6.390: Midterm Exam, Fall 2024

Solutions

- This is a closed book exam. One page (8 1/2 in. by 11 in.) of notes, front and back, are permitted. Calculators are not permitted.
- The total exam time is 2 hours.
- The problems are not necessarily in any order of difficulty.
- Record all your answers in the places provided. If you run out of room for an answer, continue on a blank page and mark it clearly.
- If a question seems vague or under-specified to you, make an assumption, write it down, and solve the problem given your assumption.
- If you have a question, raise your hand or come to the front of the room.
- **Write your name on every piece of paper.**

Name: ______________________ MIT Email: ______________________

Question	Points	Score
1	24	
2	12	
3	14	
4	12	
5	20	
6	18	
Total:	100	

Name: ______________________________

Classification

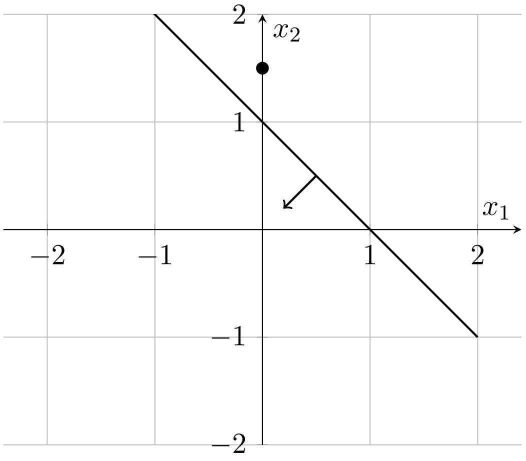
1. (24 points) Recall that a linear logistic classifier is characterized by

$$h(x; \theta, \theta_0) = \sigma(\theta^T x + \theta_0),$$

where $\sigma(\cdot)$ is the standard sigmoid function, $\sigma(z) = 1/(1 + \exp(-z))$. Define the argument of the sigmoid function in $h(\cdot)$ to be $z = \theta^T x + \theta_0$.

(a) i. On the graph below, draw the linear separator defined by the parameters $\theta = [-2, -2]^T$, $\theta_0 = 2$. Be sure to include a direction normal pointing in the direction of the positive class.

Solution:



ii. Would the corresponding linear logistic classifier assign the indicated point at $(x_1, x_2) = (0.0, 1.5)$ as positive or negative?

Solution: The indicated point is classified as negative.

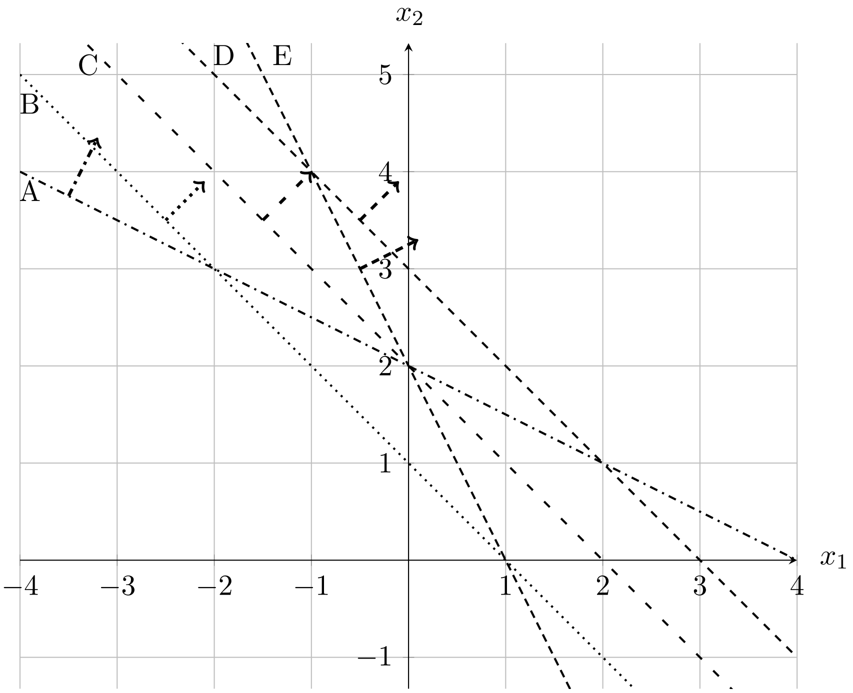
iii. What value of z does the classifier assign to the indicated point? (Your answer should be a specific number.)

Solution: $z = -2 * 0 - 2 * 1.5 + 2 = -1$

iv. What is the numerical probability output by the classifier? (You can leave a mathematical expression involving e, but your solution must not have matrix operations left to be done.)

Solution: $\sigma(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+e}$

(b) The following plot represents a two-dimensional space into which five separating hyperplanes for classifiers have been drawn, each with an associated normal vector intended to point toward the positive examples.



For each of the hypotheses parameterized by the values of the values of (θ, θ_0) below, identify the matching hyperplane from the plot above.

i. $\theta = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$, $\theta_0 = -2$

Solution: The correct answer is B. Conflict: C

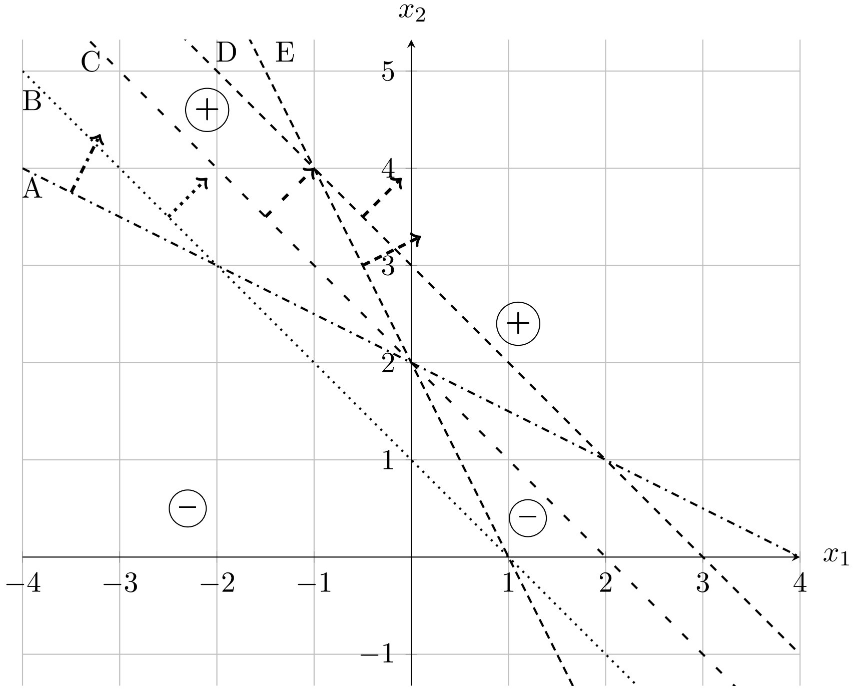
ii. $\theta = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$, $\theta_0 = -6$

Solution: The correct answer is D.

iii. $\theta = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$, $\theta_0 = -8$

Solution: The correct answer is A. Conflict: E

(c) The plot below is the same as from the previous part, but, now with some "held-out" data points that were not used in training the models. Each data point is labeled as positive (+) or negative (−).



