6.036: Final Exam, Fall 2019

Solutions

- This is a closed book exam. Two sheets of paper (8 1/2 in. by 11 in.) of notes, front and back, are permitted. Calculators are not permitted.
- 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 absolutely *have* to ask a question, come to the front.
- **Write your name on every page.**

Name: _____ Athena ID: _____
(username)

Question:	1	2	3	4	5	6	7	8	Total
Points:	12	18	8	14	16	12	12	8	100
Score:									

Rev. r10

Name: _____

Shoe Shop

1. (12 points) General Ization is consulting for a shop that sells shoes, and the General is building a model to predict what color of sneakers a given customer will buy, given information about their age and the color of the shoes they're wearing when they enter the store. The shoe shop asks for a classifier, as well as an indication of how well the classifier will perform once deployed.

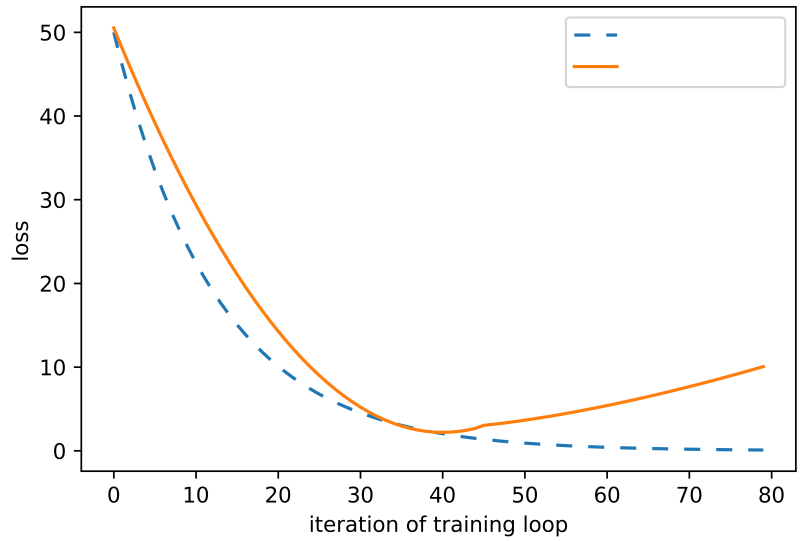
   (a) There are several sneaker colors that customers might wear or buy. The General wishes to train a neural network classifier. What representation is best for the input (the color of shoes a customer is wearing when they enter the store)?

   Solution: One-hot encoding of shoe color. Alternative approach is RGB encoding of color.

   What kind of output layer should the General use?

   Solution: Softmax is most appropriate for multiclass classification output.

   (b) The store gives the General data from the past year of sales, which she splits into three distinct parts: training data, validation data, and test data. While training the neural network classifier, the General gets the following learning curves. This graph indicates that she should use the classifier resulting from training after fewer than 80 iterations. Unfortunately, she forgot to put the legend in, but luckily you can fix it! Fill in the legend with the appropriate two among `training_time`, `training_loss`, `validation_loss`.



   Solution: Solid line = validation_loss; Dashed line = training_loss

(c) Around how many iterations should the General use to train the classifier she delivers to the shoe shop? Explain why.

Solution: Around 40, as this is when the validation loss starts to increase.

(d) The General made a grave mistake. It turns out that though she thought she had split the data into three parts, she had only split it into two and used both those splits in training and selecting her classifier. Now, she needs to collect the third split in order to indicate how well her classifier will perform when deployed. Which of the following would be the best to use? Provide a short justification for your choice.

1. Go to a nearby school and ask the students what color sneakers they used to own and note what color sneakers they are currently wearing.
2. Go to a nearby construction site and ask the workers what color shoes they used to own and note what color shoes they are currently wearing.
3. Ask the shoe store to give her more data in two months.
4. Ask a different shoe store for their data.

Solution: Either 3 or 4 would be best. 3 would better mirror the distribution they would see in that store (though there is risk of covariate shift over time), but if the store is in a rush to deploy the model, then the delay might not be possible. 4 would be faster but might not match the distribution of the original store as well.

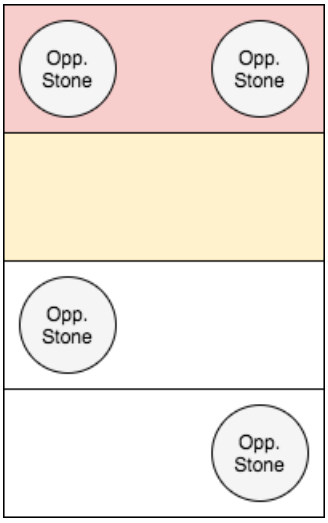
(e) The store goes back to the General and says they've discovered a new feature they think might be useful: the color of shoes that a famous celebrity, Keslie Laelbling, is wearing that day. (Due to social media, both the customer and the store know exactly what color of shoes Keslie is wearing each day.) Unfortunately, the General is close to her deadline: she has time to train a new linear model, but not to train another deep neural network like she did before. How might the General produce an augmented model that incorporates this new feature?

Solution: Train a linear classifier that uses the neural network plus the color of Keslie's shoes as input, and produces a new prediction for what color of shoes the customer will buy. For example, the linear classifier could take the softmax layer inputs a_{NN} from the last layer of the original neural network, together with the color of Keslie's shoes x_{KL}, and train θ coefficients as $a_{\text{new}} = \theta_{\text{NN}} \cdot a_{\text{NN}} + \theta_{\text{KL}} \cdot x_{\text{KL}} + \theta_0$ with a new overall output $y = \text{softmax}(a_{\text{new}})$.

Name: _____

Curling

2. (18 points) You have designed a robot curling stone to enter a modified curling contest.[1] In an attempt to get your robot stone to perform well, you have designed a state and action space, a reward structure, and a transition model. The goal of the robot stone is to slide upwards on an ice sheet and stop in a target region. Your robot stone likes to show off; after each state transition, it displays the reward it receives. In addition to your robot stone, there will be a number of opponent stones on the ice, as shown below. For simplicity's sake, we will consider the opponent stones to be fixed.

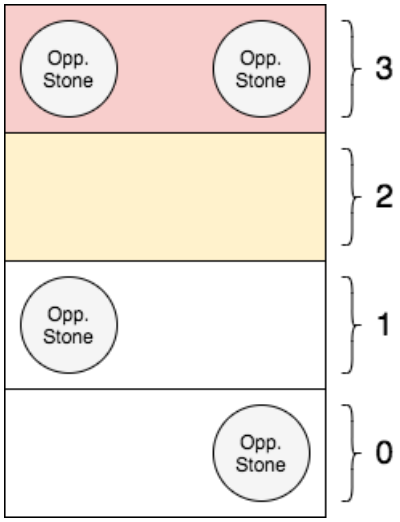


[1]Curling is an Olympic sport involving granite stones about 36 inches in circumference and 4.5 inches in height, that are directed toward target regions on an ice rink, and that might be defended or attacked by an opponent's stones. MIT has a Curling Club that you might enjoy.

