CMSC 478, Spring 2022, Midterm Exam
April 4

Name:

Directions (please read carefully)

1. Be sure to write your name above
2. This exam is closed book and closed note, and you are not to communicate with anyone but the proctors during the exam
3. There is no need for a calculator of any kind, so the use of electronics is prohibited
4. Show your work, where appropriate, for partial credit

Question 1: Suppose you have a function of two variables f(x, y) = $(x - y)^2$ that you are trying to minimize. If the initial values are x = 2 and y = 1 and the learning rate is 0.1, what are the values of x and y after one step of gradient **descent**.

To perform gradient descent, we first need to calculate the gradients of the function with respect to x and y.

∂f/∂x = 2(x-y)
∂f/∂y = -2(x-y)

Using the initial values x=2 and y=1, we can evaluate these partial derivatives:

∂f/∂x = 2(2-1) = 2
∂f/∂y = -2(2-1) = -2

Next, we update the values of x and y using the formula:

x = x - learning_rate * ∂f/∂x
y = y - learning_rate * ∂f/∂y

Substituting the values, we get:

x = 2 - 0.1 * 2 = 1.8
y = 1 - 0.1 * (-2) = 1.2

So after one step of gradient descent, the values of x and y are 1.8 and 1.2, respectively.

Question 2: Suppose the perceptron algorithm is running and the current weight vector is w = (2, -1) and b = 0. What are w and b after the algorithm processes the next instances which is x = (0, 1) and y = 1?

Assuming that the learning rate is 1, the perceptron algorithm works as follows:
Compute the activation for the input instance:
a = w · x + b = (2, -1) · (0, 1) + 0 = -1
Apply the activation function (in this case, the step function):
y_pred = step(a) = step(-1) = 0
Update the weights and bias if the prediction is incorrect:

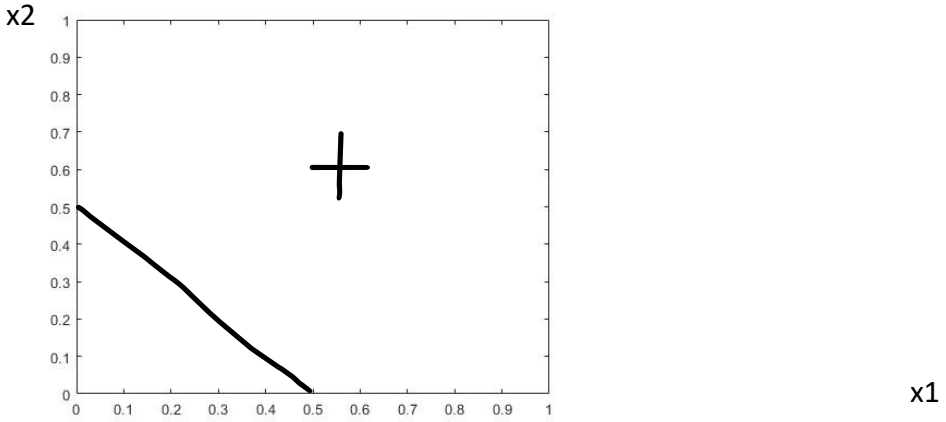
if y_pred != y:

w = w + y * x = (2, -1) + 1 * (0, 1) = (2, 0)

b = b + y = 0 + 1 = 1

In this case, the prediction is incorrect because y_pred = 0 and y = 1. Therefore, we update the weights and bias using the above equations. So the new weight vector is w = (2, 0) and the new bias is b = 1.

Question 3: Suppose the perceptron algorithm terminates with w = (w1, w2) = (1, 1) and b = -0.5. Draw the boundary between the positive and negative instances in the plot below (note that x1 is the horizontal axis and x2 is the vertical axis). And show which half plane is the positive half by putting a + in it.



The perceptron algorithm wx + b = 0 finds the weights and bias that separate the positive and negative instances in the dataset. The boundary between the two classes is a line (in 2D) defined by the equation w1x1 + w2x2 + b = 0. In this case, the weights are (w1, w2) = (1, 1) and the bias is b = -0.5. So, the boundary equation is x1 + x2 - 0.5 = 0 =>
x2 = -x1 + 0.5

To plot this line, you can choose two points on it and draw a line passing through them. For example, when x1 = 0, we have x2 = 0.5, and when x2 = 0, we have x1 = 0.5. So, the two points on the line are (0, 0.5) and (0.5, 0).

To determine the positive half-plane, you can choose any point that lies on one side of the line and plug its coordinates into the boundary equation. If the result is positive, the point is in the positive half-plane; otherwise, it is in the negative half-plane. For example, the point (1, 1) satisfies the equation 1 + 1 - 0.5 = 1.5 > 0, so it is in the positive half-plane. Therefore, you can put a + in the half-plane that contains the point (1, 1).

Question 4: Given the dataset below which has 3 features and a class label which is either 0 or 1, fill in values for the class label that would ensure that the root split of a decision tree will be x2. That is, you should put either a 0 or 1 in each row of the y column in the table below.

x1	x2	x3	y
0	0	0	**0**
0	0	1	**0**
0	1	0	**1**
0	1	1	**1**
1	0	0	**0**
1	0	1	**0**
1	1	0	**1**
1	1	1	**1**

To ensure that the root split of a decision tree will be x2, we need to choose the values of y such that they vary only with changes in x2 and remain constant for all values of x1 and x3. In other words, we need to assign a value of y such that it depends only on the value of x2 and not on the values of x1 and x3.