Imagine the linear logistic classifier model with parameter $|\theta_1| = 1$ that corresponds to each of the drawn, oriented separating hyperplanes.

i. What is the accuracy of each of the models (A–E) on the set of held out data indicated?

Solution:

Model	Accuracy
A	100%
B	75%
C	100%
D	75%
E	50%

ii. Now using a NLL measure of Loss, which model has the minimum loss on the held out data? Justify your response. Hint: you do not need to explicitly compute the NLL value.

Solution: Both models A and C classify the correctly and would be expected to have lower loss than models that misclassify some of the data. We are also told that the parameters are on the same scale for all the models, and so we expect distance from the separating hyperplane to have a roughly similar contribution to each model's NLL Loss. Comparing model A and model C, two of the data points are quite close to the separating line for model C and are expected to contribute some significant loss. Whereas for model A, all of the points are relatively far. Thus, among the given models, model A is expected to have the minimum NLL loss on the held out data.

(d) Now, we would like to analyze a multiclass linear logistic classifier with $K = 3$ classes. For this part of the problem, we are still working with only 2 input features (x_1, x_2), but we choose to fold θ_0 into the θ matrix by adding a row to the bottom of the θ matrix representing the θ_0's and we add a 1 to the end of each x column vector. So, in this framing, let x be a data point, $x = [x_1, x_2, 1]^T$. Our θ will be a 3×3 matrix and let $z = \theta^T x$ be a 3×1 vector with $z = [z_1, z_2, z_3]^T$. Then, the output of the model will be defined as,

$$g = \text{softmax}(z) = \begin{bmatrix} \exp(z_1)/\sum_{i=1}^{3} \exp(z_i) \\ \exp(z_2)/\sum_{i=1}^{3} \exp(z_i) \\ \exp(z_3)/\sum_{i=1}^{3} \exp(z_i) \end{bmatrix}.$$

Recall that the vector g represents the likelihood assigned to each of the three classes, and the class prediction is the made from the largest element of g.

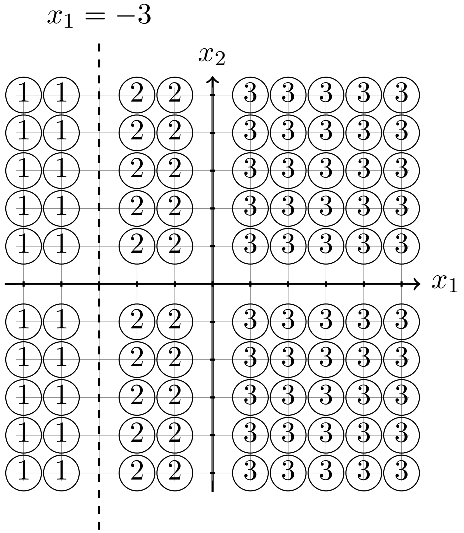
i. Suppose that we have a model defined by the following matrix:

$$\theta = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Consider the data point $x = [1, -1, 1]^T$. Compute $z = \theta^T x$ and determine which class will be assigned to x.

Solution: $z = [1, -1, 0]^T$; therefore, x will be assigned to class 1.
Conflict: $z = [-1, 1, 0]^T$; therefore, x will be assigned to class 2.

ii. Examine the classification problem represented by the graph below, where points are labelled with their class: $1, 2,$ or 3. Also given below is a model represented by the matrix θ (note that this is θ and so the first column represents $\theta_1, \theta_2,$ and θ_0 for class 1).

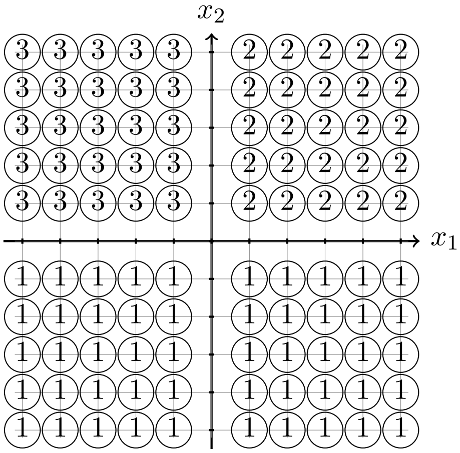


$$\theta = \begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ -3 & 0 & 0 \end{bmatrix}$$

Does the provided model defined by θ perfectly separate the data as desired in the graph above? If yes, show your reasoning. If no, identify all data points in the graph above that are misclassified.

Solution: Yes; we have that $z = [-x_1 - 3, 0, x_1]^T$. This implies that we will get a label of class 1 when $x_1 \leq -3$, class 2 when $x_1 > -3$ and $x_1 \leq 0$, class 3 when $x_1 > 0$.

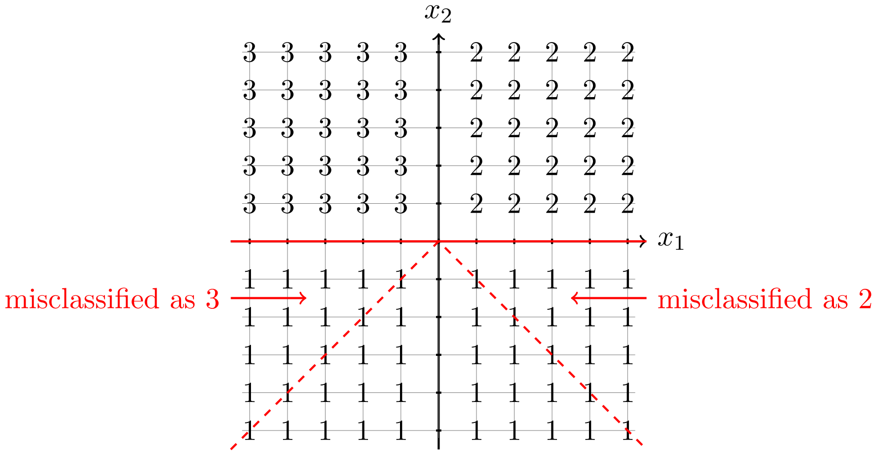
iii. Examine the classification problem represented by the graph below, where points are labelled with their class: $1, 2,$ or 3. Also given below is a model represented by the matrix θ (note that this is θ and so the first column represents $\theta_1, \theta_2,$ and θ_0 for class 1).



$$\theta = \begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Does the provided model defined by θ perfectly separate the data as desired in the graph above? If yes, show your reasoning. If no, identify all data points in the graph above that are misclassified.

Solution: No; we have that $z = [-x_2, x_1, -x_1]^T$. This implies that we will get a label of class 1 when $x_2 \leq 0$ and $|x_1| < -x_2$, class 2 when $x_1 \geq 0$ and $x_1 > -x_2$, class 3 either when $x_1 < 0$ and $x_2 > 0$ or when $x_2 < 0$ and $x_1 < x_2$. The misclassified regions are indicated in the figure below.



