

6.036: Final Exam, Spring 2021

- This is an open-book exam. You may use any materials you want (electronic or otherwise, including notes, calculators, the 6.036 online material, Python, and Wikipedia) during the exam, but you are not allowed to converse with other humans (including through text message, email, etc.) from the time you start the exam until 24 hours afterward.
- It was designed to be a 3-hour exam, but you have an extra hour to help account for upload/download time.
- You may complete it by: (a) printing the pdf, writing on it, and uploading photos or scans; (b) writing on the pdf using a tablet and uploading the pdf; (c) writing answers on blank paper and uploading scans or photos.
- 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, indicate that you are continuing your answer, write on a blank page 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.
- We will not be answering questions about the exam via email or Piazza. An exception is if you are **very sure** there is a significant error, in which case, please make a private Piazza post. Any information we provide in response will be emailed to the entire class.

Question:	1	2	3	4	5	6	7	8	9	10	11	12	13	Total
Points:	1	8	8	10	10	6	8	8	5	10	8	12	6	100
Score:														

1 Name

1. (1 point) (a) Name:

(b) Kerberos (MIT username):

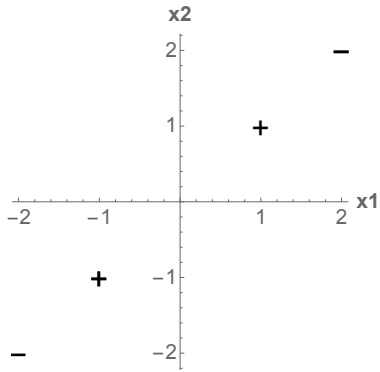
(c) I acknowledge that I am taking this examination without proctoring. I will not discuss the contents or characteristics of this examination with any person other than the 6.036 instructors before Wednesday, May 26, 2021.

Signature:

2 Transformative experience

2. (8 points) For each of the datasets below, find a transformation from the original data into a **single** new feature $\phi((x_1, x_2))$ such that the data is linearly separable in the new space, and specify the parameters θ and θ_0 of the separator in the transformed space.

(a)

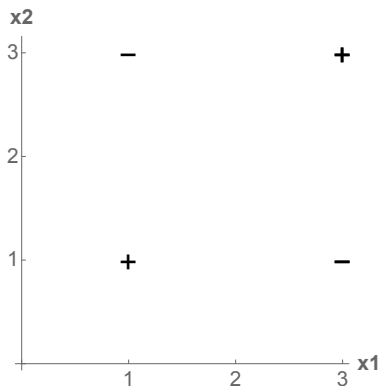


$$\phi((x_1, x_2)) =$$

$$\theta =$$

$$\theta_0 =$$

(b)



$$\phi((x_1, x_2)) =$$

$$\theta =$$

$$\theta_0 =$$

3 Feature encoding

3. (8 points) You are trying to predict whether start-up companies will succeed or fail, based on the name of the company. You have the following data set:

x	y
"Aardvarkia"	+1
"Fro"	+1
"Rodotopo"	-1
"Whoodo"	-1

You consider two different encodings of the features:

