that UCLA is biased against applicants who like the color red, so you set out to measure this. Luckily, UCLA asks each applicant if they like red on their application, so this data is available. The UCLA team shows you some statistics of their model's predictions on this year's applicants as follows:

Likes Red?	Predicted to Admit?	Number of Applicants
Yes	Yes	100
Yes	No	300
No	Yes	200
No	No	400

i. (6 points) Is this enough information to measure whether **independence** is satisfied with respect to the "likes red" and "does not like red" groups of applicants? If so, perform the necessary computations and state whether independence is satisfied. If not, explain what information is missing. *(Recall that independence measures whether predictions are independent of the protected attribute.)*

Solution: Yes, there is enough information to determine this. The admission rate of students who like red is $100/400 = 25\%$. Meanwhile, the admission rate

of students who do not like red is 200/600 = 33%. Since these are not equal, independence is **not** satisfied.

ii. (6 points) Is this enough information to measure whether **separation** is satisfied with respect to the "likes red" and "does not like red" groups of applicants? If so, perform the necessary computations and state whether separation is satisfied. If not, explain what information is missing. *(Recall that separation measures whether precision on both classes is equalized across the two groups.)*

Solution: No, this is not enough information. Separation requires a notion of ground truth to compute precision, but there is no ground truth provided here.

(d) (10 points) While testing the UCLA classifier, you notice that it was designed in an unexpected way. The UCLA team first converts all applications into images by taking screenshots of the application form, then passes these images into a convolutional neural network to make predictions. Furthermore, the application includes a photo of the applicant that the applicant uploads. You realize that this makes the classifier vulnerable to adversarial examples. You want to create a demonstration to show the UCLA team how an adversarial applicant could trick their classifier into getting admitted by imperceptibly changing their photo. Explain step-by-step how you would create an adversarial example that demonstrates this threat. (Assume you have access to the model parameters.)

Solution: The key steps would be:

- Create a screenshot of the application with the original photo, and pass that to the classifier.
- Compute the gradient of probability of outputting "accepted" with respect to the input screenshot (other solutions based on loss are also OK, but must specify loss relative to what label).
- For the part of the screenshot corresponding to the photo, change each pixel of the photo by a small amount matching the sign of the gradient.
- Upload the corresponding photo.

(e) (2 points) **True** or **False** (circle one): UCLA would eliminate the risk of adversarial examples if they instead used a pre-trained Transformer language model that was fine-tuned to predict admissions decisions based on the text of the application only.

Solution: This is false. Language models are also susceptible to adversarial attacks, as discussed in class.

Question 2: Federated Learning for Linear Regression (22 points)

Federated learning refers to a type of machine learning where we train a single model using several different computers, each of which has access to a different subset of the training data.

For example, imagine several hospitals that want to collaborate on training a machine learning classifier. Each hospital has their own private data from their patients, and does not wish to share their data with anyone else. Through federated learning, we can train a model that learns from all of the hospitals' data, which would likely be better than what any individual hospital could train using only their data.

In this problem, we will examine federated learning for linear regression. Assume there exist k datasets $D_1, \ldots, D_k$ (e.g., from k different hospitals), each of which contains (x, y) pairs where $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$. Let D denote the union of all examples from these k datasets, and let $n = |D|$.

(a) First, let's recall how linear regression works normally. Consider the following pseudocode:

Input: Dataset D, learning rate η
Initialize: Weight vector $w \in \mathbb{R}^d$
$\nabla_w \leftarrow 0$ ▷ Initialize accumulated gradient
for each $(x^{(i)}, y^{(i)}) \in D$ **do**
    $\hat{y}^{(i)} \leftarrow w^\top x^{(i)}$ ▷ Linear prediction
    $\mathcal{L}_i \leftarrow \frac{1}{2}(\hat{y}^{(i)} - y^{(i)})^2$ ▷ Mean squared error loss
    $\nabla_w \leftarrow \nabla_w + \frac{\partial \mathcal{L}_i}{\partial w}$ ▷ Accumulate gradients
end for
$w \leftarrow w - \eta \nabla_w$ ▷ Update parameters
Return w

---

i. (3 points) Which of the following best describes what this pseudocode does?