Name: ____________________

Feathery Featurizations

2. (12 points) In a popular bird-themed board game, players collect bird cards, each representing a different species. Each card has several attributes relevant to the game mechanics, such as the bird's wingspan, the type of nest it builds, the maximum number of eggs it can hold, and the food required to play the card. To use these bird cards in a machine learning model, these attributes need to be transformed into numerical features that the model can interpret.

   In this problem, you will determine how to encode the features of these birds and encode specific examples. Describe how you would encode each of the features below for use in a machine learning model and the dimensions of the encoded feature. Consider different types of encoding (e.g., binary, one-hot encoding, thermometer, normalization, etc.) and explain your reasoning for each feature.

   (a) Nest Type: Birds can build different types of nests (bowl, cavity, platform, or ground). A bird can only have one of these types of nests or a "wildcard" nest, meaning that their nest counts for any and all types of nests.

   Write the encoding of: (1) a bird that builds a bowl nest and (2) a bird that builds a wildcard nest.

   Solution: A binary length 4 encoding where [1, 0, 0, 0] (bowl) and [1, 1, 1, 1] (wild).

   (b) Habitat: Each bird can live in one or more of three habitats: forest, grassland, or wetland.

   Write the encoding of: a bird that can live in the forest or grasslands.

   Solution: Binary encoding per habitat. Bird that can live in forest or grasslands but not wetland would be [1, 1, 0].

   (c) Wingspan: A continuous feature representing the bird's wingspan (in centimeters). The minimum wingspan is 0 cm and maximum is roughly 300 cm.

   Write the encoding of: a bird with a wingspan of 50cm.

   Solution: Normalized real positive number.

(d) Egg Limit: Each bird card specifies how many eggs the bird can hold, represented as an integer. The minimum number of eggs is zero and maximum number of eggs is eight.

Write the encoding of: a bird that can hold 4 eggs.

Solution: Thermometer encoding of size 8, i.e. [1, 1, 1, 1, 0, 0, 0, 0] for 4 egg capacity encodes the fact that the number of eggs is discrete and the similarity of 4 egg capacity vs. 5 egg capacity.

We gave partial credit for real numbers (standardized and not) because this is an encoding that likely lead to an okay model, but it does allow for invalid encodings (encoding birds that are not in-distribution for the game).

(e) Food Cost: Each bird requires a specific combination of food to play the card. There are 5 types of food (e.g., Invertebrate, Seed, Fruit, Fish, Rodents) and birds cost no more than three food. Example to encode: A bird that costs 2 Fruit and 1 Seed to play.

Solution: There are multiple answers that are reasonable here.

- Integer encoding per food item, e.g. [0, 1, 2, 0, 0] or could divide by three to standardize between 0 and 1, e.g. [0, 1/3, 2/3, 0, 0].

- Thermometer encoding for each category (5 food types) × (3 max number of food items).

```
[[0, 0, 0],
 [1, 0, 0],
 [1, 1, 0],
 [0, 0, 0],
 [0, 0, 0]]
```

- (Partial Credit) One-hot encoding for each food slot (5 food types) × (3 max number of food items). Note, that this encoding is ambiguous – you need to not only order the food categories, but also the order in which the food is ordered in the slots. For example, 2 Fruit and 1 Seed could be encoded

```
[[0, 1, 0, 0, 0],
 [0, 0, 1, 0, 0],
 [0, 0, 1, 0, 0]]
```

or

```
[[0, 0, 1, 0, 0],
 [0, 1, 0, 0, 0],
 [0, 0, 1, 0, 0]]
```

or

```
[[0, 0, 1, 0, 0],
 [0, 0, 1, 0, 0],
 [0, 1, 0, 0, 0]]
```

and this ambiguity makes it an unfavorable encoding. However, one could use the convention that food is ordered in the three slots e.g. first list Invertebrate, then Seed, then Fruit, then Fish, and then Rodents, but it must be specified. Furthermore, the ordering does not easily show a relationship between needing 1 vs. 2 of a food.

- (Partial Credit) One-hot encoding of the number in each category. Similar to above, this does not show any relationship between 0, 1, 2, and 3 of that food type, so we gave partial credit.
- (Partial Credit) One-hot encoding of all possible combinations, six food options and three max food slots (6 choose $3 = 20$). Encoding does not show relationships between different food costs, so partial credit.

(f) Points: Each bird card is worth a certain number of points, which is an integer value (a minimum of 0 and maximum of 10).

Write the encoding of: a bird worth 7 points.

Solution: Thermometer encoding of size 10: [1, 1, 1, 1, 1, 1, 1, 0, 0, 0]. Standardized real positive value also received full credit. Could normalize 10 to be 1, so a bird worth 7 would be 0.7.

Random Descent

3. (14 points) Your friend Jordan is interested in learning algorithms for producing linear regression models. However, they have resolved to not compute any gradients. Instead, they decide to come up with their own iterative learning algorithm, Random Descent. At every iteration, we randomly decide to increase or decrease the parameter values by the learning rate. If this change results in a lower mean-squared error (MSE), the update is accepted; otherwise, the parameters remain unchanged. Pseudo-code for learning two parameters, a, b, with Random Descent is as follows:

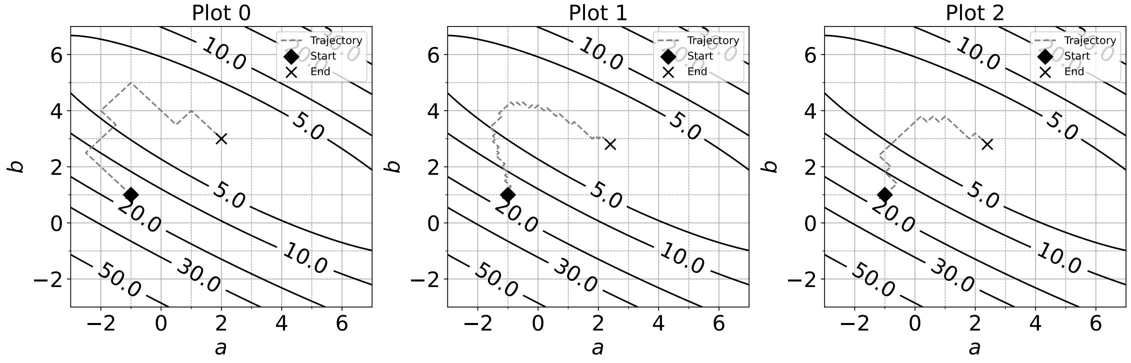
```
def random_descent(a, b, X, Y, ...
    max_iter, learning_rate, decay=1):
    # Compute initial error
    error = MSE(a, b, X, Y)

    # Iterative loop for random descent
    for iter in range(max_iter):
        # Propose new values for a and b
        a_new = a + coin_flip([-1, 1]) * learning_rate
        b_new = b + coin_flip([-1, 1]) * learning_rate

        # Compute new error
        new_error = MSE(a_new, b_new, X, Y)

        # Accept new parameters if error decreases
        if new_error < error:
            a = a_new
            b = b_new
            error = new_error
            learning_rate = learning_rate*decay

    # Return final learned parameters
    return a, b
```

Here, `coin_flip` is used to randomly pick whether to increase or decrease the parameter value with equal probability. Suppose that Jordan has some data set $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{100}$ where each $x^{(i)} \in \mathbb{R}^2$ is a 2-dimensional feature vector and $y^{(i)} \in \mathbb{R}$ is its corresponding label. Each column of `X` is a feature vector, and each "column" of `Y` is the corresponding target output value. The `error` is then computed to be

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - h(x^{(i)}; \{a, b\}) \right)^2 .$$

(a) In Jordan's first data set, the feature vectors take the form $x^{(i)} = [x_1^{(i)}, 1]^T$. Their hypotheses take the form $h_1(x; \{a, b\}) = ax_1 + b$. Jordan runs three different instances of Random Descent, each initialized with `a=-1, b=1, max_iter=1000, decay=1`, and using learning rates 0.1, 0.2, and 0.5. However, they lost track of which rate corresponds to which of the three instances.