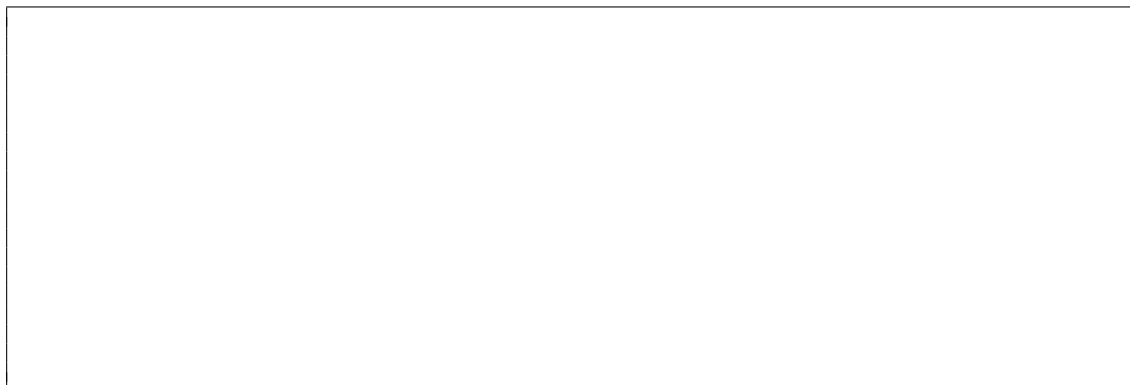
- **One-hot encoding**, with the first feature corresponding to “Aardvarkia,” the second to “Fro,” third to “Rodotopo,” and fourth to “Whoodo.”
- **Numerical encoding**, using the numerical place of the first letter of the name in the English alphabet (so 'A' is 0 and 'Z' is 25).

- (a) Provide parameters of a 0-error linear separator using one-hot encoding.

- (b) Provide parameters of a 0-error linear separator using the numeric encoding.

- (c) You add a new company with name “Zzyzygy” and class +1. If you extend the one-hot encoding to add another feature corresponding to this company name, will this new data set be linearly separable using the one-hot encoding? Explain briefly.

- (d) If you add this company to your data set but use the numeric encoding, is the new data set linearly separable? Explain briefly.



4 Off-balance logistic regression

4. (10 points) It is common in classification problems for the cost of a false positive (predicting positive when the true answer is negative) to be different from the cost of a false negative (predicting negative when the true answer is positive). This might happen, for example, when the task is to predict the presence of a serious disease.

Let's say that the cost of a correct classification is 0, the cost of a false positive is 1, and the cost of a false negative (that is, you predict 0 when the correct answer was +1) is α .

Recall that the usual logistic regression loss is:

$$\mathcal{L}_{\text{nl}}(g, y) = -(y \cdot \log g + (1 - y) \cdot \log(1 - g)) \ .$$

- (a) What is the usual loss, as a function of guess g , when the true label $y = 0$?

- (b) What is the usual loss, as a function of guess g , when the true label $y = 1$?

- (c) Jon suggests using a simple modification of the usual logistic regression loss function. Write down a loss function that penalizes false negatives α times more than false positives.

- (d) Jun proposes that we can find a classifier that optimizes the classification cost without changing our logistic regression loss function, by *rebalancing* the training data, that is, adding multiple copies of each of the points in one of the classes. For $\alpha = 3$, explain briefly how you would change the data.

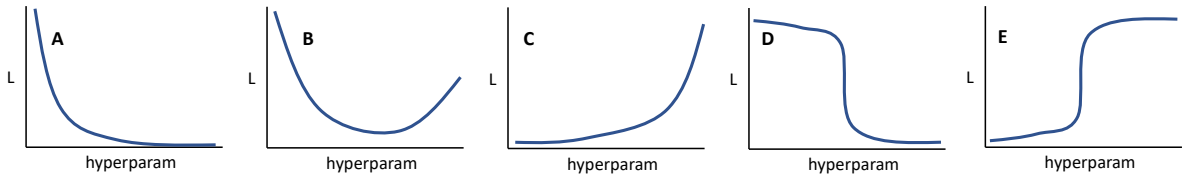
- (e) Would Jun's approach result in a classifier that optimizes the classification cost when using the Perceptron algorithm when the data are linearly separable? Explain briefly why or why not.

- (f) Would Jun's approach result in a classifier that optimizes the classification cost when using the Perceptron algorithm when the data are **not** linearly separable? Explain briefly why or why not.

- (g) Usually, in logistic regression, we predict class +1 when $a > 0.5$ and -1 otherwise. Jin proposes that we can use the standard logistic regression loss function and the same data set, but change the threshold of 0.5 that we use to select a prediction. Would you increase or decrease the threshold when $\alpha = 3$?
- Increase Decrease
- (h) Suggest a strategy that Jin can use for picking a new threshold that minimizes our average asymmetric cost of classification.