A. Train a model fully with gradient descent
B. Train a model fully with stochastic gradient descent
C. Train a model with gradient descent for one iteration
D. Train a model with stochastic gradient descent for one epoch

Solution: C. This describes one iteration of full gradient descent. It is not SGD since the gradient is summed up over the whole dataset, and only used to update the parameters once at the end.

ii. (3 points) Consider the following list of symbols and variables:

$$x^{(i)}, y^{(i)}, w, \nabla_w, \mathcal{L}_i, \eta.$$

If you were to deploy your model for others to use, which one(s) would be critical to include in the deployment?

Solution: The only thing needed for deployment is the model parameters, only w.

Now let's consider a setting where there are k "worker" computers; the j-th worker only has access to D_j. In addition, there is a central server that coordinates these k workers. To compute one gradient descent update, federated learning does the following:

1. The central server sends the current model parameters to each worker computer.

This content is protected and may not be shared, uploaded, or distributed.

2. Each worker calculates the gradient of the loss on **its own dataset** with respect to the model parameters. For our linear regression setting, worker j computes

$$g^{(j)} = \nabla_w \frac{1}{|D_j|} \sum_{(x,y)\in D_j} \frac{1}{2}(w^\top x - y)^2.$$

Each computer then sends this gradient back to the central server.

3. The central server uses the gradients $g^{(1)}, \ldots, g^{(k)}$ from the other computers to compute the overall gradient with respect to the full dataset, i.e.,

$$\nabla_w \frac{1}{n} \sum_{(x,y)\in D} \frac{1}{2}(w^\top x - y)^2.$$

4. Using the overall gradient, the central server updates the model parameters.

(b) (6 points) Write a formula that describes how the central server should compute step 3. In other words, how can it use $g^{(1)}, \ldots, g^{(k)}$ to compute the overall gradient? Assume the server knows the size $|D_j|$ of each worker's dataset.

Solution: Each worker computes the gradient of the average loss on its dataset. The gradient of the sum of the loss across dataset D_j is therefore $|D_j| \cdot g^{(j)}$. So the gradient of the sum of the loss across the entire dataset D is

$$\sum_{j=1}^{k} |D_j| \cdot g^{(j)},$$

which means the overall average is

$$\sum_{j=1}^{k} \frac{|D_j|}{n} \cdot g^{(j)}.$$

(c) The advantage of federated learning is that each worker does not need to share its private dataset. However, sending the gradient $g^{(j)}$ does leak some information about the dataset D_j. Consider an attacker who is able to listen to network traffic from worker j to the central server and is able to obtain $g^{(j)}$ at a single timestep during training. Suppose further that D_j consists of only a single example $(\tilde{x}, \tilde{y})$.[1]

i. (3 points) Write the expression for $g^{(j)}$ in terms of w, $\tilde{x}$, and $\tilde{y}$.

Solution:

$$g^{(j)} = (w^\top \tilde{x} - \tilde{y}) \cdot \tilde{x}.$$

ii. (7 points) Suppose that the attacker knows that $\tilde{x}_1 = 1$ (e.g., because all the workers agree to set the first feature of x to be a constant feature). Describe how the attacker could completely recover $\tilde{x}$.

[1] This problem shows why federated learning with a very small dataset is not very private. In practice, you would want to ensure that each worker has a relatively large dataset to avoid leaking private data.

This content is protected and may not be shared, uploaded, or distributed.

