

## 6.036: Final Exam, Fall 2021

# Solutions

- This is a closed book exam. Two pages (8 1/2 in. by 11 in.) of notes, front and back, are permitted. Computers, phones, and other electronics are not permitted.
- You have 3 hours.
- The problems are not necessarily in any order of difficulty.
- Write all your answers in the places provided. If you run out of room for an answer, indicate that you are continuing your answer, use the provided blank page at the end, and mark clearly what question is being continued.
- If a question seems vague or under-specified to you, make an assumption, write it down, and solve the problem given your assumption.

Name: \_\_\_\_\_

Kerberos (MIT username): \_\_\_\_\_

Question:	1	2	3	4	5	6	7	8	Total
Points:	14	12	12	12	14	12	12	12	100
Score:									

## App Store

1. (14 points) Mac O'Larnin is considering selling an app on Frugal Play. You have a friend with inside info at Frugal, and they're able to share data on how previous apps have performed on the store.

Mac decides that he will learn a neural network with no hidden layer (i.e., consisting only of the output layer). He needs help in figuring out the precise formulation for machine learning.

- (a) For each of the following app characteristics, suggest the best way of encoding it as feature(s) to be input to the neural network. Choose from among the following: multiple unary features (one-hot encoding), multiple binary features (thermometer encoding), an integer or real-valued feature. Also **give the exact function that maps each input to its corresponding feature(s)**.

**Solution:**

Genre (Game, Productivity, Education, Information, Social): One-hot, with a bit for each possible genre: Game  $\rightarrow$  10000, Productivity  $\rightarrow$  01000, Education  $\rightarrow$  00100, Information  $\rightarrow$  00010, Social  $\rightarrow$  00001

Suitable for people ages (2-4, 5-10, 11-15, 16 and over): Thermometer, because order should be preserved: 2-4: 1000 ; 5-10: 1100, 11-15: 1110, 16 and over: 1111

Was it banned in any previous quarter (True, False): Single binary feature, True: 1, False: 0. We also accepted a True/False encoding since Python correctly does arithmetic with it.

Price of the app (positive number): Real-value, may standardize it using  $(x - \mu)/\sigma$  for  $\mu$  being the mean and  $\sigma$  the standard deviation

Does it have in-game advertising (True, False): Single binary feature, True: 1, False: 0. We also accepted a True/False encoding since Python correctly does arithmetic with it.

Name: \_\_\_\_\_

- (b) Mac wants to predict the sales volume (how many times someone will purchase the app each month) for his new app. The sales volume can be negative if many people returned the app for a refund in a given month. What should Mac choose for the number of units in the output layer, the activation function(s) in the output layer (linear, ReLU, sigmoid, softmax), and the loss function (negative log likelihood, quadratic)?

**Solution:** One unit, because the output is an integer, a linear activation function, and a quadratic loss function.

- (c) Mac also has data on several other properties of interest, including
- whether an app was featured on the front page
  - whether it got a favorable review on the Coolest Apps Ever web site
  - whether Orange Computer offered to pay to port the app to their site

He would like to train a new neural network to predict these three properties.

For this new prediction task, what should Mac choose for the number of units in the output layer, the activation function(s) in the output layer (linear, ReLU, sigmoid, softmax), and the loss function (negative log likelihood, quadratic)?

**Solution:** Three units, sigmoid activation function for each, and training should use a negative log likelihood loss function.

Name: \_\_\_\_\_

- (d) Mac's first attempt at machine learning to predict the sales volume (setup of (b)) uses all customer data from 2020. He randomly partitions the data into train (80%) and validation (20%), and uses the same number of units, activation function(s), and loss function as in (b). To prevent overfitting, he uses ridge regularization of the weights  $W$ , minimizing the optimization objective

$$J(W; \lambda) = \sum_{i=1}^n \mathcal{L}(h(x^{(i)}; W), y^{(i)}) + \lambda \|W\|^2,$$

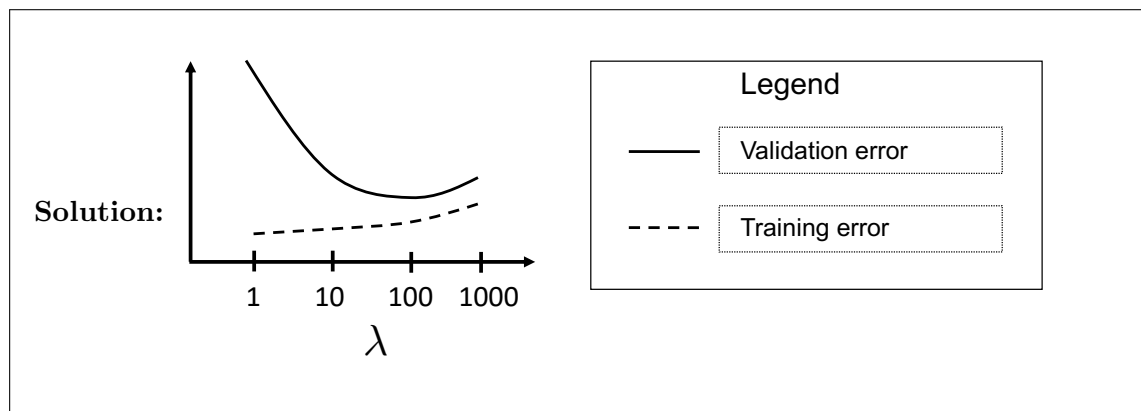
where  $\|W\|^2$  is the sum over the square of all output units' weights.

Mac discovers that it's possible to find a value of  $W$  such that  $J(W; \lambda) = 0$  even when  $\lambda$  is very large, nearing  $\infty$ . Mac suspects that he might have an error in the code that he wrote to derive the labels (i.e., the monthly sales volumes). Let's see why. First, what can Mac conclude about  $W$  from this finding? Second, what does this imply about the labels?

**Solution:** (1) Since the loss is always non-negative and the penalty is always non-negative, the only way to get 0 here is for both to be equal to 0. The only way the penalty can equal 0 is if every element of  $W$  equals 0.

(2) When  $W$  has all entries equal to 0, the prediction at every data point is a constant (the offset). The only way for the squared error to be 0 is for the label of every data point to equal that offset. It seems unlikely that every data label would be exactly the same in this data set, which we assume ranges over a wide number of apps.

- (e) Mac found and fixed the error. Now, to choose the regularization constant  $\lambda$ , Mac tried values of 1, 10, 100, and 1000, creating the below plot. Unfortunately, he forgot to label the legend! Help Mach by filling in the legend using two of the following: 'Training error', 'Validation error', 'Training time'.



Name: \_\_\_\_\_

- (f) Continuing the scenario of (e), which value of  $\lambda$  (out of 1, 10, 100, and 1000) should Mac choose to obtain the neural network that he will deploy on the app store, and why?

**Solution:**  $\lambda = 100$ , because the validation error is lowest at this value.

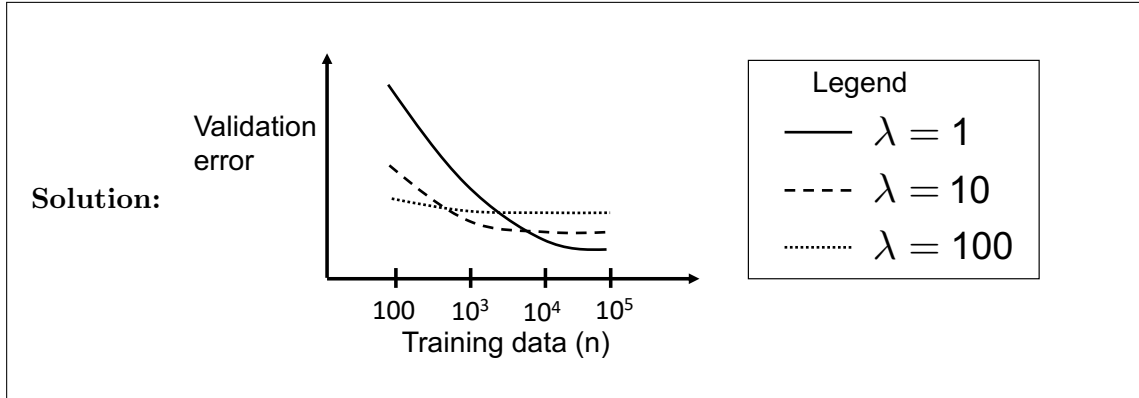
- (g) When Mac wakes up the next day, he decides to re-run learning for the  $\lambda$  selected in (f), now with a different partition of the data into train and validation sets (since he had previously forgotten to set the random seed). He finds that he gets a very different validation error! To obtain a more stable estimate, Mac decides to split the data into 5 disjoint chunks of 20% of the data. For each chunk, he evaluates on it after training on the union of the other 4 chunks. He gets the following results for the average error within each chunk: 0.15, 0.3, 0.1, 0.2, 0.25. What can Mac conclude is an estimate of the test error of the neural network, and why?

**Solution:** 0.2 (the average). This is cross-validation.

Name: \_\_\_\_\_

- (h) The initial results look promising. Mac now wants to add in data from additional, earlier, years. (He is confident his customers have been behaving similarly over many years, so the earlier data is relevant.)

Before curating the older data, Mac decides to use the training data that he has to get a sense of whether more data would help. He creates a learning curve where on the horizontal axis he varies the amount of training data used and on the vertical axis he shows the validation error, using a fixed validation set across all settings considered. He experiments with  $\lambda = 1, 10, 100$ , but again forgot to include a legend. Fill in the below legend by labeling the curves with the value of  $\lambda$  that each corresponds to:



- (i) Based on these plots does it seem likely that even more data will improve validation error (possibly for a different value of  $\lambda$ )? Explain why or why not.