Your model for the state and action spaces is as follows:

$$S \in \{t, 0, 1, 2, 3\}$$
$$a \in \{\text{“go”}, \text{“stop”}\}$$

where the states refer to the robot stone being in either a terminal state (denoted as t) or within one of the four regions below:



Name: _____

You design the following reward function and (deterministic) transition model for your robot stone:

$R(s,a)$:

	action "go"	action "stop"
state 3	0	2
state 2	1	1
state 1	1	0
state 0	1	0

$T(s,a,s')$:

$T(0,$ "go", $1) = 1$
$T(1,$ "go", $2) = 1$
$T(2,$ "go", $3) = 1$
$T(3,$ "go", t$) = 1$
$T($*, "stop", t$) = 1$

and all other transition probabilities are 0. Here * indicates any state. Note that once the robot stone enters state t the game ends: there is no transition and zero reward out of state t (and hence no action to decide once in state t.) Together with this reward function and transition model, you specify a discount factor $\gamma = 1$.

(a) In order to enable decision making by your robot stone, you need to give it the optimal policy $\pi^*(s)$. For your reward and transition structure and discount factor $\gamma = 1$, what are the optimal Q-values, $Q^*(s,a)$? What is the optimal policy $\pi^*(s)$? Fill in the following two tables.

$Q^*(s,a)$:

	action "go"	action "stop"
state 3	0	2
state 2	3	1
state 1	4	0
state 0	5	0

$\pi^*(s)$:

	$\pi^*(s)$
state 3	stop
state 2	go
state 1	go
state 0	go

Unfortunately, your competitor has also designed a robot stone. You do not know your competitor's reward structure $R(s, a)$ or transition model $T(s, a, s')$; however, you do know they use the same state and actions spaces. Instead, you decide to use Q-learning to observe their robot stone and learn from it! For your Q-learning, use discount factor $\gamma = 1$ and learning rate $\alpha = 0.5$, with a Q table initialized to zero for all (s, a) pairs.

(b) Your competitor runs their robot through a first game, exhibiting the following experience:

step #	s	a	r	s'
1	0	"go"	1	1
2	1	"stop"	0	t

You perform Q-learning updates based on the experience above. After observing steps 1 and 2 (the first game), what is the learned $Q(0, \text{"go"})$?

Solution: We know $Q(s, a) := \alpha Q(0, a) + \alpha(r + \gamma \max_{a'} Q(0, a_i)$ So step #1 causes the following update: $(1 - \alpha)Q(0, a) + \alpha\left(r + \gamma \max_{a'} Q(1, a')\right)$

$$Q(0, \text{"go"}) = 0.5 \cdot 0 + 0.5(1 + 1 \cdot 0) = 0.5$$

What is the learned $Q(1, \text{"stop"})$?

Solution:

$$Q(1, \text{"stop"}) = 0.5 \cdot 0 + 0.5(0 + 1 \cdot 0) = 0$$

(c) Your competitor runs their robot through a second game, exhibiting the following **additional** experience:

step #	s	a	r	s'
3	0	"go"	1	1
4	1	"go"	1	2
5	2	"go"	1	3
6	3	"stop"	2	t

You perform additional Q-learning updates based on this additional experience. After completion of both games (all six steps), what are the full set of Q values you have learned for their robot? Fill in the following table.

$Q(s, a)$:

state s	action "go"	action "stop"
3	0	1
2	0.5	0
1	0.5	0
0	0.75	0

(d) We can think of learning the Q-value function for a given action as a regression problem with each state s mapped to a one-hot feature vector $x = \phi_A(s)$, where $x = [1\ 0\ 0\ 0]^T$ for state 0, $x = [0\ 1\ 0\ 0]^T$ for 1, etc., and $x = [0\ 0\ 0\ 0]^T$ for state t.

We'll focus on the action "go". We would like to come up with parameters θ, θ_0 such that $Q(s, \text{"go"}) = \theta \cdot \phi_A(s) + \theta_0 = \theta \cdot x + \theta_0$. Is there in general — for arbitrary values of our $Q(s, \text{"go"})$ — a setting of θ, θ_0 that enables representation of $Q(s, \text{"go"})$ with perfect accuracy? If so, provide the corresponding θ and θ_0. If not, explain why. (Note that we do not need to model $Q(t, a)$, since the game is over once state t has been reached.)

Solution: Yes; θ_i is simply the value for $Q(s = i, \text{"go"})$ and $\theta_0 = 0$.

Note: $\theta = [5\ 4\ 3\ 0]^T$ and $\theta_0 = 0$ would work for our optimal $Q^*(s, a)$, but we seek a more general θ corresponding to arbitrary or general $Q(s, a)$.

(e) Unfortunately, your robot's GPS system suddenly breaks, and it is no longer able to tell which of the four regions it is in. However, the robot has side cameras which can detect the opponent stones as it travels through the center of the ice, encoded as [(number of stones to immediate left) (number of stones to immediate right)]^T. You decide to use this information as state, giving the following feature transformation ϕ_B on your original state:

$$\phi_B(3) = [1\ 1]^T$$
$$\phi_B(2) = [0\ 0]^T$$
$$\phi_B(1) = [1\ 0]^T$$
$$\phi_B(0) = [0\ 1]^T$$

We would still like to come up with parameters θ, θ_0 such that $Q(s, \text{"go"}) = \theta \cdot \phi_B(s) + \theta_0$, for general values of $Q(s, \text{"go"})$. Is there a setting of θ, θ_0 that enables representation of this encoding of $Q(s, \text{"go"})$ with perfect accuracy? If so, provide the corresponding θ and θ_0. If not, explain why this is not possible, and provide a feature transformation $\phi_C(\cdot)$ that does enable representation of $Q(s, \text{"go"}) = \theta \cdot \phi_C(\phi_B(s)) + \theta_0$ with perfect accuracy.

Solution: No. Let $[x_1\ x_2] = \phi_B(s)$, so $\theta_1 x_1 + \theta_2 x_2 + \theta_0 = Q(s, \text{"go"})$. $\phi_B(2)$ forces $\theta_0 = Q(2, \text{"go"})$; $\phi_B(1)$ forces θ_1; $\phi_B(0)$ forces θ_2; and we no longer have the ability to find θ for $\phi_B(3)$.