5 Hyper active

5. (10 points) We have looked at many machine-learning algorithms with hyper-parameters. Varying each of them has an effect on the loss on both the *training* data and on unseen *testing* data. Here are some plots of plausible ways that training or testing error might depend on the value of a hyperparameter.



For each of the following hyper-parameters, indicate which **one** of the plots above would be the most typical behavior, for both training and testing error. If none of them is appropriate, explain.

- (a) λ : the weight on the regularization term in logistic regression

i. Training: A B C D E None

If None, explain briefly.

ii. Testing: A B C D E None

If None, explain briefly.

- (b) η : the step-size in gradient descent for neural networks (assuming a fixed number of iterations)

i. Training: A B C D E None If None, explain briefly.

ii. Testing: A B C D E None

If None, explain briefly.

(c) D : the maximum depth of a decision tree

- i. Training: A B C D E None
If None, explain briefly.

- ii. Testing: A B C D E None
If None, explain briefly.

(d) k : the number of neighbors in nearest-neighbor classification

- i. Training: A B C D E None
If None, explain briefly.

- ii. Testing: A B C D E None
If None, explain briefly.

(e) T : the number of epochs of gradient-descent to perform

- i. Training: A B C D E None
If None, explain briefly.

- ii. Testing: A B C D E None
If None, explain briefly.

6 HairNet and PairNet

6. (6 points) In this question, we'll look at two neural-network structures and compare and contrast to convolutional neural networks. The PairNet is the same as from NQ4, but you don't need to remember it.

PairNet definition: The PairNet is parameterized by (a) the weights W of a small network NN and (b) one more scalar w_0 . If there are a total of d features in the input, it has the final form of:

$$y = \sigma(w_0 + \sum_{j \in \{1, \dots, d\}, k \in \{1, \dots, d\}, j \neq k} \text{NN}([x_j, x_k]; W))$$

The small neural network, parameterized by weights W , takes a two-dimensional vector as input and generates a scalar output. We write it as $\text{NN}([x_i, x_j]; W)$. If this smaller neural network has multiple layers, then W includes all the weights of all the layers, including offsets. We apply *PairNet* to an image by letting x_i and x_j be pairs of pixel values drawn from throughout the input image, and d is the total number of pixels.

HairNet definition: A hairnet has *hairny* layers and max pooling layers. A hairy layer is a lot like a convolutional layer, but it uses a different set of weights on each image patch.

Define the local 3 x 3 region of the (zero-padded) **input** image I around pixel i, j :

$$R(I, i, j) = (I_{i-1, j-1}, I_{i-1, j}, I_{i-1, j+1}, \\ I_{i, j-1}, I_{i, j}, I_{i, j+1}, \\ I_{i+1, j-1}, I_{i+1, j}, I_{i+1, j+1})$$

where $I_{i,j}$ is the pixel i, j of I . Pixel i, j of the **output** image is computed as the dot product of $R(I, i, j)$ and a weight vector and offset *for each image location*, $W^{i,j}$ and $W_0^{i,j}$. So output pixel $O_{i,j} = W^{i,jT} R(I, i, j) + W_0^{i,j}$.

- (a) Consider the number of parameters in a HairNet. Is it bigger or smaller than a fully connected network on an image of size 100 x 100? Explain briefly.

- (b) For a 100 x 100 image, is the number of parameters in a HairNet bigger or smaller than a CNN with a single convolutional layer with a 3 x 3 filter? Explain briefly.

CNNs are often described as exploiting spatial locality and translation invariance.

- (c) PairNets exploit spatial locality translation invariance both neither
Explain your answer briefly.

- (d) HairNets exploit spatial locality translation invariance both neither
Explain your answer briefly.