In each panel below is a contour plot showing lines of constant `error`, visualizing the trajectory of one instance of minimizing the MSE with random descent:



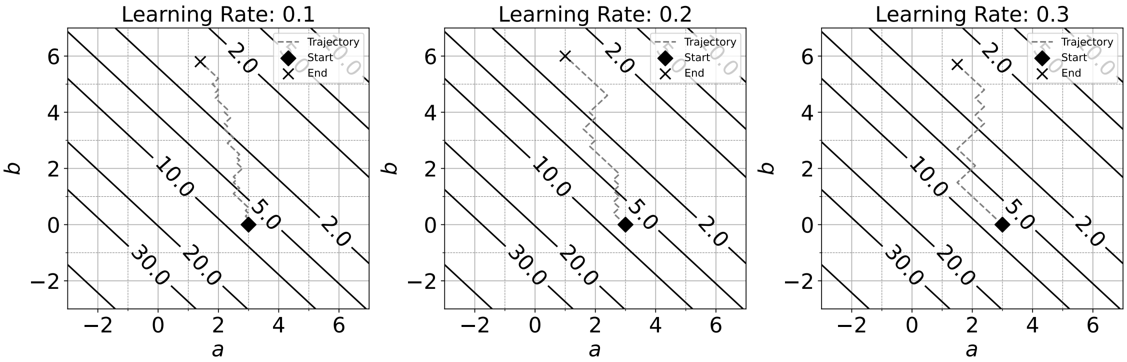
Match each of the three plots with the learning rate that was used to generate it. Each choice of learning rate was used exactly once.

Solution: Plot 0: 0.5, Plot 1: 0.1, Plot 2: 0.2

(b) Consider the left-most plot in Part (a). What are the learned parameter values for a, b? Round a, b to the closest integer.

Solution: $a = 2, b = 3$

(c) In Jordan's second data set the feature vectors take the form $x^{(i)} = [x_1^{(i)}, x_2^{(i)}]^T$. Their hypotheses take the form $h_2(x; \{a, b\}) = ax_1 + bx_2$. Jordan runs three instances of Random Descent, each initialized with `a=3, b=0, max_iter=1000, decay=1`, but with different learning rates. Three contour plots showing lines of constant `error` are shown below, corresponding to these three runs, with the learning rate as indicated in the title:



An oracle approaches, who claims to be omniscient. She tells Jordan that the best hypothesis takes the form $h(x) = 4x_1 + 3x_2$.

Does this claim have any merit? Should we trust this oracle? Mark all that are true and explain your reasoning.

Solution:

○ The results of our experiments are enough to disprove the oracle's claim. Random Descent learned the uniquely best model.

√ **Based on the contour plots, parameter values of $a = 4, b = 3$ would correspond to a hypothesis which minimizes the MSE.**

○ Given more iterations, every instance of Random Descent ran on this data set would eventually converge to $a = 4, b = 3$.

○ We need access to a larger training data set so that we can be more confident in our learned parameter values.

√ **We need to evaluate the hypotheses on a held-out data set.**

From the contour plot, we can see that the features are linearly dependent; more training data will not fix this issue. This is because the contour plot demonstrates that our objective function has a "half-pipe" shape, and there will be infinitely many points which minimize the MSE, including (4,3) and (1,6). The half-pipe is a convex shape; there is no concern for local optima. In any case, the only way to determine which model performs best is to analyze its performance on unseen data.

(d) Jordan would like to start using a `decay` with the learning rate such that progressively smaller steps will be taken at each iteration. They would like to utilize 4-fold cross validation in order to find a `decay` value that produces hypotheses that generalize to held-out validation data.

Fill in the blanks in the pseudocode below to implement 4-fold cross validation.

```
1.  decays = [0.8,0.9,0.99,0.999]

2.  for i = 1 to 4:

3.      Divide X,Y into _________________________

4.      for j = 1 to 4:

5.          a[j],b[j] = random_descent(0,0,_______,______,1000,0.3,__________)

6.          error[j] = ______________

7.      avg_error[i] = ________________

8.  best_decay = _____
```

Solution:

```
1.  decays = [0.8,0.9,0.99,0.99]
2.  for i = 1 to 4:
3.      Divide X,Y randomly into four equal subsets,
        (Xi,Yi) i = 1, ..., 4
4.      for j = 1 to 4:
5.          a[j],b[j] = random_descent(0,0,X without Xj,Y without Yj,
            1000,0.3,decays[i])
6.          error[j] = MSE(a[j],b[j],Xj,Yj)
7.      avg_error[i] = sum(error)/4
8.  best_decay = decays[argmin(avg_error)]
```

We're very generous with how "pseudo" the filled-in code may be. It was critical to identify the key components of K-fold cross validation: splitting the data set in K equally sized folds, ideally selected at random. We train a model on $K - 1$ folds and test on the held-out fold.

Logistic Mysteries

4. (12 points) The standard stochastic gradient descent algorithm is defined as

$$\Theta^{(t)} = \Theta^{(t-1)} - \eta(t)\nabla_\Theta f_{i(t)}(\Theta^{(t-1)}) \ (t = 1, 2, 3, \dots).$$

Here, $\Theta^{(t)} = [\theta^{(t)}, \theta_0^{(t)}]^T \in \mathbb{R}^2$ are two-dimensional vectors. Our objective is the negative log-likelihood function,

$$f_k(\Theta) = \mathcal{L}_{nll}(h(x^{(k)}, \Theta), y^{(k)}), \quad h\left(x, \begin{bmatrix}\theta \\ \theta_0\end{bmatrix}\right) = \sigma(\theta x + \theta_0).$$

$\sigma(\cdot)$ is the standard sigmoid function. At each iteration t, $i(t)$, is an integer selected randomly from $\{1, 2, \dots, n\}$. A variable learning rate $\eta(t) > 0$ is used.