We can create ϕ_C as a one-hot encoding of state such that $\phi_C(\phi_B(s)) = \phi_A(s)$ to uniquely identify our four states (with corresponding θ and θ_0 as in the previous part) to regain perfect representational power.

Name: _____

The Missing Rating

3. (8 points) Mitch Mitdiddle has kept track of ratings of three items from three different users of his new e-commerce website. These items are essential, and so each user has purchased and rated all three of the items. However, Mitch has lost one of the ratings, r. The (almost) complete ratings matrix is here:

$$Y = \begin{bmatrix} 6 & 8 & 10 \\ 9 & 12 & r \\ 3 & 4 & 5 \end{bmatrix}$$

Note, users a and items i are indexed from 1, i.e., the first row of Y corresponds to user 1 ($a = 1$), and the first column of Y corresponds to item 1 ($i = 1$).

(a) Treating r as a missing value, is there a rank-1 representation of Y as UV^T (i.e., such that UV^T produces a matrix that perfectly matches the non-missing elements of Y)? If yes, provide matrices U and V of shape 3×1 such that $Y = UV^T$. If no, explain why not.

Solution: $U = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}, V = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}$.
Other solutions exist if student scales U by s and V by $\frac{1}{s}$.

An e-commerce expert explains to you that users only care about one particular feature when it comes to rating products, and provides you with the value of this feature for each item, which we set as V:

$$V = \begin{bmatrix} 6 \\ 8 \\ 10 \end{bmatrix}$$

Mitch remembers that we can use the alternating least squares method to solve for U, minimizing:

$$J(U, V) = \frac{1}{2} \sum_{(a,i) \epsilon D} (U^{(a)} \cdot V^{(i)} - Y_{a,i})^2$$

where D is the set of all user a item i rating pairs (a, i). Here $U^{(a)}$ is the a^{th} row of U, and $V^{(i)}$ is the i^{th} row of V. Note that offsets are fixed at $b_U = 0$ and $b_V = 0$ in this problem.

(b) Using the same data matrix Y with missing value r and holding V constant, what is the value of $U^{(2)}$ (the second row of U) that minimizes J? Identify what (a, i) pairs and $Y_{a,i}$ values matter in this minimization, remembering that r (value of $Y_{2,3}$) is not involved.

Solution: First, we have $a = 2$, the second row of Y, so we only have $(a, i) \in \{(2, 1), (2, 2)\}$ since $(a, i) = (2, 3)$ is missing (our r value). We have a minimum at

$$\frac{dJ}{dU^{(2)}} = 6(6U^{(2)} - 9) + 8(8U^{(2)} - 12) = 0$$

$$100U^{(2)} - 150 = 0$$
$$U^{(2)} = 1.5$$

(c) What is our prediction for r, given the V and $U^{(2)}$?

Solution: $r = 10 \cdot 1.5 = 15$. Note that this is the same value as generated by our rank-1 representation, UV^T.

(d) Mark all that are true for our U, V, and Y above:

- √ **There are infinitely many settings of U and V that minimize J.**
- √ **For any constant (non-zero) V, $J(U)$ has a unique global minimum.**
- ◯ For any constant (non-zero) V, there exists a U such that $J(U, V) = 0$.
- √ **For any $m \times n$ matrix Y of rank 1, there exist matrices U and V of sizes $m \times 1$ and $n \times 1$ such that $J(U, V) = 0$.**

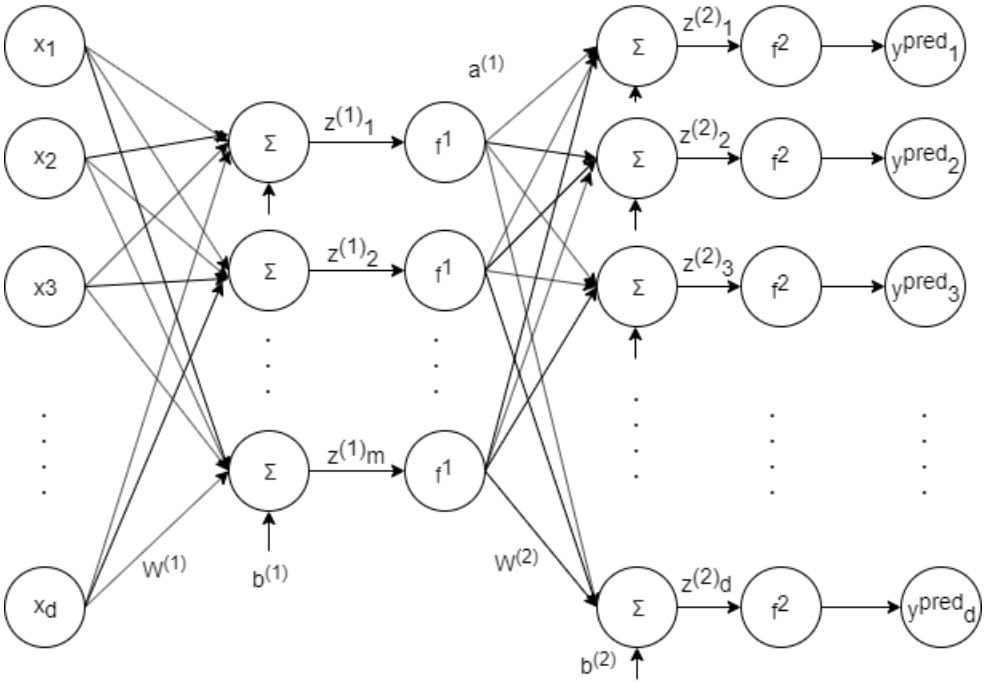
Solution:

- True: We can always scale U by s and V by $\frac{1}{s}$ and get equivalent product UV^T.
- True: The problem becomes a convex least squares linear regression problem. We exclude V being all zeros, as that results in a trivial J that is always zero.
- False: Imagine we had $V = [1, 1, 1]^T$. We cannot find U such that $UV^T = Y$ for all available Y values.
- True: In general, when Y has rank k, we can perfectly reconstruct Y from UV^T where U and V^T have inner dimension k.