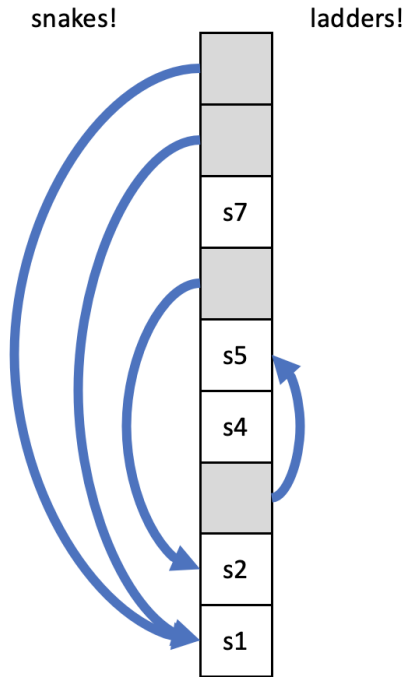
The parameters of a CNN trained on images of one size can often be applied successfully to images of another size.

(e) Is this true of PairNet? Yes No Explain your answer briefly.

(f) Is this true of HairNet? Yes No Explain your answer briefly.

7 Snakes and ladders

7. (8 points) You are playing a simplified version of a classic game, sometimes called “Snakes and Ladders” in English.



- You are moving up a 1-dimensional track with squares s_1, s_2, s_4, s_5, s_7 , as shown above.
- You have two actions: **climb** and **quit**.
- If you **climb** from state s_i then with probability 0.5 you go up one square, and with probability 0.5 you go up two squares.
- **However!** if the square you land on has a ladder going *up* from it, then you automatically, instantaneously, with probability 1 move to the square the ladder goes to.
- **And!** if the square you land on has a snake going *down* from it, then you automatically, instantaneously, with probability 1 move to the square the snake goes to.
- **So:** for example, in our case, if you start in state s_5 and **climb** there is a .5 chance you'll end up in square s_2 (because you move up one but hit a snake and fall back down) and a .5 chance you'll end up in square s_7 . If you **climb** from s_7 then you will *go back to square 1* with probability 1.0.
- If you **quit** then the game is over and you get to take no further actions.
- Each new episode starts in state s_1 .
- The reward for choosing **climb** in any state is 0.
- The reward for choosing **quit** in state s_i is i .

In the following, you need to supply actions or Q-values for the states s_1, s_2, s_4, s_5 and s_7 .

(a) What is the optimal horizon 1 policy?

s_1	s_2	s_4	s_5	s_7

(b) If you initialize the Q values of all the states to 0, and do one iteration of undiscounted ($\gamma = 1$) value iteration, what is the resulting Q value function?

	s_1	s_2	s_4	s_5	s_7
$Q(s, \mathbf{quit})$					
$Q(s, \mathbf{climb})$					

(c) If $\gamma = 1$ (that is, there is no discounting) what is the optimal infinite-horizon policy?

s_1	s_2	s_4	s_5	s_7

(d) Now let's consider discounting. State an inequality involving numeric values, γ , $Q(s_2, \mathbf{climb})$, and $Q(s_7, \mathbf{climb})$, specifying the condition under which the optimal action in s_5 is to **quit**.

8 Sneaking snakes and lurking ladders

8. (8 points) *Note that this is a continuation of question 7. The answers are independent, but the problem set-up is shared.*

What if we have to play snakes and ladders, but we don't know where the snakes and ladders actually are? Assume we know the rewards, but not the transition model. We'll use Q learning to address this problem.

During learning, after executing the **quit** action and getting a reward, a new learning episode is initialized, after *we go back to square 1*. We will use a discount factor of $\gamma = 1$ throughout.

- (a) Why is value iteration not a good choice of algorithm for this problem?

- (b) If we do purely greedy action selection *during* Q-learning (that is $\epsilon = 0$), starting from all 0's in our Q table and where ties are broken in favor of the **climb** action, what (roughly) will the Q function be after 1000 steps?