**Solution:** Yes, because the validation error continues to decline as the amount of regularization decreases and amount of data increases. With more data and  $\lambda = 0$ , it is conceivable that the validation error will be even smaller.

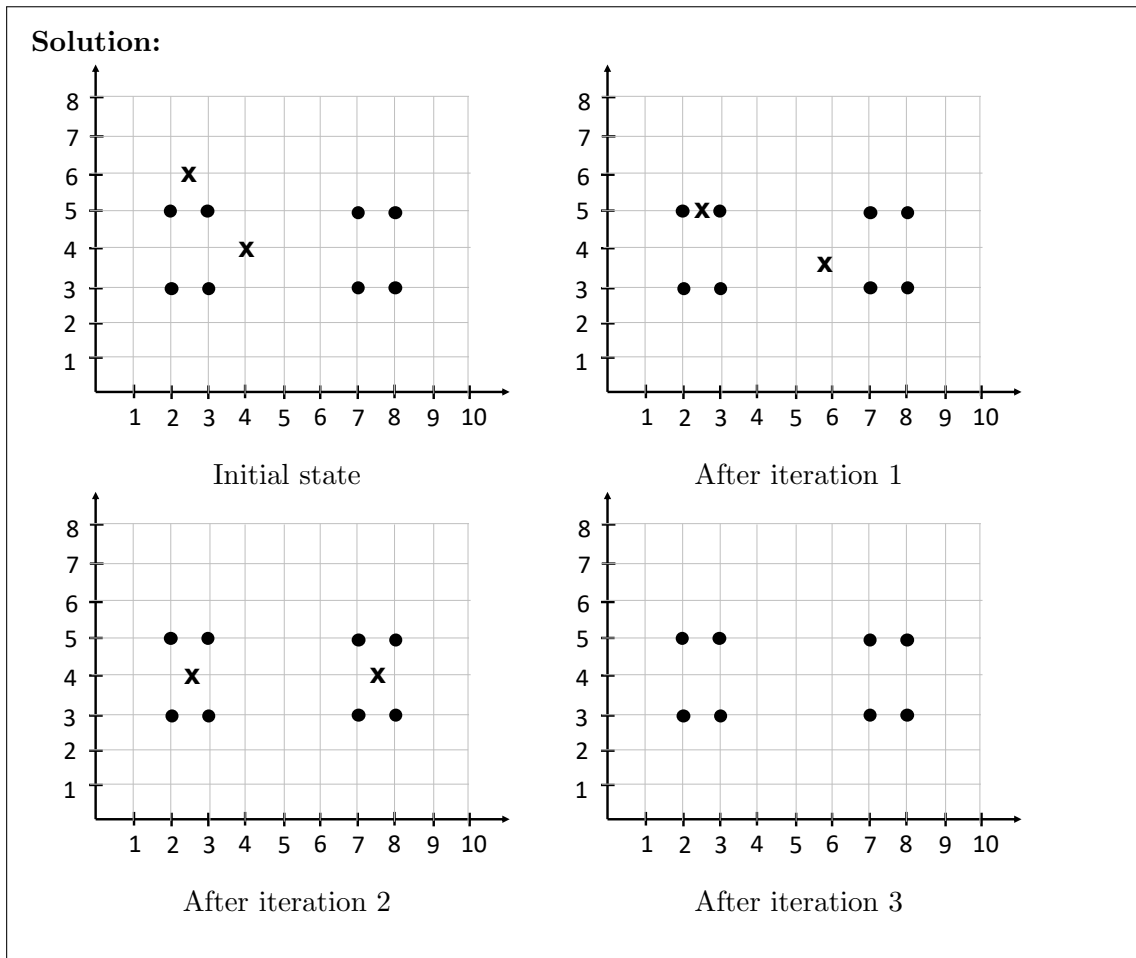
- (j) Mac experiments with even more training data and additional values of  $\lambda$ , but finds that he cannot decrease the validation error further. Are there changes to the neural network architecture that Mac could make to try to improve prediction performance? Explain.

**Solution:** Mac could add hidden layers with nonlinear activation functions to the neural network.

## Clustering

2. (12 points) Assume that the number of clusters  $k = 2$  for all of the following questions.

- (a) Walk through each step of the  $k$ -means algorithm, beginning with the initialization shown in the plot in the top left of the box below. Dots show the observed data. In each plot (go left to right, top to down), mark with two 'x' symbols where the cluster centers are in that iteration of  $k$ -means. These are already shown in the initial state. Once the  $k$ -means algorithm has converged, you can leave all subsequent plots unmarked.

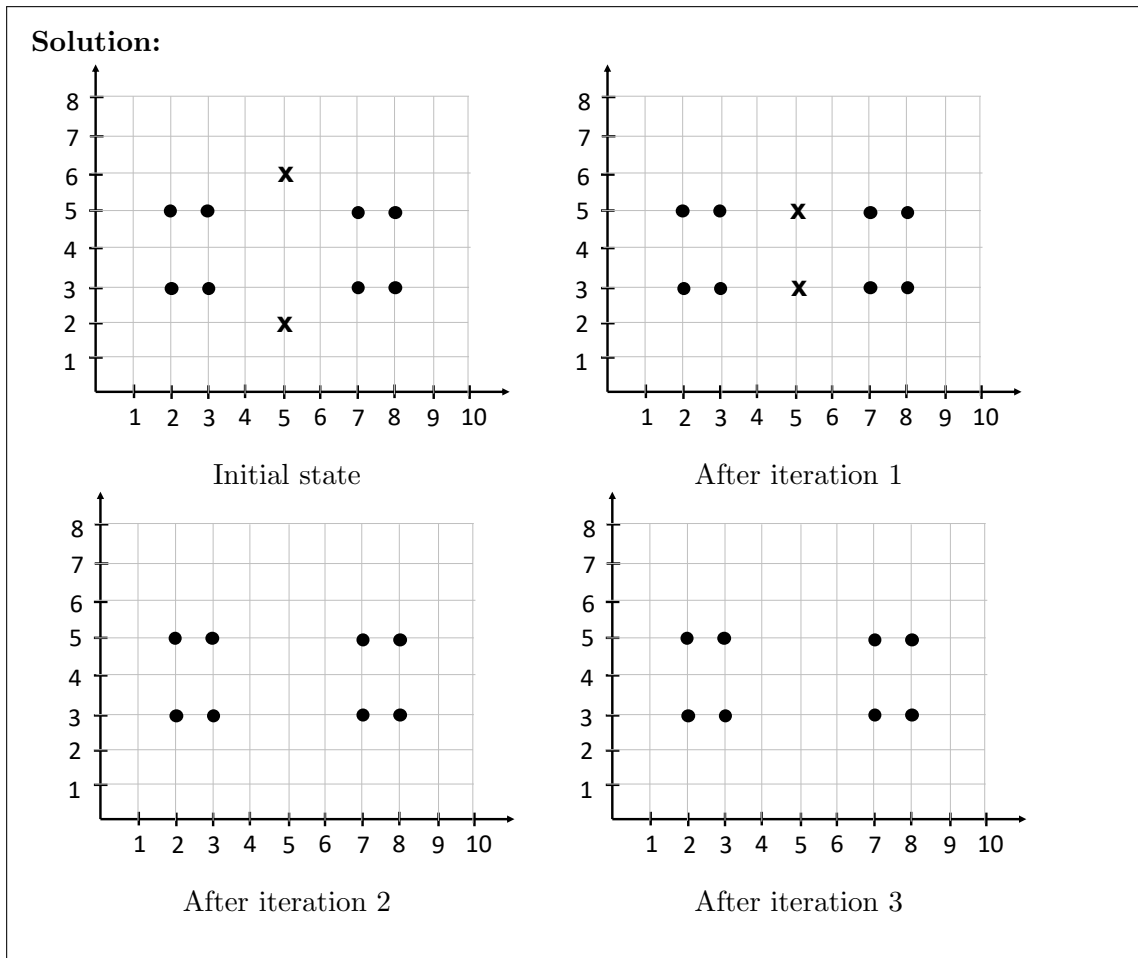


- (b) What is the numerical value of the  $k$ -means objective for the clustering found in (a), after the algorithm has finished running?

**Solution:**  $8 \cdot (1^2 + 0.5^2) = 8 \cdot (1.25) = 10.$

Name: \_\_\_\_\_

- (c) Just as in (a), walk through each step of the  $k$ -means algorithm, beginning with the initialization shown in the plot in the top left. In each plot (go left to right, top to down), mark with two 'x' symbols where the cluster centers are in that iteration of  $k$ -means. Once the  $k$ -means algorithm has converged, you can leave all subsequent figures unmarked.



- (d) What is the numerical value of the  $k$ -means objective for the clustering found in (c), after the algorithm has finished running?