Name: _____

Autoencoder

4. (14 points) Otto N. Coder is exploring different autoencoder architectures. Consider the following autoencoder with input $x \in \mathbb{R}^d$ and output $y^{pred} \in \mathbb{R}^d$. The autoencoder has one hidden layer with m hidden units: $z^{(1)}$, $a^{(1)} \in \mathbb{R}^m$.



$$z^{(1)} = W^{(1)}x + b^{(1)}$$
$$a^{(1)} = f^{(1)}(z^{(1)}) \quad \text{element-wise}$$
$$z^{(2)} = W^{(2)}a^{(1)} + b^{(2)}$$
$$y^{pred} = f^{(2)}(z^{(2)}) \quad \text{element-wise}$$

(a) Assume x, $z^{(2)}$, and y^{pred} have dimensions $d \times 1$. Also let $z^{(1)}$ and $a^{(1)}$ have dimensions $m \times 1$. What are the dimensions of the following matrices?

$W^{(1)}$	$b^{(1)}$	$W^{(2)}$	$b^{(2)}$
$m \times d$	$m \times 1$	$d \times m$	$d \times 1$

Name: _____

Otto trains the autoencoder with back-propagation. The loss for a given datapoint x, y is:

$$J(x, y) = \frac{1}{2}||y^{pred} - y||^2 = \frac{1}{2}(y^{pred} - y)^T(y^{pred} - y) \ .$$

Compute the following intermediate partial derivatives. For the following questions, write your answer in terms of x, y, y^{pred}, $W^{(1)}$, $b^{(1)}$, $W^{(2)}$, $b^{(2)}$, $f^{(1)}$, $f^{(2)}$ and **any previously computed or provided partial derivative**. Also note that:

1. Let $\partial f^{(1)}/\partial z^{(1)}$ be an $m \times 1$ matrix, provided to you.
2. Let $\partial f^{(2)}/\partial z^{(2)}$ be a $d \times 1$ matrix, provided to you.
3. If $Ax = y$ where A is a $m \times n$ matrix and x is $n \times 1$ and y is $m \times 1$, then let $\partial y/\partial A = x$.
4. In your answers below, we will assume multiplications are matrix multiplication; to indicate element-wise multiplication, use the symbol $*$.

(b) Find $\partial J/\partial y^{pred}$, a $d \times 1$ matrix.

Solution:

$$\frac{\partial J}{\partial y^{pred}} = (y^{pred} - y)$$

(c) Find $\partial J/\partial z^{(2)}$, a $d \times 1$ matrix. You may use $\partial J/\partial y^{pred}$ and $*$ for element-wise multiplication.

Solution:

$$\begin{aligned}\frac{\partial J}{\partial z^{(2)}} &= \frac{\partial J}{\partial y^{pred}} \frac{\partial y^{pred}}{\partial z^{(2)}} \\ &= \frac{\partial J}{\partial y^{pred}} * \frac{\partial f^{(2)}}{\partial z^{(2)}}\end{aligned}$$

Check: $(d \times 1) = (d \times 1) * (d \times 1)$ dimensioned arrays; element-wise multiplication.

(d) Find $\partial J/\partial W^{(2)}$, a $d \times m$ matrix. You may use $\partial J/\partial z^{(2)}$.

Solution:

$$\begin{aligned}\frac{\partial J}{\partial W^{(2)}} &= \frac{\partial J}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial W^{(2)}}^T \\ &= \frac{\partial J}{\partial z^{(2)}} \ a^{(1)^T} = \frac{\partial J}{\partial z^{(2)}} \ f^{(1)}(W^{(1)}x + b^{(1)})^T\end{aligned}$$

Check: $(d \times m) = (d \times 1)(1 \times m)$ dimensioned arrays. Note that we also accept $a^{(1)}$ even though it was not explicitly provided.

(e) Write the gradient descent update step for just $W^{(2)}$ for one datapoint (x, y) given learning rate η and $\partial J/\partial W^{(2)}$.

Solution:

$$W^{(2)} := W^{(2)} - \eta \frac{\partial J(x, y)}{\partial W^{(2)}}$$

(f) Otto's friend Bigsby believes that bigger is better. He takes a look at Otto's neural network and tells Otto that he should make the number of hidden units m in the hidden layer very large: $m = 10d$. (Recall that $z^{(1)}$ has dimensions $m \times 1$.) Is Bigsby correct? What would you expect to see with training and test accuracy using Bigsby's approach?

Solution: No; training accuracy might be high, but this would likely be due to overfitting and lead to worse test accuracy.

(g) Otto's other friend Leila says having more layers is better. Let m be much smaller than d. Leila adds 10 more hidden layers all with linear activation before Otto's current hidden layer (which has sigmoid activation function $f^{(1)}$) such that each hidden layer has m units. What would you expect to see with your training and test accuracy, compared to just having one hidden layer with activation $f^{(1)}$?

Solution: The intermediary hidden layers do not add any expressivity to the network, and we would expect similar training and test accuracy as compared to the single $f^{(1)}$ hidden layer network. This may, however, require different number of training iterations with the same available data, in order to achieve similar accuracy.

(h) Another friend Neil suggests to have several layers with non-linear activation function. He says Otto should regularize the number of active hidden units. Loosely speaking, we consider the average activation of a hidden unit j in our hidden layer 1 (which has sigmoid activation function $f^{(1)}$) to be the average of the activation of $a_j^{(1)}$ over the points x_i in our training dataset of size N:

$$\hat{p}_j = \frac{1}{N}\Sigma_{i=1}^{N}\, a_j^{(1)}(x_i)\ .$$

Assume we would like to enforce the constraint that the average activation for each hidden unit $\hat{p}_j$ is close to some hyperparameter p. Usually, p is very small (say $p < 0.05$).

What is the best format for a regularization penalty given hyperparameter p and the average activation for all our hidden units: $\hat{p}_j$? Select one of the following:

- ◯ Hinge loss: $\Sigma_j \max(0, (1-\hat{p}_j)p)$
- √ **NLL:** $\Sigma_j \left(-p \log \frac{p}{\hat{p}_j} - (1-p)\log \frac{(1-p)}{(1-\hat{p}_j)}\right)$
- √ **Squared loss:** $\Sigma_j (\hat{p}_j - p)^2$
- ◯ $l2$ norm: $\Sigma_j (\hat{p}_j)^2$

Solution: Either NLL or squared loss should work, encouraging p and $\hat{p}_j$ to be close. NLL loss might better handle wide range in the magnitudes of $\hat{p}_j$.