Solution: The attacker knows that $g_1^{(j)}$ is equal to $(w^\top \tilde{x} - \tilde{y})$. Thus, they now know that

$$\tilde{x} = \frac{1}{g_1^{(j)}} \cdot g^{(j)}.$$

Question 3: Implementing Backpropagation (10 points)

In Homework 3, you extended the backpropagation library we created in lecture. Below, fill in the missing lines of code to implement backpropagation for the square root function. We have provided the definition of the `Node` class to remind you how it works. You may use the numpy function `np.sqrt()` to compute the square root of a number.

```
class Node(object):
    """Node in a computation graph. Every Node object must store the following:
        - self.value: Numerical value, computed during forward pass
        - self.grad: Gradient, computed during backward pass
        - self.topo_order: Topologically-sorted list of nodes
    """
    def backward(self):
        """Does backward pass by propagating gradient to all child nodes.

        Assumption: self.grad has been set by parent nodes already.
        """
        raise NotImplementedError

    def compute_grad(self):
        # This function only gets called on the output node.
        self.grad = np.float64(1)   # Gradient of output w.r.t itself is 1
        for node in self.topo_order[::-1]:  # Reverse the list
            node.backward()  # Calls backward() in reverse topological order

class SquareRootNode(Node):
    def __init__(self, n):
        self.n = n  # Save a reference to the child node
        """TODO(Part a, 4 points): Fill in these two lines."""
        self.value =

        self.grad =

        self.topo_order = n.topo_order  # Standard bookkeeping
        self.topo_order.append(self)  # Standard bookkeeping

    def backward(self):
        """TODO(Part b, 6 points): Compute the backward pass below."""
```

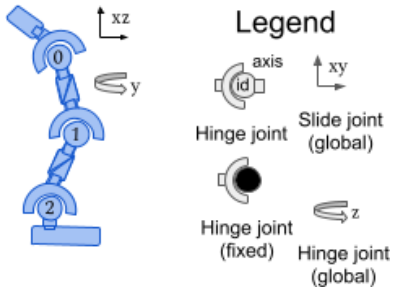
Solution: The correct code is as follows:

```
class SquareRootNode(Node):
    def __init__(self, n):
        self.n = n  # Save a reference to the child node
        """TODO(Part a, 4 points): Fill in these two lines."""
        self.value = np.sqrt(n.value)
        self.grad = np.float64(0)  # We also accepted just writing 0
        self.topo_order = n.topo_order  # Standard bookkeeping
        self.topo_order.append(self)  # Standard bookkeeping

    def backward(self):
        """TODO(Part b, 6 points): Compute the backward pass below."""
        # Compute derivative: d sqrt(n)/dn = 1/(2 * sqrt(n))
        # Then apply the chain rule:
        self.n.grad += self.grad / (2 * self.value)
```

Question 4: Clustering for Tabular Q-Learning (32 points)

Alice wants to use reinforcement learning to control a simple "Hopper" robot (illustrated below). The robot stands on one "foot" and has three joints that can be rotated clockwise or counterclockwise. The robot's state is described by a vector of 11 real numbers, which encode the robot's position, the angles of its joints, and the velocities of its different parts. The robot has one motor on each joint that can control that joint's movement.



(a) (5 points) Alice has her robot start at one end of her desk. She wants to train it to move to the other end of her desk while taking as few actions as possible. Describe a possible reward function that would be appropriate to incentivize this goal. Assume Alice uses a discount factor $\gamma = 1$.

Solution: A reasonable reward function would be:

- Receive a small negative reward (e.g., -1) for every action taken.
- Receive a large reward (e.g., 1000) for reaching the end of the desk.

(b) Alice wants to use Tabular Q-Learning for this problem. She comes up with the idea of clustering the states using k-Means clustering, and treating each cluster as a single state during Tabular Q-Learning. She chooses to run k-Means with $k = 10$ clusters.

This content is protected and may not be shared, uploaded, or distributed.

i. (5 points) Explain why Tabular Q-Learning would not be applicable to Alice's problem without doing this clustering step.

Solution: The original state space is continuous, but Tabular Q-Learning assumes that it is possible to enumerate all the possible states.

ii. (2 points) **True** or **False** (circle your choice): To carry out her idea, Alice will need to first collect a dataset of possible states of her robot before running Tabular Q-Learning.

Solution: True, she will first need a dataset to run k-means clustering.

iii. (5 points) Compared to using a larger k such as $k = 10000$, name one way in which choosing $k = 10$ makes Tabular Q-Learning **easier**.