- (c) If we do purely greedy action selection *during* Q-learning (that is $\epsilon = 0$), starting from all 0's in our Q table and where ties are broken in favor of the **quit** action, what (roughly) will the Q function be after 1000 steps?

- (d) Assume we start with a tabular Q function on states s_1, s_2, s_4, s_5 and s_7 , initialized to all 0's. Then we get the following experience, consisting of three trajectories, shown below as (state, action, reward) tuples.

Using learning rate $\alpha = 0.5$, what is the Q function after each of these trajectories? (You only need to fill in the non-zero entries). *Do not reset all your Q values back to 0 after each trajectory—assume this is all a single execution of the Q learning algorithm on 3 episodes of experience.*

Trajectory 1: $((s_1, \text{climb}, 0), (s_2, \text{climb}, 0), (s_5, \text{climb}, 0), (s_7, \text{quit}, 7))$

	s_1	s_2	s_4	s_5	s_7
$Q(s, \text{quit})$					
$Q(s, \text{climb})$					

Trajectory 2: $((s_1, \text{quit}, 1))$

	s_1	s_2	s_4	s_5	s_7
$Q(s, \text{quit})$					
$Q(s, \text{climb})$					

Trajectory 3: $((s_1, \text{climb}, 0), (s_2, \text{climb}, 0), (s_5, \text{climb}, 0), (s_7, \text{quit}, 7))$

	s_1	s_2	s_4	s_5	s_7
$Q(s, \text{quit})$					
$Q(s, \text{climb})$					

9 Grid bug?

9. (5 points) Kim is running Q learning on a simple 2D grid-world problem and visualizes the current Q value estimates and greedy policy with respect to the current Q value estimates at each location. States correspond to squares in the grid, actions are **north**, **south**, **east**, and **west**, and there is a single state with non-zero reward. Assume the transition model is deterministic and action **north** moves you one square up, etc., except at the boundaries of the domain. Diagonal moves are not possible.

(a) Define the following in terms of the current estimated action-value function, \hat{Q} .

- i. The greedy policy with respect to \hat{Q} for state s :

$$\pi(s) =$$

- ii. The estimated value of state s : $\hat{V}(s)$

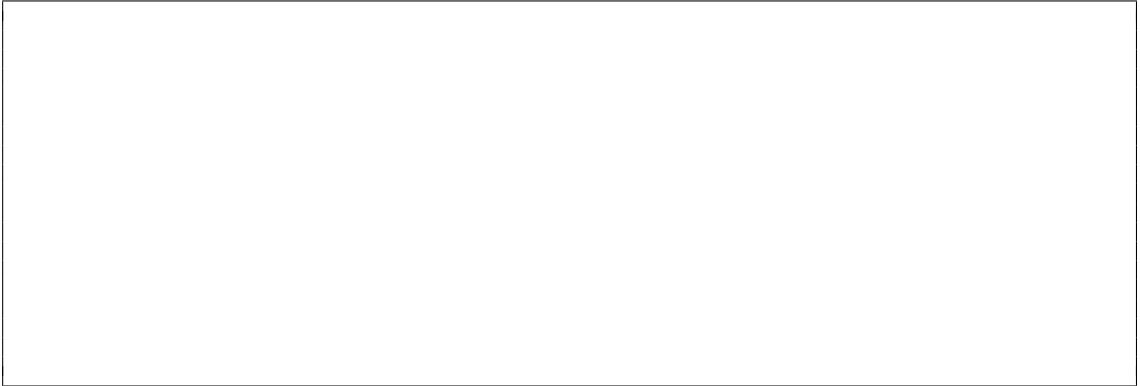
$$\hat{V}(s) =$$

- (b) Kim sees the situation below while their algorithm is running. The numbers in the boxes correspond to the estimated \hat{V} values for the states neighboring state s , and the arrow indicates the greedy action with respect to \hat{Q} for state s . All of the states shown have 0 reward values.