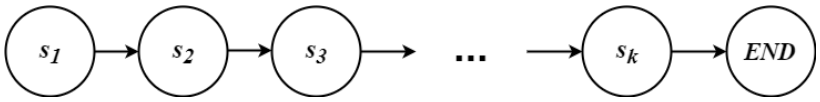
(i) Which pass should Otto compute $\hat{p}_j$ on? Select one of the following:

- √ **Forwards pass**
- ◯ Backwards pass
- ◯ Gradient descent step (weight update) pass

Name: _____

Go Positive, or Go Negative

5. (16 points) Consider the following simple MDP: **Positive Reward**



First consider the case where the MDP has positive reward. In this scenario, there is only one action ($next$); we name this decision policy π_A with $\pi_A(s) = next$ for all s. The reward is $R(s, next) = 0$ for all states s, except for state s_k where reward is $R(s_k, next) = 10$. We always start at state s_1 and each arrow indicates a deterministic transition probability $p = 1$. There is no transition out of the end state END, and 0 reward for any action from the end state.

(a) Calculate $V_\pi(s)$ for each state in the finite-horizon case with horizon $h = 1$, $k = 4$, and discount factor $\gamma = 1$.

Solution:

$$V_\pi^1(s_4) = 10$$
$$V_\pi^1(s_3) = 0$$
$$V_\pi^1(s_2) = 0$$
$$V_\pi^1(s_1) = 0$$

(b) Calculate $V_\pi(s)$ for each state in the infinite horizon case with $k = 4$ and discount factor $\gamma = 0.9$.

Solution:

$$V_\pi(s_4) = 10$$
$$V_\pi(s_3) = 0 + \gamma * 10 = 0.9 * 10 = 9$$
$$V_\pi(s_2) = 0.9 * 9 = 8.1$$
$$V_\pi(s_1) = 0.9 * 8.1 = 7.29$$

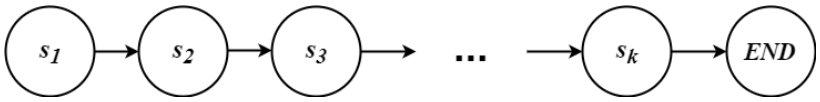
(c) Derive a formula for $V_\pi(s_1)$ that works for any value of (is expressed as a function of) k and γ for the above positive reward MDP, in the infinite horizon case.

Solution: At each step, we receive a reward of 0, except after the k^{th} step, when we get a reward of 10. Therefore, the summation is

$$\sum_{i=0}^{k-1} 0 * \gamma^i + 10 * \gamma^{k-1} = 0 * \gamma^0 + 0 * \gamma^1 + 0 * \gamma^2 + 0 * \gamma^3 + ... + 10 * \gamma^{k-1} = 10\gamma^{k-1}.$$

Negative Reward

Now consider the case where this MDP has negative reward. In this scenario, the reward is $R(s, next) = -1$ for all states, except for state s_k where the reward is $R(s_k, next) = 0$. Again, there is only one action, $next$, and the decision policy remains $\pi_A(s) = next$ for all s. We always start at state s_1 and each arrow has a deterministic transition probability $p = 1$. There is no transition out of the end state END, and zero reward for any action from the end state, i.e., $R(END, next) = 0$.



(d) Calculate $V_\pi(s)$ for each state in the finite-horizon case with horizon $h = 1$, $k = 4$, and discount factor $\gamma = 1$.

Solution:

$$V_\pi^1(s_4) = 0$$
$$V_\pi^1(s_3) = -1$$
$$V_\pi^1(s_2) = -1$$
$$V_\pi^1(s_1) = -1$$

(e) Calculate $V_\pi(s)$ for each state in the infinite horizon case with $k = 4$ and discount factor $\gamma = 0.9$.

Solution:

$$V_\pi(s_4) = 0$$
$$V_\pi(s_3) = -1 + \gamma * 0 = -1$$
$$V_\pi(s_2) = -1 + 0.9(-1) = -1.9$$
$$V_\pi(s_1) = -1 + 0.9(-1.9) = -2.71$$

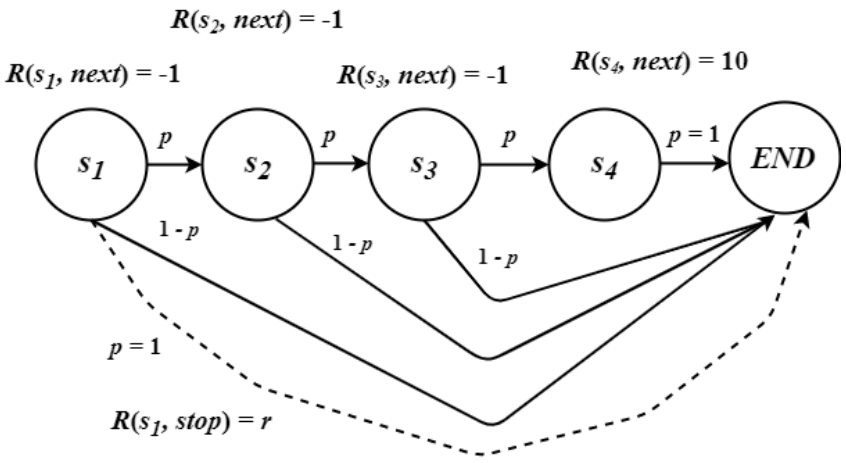
(f) Derive a formula for $V_\pi(s_1)$ that works for any value of (is expressed as a function of) k and γ for this negative reward MDP with infinite horizon. Recall that $\sum_{i=0}^{n} \gamma^i = \frac{(1-\gamma^{n+1})}{(1-\gamma)}$.

Solution: At every step, we receive a reward of -1, except for the k^{th} step, where we receive a reward of 0. Therefore, the summation is

$$\sum_{i=0}^{k-2} -1 * \gamma^i + 0 * \gamma^{k-1} = -1 * \gamma^0 - 1 * \gamma^1 - 1 * \gamma^2 + ... - 1 * \gamma^{k-2} + 0 * \gamma^{k-1} = -\frac{1 - \gamma^{k-1}}{1 - \gamma}.$$

Positive and Negative Reward

Consider the MDP below with negative rewards for some $R(s, a)$ and positive rewards for others. Now there are two actions, *next* and *stop*. The solid arrows show the probabilities of state transitions under action *next*; the dashed arrows show the probability of state transitions under action *stop*. (If there is no dashed arrow from a state, that indicates a probability $p = 0$ of transitioning out of that state under action *stop*.) The corresponding rewards $R(s_i, a)$ are also indicated on the figure below. Note that the rewards are $R(s_i, next) = -1$ for all s_i, except for state s_4, where the reward is $R(s_4, next) = 10$. Finally, under action *stop*, we have reward $R(s_1, stop) = r$ (some unknown value r), and $R(s, stop) = 0$ for all other states. As before, we always start in state s_1. There is no transition out of the end state END, and zero reward for any action from the end state, i.e., $R(END, next) = R(END, go) = 0$. Assume discount factor γ and infinite horizon.