Solution: Two possible answers are:

- Because the table you have to learn for Q-learning is smaller, the final model will be more memory efficient.
- Because the table you have to learn for Q-learning is smaller, it will require less data to learn the Q value for every (state, action) pair.

iv. (5 points) Compared to using a larger k such as $k = 10000$, name one way in which choosing $k = 10$ makes Tabular Q-Learning **less likely to achieve high reward**.

Solution: Using a small number of clusters means that many different states will be combined into the same state for the model. If different actions are optimal in those different underlying states, the model cannot distinguish them. Therefore the model cannot learn a very precise policy.
Another possible answer is that with a small number of clusters, taking an action may often cause a small enough change to the underlying state that the robot stays within the same cluster. This makes it harder to learn the dynamics of the environment.

v. (5 points) Alice notices that in the 11-dimensional state vectors, some values are highly correlated. For example, the angle of one joint constrains the angle of another one, so those two values are tightly correlated. Based on this observation, is there a more suitable approach than k-Means clustering? Explain your reasoning.

Solution: She could use Gaussian Mixture Models, which could produce better clusters by taking into account the covariance/correlation between different features. In contrast k-Means only produces spherical clusters, which ignore correlations between features.
Another acceptable answer is that she could first use PCA to reduce the dimensionality before running clustering. Very correlated dimensions would be collapsed to a single dimension.

(c) (5 points) Alice notices that the robot's hardware is somewhat unreliable. When she positions the robot in the same state and activates one of the motors, sometimes the

This content is protected and may not be shared, uploaded, or distributed.

corresponding joint moves more and other times it moves less. Will this prevent Alice from using reinforcement learning for her problem? Explain your reasoning using terminology from class.

Solution: No, this is still fine for a reinforcement learning problem. This just means that the transitions from one state to another are random and not deterministic, which is allowed.

Question 5: Unsupervised Learning (21 points)

You have a dataset of 8 examples $\{x^{(1)}, \ldots, x^{(8)}\} \in \mathbb{R}^2$ as follows:

i	$x_1^{(i)}$	$x_2^{(i)}$
1	1	-3
2	1	-2
3	1	2
4	1	3
5	-1	-3
6	-1	-2
7	-1	2
8	-1	3

The covariance matrix of this data is:

$$\Sigma = \begin{pmatrix} 1.1 & 0 \\ 0 & 7.4 \end{pmatrix}.$$

(a) (4 points) What are the eigenvalues λ_1 and λ_2 of Σ, where $\lambda_1 \geq \lambda_2$? What are the corresponding unit-length eigenvectors v_1 and v_2?

Solution: Since Σ is a diagonal matrix, the eigenvalues are simply the diagonal entries, so $\lambda_1 = 7.4$ and $\lambda_2 = 1.1$. The corresponding eigenvectors are $v_1 = [0, 1]$ and $v_2 = [1, 0]$.

(b) Now let's see what happens when we run k-means before and after projecting the data onto different principal components. *(Note: You can answer these questions just by visualizing the data; there is no need to actually run the k-means algorithm)*

i. (3 points) Suppose you ran k-means on the original dataset with $k = 2$. Which points would (most likely) be clustered together? Give your answer as two lists of indices.

Solution: One cluster is $\{1, 2, 5, 6\}$ and the other is $\{3, 4, 7, 8\}$.

ii. (3 points) Now suppose you first projected the dataset onto the **first** principal component, and then ran k-means. Which points would (most likely) be clustered together?

Solution: The first principal component is the $[0, 1]$ direction, so we would just cluster based on the 2nd feature. This yields the same clusters: $\{1, 2, 5, 6\}$ and $\{3, 4, 7, 8\}$.

This content is protected and may not be shared, uploaded, or distributed.

iii. (5 points) Are your answers to parts (i) and (ii) the same or different? Is this expected in light of the fact that the first principal component is chosen to minimize the reconstruction error? Explain your reasoning.

Solution: The answers are the same. This makes sense since projecting onto the first principal is supposed to retain as much structure of the original dataset as possible.