Stochastic gradient descent was applied to minimize (with respect to Θ) the (non-regularized) logistic regression objective function

$$J(\Theta) = \sum_{k=1}^{n} f_k(\Theta)$$

for the training dataset $\{(x^{(j)}, y^{(j)})\}_{j=1}^{n}$ containing n training samples $(x^{(j)}, y^{(j)})$ where $x^{(j)} \in \mathbb{R}$ are input samples, and $y^{(j)} \in \{0, 1\}$ are the corresponding labels, for $j \in \{1, 2, \dots, n\}$. The resulting first few values of Θ are:

$$\Theta^{(0)} = \begin{bmatrix}3 \\ 0\end{bmatrix}, \quad \Theta^{(1)} = \begin{bmatrix}0 \\ 2\end{bmatrix}, \quad \Theta^{(2)} = \begin{bmatrix}2 \\ 3\end{bmatrix}, \quad \Theta^{(3)} = \begin{bmatrix}2 \\ 2\end{bmatrix}.$$

Recall that

$$\frac{\partial}{\partial\theta} f_k(\Theta) = x^{(k)}(h(x^{(k)}, \Theta) - y^{(k)}), \qquad \frac{\partial}{\partial\theta_0} f_k(\Theta) = h(x^{(k)}, \Theta) - y^{(k)}.$$

(a) Find $x^{(i(1))}, x^{(i(2))}, x^{(i(3))}$. Show your reasoning.

Solution: $x^{i(1)} = -1.5, \ y^{i(1)} = 1, \ x^{i(2)} = 2, \ y^{i(2)} = 1, \ x^{i(3)} = 0, \ y^{i(3)} = 0$

$\frac{\partial}{\partial\theta_0} f_i(\Theta) = g^{(i)} - y^{(i)}, \ \frac{\partial}{\partial\theta} f_i(\Theta) = (g^{(i)} - y^{(i)})x^{(i)}$

At iteration 1,

$2 = 0 - \eta(g^{(1)} - y^{(1)})$

$0 = 3 - \eta(g^{(1)} - y^{(1)})x^{(1)}$

$\Rightarrow -2 = \eta(g^{(1)} - y^{(1)}) < 0 \Rightarrow y^{(1)} = 1$

$0 = 3 + 2x^{(1)} \Rightarrow x^{(1)} = -\frac{3}{2}$

At iteration 2,

$3 = 2 - \eta(g^{(2)} - y^{(2)})$

$2 = 0 - \eta(g^{(2)} - y^{(2)})x^{(2)}$

$\Rightarrow -1 = \eta(g^{(2)} - y^{(2)}) < 0 \Rightarrow y^{(2)} = 1$

$2 = 0 + x^{(1)} \Rightarrow x^{(2)} = 2$

At iteration 3,
$2 = 3 - \eta(g^{(3)} - y^{(3)})$
$2 = 2 - \eta(g^{(3)} - y^{(3)})x^{(3)}$
$\Rightarrow 1 = \eta(g^{(i)} - y^{(3)}) > 0 \Rightarrow y^{(3)} = 0$
$2 = 2 - x^{(3)} \Rightarrow x^{(3)} = 0$
Conflict:
At iteration 3,
$1 = 3 - \eta(g^{(3)} - y^{(3)})$
$6 = 2 - \eta(g^{(3)} - y^{(3)})x^{(3)}$
$\Rightarrow 2 = \eta(g^{(i)} - y^{(3)}) > 0 \Rightarrow y^{(3)} = 0$
$6 = 2 - 2x^{(3)} \Rightarrow x^{(3)} = -2$ Alternate approach:
The update rule for SGD is given as

$$\Theta^{(t)} = \Theta^{(t-1)} - \eta(t)\,\nabla_\Theta f_{i(t)}(\Theta^{(t-1)}) \tag{1}$$

First, let's take this rule and break it up into θ and θ_0 so that we can see things more clearly.

$$\begin{aligned}\theta^{(t)} &= \theta^{(t-1)} - \eta(t)\,\frac{\partial}{\partial\theta} f_{i(t)}(\Theta) \\ &= \theta^{(t-1)} - \eta(t)\,x^{i(t)}\left(h(x^{i(t)}) - y^{i(t)}\right)\end{aligned} \tag{2}$$

$$\begin{aligned}\theta_0^{(t)} &= \theta_0^{(t-1)} - \eta(t)\,\frac{\partial}{\partial\theta_0} f_{i(t)}(\Theta) \\ &= \theta_0^{(t-1)} - \eta(t)\,\left(h(x^{i(t)}) - y^{i(t)}\right)\end{aligned} \tag{3}$$

Equation (2) and (3) seems very similar except for the $x^{i(t)}$ term, so let's rearrange these two equations to

$$\theta^{(t)} - \theta^{(t-1)} = -\eta(t)\,x^{i(t)}\left(h(x^{i(t)}) - y^{i(t)}\right) \tag{4}$$

and

$$\theta_0^{(t)} - \theta_0^{(t-1)} = -\eta(t)\,\left(h(x^{i(t)}) - y^{i(t)}\right) \tag{5}$$

Now, if we divide Equation (4) by (7), we can isolate the $x^{(i(t)}$ term!

$$\frac{\theta^{(t)} - \theta^{(t-1)}}{\theta_0^{(t)} - \theta_0^{(t-1)}} = x^{i(t)} \tag{6}$$

For your convenience, we repeat that the resulting first few values of Θ are:

$$\Theta^{(0)} = \begin{bmatrix} 3 \\ 0 \end{bmatrix}, \quad \Theta^{(1)} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}, \quad \Theta^{(2)} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \quad \Theta^{(3)} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}.$$

(b) Find $y^{(i(1))}, y^{(i(2))}, y^{(i(3))}$. Show your reasoning. Hint: recall that $y^{(j)} \in \{0, 1\}$.

Solution: $x^{i(1)} = -1.5, \;\; y^{i(1)} = 1, \;\; x^{i(2)} = 2, \;\; y^{i(2)} = 1, \;\; x^{i(3)} = 0, \;\; y^{i(3)} = 0$
$\frac{\partial}{\partial \theta_0} f_i(\Theta) = g^{(i)} - y^{(i)}$, $\frac{\partial}{\partial \theta} f_i(\Theta) = (g^{(i)} - y^{(i)}) x^{(i)}$
At iteration 1,
$2 = 0 - \eta(g^{(1)} - y^{(1)})$
$0 = 3 - \eta(g^{(1)} - y^{(1)}) x^{(1)}$
$\Rightarrow -2 = \eta(g^{(1)} - y^{(1)}) < 0 \Rightarrow y^{(1)} = 1$
$0 = 3 + 2x^{(1)} \Rightarrow x^{(1)} = -\frac{3}{2}$
At iteration 2,
$3 = 2 - \eta(g^{(2)} - y^{(2)})$
$2 = 0 - \eta(g^{(2)} - y^{(2)}) x^{(2)}$
$\Rightarrow -1 = \eta(g^{(2)} - y^{(2)}) < 0 \Rightarrow y^{(2)} = 1$
$2 = 0 + x^{(1)} \Rightarrow x^{(2)} = 2$
At iteration 3,
$2 = 3 - \eta(g^{(3)} - y^{(3)})$
$2 = 2 - \eta(g^{(3)} - y^{(3)}) x^{(3)}$
$\Rightarrow 1 = \eta(g^{(i)} - y^{(3)}) > 0 \Rightarrow y^{(3)} = 0$
$2 = 2 - x^{(3)} \Rightarrow x^{(3)} = 0$
Conflict:
At iteration 3,
$1 = 3 - \eta(g^{(3)} - y^{(3)})$
$6 = 2 - \eta(g^{(3)} - y^{(3)}) x^{(3)}$
$\Rightarrow 2 = \eta(g^{(i)} - y^{(3)}) > 0 \Rightarrow y^{(3)} = 0$
$6 = 2 - 2x^{(3)} \Rightarrow x^{(3)} = -2$
Alternate approach:
Now that we can solve for $x^{i(t)}$, let's solve for $y^{i(t)}$. First, note that $g^{i(t)} = h(x^{i(t)}, \Theta^{(t-1)}) = \sigma(\theta x^{i(t)} + \theta_0) \in (0, 1)$ due to the nature of sigmoid. Let's go back to Equation (7) since it contains y and is simpler than Equation (4).