(g) We consider two possible policies: $\pi_A(s) = next$ for all s, and $\pi_B(s) = stop$ for all s. Your goal is to maximize your reward. When you start at s_1, you have reward 0 before taking any actions. Determine what r should be, so that it is best to run this MDP under policy π_B rather than policy π_A. Give your answer as an expression for r involving p and γ.

Solution: Under policy π_A:

$$
\begin{aligned}
V_\pi(s_4) &= 10 \\
V_\pi(s_3) &= -1 + p\gamma V_\pi(s_4) + (1-p)\gamma V_\pi(end) = -1 + p\gamma \cdot 10 \\
V_\pi(s_2) &= -1 + p\gamma V_\pi(s_3) = -1 - p\gamma + (p\gamma)^2 \cdot 10 \\
V_\pi(s_1) &= -1 + p\gamma V_\pi(s_2) = -1 - p\gamma - (p\gamma)^2 + (p\gamma)^3 \cdot 10
\end{aligned}
$$

Under policy π_B, we simply have $V_\pi(s_1) = r$. So we should choose policy π_B when

$$r > -1 - p\gamma - (p\gamma)^2 + (p\gamma)^3 \cdot 10$$

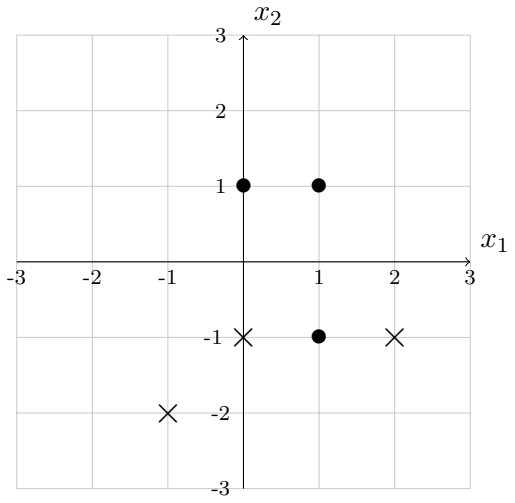
As an example, for $\gamma = 1$ and $p = 0.9$, r is 4.58.

Name: _____

Trees in a Forest

6. (12 points) Consider the following 2D dataset. Positively labeled points (+1) are solid points (•) and negatively labeled points (-1) are X marks (×).

x	y
$(0, 1)$	+1
$(1, 1)$	+1
$(-1, -2)$	-1
$(0, -1)$	-1
$(1, -1)$	+1
$(2, -1)$	-1



Consider the following splits:

Split A: $x_2 \geq 0$
Split B: $x_1 \geq 0.5$
Split C: $x_1 \geq -0.5$

Paul Bunyan works to construct trees using the algorithm discussed in the lecture notes, i.e., a greedy algorithm that recursively minimizes weighted average entropy, considering only combinations of the three splits mentioned above. He wants the output of the tree for any input (x_1, x_2) to be the probability that the input is a positive (+1) example.

Recall that the weighted average entropy $\bar{H}$ of a split into subsets R_1 and R_2 is:

$$\bar{H}(\text{split}) = (\text{fraction of points in } R_1) \cdot H(R_1) + (\text{fraction of points in } R_2) \cdot H(R_2)$$

where the entropy $H(R_m)$ of data in a region R_m is given by

$$H(R_m) = -\sum_k \hat{P}_{mk} \log_2 \hat{P}_{mk}.$$

Here $\hat{P}_{mk}$ is the *empirical probability*, which is in this case the fraction of items in region m that are of class k.

Some facts that might be useful to you:

$$0 \log_2(0) = 0 \qquad \log_2(2) = 1$$
$$\log_2(1) = 0 \qquad \log_2(1/2) = -1$$
$$\log_2(1/3) = -1.58 \qquad \log_2(1/4) = -2.00$$
$$\log_2(2/3) = -0.58 \qquad \log_2(3/4) = -0.42$$

Name: _____

(a) Considering the entire data set, Paul finds that the best first split of these three is Split A, with $\bar{H}(A) = 0.54$, compared to $\bar{H}(B) = 0.92$ and $\bar{H}(C) = 0.81$, resulting in a region R_{A^+} with all positive examples, and a region R_{A^-} with mixed positive and negative examples. Given Split A, however, Paul is not sure which is the next split to include in his tree. Calculate the weighted average entropy of Split B for region R_{A^-}, $\bar{H}(B|R_{A^-})$, versus Split C for the same region, $\bar{H}(C|R_{A^-})$, and identify which of Split B or Split C Paul should choose for his second split.

Solution:

$$\bar{H}(B|R_{A^-}) = \frac{2}{4} \cdot 0 + \frac{2}{4} \cdot 1 = 0.5$$

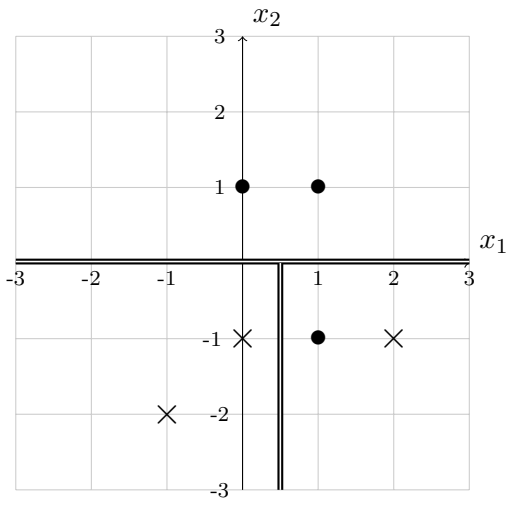
since the left split for B in region R_{A^-} has only -1 examples in it, so H of that side is 0, and the right split has equal number of +1 and -1 examples in it, so H of that side is 1.

$$\bar{H}(C|R_{A^-}) = \frac{1}{4} \cdot 0 + \frac{3}{4}\left[-\frac{2}{3}\log_2\left(\frac{2}{3}\right) - \frac{1}{3}\log_2\left(\frac{1}{3}\right)\right] = 0.69$$

So Paul should pick Split B as the second split.

(b) Draw the decision tree boundaries represented by this decision tree (with two splits) on the data plot figure below.

x	y
$(0, 1)$	+1
$(1, 1)$	+1
$(-1, -2)$	-1
$(0, -1)$	-1
$(1, -1)$	+1
$(2, -1)$	-1



(c) Draw the decision tree corresponding to this tree with two splits. Clearly label the test in each node, which case (yes or no) each branch corresponds to, and the output at a leaf node **represented as a probability of having a positive label, +1**.