(c) (6 points) For any dataset and datapoint x within that dataset, we can compute the nearest neighbor of x in the dataset in terms of Euclidean distance (excluding x itself). We can also first run PCA on the dataset, project the data onto the first principal component, and then compute the nearest neighbor of x. Are these two processes guaranteed to return the same result? Justify your answer. *(Hint: It may help to consider the 8-example dataset used in this problem.)*

Solution: No, this does not always yield the same result. For instance, in our dataset, the closest point to $[1, -3]$ is $[1, -2]$. But after projecting onto the first principal component, $[1, -3]$ and $[-1, -3]$ get mapped to the same point, so they are the nearest neighbors for each other.

Question 6: Language Models and Saving Work (14 points)

When generating text with an autoregressive language model, some information from past tokens needs to be saved, while other information can be discarded. In this question, assume your language model has just generated the t-th word in a document, and now you are predicting the $t+1$-st word.

(a) First, suppose we are using an LSTM-based autoregressive language model. Is each of the following needed to generate the model's prediction at the current timestep? Write Y (yes) or N (no).

i. (2 points) The previous hidden state h_{t-1}

i. _____

ii. (2 points) The word vector for the $t-1$-th word

ii. _____

iii. (2 points) The first hidden state h_1 (assume $t > 2$).

iii. _____

Solution: The correct answer is Y, N, N. You need the previous hidden state and the *current* word vector to compute the current hidden state h_t.

(b) Now, suppose we are using a Transformer-based autoregressive language model. Is each of the following needed to generate the model's prediction at the current timestep? Write Y (yes) or N (no).

i. (2 points) The key vectors from all timesteps $< t$

i. _____

This content is protected and may not be shared, uploaded, or distributed.

ii. (2 points) The query vectors from all timesteps $< t$

ii. _____

iii. (2 points) The value vectors from all timesteps $< t$

iii. _____

iv. (2 points) The final hidden state from timestep $t - 1$ (the previous timestep)

iv. _____

Solution: The correct answer is Y, N, Y, N. You need the key and value vectors from all previous timesteps for attention. The query vector is only needed for the current timestep. The hidden state from the previous timestep is also not needed for a Transformer, as all information flow between timesteps happens only through attention.

Question 7: Short Answer (19 points)

In the following questions, circle the correct answer or write your choices on the provided lines.

(a) (2 points) **True** or **False**: The 2017 paper *Attention is All You Need* introduced the concept of attention for neural networks.

Solution: False. It only introduced Transformers, but attention was used previously (e.g. in combination with RNNs for sequence-to-sequence tasks).

(b) (2 points) **True** or **False**: If your supervised learning method cannot fit the training data well, it likely has high variance.

Solution: False. This is a sign of high bias.

(c) (2 points) **True** or **False**: In a bandit problem, the set of possible actions is the same at every timestep.

Solution: True. This is different from full reinforcement learning, when different states could have a different set of allowed actions.

(d) (2 points) **True** or **False**: Pre-training is a general technique that can be applied to language or image data.

Solution: True.

(e) (2 points) Which of the following accurately describes the motivation for Byte Pair Encoding?

A. It helps the model process rare words.

B. It makes the model more efficient by making each sentence consist of fewer tokens.

C. It adds regularization to prevent overfitting to common words.

D. It enables a Transformer to know the order of words in a document.

This content is protected and may not be shared, uploaded, or distributed.

Solution: A.

(f) Consider the following methods from class:

A. Decision Trees
B. Gaussian Mixture Models
C. Logistic Regression
D. Naive Bayes
E. Principal Components Analysis
F. Q-Learning
G. Upper Confidence Bound Algorithm

For each description below, select **all** methods that fit the description.

i. (3 points) Makes predictions with a series of learned "if-then" rules.

i. _____

Solution: A only. This describes decision trees.

ii. (3 points) Is an unsupervised learning method.

ii. _____

Solution: B and E.

iii. (3 points) The Exploration vs. Exploitation trade-off is relevant to the method.

iii. _____

Solution: F and G. This is relevant only for reinforcement learning methods.

This content is protected and may not be shared, uploaded, or distributed.

[This page provides extra space for answers]

This content is protected and may not be shared, uploaded, or distributed.

[This page provides extra space for answers]

This content is protected and may not be shared, uploaded, or distributed.