		2.2	
1.7	$s \uparrow$	0.9	
	3.1		

Explain briefly why this situation might be concerning.

- (c) Does this situation mean that there is a bug in Kim's Q-learning implementation? Explain briefly why or why not.



10 Heavy neighbors

10. (10 points) Given a set of data $\mathcal{D}_{\text{train}} = \{(x^{(i)}, y^{(i)})\}$, a weighted nearest neighbor regressor has the form

$$h(x; \theta) = \frac{\sum_{(x^{(i)}, y^{(i)}) \in \mathcal{D}_{\text{train}}} f(x, x^{(i)}; \theta) y^{(i)}}{\sum_{(x^{(i)}, y^{(i)}) \in \mathcal{D}_{\text{train}}} f(x, x^{(i)}; \theta)} .$$

A typical choice for f is

$$f(x, x'; \theta) = e^{-\theta \|x - x'\|^2}$$

where θ is a scalar and $\|x - x'\|^2 = \sum_{j=1}^d (x_j - x'_j)^2$. One way to think about this is that the prediction at a new point x is a weighted combination of the outputs at all the training points, with training points closer to x having more influence than those that are far away.

- (a) Assume our training data $\mathcal{D}_{\text{train}} = ((1, 1), (2, 2), (3, 6))$. What is $h(10; 0)$? That is, letting $\theta = 0$, what is our prediction for $x = 10$?

- (b) On this same data set, **approximately** what is $h(10; 1)$?
Just give us the closest integer—calculator should not be required.

- (c) How does this prediction (for $\theta = 1$) compare (qualitatively) to the prediction that would result from linear regression on this same model? Why might we prefer one over the other?

- (d) If we were only ever going to have to make predictions on the training data, what value of θ would tend to minimize our prediction error?

- (e) Dino thinks the denominator in the definition of h is not useful and it would be fine to remove it. Is Dino right? Explain briefly.

11 DeepRNN

11. (8 points) Ronnie makes a simple RNN with state dimension 1 and a *step* function for f_1 , so that

$$s_t = \text{step}(w_1 x_t + w_2 s_{t-1} + b)$$

where $\text{step}(z) = 1$ if $z > 0.0$ and equals 0 otherwise, and where the output

$$y_t = s_t .$$

- (a) Assuming $s_0 = 0$, what values of w_1 , w_2 and b would generate output sequence

$$[0, 0, 0, 1, 1, 1, 1]$$

given input sequence

$$[0, 0, 0, 1, 0, 1, 0]$$

$w_1 =$
$w_2 =$
$b =$

- (b) Now Ronnie wants to make their machine generate output sequence

$$[1, 1, 1, 0, 0, 0, 1, 1]$$

given input sequence

$$[0, 0, 0, 1, 0, 0, 1, 0]$$

Assuming $s_0 = 1$, provide the desired s_t value for each possible combination of x_t and s_{t-1} values, for this example sequence.

x_t	s_{t-1}	
0	0	
0	1	
1	0	
1	1	

(c) Rennie thinks this is not possible using Ronnie's architecture. Rennie makes an argument based on the relationships in the table above. Is Rennie right?

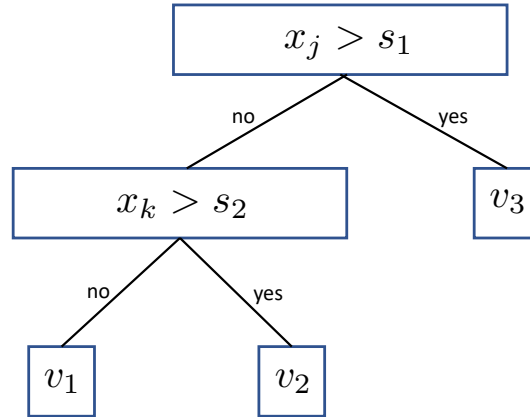
This is **not possible** This is **possible**

If you answered **not possible**, explain why and describe a change to this architecture that can implement this mapping. If you answered **possible**, provide parameters w_1 , w_2 , and b that implement this mapping.