**Solution:**  $4 \cdot (2^2) + 4 \cdot (3^2) = 16 + 36 = 52$ .

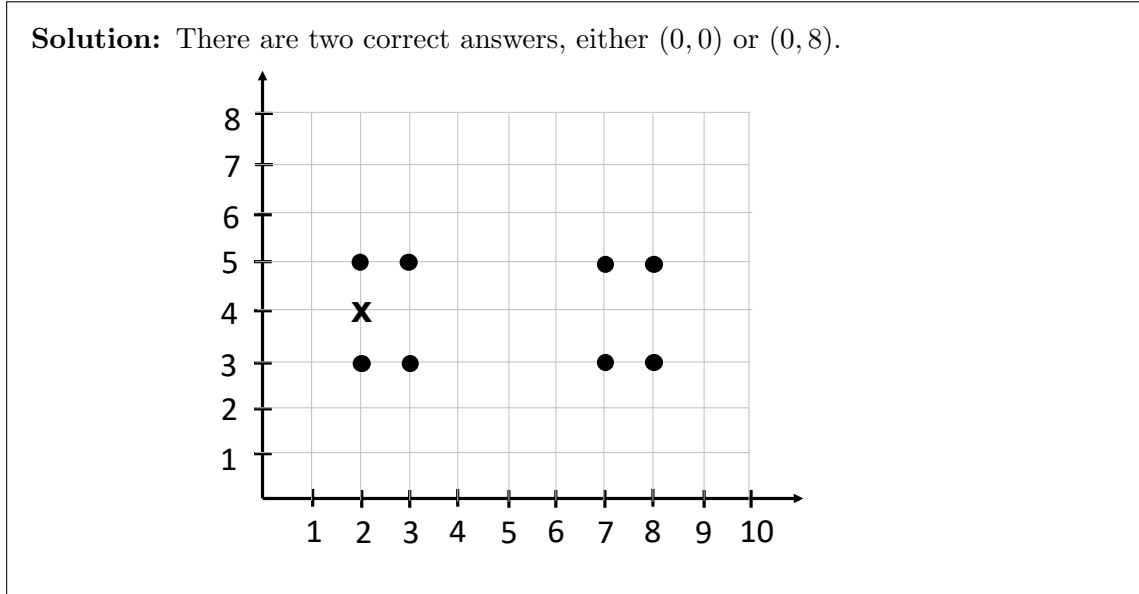
- (e) According to the  $k$ -means objective of the learned clusters, which initialization was better?

**Solution:**  Initialization (a)     Initialization (c)



Name: \_\_\_\_\_

- (f) Consider the data in black dots shown in the plot below. We drew one cluster center with an  $\mathbf{x}$  symbol at (2,4). Draw the second cluster center to satisfy the following property. When we initialize the clusters centers at the two  $\mathbf{x}$ 's and run the  $k$ -means algorithm to convergence, the final state will be such that one cluster will have all the data points assigned to it, and the other cluster will have no data points assigned to it.



- (g) Christy thinks she came up with a compelling new initialization method for the  $k$ -means algorithm. Looking at her code below, explain why it is unlikely to give good results.

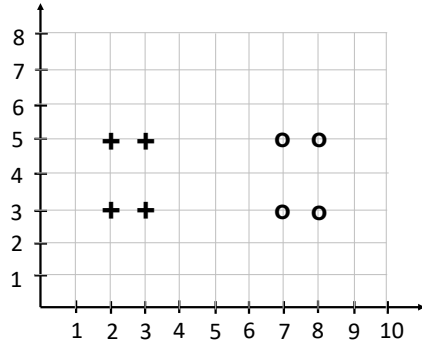
```
def kmeans_init(X, n_clusters):  
    centers = []  
    for i in range(n_clusters):  
        centers.append(X[:, X.shape[1]-1-i])  
    return np.asarray(centers).T
```

**Solution:** Christy's method selects the last  $n\_clusters$  data points as the cluster centers. These points may be very close to each other, leading to the  $k$ -means algorithm finding a poor local optima of the  $k$ -means objective.

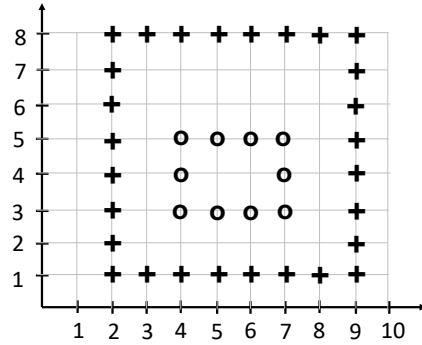
Name: \_\_\_\_\_

- (h) Each of the following five data sets has two ground truth clusters, whose points are denoted as '+' and 'o'. For which of these would the clustering with the smallest  $k$ -means objective value **not** recover the ground truth? Assume  $k = 2$ . (Select all that apply.)

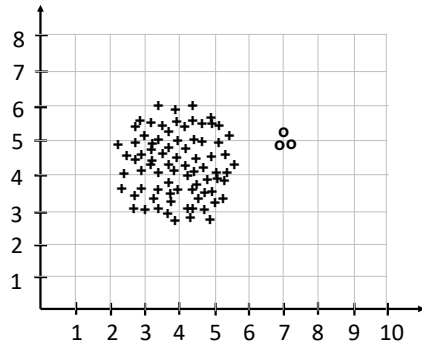
**Solution:**  (I)     (II)     (III)     (IV)     (V)



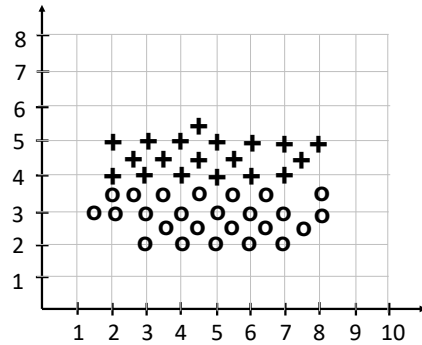
(I)



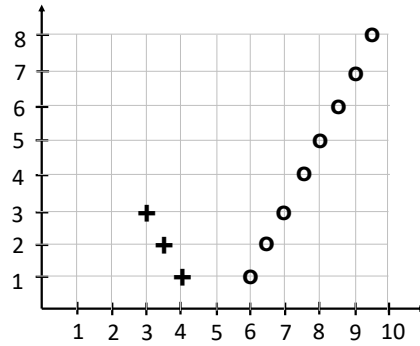
(II)



(III)



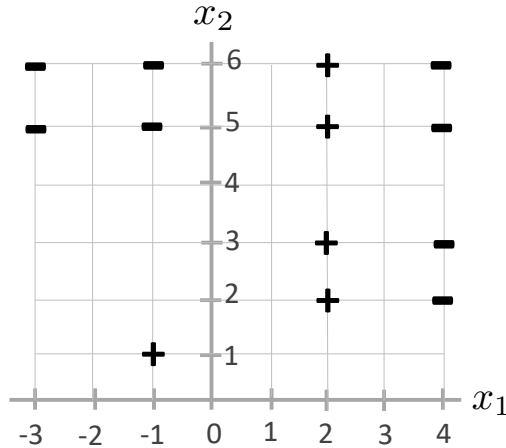
(IV)



(V)

### Decision trees

3. (12 points) We seek to learn a classifier on the data set shown on the left with 13 data points labeled +1 or -1. For your convenience, we include some helpful calculations in the table to the right.



For reference:

$-\frac{2}{7}\log_2\left(\frac{2}{7}\right) - \frac{5}{7}\log_2\left(\frac{5}{7}\right) \approx 0.86$
$\frac{7}{9} \times 0.86 \approx 0.67$
$-\frac{4}{5}\log_2\left(\frac{4}{5}\right) - \frac{1}{5}\log_2\left(\frac{1}{5}\right) \approx 0.72$
$\frac{5}{9} \times 0.72 \approx 0.40$
$-\frac{4}{6}\log_2\left(\frac{4}{6}\right) - \frac{2}{6}\log_2\left(\frac{2}{6}\right) \approx 0.92$
$\frac{6}{9} \times 0.92 \approx 0.61$

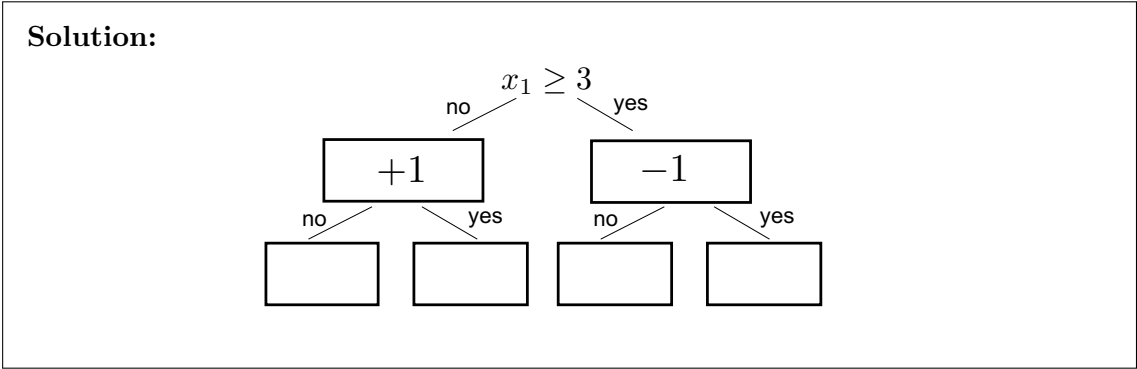
- (a) We first learn a linear logistic classifier with offset on this data set, with no regularization. Will it obtain zero training error? Write “yes” or “no” and explain your answer.

**Solution:** No, this data set is not linearly separable.

- (b) We now learn a depth-2 decision tree, with `min_samples_split=2`. We give you a partially completed tree below, where the first split is  $x_1 \geq 3$ . Complete the rest of the tree by filling in the boxes with the splits on the second level and the classifications (either +1 or -1) at the leaves. Use the entropy criterion to choose the splits, and leave empty any boxes that are unused. As a reminder, `min_samples_split` is the minimum number of data points required to split an internal node.

**Solution:**

- (c) Now suppose that we set `min_samples_split=10`. Again, complete the rest of the tree by filling in the boxes. Leave empty any boxes that are unused.



(d) What is the purpose of increasing min\_samples\_split?

**Solution:** It improves generalization (i.e., prevents overfitting to the training data) by requiring more samples to split a node, resulting in smaller tree depth.