Solution:

```
x_2 >= 0
  Yes branch: leaf +1 with probability 1.0
  No branch:
```

```
x_1 >= 0.5
  Yes branch: leaf +1 with probability 0.5
  No branch: leaf +1 with probability 0.0
```

(d) What probability of being a positive example does Paul's decision tree from part (a) above return for the new point (-1, 1)?

Solution: 1.0

(e) What probability of being a positive example does Paul's decision tree from part (a) above return for the new point (1, -2)?

Solution: 0.5

(f) Paul decides to consider a particular type of "random forest," which is an ensemble or collection of decision trees, where each tree might only have a subset of split features. Paul restricts his trees to only use Splits A, B, C, or some combination of these splits. The final output of the random forest is the average of the output across the collection of n trees (i.e., with equal weight $1/n$ for each tree in the random forest). Paul's random forest consists of three trees:

- The tree consisting of the best *single* split using feature x_2 only.
- The tree consisting of the best *single* split using feature x_1 only.
- The tree consisting of the best *two* splits (in total) using both features x_1 and x_2 (this is the tree from part (a) in this problem).

For this random forest, what is the output for the probability that an input point at $(-1, 1)$ is a positive $(+1)$ example? Note: Paul's calculations in part (a) may be of help.

Solution: The first tree corresponds to just Split A on x_2 from Paul's original tree; this tree gives $p = 1.0$ for the point being a positive example. As noted in part (a) the best tree splitting only on x_1 is Split C, since $\bar{H}(C) = 0.81$ is less than $\bar{H}(B) = 0.92$);this tree has $p = 0.0$ for the point $(-1, 1)$ being a positive example. Finally, the two-split tree as derived in part(a) had $p = 1.0$. Thus the aggregate (average) probability is that $(-1, 1)$ is a positive example is $p = 2/3$.

(g) Would you expect the accuracy for Paul's random forest generated decision to be better, or for the decision made by Paul's single two-split decision tree from part (a) to be better, when evaluated against held-out test data? Explain.

Solution: We would expect that the random forest generated decision will generalize better. Using all the features available to us can lead to over-fitting. For random forests, although each individual decision tree can have a higher error rate on the training data, the averaging effect (or majority vote for classification trees) can serve as a filter on noise vs. true signal.

Name: _____

We Recur!

7. (12 points) We have seen in class recurrent neural networks (RNNs) that are structured as:

$$z_t^1 = W^{ss} s_{t-1} + W^{sx} x_t$$
$$s_t = f_1(z_t^1)$$
$$z_t^2 = W^o s_t$$
$$p_t = f_2(z_t^2)$$

where we have set biases to zero. Here x_t is the input and y_t the actual output for (x_t, y_t) sequences used for training, with p_t as the RNN output (during or after training).

Assume our first RNN, call it RNN-A, has s_t, x_t, p_t all being vectors of shape 2×1. In addition, the activation functions are simply $f_1(z) = z$ and $f_2(z) = z$.

(a) For RNN-A, give dimensions of the weights:

W^{ss}: ___2×2___ W^{sx}: ___2×2___ W^0: ___2×2___

(b) We have finished training RNN-A, using some overall loss $J = \sum_t Loss(y_t, p_t)$ given the per-element loss function $Loss(y_t, p_t)$. We are now interested in the derivative of the overall loss with respect to x_t; for example, we might want to know how sensitive the loss is to a particular input (perhaps to identify an outlier input). What is the derivative of overall loss at time t with respect to x_t, $\partial J / \partial x_t$, with dimensions 2×1, in terms of the weights W^{ss}, W^{sx}, W^0 and the input x_t? Assume we have $\partial Loss / \partial z_t^2$, with dimensions 2×1. Use $*$ to indicate element-wise multiplication.

Solution:

$$\frac{\partial J}{\partial x_t} = \frac{\partial Loss}{\partial x_t} = \frac{\partial z_t^2}{\partial x_t} \frac{\partial Loss}{\partial z_t^2}$$
$$= \frac{\partial z_t^1}{\partial x_t} \frac{\partial z_t^2}{\partial z_t^1} \frac{\partial Loss}{\partial z_t^2}$$
$$= W^{sxT} W^{oT} \frac{\partial Loss}{\partial z_t^2}$$

Check: $(2 \times 1) = (2 \times 2)(2 \times 2)(2 \times 1)$ matrix dimensions.

Or more generally, for x_t being $(d \times 1)$, s_t and z_t^1 being $(m \times 1)$, and p_t and z_t^2 being $(n \times 1)$, then W^{sx} has dimensions $(m \times d)$ and W^o has dimensions $(n \times m)$. Then the above derivative dimension check is $(d \times 1) = (d \times m)(m \times n)(n \times 1)$ dimensions.

Now consider a modified RNN, call it RNN-B, that does the following:

$$z_t^1 = W^{ssx} \begin{bmatrix} s_{t-1} \\ x_t \end{bmatrix}$$
$$s_t = z_t^1$$
$$z_t^2 = W^{ox} \begin{bmatrix} s_t \\ x_t \end{bmatrix}$$
$$p_t = f_2(z_t^2)$$

where s_t, x_t, p_t are all vectors of shape 2×1, $\begin{bmatrix} s_{t-1} \\ x_t \end{bmatrix}$ and $\begin{bmatrix} s_t \\ x_t \end{bmatrix}$ are vectors of shape 4×1.

(c) For RNN-B, give dimensions of the weights:

W^{ssx}: 2×4 W^{ox}: 2×4

(d) Imagine we are using RNN-B to generate a description sentence given an input word, as in language modeling. The input is a single 2×1 vector embedding, x_1, that encodes the input word. The output will be a sequence of words $p_1, p_2, ..., p_n$ that provide a description of that word. In this setting, what would be an appropriate activation function f_2?

Solution: Arguably, linear is best if we interpret both x_1 and p_t as 2×1 vector embeddings of words. Alternatively, one could interpret p_t as the "best" next word, in which case softmax (or sigmoids) as the output would be reasonable.

(e) Continuing with RNN-B for one-to-many description generation using our language modeling approach, we calculate p_1 in a forward pass. How do we calculate x_2 (what is x_2 equal to)?

Solution: Treating both x_2 and p_1 as vector embeddings (with a linear activation function in the previous part), then the new input is just the previous output, $x_2 = p_1$. With a softmax output, technically one would need to include an extra step to take that previous output word and convert to the embedding expected of inputs x_i (though we accepted $x_2 = p_1$ here as well).