$$\theta_0^{(t)} - \theta_0^{(t-1)} = -\eta(t) \left(g^{i(t)} - y^{i(t)} \right) = \eta(t) \left(y^{i(t)} - g^{i(t)} \right) \tag{7}$$

Since $\eta > 0$ (else this wouldn't be gradient descent!), we can state that

$$\text{sign}\left(\theta_0^{(t)} - \theta_0^{(t-1)} \right) = \text{sign}\left(y^{i(t)} - g^{i(t)} \right) = \begin{cases} \text{positive} & \text{if } y^{i(t)} = 1, \\ \text{negative} & \text{if } y^{i(t)} = 0, \\ 0 & \text{never} \end{cases} \tag{8}$$

The second inequality is true since $y^{i(t)} \in \{0, 1\}$ and $g^{i(t)} \in (0, 1)$. Note the difference between curly brackets (a finite set) and parenthesis (open interval) and square bracket (close interval). You should think about why this value cannot be 0 before convergence!

Plugging the given values from the main test in, we get that

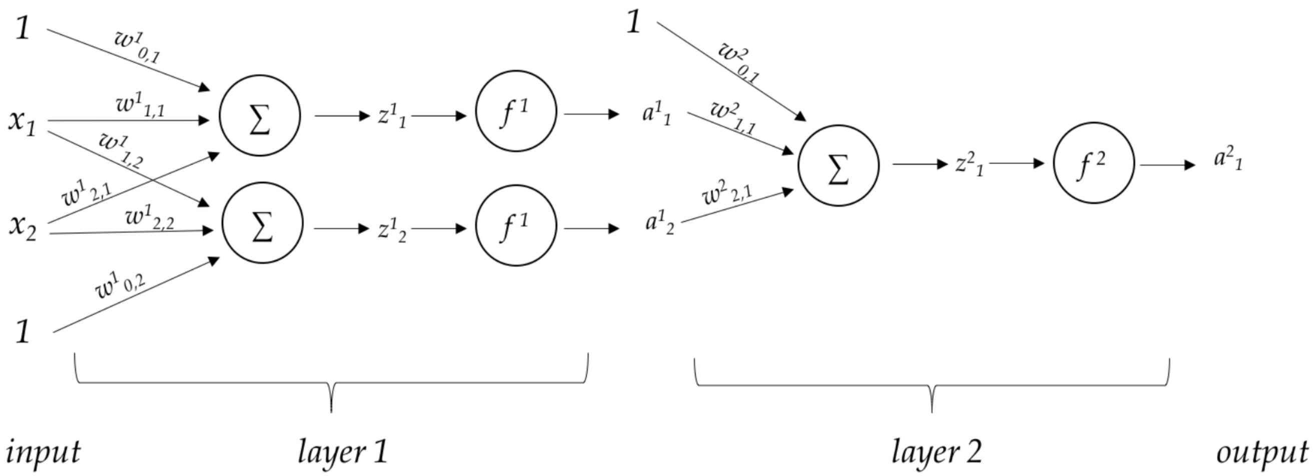
t	0	1	2	3
$\theta^{(t)}$	3	0	2	2
$\theta_0^{(t)}$	0	2	3	2
$\theta^{(t)} - \theta^{(t-1)}$		-3	2	0
$\theta_0^{(t)} - \theta_0^{(t-1)}$		2	1	-1
$x^{i(t)} = \frac{\theta^{(t)} - \theta^{(t-1)}}{\theta_0^{(t)} - \theta_0^{(t-1)}}$		-1.5	2	0
$y^{i(t)} = \text{sign}(\theta_0^{(t)} - \theta_0^{(t-1)})$		1	1	0

Slippery Slope

5. (20 points) Alice, Bob, and Carl are debating the utility of non-linear activation functions:

- Alice believes that ReLUs rule! (After all, it's right there in the name.)
- Bob thinks that we should always use sigmoids.
- Carl wants to invent their own activation function.

Consider the following neural network:



(a) We follow Alice's suggestion and keep layer-1 activation as ReLUs. Recall that the ReLU function is defined as $f(z) := \max(z, 0)$. Assume that we let the derivative of ReLU at $z = 0$ be zero. For this part, answer with a real number; if you believe the set up isn't enough to determine a real-valued answer, answer "it depends".

i. If $a_1^1 = 0.5$, what is $\frac{\partial a_1^1}{\partial z_1^1}$?

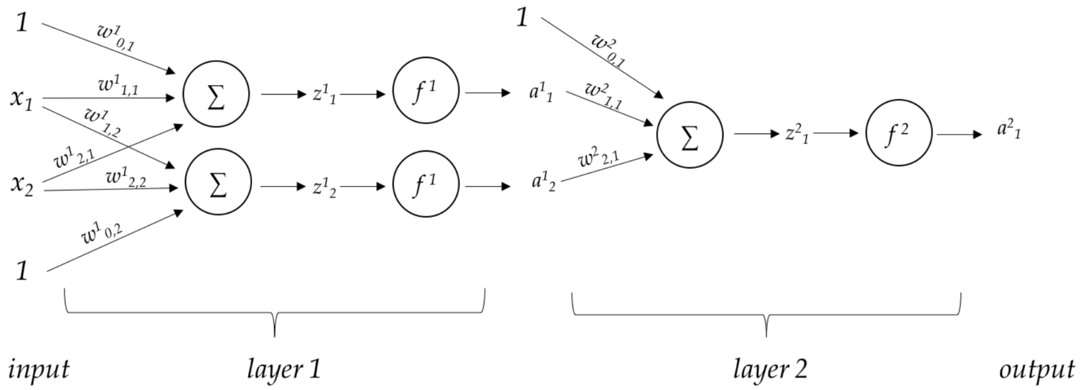
Solution: Gradient is 1

ii. If $a_1^1 = 0.5$, what is $\frac{\partial a_1^1}{\partial x_1}$?

Solution: It depends (on w_{11}^1.)

iii. If $a_1^1 = 0$, what is $\frac{\partial a_1^1}{\partial x_1}$?

Solution: Gradient is 0



(b) We now follow Alice's preference to have the layer-2 activation as ReLUs as well. Suppose the current weights are $w^2_{0,1} = 1, w^2_{1,1} = 1, w^2_{2,1} = 1$. The final output a^2_1 is 0. Using a step-size of 0.1, what would be the updated values of these weights?

Solution: The gradient is 0, so the weights are unchanged

(c) Suppose instead we follow Bob's desire to keep layer-1 activation as sigmoids. Recall that the sigmoid function is defined as $f(z) := \frac{1}{1+\exp(-z)}$. For this part, answer with a real number; if you believe the setup isn't enough to determine a real-valued answer, answer "it depends".

i. If $a^1_1 = 0.5$, what is $\frac{\partial a^1_1}{\partial z^1_1}$?