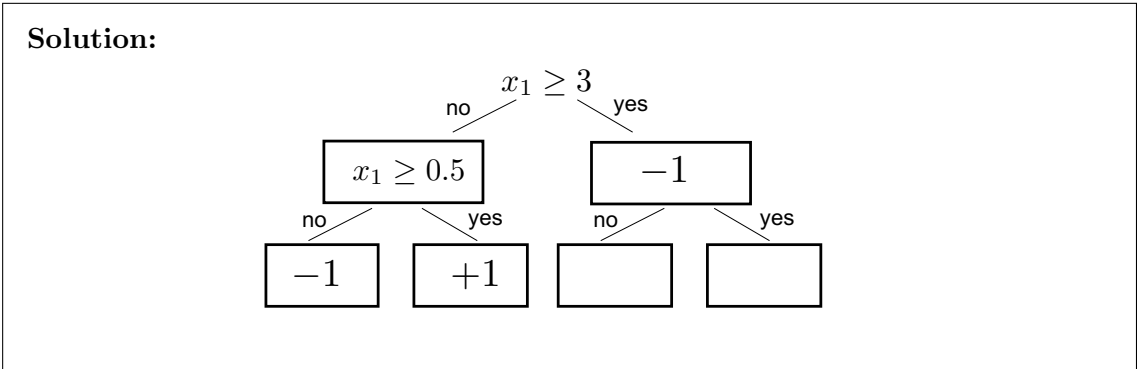
(e) What is the training error of the trees learned in parts (b) and (c)?

**Solution:**  
 Tree (b): 1/13.  
 Tree (c): 4/13.

(f) With min\_samples\_split=2, if we were to continue building the tree without any restriction to its depth, what would be the training error of the resulting tree?

**Solution:** 0.

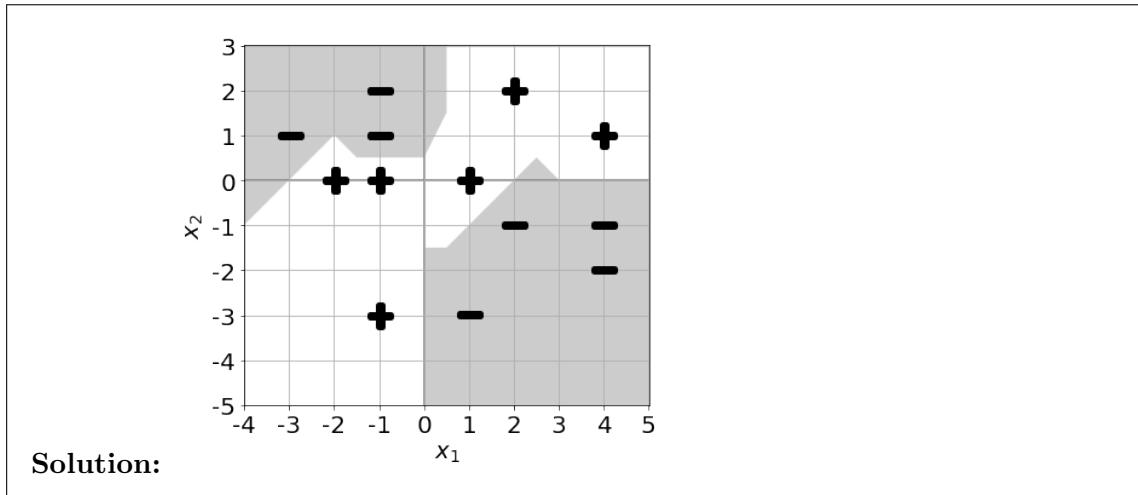
(g) Suppose we give as new features  $x_i^3$ , using these in addition to the original features  $x_i$ . Draw the new depth-2 tree that would be learned. Assume the features are organized  $x_1, x_2, x_1^3, x_2^3$  and if two features are equally good for the split according to the entropy criterion, then we choose the first one in this order. As in part (b), assume min\_samples\_split=2.



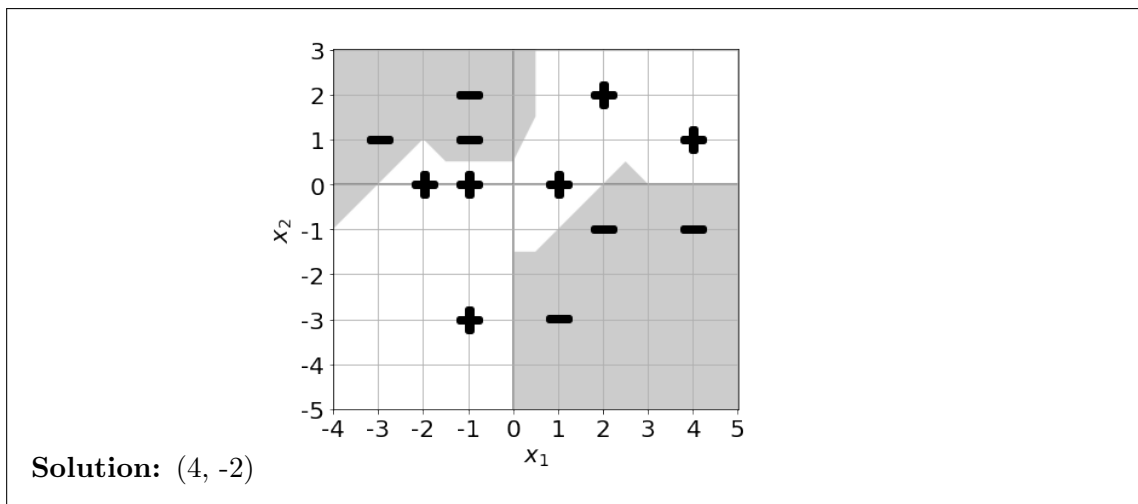
## Nearest neighbor classifiers

4. (12 points) This question asks about learning nearest neighbor (NN) classifiers. Assume that we are using Euclidean distance squared as the distance metric, i.e.  $d(x, x') = \|x - x'\|^2$ .

- (a) Draw on the below figure the decision boundary for a 1-NN classifier on this data set. In each region, denote whether the classification of any point (*any* point, not just the training data) in that region would be +1 or -1. (Note, all data points are assumed to be on integer coordinates.)

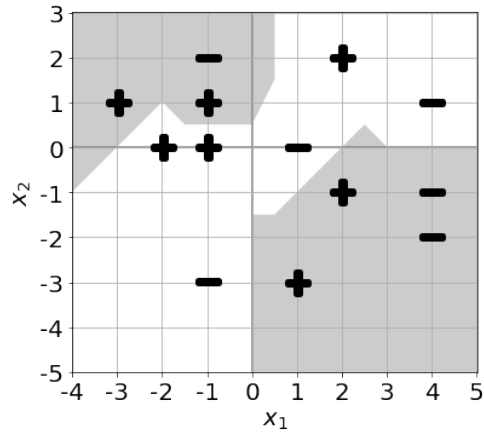


- (b) Which training data points, if any, could you remove and keep the decision boundary identical? Answer using their  $(x_1, x_2)$  coordinates.

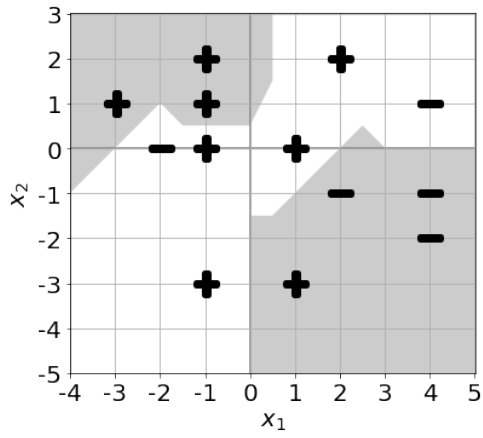


- (c) You perform leave-one-out cross-validation of the 1-NN and 3-NN classifiers on this data set, i.e. you use use cross-validation with a chunk size of 1 data point. Assume ties go to the +1 region. What cross-validation errors do you obtain?

Name: \_\_\_\_\_



Solution: 1-NN: 7 / 13



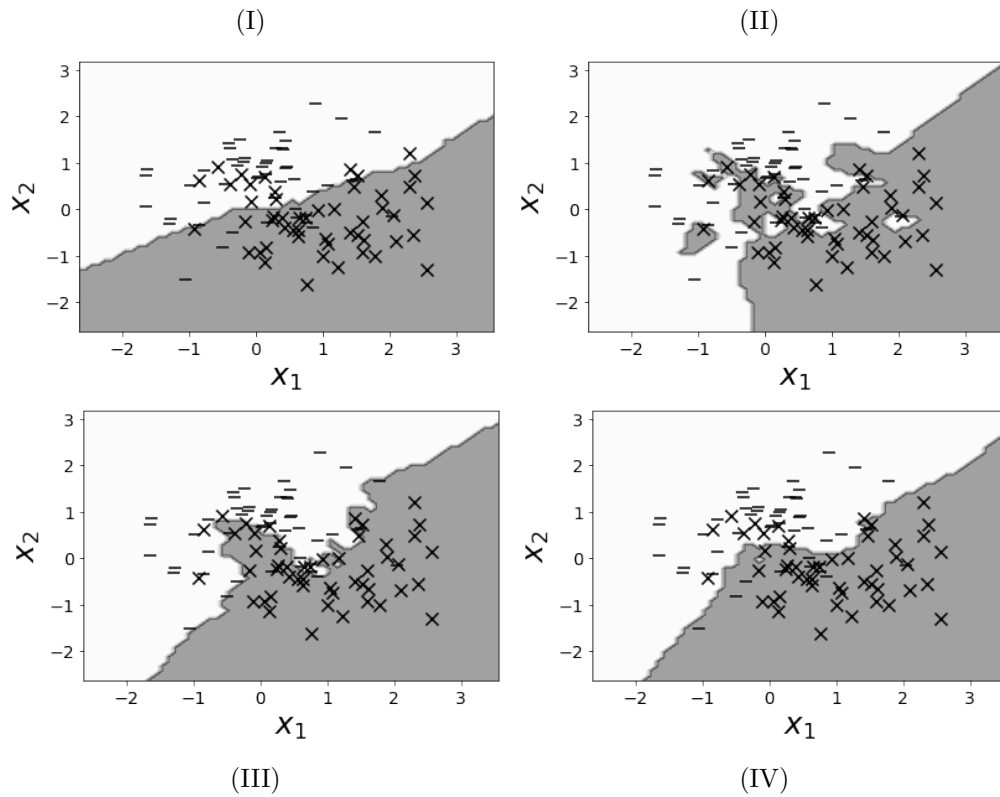
3-NN: 6 / 13

Name: \_\_\_\_\_

- (d) Suppose we now use the following feature transformation,  $\phi(x_1, x_2) = x_1x_2$ , and seek to learn a nearest neighbor classifier in the transformed space. This is equivalent to using a different distance metric,  $d(x, x') = \|\phi(x) - \phi(x')\|^2$ . What is the average leave-one-out cross-validation error of a 3-NN classifier using this new distance metric? Which points would be misclassified (specified using their  $(x_1, x_2)$  coordinates)?