12 Trees that bend

12. (12 points) Here is a standard regression tree of a fixed size. It has 5 scalar parameters $(s_1, s_2, v_1, v_2, v_3)$ and two discrete choices of feature to split on, denoted by integers j and k .



We are given a training data set $\mathcal{D}_{\text{train}} = \{(x^{(j)}, y^{(j)})\}$ where the dimension of $x^{(j)}$ is d .

- (a) Explain briefly why we cannot use gradient descent on a squared loss to optimize all the parameters of this predictor.

Terry would like to make a “smoother” tree by replacing the tests at the nodes with neural-network logistic classifiers and by combining predictions from the branches, so that we can think of the tree as a parametric model and optimize the parameters using gradient descent. More concretely, at each internal node, the test will be replaced by $\text{NN}(x; \theta)$, a neural network that takes an entire input vector x , of dimension d , as input and generates an output in the range $[0, 1]$ by using a sigmoid unit on the output.

You can think of any node T_i of a tree as producing an output value as follows:

- If T_i is a leaf, then the output on input x , $T_i(x)$, is a constant v_i .
- If T_i is an internal node with children T_{no} (corresponding to the “no” branch) and T_{yes} (corresponding to “yes” branch), then the output on input x is

$$T_i(x) = (1 - \text{NN}(x; \theta^{(i)}))T_{\text{no}}(x) + \text{NN}(x; \theta^{(i)})T_{\text{yes}}(x) .$$

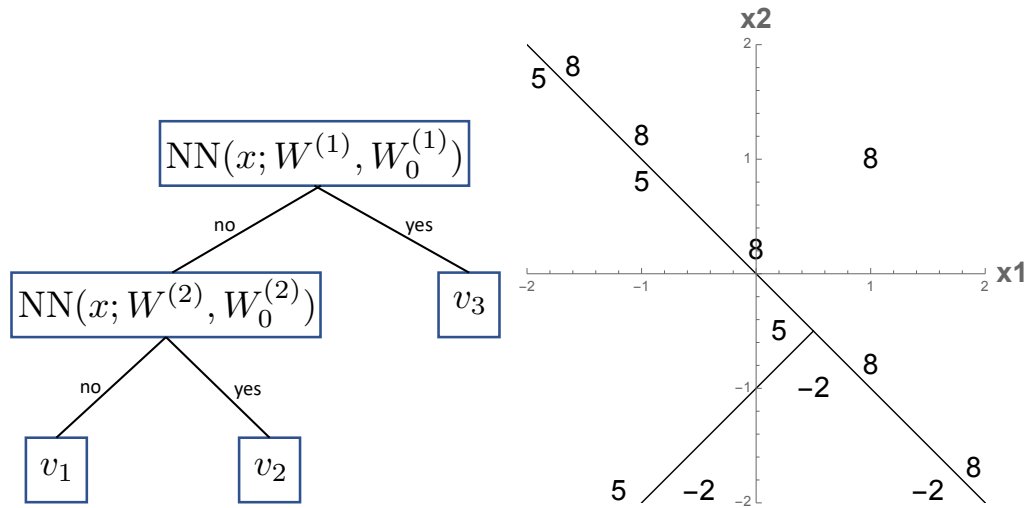
That is, it is a weighted combination of the results of the children, where the neural network at the parent node, with parameters $\theta^{(i)}$, modulates the combination of the results of the children.

We will consider the specific case where NN is a single unit with a sigmoidal activation function, so that

$$\text{NN}(x; W^{(i)}, W_0^{(i)}) = \sigma(W^{(i)T} x + W_0^{(i)})$$

where $W^{(i)}$ is a vector of length d and $W_0^{(i)}$ is a scalar and σ is the sigmoid function.