One way to achieve this is to set the value of y to be equal to x2 for all rows. With this assignment of y, the root split of a decision tree would always be based on the value of x2.

Question 5: Explain why logistic regression is trained using **gradient ascent** instead of gradient descent.

Tries to maximize the probability of the class label.

Logistic regression is a type of machine learning algorithm used for binary classification problems. In logistic regression, the goal is to find the optimal weights that maximize the likelihood of the observed data.

Unlike in linear regression, where the goal is to minimize the sum of squared errors, logistic regression involves the use of a sigmoid function to transform the output of a linear regression equation into a probability between 0 and 1.

Since the log-likelihood function for logistic regression is concave, maximizing it is equivalent to minimizing its negative. Therefore, we can use gradient ascent to update the

weights iteratively and find the maximum likelihood estimates. Gradient ascent moves the weights in the direction of the gradient that increases the likelihood of the observed data, leading to convergence to a local maximum.

In contrast, gradient descent would move the weights in the direction of decreasing the negative log-likelihood, which would lead to convergence to a local minimum instead of the desired local maximum. Therefore, logistic regression is trained using gradient ascent instead of gradient descent.

Question 6: Both k-means and kNN have a k parameter. Say for each algorithm whether overfitting is more likely with larger or smaller values of k and why.

For k-means, overfitting is more likely with smaller values of k because the model may assign too few clusters and end up fitting the noise in the data. For kNN, overfitting is more likely with larger values of k because the model may capture the idiosyncrasies of the training data too closely and not generalize well to new data.

(**Idiosyncrasies refer to unique or unusual features or patterns in a particular dataset that may not be representative of the underlying population**)

Question 7: Given the 2x2 contingency table below, what are recall and precision if y = 1 is the positive class? Given each of them as a ratio of whole numbers. Note that y is the true label and y' is the predicted label.

	y' = 1	y' = 0
y = 1	45	7
y = 0	4	22

Recall = TP / (TP + FN)
Precision = TP / (TP + FP)

Recall = 45 / (45 + 7) = 0.865
Precision = 45 / (45 + 4) = 0.918

Question 8: There is an MDP with 4 states (s1, s2, s3, s4) and two actions (A, B). Consider the Q-table below. What cell changes and to what value if the agent does a Q update after taking

action A in state s3, gets a reward of 4 and winds up in state s1? The learning rate and discount factor are both 0.5. Give your final answer as a single number.

	A	B
S1	4	8
S2	8	7
S3	16	8
S4	5	5

When the agent takes action A in state s3, gets a reward of 4, and ends up in state s1, the Q-value for the (s3, A) pair needs to be updated. To update the Q-value, we use the Q-learning update rule:

$Q(s, a) = Q(s, a) + \alpha [r + \gamma \max Q(s', a') - Q(s, a)]$

where:

s is the current state (s3 in this case)
a is the action taken (A in this case)
r is the reward received (4 in this case)
s' is the next state (s1 in this case)
a' is the action taken in the next state (unknown from the given information)
α is the learning rate (0.5 in this case)
γ is the discount factor (0.5 in this case)
From the Q-table, we can see that the initial Q-value for the (s3, A) pair is 16.

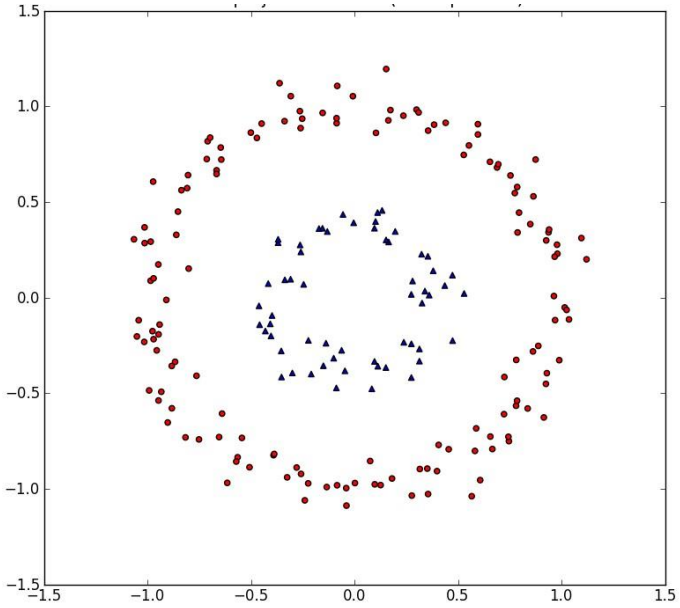
We don't know what the Q-value for the next state-action pair (s1, a') is, so we take the maximum Q-value over all actions in state s1 from the Q-table. From the table, we can see that the maximum Q-value in state s1 is 8 for action B.

Substituting the values into the Q-learning update rule, we get:

$Q(s3, A) = 16 + 0.5 [4 + 0.5 (8) - 16] = 12$

Therefore, the cell for (s3, A) in the Q-table changes from 16 to 12.

Question 9: Consider the dataset in the figure which has two features (x1, x2) and two classes (denoted with red and blue). Given the polynomial kernel $K(a, b) = (a \bullet b + 1)^d$, what is a value of d for which the data will be linearly separable in the higher dimensional space? Explain why that value of d works.



Scratch Paper