**Solution:**  
 3-NN:  
 1 / 13 Misclassified points:  
 (-1, 1)

- (e) The plots below show the decision boundaries as predicted by a k-NN classifier for four different values of k: 1, 5, 20, 40. Map each plot to the corresponding value of k.



$k = 1$ :	<input type="radio"/> (I)	<input checked="" type="radio"/> (II)	<input type="radio"/> (III)	<input type="radio"/> (IV)
$k = 5$ :	<input type="radio"/> (I)	<input type="radio"/> (II)	<input checked="" type="radio"/> (III)	<input type="radio"/> (IV)
$k = 20$ :	<input type="radio"/> (I)	<input type="radio"/> (II)	<input type="radio"/> (III)	<input checked="" type="radio"/> (IV)
$k = 40$ :	<input checked="" type="radio"/> (I)	<input type="radio"/> (II)	<input type="radio"/> (III)	<input type="radio"/> (IV)

## Championship Material

5. (14 points) Ser Ena is a professional athlete who plays an individual sport. Ser is either **fully fit**, **partially fit** or **injured**. Regardless of her state, Ser can choose to **play** a tournament, take time to **train** or decide to take a complete **break** to rest. Ser's coaching team formulates an MDP to keep track of the states, actions, rewards and transitions. It is assumed that the discount factor is 1 (unless stated otherwise) and Ser is **fully fit** right before the next big tournament. The team comes up with the following MDP for Ser.

(a) First the **Rewards** for the state, action pairs:

When **fully fit**:

- if Ser decides to **play**, there is a reward of **+100**;
- if Ser decides to **train**, there is a reward of **-10**;
- if Ser decides to take a **break**, there is **no reward**.

When **partially fit**:

- if Ser decides to **play**, there is a reward of **+20**;
- if Ser decides to **train**, there is a reward of **-10**;
- if Ser decides to take a **break**, there is a reward of **-20**.

When **injured**:

- if Ser decides to **play**, there is a reward of **-60**;
- if Ser decides to **train**, there is a reward of **-30**;
- if Ser decides to take a **break**, there is a reward of **0**.

(b) Next, the **transition probabilities**:

When **fully fit**:

- if Ser decides to **play**, there is a **80%** chance of remaining fully fit, **20%** chance of getting injured.
- if Ser decides to **train**, there is a **90%** chance of remaining fully fit, **10%** chance of getting injured.
- if Ser decides to take a **break**, there is a **50%** chance of remaining full fit, **50%** chance of being partially fit.

When **partially fit**:

- if Ser decides to **play**, there is a **50%** chance of remaining partially fit; and a **50%** chance of getting injured.
- if Ser decides to **train**, there is a **40%** chance of remaining partially fit, **60%** chance of getting fully fit;.
- if Ser decides to take a **break**, Ser will remain partially fit.

When **injured**:

- if Ser decides to **play**, Ser will remain injured.
- if Ser decides to **train**, Ser will remain injured.
- if Ser decides to take a **break**, there is a **50%** chance Ser will remain injured and a **50%** chance of being partially fit.



Name: \_\_\_\_\_

- (a) What is the horizon 1 optimal policy for Ser? Assume discount is 1.

**Solution:**

$$\begin{aligned}\pi_1^*(\text{fully fit}) &= \text{play} \\ \pi_1^*(\text{partially fit}) &= \text{play} \\ \pi_1^*(\text{injured}) &= \text{break}\end{aligned}$$

This optimal policy is determined by taking the action with the maximum reward for each state.

- (b) For horizon 2, what is the best action to take when Ser is **partially fit** and what is the horizon 2 expected reward for taking that best action when **partially fit**? Assume discount of 1. Please show how you arrived at the answer.

**Solution:**

$$\begin{aligned}Q_2(\text{partially fit, play}) &= R(\text{partially fit, play}) + \sum_{s'} T(\text{partially fit, play, } s') \max_{a'} Q_1(s', a') \\ &= 20 + (0.5 * 20 + 0.5 * 0) \\ &= 30 \\ Q_2(\text{partially fit, train}) &= R(\text{partially fit, train}) + \sum_{s'} T(\text{partially fit, train, } s') \max_{a'} Q_1(s', a') \\ &= -10 + (0.4 * 20 + 0.6 * 100) \\ &= 58 \\ Q_2(\text{partially fit, rest}) &= R(\text{partially fit, rest}) + \sum_{s'} T(\text{partially fit, rest, } s') \max_{a'} Q_1(s', a') \\ &= -20 + (1 * 20) \\ &= 0 \\ \pi_2^*(\text{partially fit}) &= \arg \max_a Q_2(\text{partially fit, } a) \\ &= \text{train}\end{aligned}$$

Name: \_\_\_\_\_

(c) Does the answer to (b) change if the discount factor was 0.5? Explain why or why not.

**Solution:** Yes it changes. The new best  $Q$ (partially fit, a) for all actions  $a$  is now **play** because the discount of 0.5 ensures that the large expected reward on step is not enough to overcome the negative immediate negative reward of training in **partially fit** state.

$$\begin{aligned} Q_2(\text{partially fit, play}) &= R(\text{partially fit, play}) + \delta \sum_{s'} T(\text{partially fit, play}, s') \max_{a'} Q_1(s', a') \\ &= 20 + 0.5(0.5 * 20 + 0.5 * 0) \\ &= 25 \end{aligned}$$

$$\begin{aligned} Q_2(\text{partially fit, train}) &= R(\text{partially fit, train}) + \delta \sum_{s'} T(\text{partially fit, train}, s') \max_{a'} Q_1(s', a') \\ &= -10 + 0.5(0.4 * 20 + 0.6 * 100) \\ &= 24 \end{aligned}$$

$$\begin{aligned} Q_2(\text{partially fit, rest}) &= R(\text{partially fit, rest}) + \delta \sum_{s'} T(\text{partially fit, rest}, s') \max_{a'} Q_1(s', a') \\ &= -20 + 0.5(1 * 20) \\ &= -10 \end{aligned}$$

$$\begin{aligned} \pi_2^*(\text{partially fit}) &= \arg \max_a Q_2(\text{partially fit}, a) \\ &= \text{play} \end{aligned}$$

(d) What is the infinite horizon optimal policy for Ser? Assume discount is 1.

**Solution:**

$$\begin{aligned} \pi_\infty^*(\text{fully fit}) &= \text{play} \\ \pi_\infty^*(\text{partially fit}) &= \text{train} \\ \pi_\infty^*(\text{injured}) &= \text{break} \end{aligned}$$

The solution given above is optimal for a wide range of  $\gamma$  (it is straightforward to see for  $\gamma > 30/58$ , from looking at the horizon 2 case; a transition actually happens around  $\gamma = 0.37$ , but that's beyond this question).

When  $\gamma = 1$ , the value of a policy may diverge to  $+\infty$  or  $-\infty$ . Therefore, for grading purposes, we accepted any well-specified policy for which the value (the total expected reward) is positive and growing toward  $+\infty$  as the horizon is increased.

Note that technically,  $0 < \gamma < 1$  is the range for which an infinite horizon optimal policy is defined, according to the lecture notes.

Name: \_\_\_\_\_

- (e) Is there any policy which maximizes the expected reward in the infinite horizon under which Ser should play if injured? Explain.

**Solution:** No there isn't. Both other actions have a negative reward and they both keep Ser in the injured state.

Name: \_\_\_\_\_

- (f) Djo Ko is another athlete who plays the same sport. Djo Ko has the exact same MDP as Ser Ena's, except Djo's team has **forgotten the reward for playing when in the fully fit** state. Djo's team also remember that the horizon 2 best action to take in the **partially fit** state is exactly the same as that for Ser Ena (determined in part b). Given this information, what are the range of possible values for  $R(\text{fully fit, play})$  for Djo Ko? Assume discount of 1.

**Solution:** If the horizon 2 optimal policy in state **partially fit** is the same for both Ser and Djo, then  $\pi_2^*(\text{partially fit}) = \text{train}$ . This implies that  $Q_2(\text{partially fit, train}) > Q_2(\text{partially fit, play})$  and  $Q_2(\text{partially fit, train}) > Q_2(\text{partially fit, rest})$ .