- (b) Consider the dataset shown in the plot below right, where $d = 2$. Each integer value on the plot (one of 5, -2, or 8) corresponds to a datapoint whose input x features are the coordinates of the point on the plot and whose output y value is the printed number.



Provide the parameters of a tree-predictor, corresponding to the model shown above left, that make accurate predictions on the dataset.

$W^{(1)} =$

$W_0^{(1)} =$

$W^{(2)} =$

$W_0^{(2)} =$

$v_1 =$

$v_2 =$

$v_3 =$

(c) What is $\partial T_1(x)/\partial W^{(1)}$ in this particular model? Please use the following shorthand:

- $T = T_1(x)$
- $O = \text{NN}(x; W^{(1)}, W_0^{(1)})$
- $T_{\text{no}} =$ the output of the “no” branch of T_1
- $T_{\text{yes}} =$ the output of the “yes” branch of T_1

Express your answer in terms of these quantities, x , and parameters $(W^{(1)}, W^{(2)}, W_0^{(1)}, W_0^{(2)}, v_1, v_2, v_3)$, as needed, but do not leave any derivatives in it.

(d) Tori thinks that since regression trees have repeated structure, similar to a CNN, that we should use the same weight vector W and offset W_0 at all the internal nodes. Explain the hypothesis class that results.

13 Discretization

13. (6 points) Sam wants to build a neural network that takes in a scalar value x in the range $[0, 1]$ and generates a one-hot output vector y of dimension K , where, for $k \in \{0, 1, \dots, K-1\}$, $y_k = 1$ if and only if $k/K < x \leq (k+1)/K$; that is, it discretizes the interval into K equally sized sequential ranges. Please don't worry about precisely what the output is at the boundaries of the intervals.

They choose an architecture with a single linear layer with weights W and W_0 and a softmax activation function, so that the output

$$a = \text{softmax}(z)$$

where

$$z = W^T x + W_0 .$$

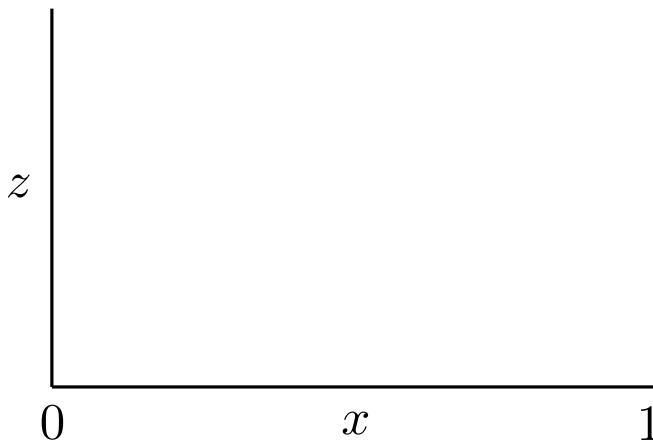
Assume that, for prediction purposes, we are going to take the output of the network, a , and convert it into a K -dimensional one-hot vector (y_0, \dots, y_{K-1}) where

$$y_i = \begin{cases} 1 & \text{if } i = \arg \max_j a_j \\ 0 & \text{otherwise} \end{cases}$$

That is, it has a value of 1 at the index corresponding to the maximal element of a and value 0 everywhere else.

- (a) How many trainable weights does this network have when $K = 10$?

- (b) Let's consider the case of $K = 3$. On the axes below, draw the three components of the z vector, z_0 , z_1 , and z_2 , as a function of x so that the resulting y will provide a correct discretization of the interval into three equal regions. (There are many correct solutions.)



- (c) Provide a set of weight values that will discretize the unit interval into 3 equal parts, with output predictions $y = [1, 0, 0]$ for $x \in [0, 1/3]$, $y = [0, 1, 0]$ for $x \in [1/3, 2/3]$, and $y = [0, 0, 1]$ for $x \in [2/3, 1]$. *Please don't worry about exactly what happens at the boundaries!!!*

$W_0 =$

$W =$