(f) For RNN-B, we are also interested in the derivative of loss at time t with respect to x_t, $\partial Loss/\partial x_t$. Indicate all of the following that are true about RNN-B, and the derivative of loss with respect to x_t :

- √ $\partial Loss/\partial x_t$ **depends on** W^{ox}
- √ $\partial Loss/\partial x_t$ **depends on all elements of** W^{ox}
- √ $\partial Loss/\partial x_t$ **depends on** W^{ssx}
- ◯ $\partial Loss/\partial x_t$ depends on all elements of W^{ssx}

Name: _____

Solution: The stacking of s_t and s_{t-1} with x means we need to carry through the differentiation carefully.

$$\frac{\partial Loss}{\partial x_t} = \frac{\partial Loss}{\partial z_t^2}\frac{dz_t^2}{dx_t}$$

Since $z_t^2 = W^{ox}\begin{bmatrix} s_t \\ x_t \end{bmatrix}$ we might think that only the third and fourth columns of W^{ox} multiply by the two elements of x_t and so only a subset of the elements of W^{ox} come into play when we take dz_t^2/dx_t. However, s_t *also* depends on x_t through W^{ssx}, and so all elements of W^{ox} matter when we take dz_t^2/dx_t. However, it actually is the case that only the third and fourth columns of W^{ssx} multiply x_t, so when we carry through the derivative there, only some elements of W^{ssx} matter in this overall gradient.

Name: _____

With Some Regularity

8. (8 points) We previously examined ridge regression, where a regularizer term $R_\lambda(\theta)$ is added to a sum of squares loss to form the J_1 objective function as below. Throughout this problem, we will assume zero offset $\theta_0 = 0$ and linear models of output y as a function of input x.

$$J_1(\theta) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \theta \cdot x_i)^2 + R_\lambda(\theta)$$
$$= \frac{1}{2}\sum_{i=1}^{n}(y_i - \theta \cdot x_i)^2 + \frac{\lambda}{2}||\theta||^2$$

Laz Zo prefers an alternative approach (called "lasso" regularization), where a different regularizer $R_\alpha(\theta)$ is added to the sum of squares loss:

$$J_2(\theta) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \theta \cdot x_i)^2 + R_\alpha(\theta)$$
$$= \frac{1}{2}\sum_{i=1}^{n}(y_i - \theta \cdot x_i)^2 + \alpha\sum_{j=1}^{k}|\theta_j|$$

Consider the two-dimensional case, $k = 2$, so that our vector θ has just two components, θ_1 and θ_2. **Suppose also that $\theta_1 > \theta_2$ and both are positive ($\theta_1, \theta_2 > 0$).** We are interested in the behavior of $R_\alpha(\theta)$ and $R_\lambda(\theta)$. Assume both λ and α are positive.

(a) First consider the lasso regularizer for this specific case:

$$R_\alpha(\theta) = \alpha\sum_{j=1}^{k}|\theta_j| = \alpha(\theta_1 + \theta_2)$$

where $R_\alpha(\theta) = \alpha(\theta_1 + \theta_2)$ in this case since both θ_1 and θ_2 are positive. We consider reducing θ_1 by a small δ where $\delta > 0$, versus reducing θ_2 by δ. (By "reducing" we mean subtracting δ from θ_1. You can assume δ is smaller than θ_1 and θ_2.) What is true, if our goal is to minimize $R_\alpha(\theta)$?

◯ It is better to reduce θ_1 by δ
◯ It is better to reduce θ_2 by δ
√ **It is equally beneficial to reduce θ_1 or θ_2 by δ.**

(b) Now we are interested in the behavior of $R_\lambda(\theta)$ for this specific case:

$$R_\lambda(\theta) = \frac{\lambda}{2}||\theta||^2 = \frac{\lambda}{2}(\theta_1^2 + \theta_2^2).$$

We consider reducing θ_1 by a small δ, where $\delta > 0$, versus reducing θ_2 by δ. (You can assume δ is smaller than θ_1 and θ_2.) What is true, if our goal is to minimize $R_\lambda(\theta)$?

√ **It is better to reduce θ_1 by δ**
◯ It is better to reduce θ_2 by δ
◯ It is equally beneficial to reduce θ_1 or θ_2 by δ

Rega Lizer is interested in the behavior of these two regularizers, when used to fit a linear model by minimizing J_1 and J_2. We compare the ridge regularizer R_λ and the lasso regularizer R_α, for general k. Assume α and λ are positive.

(c) Check all of the following that are true about R_λ when minimizing J_1 (with sum of squares loss and $R_\lambda(\theta)$ terms):

- √ R_λ **pushes θ to have smaller magnitude θ_i**
- √ R_λ **favors reducing the magnitude of the largest magnitude θ_i over reducing the magnitude of smaller magnitude θ_i**
- √ R_λ **inhibits sparsity (i.e., disfavors finding θ such that some θ_i are zero) for θ with equivalent sum of squares loss**

(d) Check all of the following that are true about R_α when minimizing J_2 (with sum of squares loss and $R_\alpha(\theta)$ terms):

- √ R_α **pushes θ to have smaller magnitude θ_i**
- ◯ R_α favors reducing the magnitude of the largest magnitude θ_i over reducing the magnitude of smaller magnitude θ_i
- ◯ R_α inhibits sparsity (i.e., disfavors finding θ such that some θ_i are zero) for θ with equivalent sum of squares loss

(e) Rega proposes combining the two regularizers with a sum of squares loss to form the J_3 objective:

$$J_3(\theta) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \theta \cdot x_i)^2 + R_\alpha(\theta) + R_\lambda(\theta)$$
$$= \frac{1}{2}\sum_{i=1}^{n}(y_i - \theta \cdot x_i)^2 + \alpha\sum_{j=1}^{k}|\theta_j| + \frac{\lambda}{2}||\theta||^2$$

Check all of the following that are true about using both of these regularizers when minimizing J_3:

- ◯ This is a bad idea, as the two regularizers will compete against each other.
- √ **This is a reasonable idea, to achieve some controllable mixture of the behavior of the two regularizers based on the two hyperparameters, α and λ.**
- ◯ This is a bad idea, as the two regularizers are redundant, and only add complexity in training because now there are two hyperparameters, α and λ, that need to be decided.

Solution: Rega's approach is often used in practice, and is referred to as "elastic net" regression. It seeks to achieve sparsity (setting some ~~θ_i~~ θ_j to zero) while also keeping θ coefficients small in general.

(original solution had typo)
i ranges over samples
j ranges over features
should be some $\theta_j = 0$