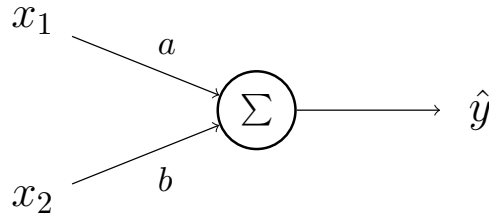
Therefore, we have:

$$\begin{aligned} Q_2(\text{partially fit, train}) &> Q_2(\text{partially fit, play}) \\ R(\text{partially fit, train}) + \sum_{s'} T(\text{partially fit, train, } s') \max_{a'} Q_1(s', a') &> \\ R(\text{partially fit, play}) + \sum_{s'} T(\text{partially fit, play, } s') \max_{a'} Q_1(s', a') & \\ -10 + (0.4 * 20 + 0.6 * R(\text{fully fit, play})) &> 20 + (0.5 * 20 + 0.5 * 0) \\ 0.6 * R(\text{fully fit, play}) &> 32 \\ R(\text{fully fit, play}) &> \frac{32 * 10}{6} = 53.33. \end{aligned}$$

## Harmony in Descent

6. (12 points) Years ago, MIT student Itu Nes learned about neural networks and how to train them, from taking 6.036. Now Itu is an engineer at Orange Computer, a hot tech company employing machine learning to revolutionize music. Looking back at her notes, Itu realizes that she once wrote down exactly what she now needs to do in her job, but unfortunately some key details are lost. Can you help her figure things out?

Specifically, Itu wants to train this simple single-node neural network:



The network accepts two inputs  $x_1$  and  $x_2$ , and outputs a prediction  $\hat{y}$  based on weights  $a$  and  $b$ . Itu's dataset has points  $(x, y)$  where  $x = (x_1, x_2)$ , and  $y$  are the true labels. Itu employs the squared error loss function

$$L(\hat{y}, y) = (y - \hat{y})^2 .$$

In her notes, Itu wrote about using gradient descent to obtain the optimal weights for the network, by minimizing this loss. Moreover, for each run of the gradient descent, she used a single data point to train the weights. Afterwards, Itu learns that the true labels are  $y = x_1 + x_2$ .

- (a) Suppose  $a_0$  and  $b_0$  are the initial values of the weights, and  $a_k$  and  $b_k$  are the weights at iteration  $k$ . Give equations for the updated weights  $a_{k+1}$ ,  $b_{k+1}$  in terms of current iteration's weights  $a_k$ ,  $b_k$ , the step size parameter  $\eta$ , and the inputs  $x_1$ ,  $x_2$ .

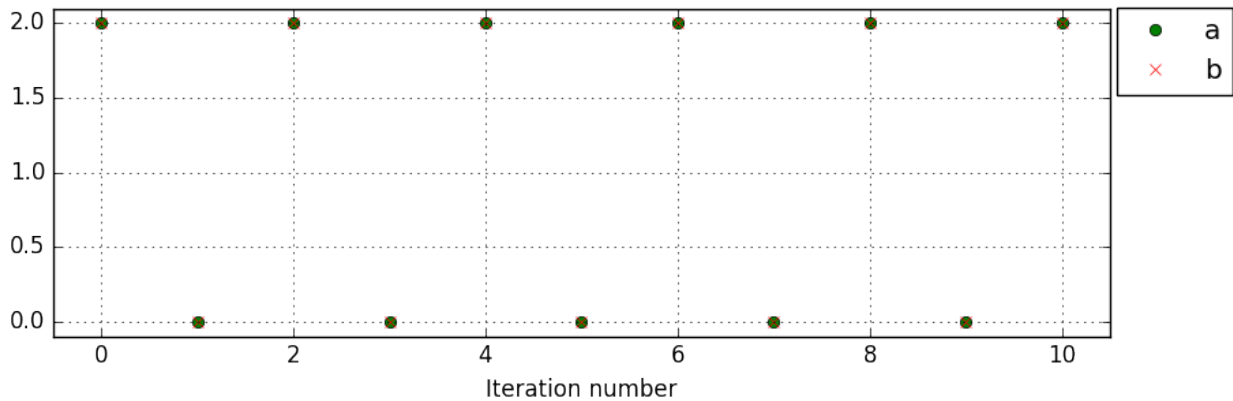
$$a_{k+1} = a_k - \eta \frac{dL}{da} = a_k - 2\eta [(a_k - 1)x_1^2 + (b_k - 1)x_1x_2]$$

**Solution:**

$$b_{k+1} = b_k - \eta \frac{dL}{db} = b_k - 2\eta [(b_k - 1)x_2^2 + (a_k - 1)x_1x_2]$$

Name: \_\_\_\_\_

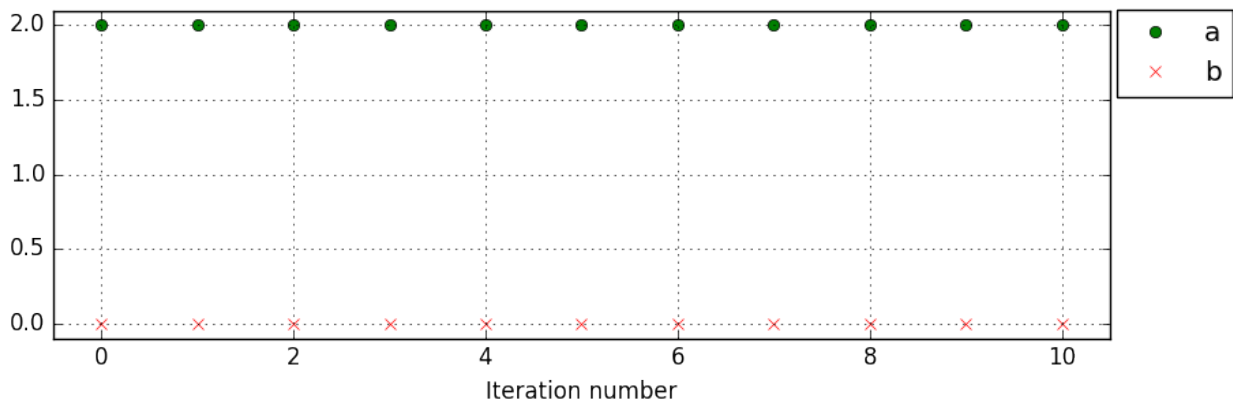
- (b) Itu sees that when she fixed  $x_1 = 1$ ,  $x_2 = 1$  and ran 10 iterations of gradient descent starting with  $a_0 = 2$ ,  $b_0 = 2$ , she recorded that the two weights oscillated back and forth, as captured in this plot pasted into her notebook:



Note that in this plot, the  $a$  and  $b$  points lay on top of each other. Unfortunately, Itu forgot to write down her code, nor did she write down what value of  $\eta$  may have been used to generate this plot. Help her figure out: was this plot a mistake (and explain why), or if not, what value of  $\eta$  could have generated it?

**Solution:** This oscillation happens when  $\eta = 1/2$ , because  $dL/da = dL/db = -4$

- (c) Itu sees that when she fixed  $x_1 = 1$ ,  $x_2 = 1$  and ran 10 iterations of gradient descent starting with  $a_0 = 2$ ,  $b_0 = 0$ , she recorded that the two weights remained unchanged, as captured in this plot pasted into her notebook:

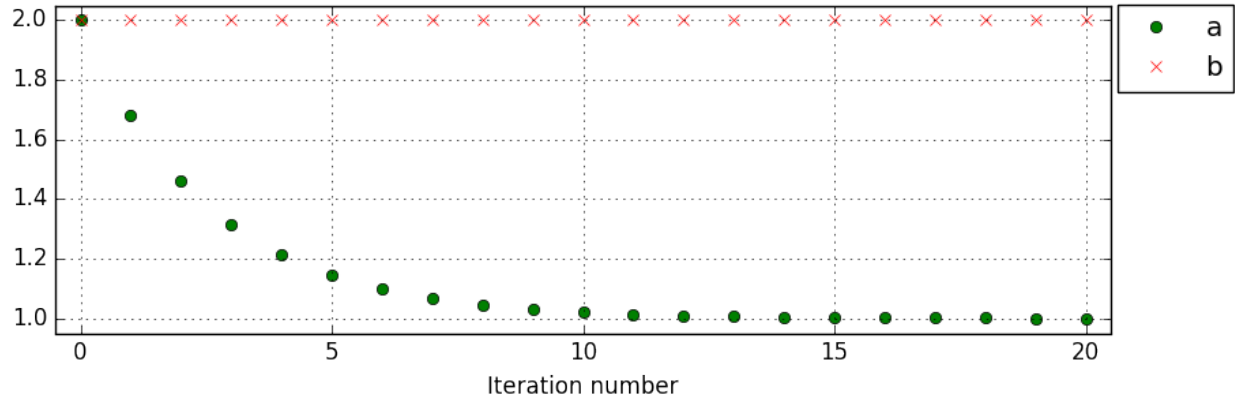


Again: was this plot a mistake (and explain why), or if not, what value of  $\eta$  could have generated it?

**Solution:** Any  $\eta$ , e.g.  $\eta = 5$ , because  $dL/da = 0$  and  $dL/db = 0$  for these parameters. Alternatively  $\eta = 0$  will also leave  $a$  and  $b$  at their initial values.