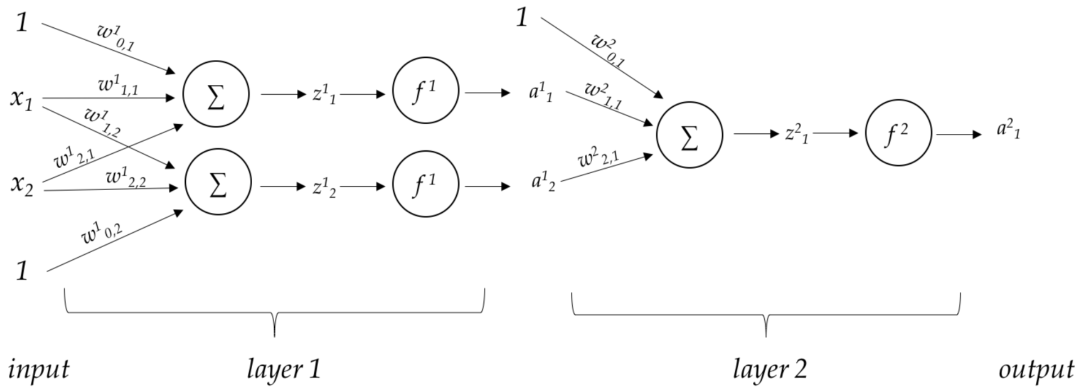
Solution: (1-0.5)*0.5 = 0.25

ii. If $a^1_1 = 0.2$, what is $\frac{\partial a^1_1}{\partial x_1}$?

Solution: It depends.

iii. If $a^1_1 = 0.8$, what is $\frac{\partial a^1_1}{\partial x_1}$?

Solution: It depends.



(d) Carl argues that both sigmoid and ReLU suffer from having a large portion of the input space where the gradients are zero (or almost zero) – which can make backpropagation very difficult.

i. Suppose that we used sigmoids in our network (reproduced above for your convenience). In particular, during an SGD update, these sigmoid units may have (nearly) zero gradient, that is,

$$\frac{\partial a^i_j}{\partial z^i_j} \approx 0$$

where $i, j = 1, 2$.
Would you agree with Carl's argument that having near-zero gradients could be troublesome for learning? Explain your reasoning.

Solution: Zero-gradient issues can be troublesome during training because they halt learning. In an SGD update, if all of the activation functions contribute a zero gradient, the partial derivative of the loss with respect to any of the model weights will be zero, due to the chain rule. Therefore, no updates will be made to the model parameters, meaning the model stops learning.

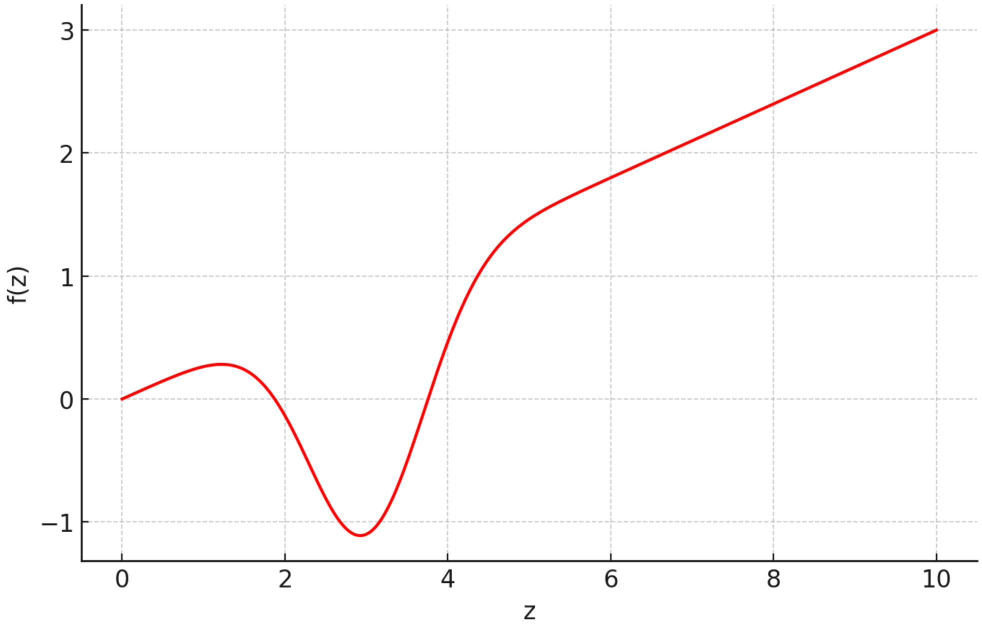
ii. Alice and Bob argue that in reality people use either sigmoids or ReLU for good reason, and that the scenario that Carl described do not happen that frequently in practice. What might be the reason? Explain your reasoning.

Solution: In practice, there are multiple data points and multiple units, and we randomly sample in SGD – we may not frequently get "unlucky" data points such that all activation functions operate in a region with zero gradient (this is especially true if the network is deep, and/or if there're lots of data points).

(e) Carl proposes a "Nike"-like swoosh type of activation function, defined as

$$f(z) = -2 \cdot e^{-1\cdot(z-3)^2} + 0.3 \cdot z$$

and graphed below:



If we use this activation function in the first layer, and let $z^1 = [z^1_1, z^1_2]^T$ be the first layer pre-activation output, and $a^1 = [a^1_1, a^1_2]^T$ be the first layer post-activation output, what would be $\frac{\partial a^1}{\partial z^1}$? (We are looking for a symbolic answer only, not a specific number.)

Solution: A 2-by-2 diagonal matrix:

$$\begin{pmatrix} -2(6-2z^1_1)\cdot e^{-(z^1_1-3)^2} + 0.3 & 0 \\ 0 & -2(6-2z^1_2)\cdot e^{-(z^1_2-3)^2} + 0.3 \end{pmatrix}$$

Judging by the First Steps

6. (18 points) Let $f : \mathbb{R} \to \mathbb{R}$ be an *unknown* function. Consider the following additional assumptions (A)-(C) one *could* make about $f(\cdot)$:

   (A) $f(\cdot)$ is differentiable everywhere;

   (B) $f(\cdot)$ is differentiable everywhere and convex;

   (C) $f(\cdot)$ is the regularized linear regression objective function

   $$f(\theta) = \lambda\theta^2 + \frac{1}{n}\sum_{k=1}^{n}(\theta x_k - y_k)^2$$

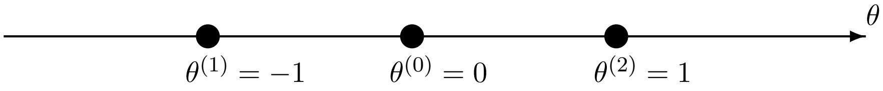
   for some $\lambda > 0$, $n \in \{1, 2, 3, \dots\}$, and real $x_1, \dots, x_n$, $y_1, \dots, y_n$.