- (d) Itu sees that when she fixed  $x_1$  and  $x_2$  and ran 10 iterations of gradient descent with  $\eta = 0.01$  starting with  $a_0 = b_0 = 2$ , she recorded that  $b$  stayed unchanged, but  $a$  decayed to 1, as captured in this plot pasted into her notebook:

Name: \_\_\_\_\_

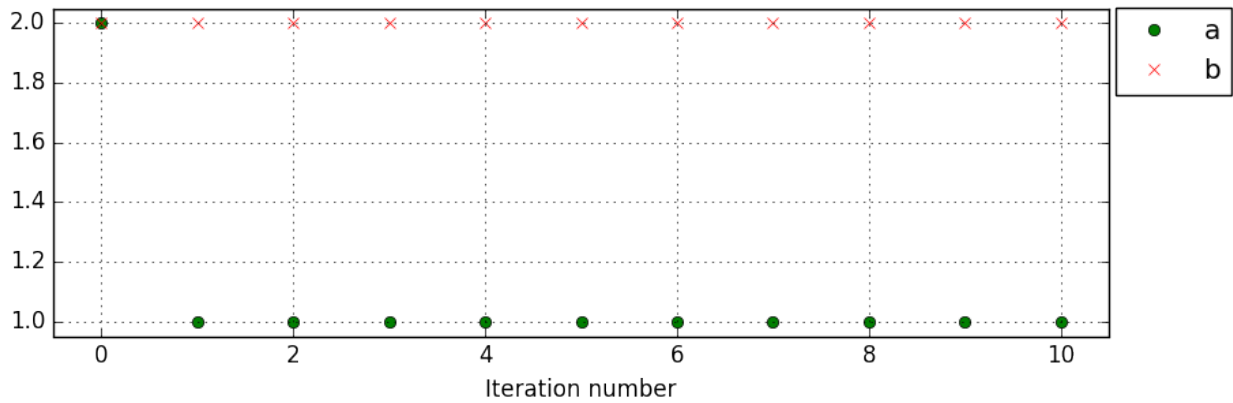


Again: was this plot a mistake (and explain why), or if not, what values of  $x_1, x_2$  could have generated it?

**Solution:** Resulted from choosing  $x_1 = 4, x_2 = 0$  (other nonzero, positive values of  $x_1$  also work).

Name: \_\_\_\_\_

(e) Itu sees that when she fixed  $x_1 = 4$  and  $x_2 = 0$  and ran 10 iterations of gradient descent starting with  $a_0 = b_0 = 2$ , she recorded that  $b$  stayed unchanged, but  $a$  jumped immediately on the first iteration to 1, as captured in this plot pasted into her notebook:



Again: was this plot a mistake (and explain why), or if not, what value of  $\eta$  could have generated it?

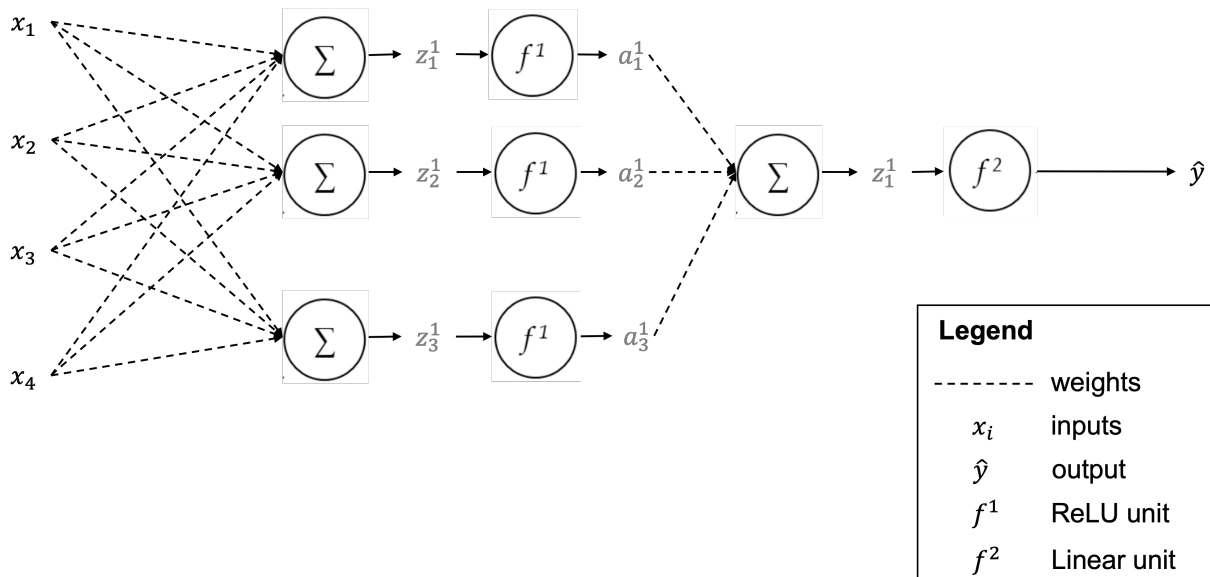
**Solution:** This results from choosing  $\eta = 1/32$ , because for  $x_1 = 4$  and  $x_2 = 0$ , the gradient descent update equations become  $a_{k+1} = a_k - \eta(32a_k - 32)$  and  $b_{k+1} = b_k$ . Note that choosing  $\eta = 1/16$  will bring  $a_1 = 0$ , but then it oscillates:  $a_2 = 2$ ,  $a_3 = 0$ , etc.



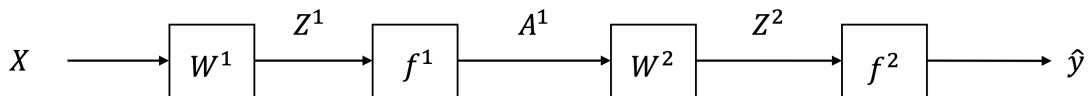
## Forward and Backwards

7. (12 points) A feed-forward neural network is shown below. As noted in the Legend, and following 6.036 conventions,

- The dashed lines represent the weights that the neural network learns
- There are no bias/constant terms among the weights
- We are using squared-loss, i.e.  $L(\hat{y}, y) = (y - \hat{y})^2$
- Inputs are represented by  $X = [x_1, x_2, x_3, x_4]^T$
- The *true* output value is denoted by  $y$ , and the estimation output by the network is  $\hat{y}$
- $f^1$  are ReLU units
- $f^2$  is a linear unit



(a) The neural network shown above can be also be represented by the network shown below, which uses matrix notation. Specifically, the input  $X$  is a  $4 \times 1$  column vector,  $\hat{y}$  is a  $1 \times 1$  scalar.  $W^2$  is a  $3 \times 1$  vector. We also know that,  $Z^1 = (W^1)^T X$  and  $Z^2 = (W^2)^T A^1$ .



i. What are the dimensions of the matrix  $W^1$ ?

**Solution:**  $4 \times 3$ .

ii. What are the dimensions of  $Z^2$ ?

Name: \_\_\_\_\_

**Solution:**  $1 \times 1$  or scalar.

(b) For both parts below, you are told that there is only **one data point** which is:  $X = [1, 1, 1, 1]^T$  and  $y = [1]$ .

i. If  $W^1$  and  $W^2$  are both matrices/vectors of all ones, what is the resulting Loss?

**Solution:**  $([12]-[1])^2 = [121]$  or just 121.

ii. If  $W^1$  is a matrix of all  $-1$ 's (all negative ones) and  $W^2$  is a vector of all  $1$ 's (positive ones), what is the resulting Loss?

**Solution:**  $([0]-[1])^2 = [1]$  or just 1.

The negatives ones in  $W^1$  will ensure that the input to the ReLU unit is negative. This leads to  $A^1 = 0$  leading to  $Z^2 = \hat{y} = 0$ .

Name: \_\_\_\_\_

- (c) i. Determine the expression for  $\frac{\partial L}{\partial W^1}$ . You may leave your expression in terms of  $X, y, \hat{y}, W^2$  and  $\frac{\partial A^1}{\partial Z^1}$ .

**Solution:**

$$\begin{aligned}\frac{\partial L}{\partial W^1} &= \left(\frac{\partial Z^1}{\partial W^1}\right) \left(\frac{\partial A^1}{\partial Z^1} * \frac{\partial Z^2}{\partial A^1} * \frac{\partial \hat{y}}{\partial Z^2} * \frac{\partial L}{\partial \hat{y}}\right)^T \\ &= X \left(\frac{\partial A^1}{\partial Z^1} * W^2 * 1 * -2(y - \hat{y})\right)^T \\ &= -2X \left(\frac{\partial A^1}{\partial Z^1} W^2 (y - \hat{y})\right)^T\end{aligned}$$

- ii. What are the dimensions of  $\frac{\partial L}{\partial W^1}$ ?

**Solution:**  $4 \times 3$ .

- (d) We now use back-propagation to update the weights during each iteration. For all questions below, assume that we only have one data point  $(X, y)$  available to use, and the stepsize parameter is 0.01. You are asked to determine how many components of  $W^1$  will get updated (i.e. have their value changed) in each scenario below.

- i. Assume  $X = [1, 1, 1, 1]^T, y = [1]$ . Further assume that we start with  $W^1$  as a matrix of  $-1$ 's (negative ones) while  $W^2$  is a vector of  $1$ 's (positive ones). How many components of  $W^1$  will get updated (i.e. have their value changed) after one iteration of backprop? Explain your answer.

**Solution:** 0 components will be updated.

Similar to part (b)(ii) above, the negative ones in  $W^1$  will lead to the outputs of the ReLU unit being zero which leads to  $\frac{\partial A^1}{\partial Z^1} = 0$ , and that will zero out the gradient,  $\frac{\partial L}{\partial W^1}$ .

- ii. Assume  $X = [0, 0, 0, 0]^T, y = [0]$ . Further assume that we start off with  $W^1$  and  $W^2$  as matrices/vectors of all ones. How many components of  $W^1$  will get updated (i.e. have their value changed) after one iteration of back-propagation? Explain your answer.

**Solution:** 0 components will be updated.

The input  $X = 0$  which will zero out the gradient,  $\frac{\partial L}{\partial W^1}$ .

- iii. Assume  $X = [1, 1, 1, 1]^T, y = [1]$ . Further assume that we start off with  $W^1$  and  $W^2$  as matrices/vectors of all ones. How many components of  $W^1$  will get updated (i.e. have their value changed) after one iteration of backprop? Explain your answer.

Name: \_\_\_\_\_

**Solution:** All components (= 12) will be updated.

No component of the gradient is zero.

- iv. Assume  $X = [1, 1, 1, 1]^T, y = [1]$ . Further assume that we start off with  $W^1$  as a matrix of all ones.  $W^2 = [0, 1, 0]^T$ . How many components of  $W^1$  will get updated (i.e. have their value changed) after one iteration of backprop? Explain your answer.

**Solution:** 4 components will be updated.

The only non-zero gradient components correspond to  $W_{1,2}^1, W_{2,2}^1, W_{3,2}^1, W_{4,2}^1$  because  $W_1^2 = W_3^2 = 0$ .

**A Tiny CNN for Tetris**

8. (12 points) MIT grad student Rec Urrant would like to submit an entry to win this year's Grand ML Tetris Competition, which gives awards to the smallest neural networks which can identify tetris pieces with the highest accuracy. Rec seeks to make a convolutional neural network that can accurately classify single-channel  $3 \times 3$  images of 2D tetris pieces as being either a line-shaped piece, or a corner-shaped piece, using just one  $2 \times 2$  filter. Let's help Rec win this competition.

- (a) What are the spatial dimensions of the output image if a  $2 \times 2$  filter is convolved with a  $3 \times 3$  image for paddings of 0, 1, and 2, and strides of 1 and 2? Fill in the dimensions below:

**Solution:**  $2 \times 2$  —  $4 \times 4$  —  $6 \times 6$

$1 \times 1$  —  $2 \times 2$  —  $3 \times 3$

- (b) Rec writes a bit of python code to implement their tiny CNN classifier for images of 2D tetris pieces, following examples they have seen in 6.036. They include in the comments the dimensions of the numpy arrays, where known.

```

1  def tinycnn(x, fcoef, w, b, final_act):
2      """
3      x: (numpy array, dimensions [1,3,3]) input image
4      fcoef: (numpy array, dimensions [1,1,2,2]) conv filter coefficients
5      w: (numpy array, dimensions [?, ?]) weights for classifier network
6      b: (numpy array, dimensions [?]) bias for classifier network
7      final_act: (function) final output activation
8      """
9      z = conv2d(x, fcoef, padding=0, stride=1) # [1, ?, ?]
10     a = ReLU(z)
11     a_sum = z1.sum(dim=-1).sum(dim=-1) # [1] sum spatial dim
12     z2 = w.T @ a_sum + b
13     return final_act(z2)

```

For performing binary classification, what activation function should Rec use for `final_act` and which loss function should Rec use?

**Solution:** Sigmoid + Negative Log Likelihood Loss

- (c) If Rec wants to allow for more than two classes, which activation function should they use for `final_act` and which loss function?



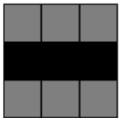
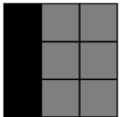
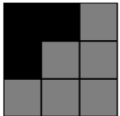
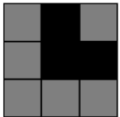
**Solution:** Softmax + Cross Entropy

- (d) What are dimensions of `w` and `b` for i) binary classification vs. ii)  $k$ -class classification?

**Solution:** For binary classification  $w$  is:  $[1,1]$  and  $b$  is:  $[1]$

For  $k$ -class classification  $w$  is:  $[1,k]$  and  $b$  is:  $[k]$

- (e) To debug their code, Rec runs `tinycnn` on four different  $3 \times 3$  input images  $x$ , and two different  $2 \times 2$  filters  $fcoef$ . They add print statements to see  $z$ ,  $a$ , and  $a\_sum$  for the 8 cases. Fill in the table below with numbers giving what Rec obtains.  $fcoef$  and  $x$  are depicted with 1 = black, 0 = grey, and -1 = white.

	$fcoef \rightarrow$	$x \downarrow$																						
			$z$	$a$	$a\_sum$	$z$	$a$	$a\_sum$																
	<table border="1" style="font-size: 0.8em;"><tr><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td></tr></table>	0	0	0	0	<table border="1" style="font-size: 0.8em;"><tr><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td></tr></table>	0	0	0	0	$0$	<table border="1" style="font-size: 0.8em;"><tr><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td></tr></table>	0	0	0	0	$0$	<table border="1" style="font-size: 0.8em;"><tr><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td></tr></table>	0	0	0	0	$0$	$0$
0	0																							
0	0																							
0	0																							
0	0																							
0	0																							
0	0																							
0	0																							
0	0																							
	<table border="1" style="font-size: 0.8em;"><tr><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td></tr></table>	0	0	0	0	<table border="1" style="font-size: 0.8em;"><tr><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td></tr></table>	0	0	0	0	$0$	<table border="1" style="font-size: 0.8em;"><tr><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td></tr></table>	0	0	0	0	$0$	<table border="1" style="font-size: 0.8em;"><tr><td>2</td><td>0</td></tr><tr><td>2</td><td>0</td></tr></table>	2	0	2	0	$2$	$4$
0	0																							
0	0																							
0	0																							
0	0																							
0	0																							
0	0																							
2	0																							
2	0																							
	<table border="1" style="font-size: 0.8em;"><tr><td>-1</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	-1	1	1	0	<table border="1" style="font-size: 0.8em;"><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	0	1	1	0	$2$	<table border="1" style="font-size: 0.8em;"><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	0	1	1	0	$2$	<table border="1" style="font-size: 0.8em;"><tr><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	1	1	1	0	$2$	$3$
-1	1																							
1	0																							
0	1																							
1	0																							
0	1																							
1	0																							
1	1																							
1	0																							
	<table border="1" style="font-size: 0.8em;"><tr><td>0</td><td>1</td></tr><tr><td>-1</td><td>0</td></tr></table>	0	1	-1	0	<table border="1" style="font-size: 0.8em;"><tr><td>0</td><td>1</td></tr><tr><td>0</td><td>0</td></tr></table>	0	1	0	0	$1$	<table border="1" style="font-size: 0.8em;"><tr><td>0</td><td>1</td></tr><tr><td>0</td><td>0</td></tr></table>	0	1	0	0	$1$	<table border="1" style="font-size: 0.8em;"><tr><td>-2</td><td>1</td></tr><tr><td>-1</td><td>0</td></tr></table>	-2	1	-1	0	$1$	$1$
0	1																							
-1	0																							
0	1																							
0	0																							
0	1																							
0	0																							
-2	1																							
-1	0																							

**Solution:**

- (f) What does each filter do? Which filter is best for distinguishing line-shaped tetris pieces vs. corner-shaped pieces? Why?

**Solution:** The first filter detects pixels on the diagonal and ignores vertical and horizontal lines. The second filter only detects vertical lines. The first filter is best for distinguishing corners and lines after applying ReLU to the output, it linearly separates corners and lines whereas the second filter does not.

- (g) Rec labels corner-shaped tetris pieces as “1” and line-shaped tetris pieces as “0”. Using this labeling, what values of  $w$  and  $b$  of the output layer give perfect classification and outputs that are close to 1 for corners and close to 0 for lines? (Assume the examples in (e) are representative of the entire dataset.) You may find the plots provided on the last page of this exam helpful.

Name: \_\_\_\_\_

**Solution:**  $\sigma(2.95) \rightarrow 0.95$ , so we need to have our output be at least  $\sim 3$  for corners and equal to or less than  $-3$  for lines. The max and min value of  $\mathbf{a\_sum}$  for lines is 0 and for corners the max is 2 and the min is 1. Therefore, distance between 0 and 1 needs to be 6, so we scale by 6 and subtract by 3. Therefore,  $w > 6$  with  $b = w/2$ . Full credit will be given for “large and positive” for  $\mathbf{w}$  and  $\mathbf{b} = -w/2$ .

- (h) If Rec instead labeled line-shaped pieces as “1” and corner-shaped pieces as “0” then what values of  $\mathbf{w}$  and  $\mathbf{b}$  of the output layer give perfect classification and outputs that are close to 0 for corners and close to 1 for lines?

**Solution:** The same as above with opposite sign.

Name: \_\_\_\_\_

- (i) Write an expression for the derivative of the binary classification loss with respect to  $z_2$ , the input of `final_act`. You may express your answer using  $g$  for the output of `final_act` and  $y$  for the example label.

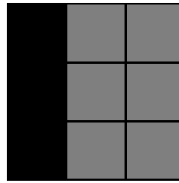
**Solution:** The derivative of the negative log likelihood loss with respect to the argument of the Sigmoid function is very elegant. It is  $g - y$ . (This is derived in the notes.)

- (j) Using your answers from above, write an expression for gradient of the loss with respect to  $w$  and  $b$  of the output layer. You may express your answers in terms of `a_sum`.

**Solution:**  $\frac{\partial \mathcal{L}}{\partial w} = z1_{sum}(g - y)$

$\frac{\partial \mathcal{L}}{\partial b} = (g - y)$

- (k) Assume we apply a filter with weights  $[[f_1, f_2], [f_3, f_4]]$  to this  $3 \times 3$  image:



with stride 1 and padding 0 and perform back propagation. Which filter weights may have non-zero gradients? Why? Under what conditions will those gradients be non-zero?

**Solution:**  $f_1$  and  $f_3$  are the only weights that will receive gradients because only those weights get multiplied by non-zero features. The gradients to those weights will be non-zero if  $2 * (f_1 + f_3) > 0$  because of the ReLU activation function.



Name: \_\_\_\_\_