   The standard gradient descent algorithm

   $$\theta^{(t)} = \theta^{(t-1)} - \eta\nabla_\theta f(\theta^{(t-1)}) \quad (t = 1, 2, \dots)$$

   with initial guess $\theta^{(0)} = 0$ and some positive, fixed learning rate $\eta > 0$ is applied to try to find an argument of minimum of $f(\cdot)$ numerically. Knowing that the first two steps of this algorithm have resulted in $\theta^{(1)} = -1$ and $\theta^{(2)} = 1$, as shown below, what can be learned about $f(\cdot)$ based on this information?

t	$\theta^{(t-1)}$	$\theta^{(t)}$
1	0	-1
2	-1	1
3	1	?



   (a) First, consider assumption (C). As the objective function is quadratic, its gradient $\nabla_\theta f(\theta)$ will be a linear function of the form $\nabla_\theta f(\theta) = \theta a + b$. Write down a and b in terms of the data $\{x_i, y_i\}_{i=1}^{n}$ and regularization hyperparameter λ.

Solution:

$$a = 2\lambda + \frac{2}{n}\sum_{k=1}^{n} x_k^2$$

$$b = -\frac{2}{n}\sum_{k=1}^{n} x_k y_k$$

Explanation:
Under assumption (C), f is

$$f(\theta) = \lambda\theta^2 + \frac{1}{n}\sum_{k=1}^{n}(\theta x_k - y_k)^2 \tag{9}$$

So the gradient is

$$\begin{aligned}\nabla_\theta f(\theta) &= 2\lambda\theta + \frac{2}{n}\sum_{k=1}^{n}(\theta x_k - y_k)x_k \\ &= \left(2\lambda + \frac{2}{n}\sum_{k=1}^{n} x_k^2\right)\theta + \left(-\frac{2}{n}\sum_{k=1}^{k} x_k y_k\right)\end{aligned} \tag{10}$$

From here, we can infer that

$$a = 2\lambda + \frac{2}{n}\sum_{k=1}^{n} x_k^2 \tag{11}$$

and

$$b = -\frac{2}{n}\sum_{k=1}^{k} x_k y_k \tag{12}$$

(b) Find the set Θ_3 of all possible values of $\theta^{(3)}$.

i. under assumption (C):
Hint: Your answer should be a real number.

Solution: $\Theta_3 = \{-3\}$ Conflict: $\Theta_3 = \{-\frac{5}{2}\}$
Explanation: Under assumption (C) and from part (a), we can state that the gradient update rule is

$$\theta^t = \theta^{t-1} - \eta\nabla_\theta f(\theta^{t-1}) = \theta^{t-1} - \eta(a\theta^{t-1} + b) = (1 - \eta a)\theta^{t-1} - \eta b \tag{13}$$

Note that a and b are constant since this is not stochastic gradient descent and η is fixed, so from the given iterations of θ, we can solve for ηa and ηb.
At $t = 1$, the update rule is

$$\begin{aligned}\theta^{(1)} &= \theta^{(0)} - \eta a\theta^{(0)} - \eta b\\ -1 &= 0 - \eta a(0) - \eta b\\ \eta b &= 1\end{aligned} \tag{14}$$

At $t = 2$, the update rule is

$$\begin{aligned}\theta^{(2)} &= \theta^{(1)} - \eta a\theta^{(1)} - \eta b\\ 1 &= -1 + \eta a - 1\\ \eta a &= 3\end{aligned} \tag{15}$$

Now we can plug the values for ηa and ηb into the update rule to get

$$\theta^t = (1 - \eta a)\theta^{t-1} - \eta b = -2\theta^{t-1} - 1 \tag{16}$$

When $t = 2$, we have that

$$\theta^{(3)} = -2\theta^{(2)} - 1 = -2(1) - 1 = -3 \tag{17}$$

Therefore, $\Theta_3 = \{-3\}$.

ii. under assumption (B):
Hint: Use the fact that the derivative of a convex scalar function is non-decreasing.

Solution: $\Theta_3 = (-\infty, 0]$
Explanation:
Using the hint, we know that the gradient at $\theta = 1$ is weakly greater than the gradient at $\theta = 0$. This means that, using the fixed η, the magnitude of the update step at at $\theta = 1$ is bigger or equal to the magnitude of the update step at $\theta = 0$, which was 1 unit to the left. Therefore, after the update step for $\theta = 1$, we will end up somewhere to the left of $\theta = 0$. The final solution is then $\Theta_3 = (-\infty, 0]$, inclusive of 0 because the gradient is *weakly* greater instead of strictly greater.

iii. under assumption (A):

Solution: $\Theta_3 = (-\infty, +\infty)$
Explanation:
Under assumption (A), we can design the gradient of f at $\theta = 1$ such that the next θ value after update could be any number on the real, so the answer is $\Theta_3 = \mathbb{R} = (-\infty, \infty)$

(c) Find the set $\Theta_{\min}$ of all possible values of the argument of minimum of f (if it has one).

i. under assumption (C):
Hint: Your answer should be a real number.

Solution: $\Theta_{\min} = \{-1/3\}$ Conflict: $\Theta_3 = \{-\frac{4}{5}\}$
Explanation: The gradient of f is $a\theta + b$, and it equals zero only when θ is at its minimum under assumption (C). Therefore, $\theta_{min} = \frac{-b}{a} = \frac{-\eta b}{\eta a} = \frac{-1}{3}$, plugging in the values we found in (15) and (14). The final answer is then $\Theta_{min} = \{\frac{-1}{3}\}$.

ii. under assumption (B):
Hint: You may want to try drawing a rough sketch of $\nabla_\theta f(\theta)$ from the information you have.

Solution: $\Theta_{\min} = (-1, 0)$
Explanation:
The update step at $\theta = 0$ moves θ to the left by one unit, and then the update step at $\theta = 1$ moves it to the right by two units, overshooting the minimum and will eventually zigzag its way to positive infinity. Therefore, the minimum is somewhere in the interval $\Theta_{min} = (-1, 0)$. This is non inclusive since the gradient at the end points are not zero (else we would not move after update step).

iii. under assumption (A):

Solution: $\Theta_{\min} = (-\infty, -1) \cup (-1, 0) \cup (0, +\infty)$
Explanation:
As with the case in part (a), we can design the function f to have the minimum point anywhere on the real except for $\theta = -1$ and $\theta = 0$. This is because from the set up, the gradient at these points are non-zero (else we would not move after update step), implying that the minimum is not here. Therefore, the answer is $\Theta_{min} = (-\infty, -1) \cup (-1, 0) \cup (0, \infty)$.

(d) Is it possible that, as the sequence of gradient descent steps continues starting from $\theta^{(0)} = 0$, $\theta^{(1)} = -1$, $\theta^{(2)} = 1$, that $\theta^{(t)}$ will converge to the argument of minimum of $f(\cdot)$ as $t \to +\infty$?

i. under assumption (C):

Solution:

○ Possible

√ **Impossible**

Explanation: From the first two iterations, we can see that the learning rate η is too large and we are diverging.

ii. under assumption (B):

Solution:

√ **Possible**

○ Impossible

iii. under assumption (A):

Solution:

√ **Possible**

○ Impossible