6.036: Final Exam: Spring 2022

Solutions

- This is a closed book exam. Calculators 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 the back of the page. Show your work neatly.
- If a question seems vague or under-specified to you, make an assumption, write it down, and solve the problem given your assumption. **Try not to ask for clarification.**
- **Write your initials on every piece of paper.**

Name: _____ MIT email: _____

Question	Points	Score
1	6	
2	8	
3	12	
4	18	
5	18	
6	10	
7	8	
8	10	
9	10	
Total:	100	

Name: _____

1 A single sigmoid

1. (6 points) Consider the simplest of all neural networks, consisting of a single unit with a sigmoid activation function:

$$h(x; w) = \sigma(w_0 + w_1 x)$$

where $\sigma(z) = (1 + \exp(-z))^{-1}$ is the sigmoid function.

(a) Let's start with a classifier defined by $w_0 = 0$ and $w_1 = 1$. Which range of input values x are classified as positive? Which as negative?

Solution: Positive if $x > 0$; negative otherwise.

(b) If you change w_1 to have value 2, which range of input values x are classified as positive? If the answer is the same as in part 1, what, if anything, about the output of the network has changed?

Solution: Positive if $x > 0$; negative otherwise. The difference is that output curve will be "steeper" near 0; output values of this model will be closer to -1 and +1 than those of the previous model.

(c) If you leave $w_1 = 1$ and set $w_0 = -1$, which range of input values x are classified as positive? If the answer is the same as in part 1, what, if anything, about the output of the classification model has changed?

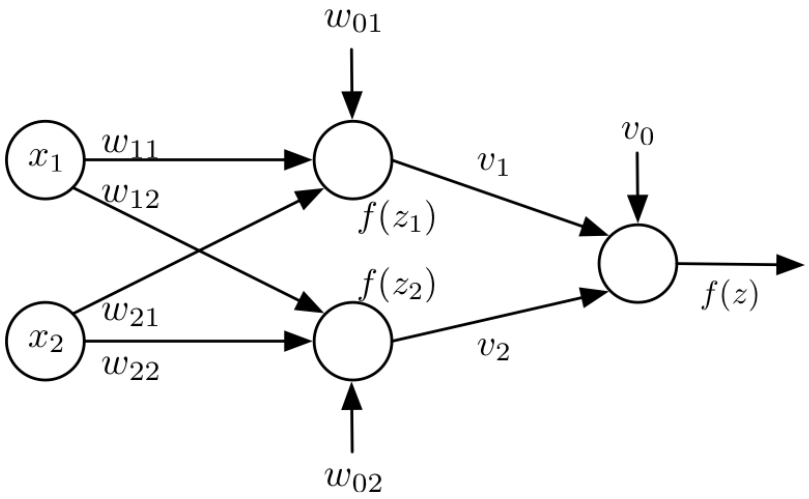
Solution: Positive if $x > 1$; negative otherwise.

Name: _____

2 A DARC and stormy night

2. (8 points) A recent paper gives us reason to think that, rather than regularizing the weights used in a deep network that is used for classification, it is good to regularize the z values, before the activation function is applied, in the last layer.

   Consider a neural network with one hidden layer and a single output unit where the activation function f is a sigmoid, as shown below.



   We can specify the DARC objective function $J(\theta, \lambda)$, where the parameters $\theta = (w_{11}, w_{12}, w_{21}, w_{22}, v_1, v_2)$, which depends on data set $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$ as

$$J(\theta, \lambda) = \sum_i \mathcal{L}_{nll}(f(z^{(i)}), y^{(i)}) + \lambda \sum_i (z^{(i)})^2$$

   where $z^{(i)}$ is the value of the output unit on example i *before* it goes into the activation function.

   (a) What is the partial derivative of this unusual regularization term with respect to the weight w_{11}, for a single (x, y) training point?

$$\frac{\partial}{\partial w_{11}} \lambda (z)^2$$

   Write it in terms of x, y, z_1, z_2, z, w and v values. You can use f' for derivative of f.

   Solution:

$$2\lambda z v_1 x_1 f'(z_1)$$

   (b) What is the derivative with respect to w_{11} of the typical regularization term, which penalizes the squares of the weights? How do these two regularizers differ?

   Solution: $2\lambda w_{11}$. One depends on the input.

   (c) Describe a situation in which it is possible for w_{11} to be extremely large, but for z to have small magnitude.

Solution: Maybe v_1 is very small.

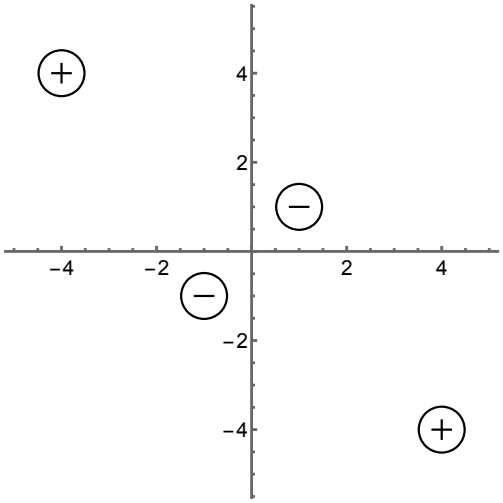
(d) Would the DARC strategy of regularizing z be good if we were, instead, doing regression and $f(x) = x$? Explain why or why not.

Solution: No, because we need the output to be able to attain its target value, which will be made impossible by penalizing the magnitude of the output.

Name: _____

3 Kopy Kat

3. (12 points) Consider the following data. Clearly, the points are not linearly separable, so we will try three alternative approaches to solving the problem.



Approach 1: Nested linear classifiers

Let $w = \begin{bmatrix} +1 \\ -1 \end{bmatrix}$ and

$$a_1 = \text{sign}(w^T x + 4)$$
$$a_2 = \text{sign}(w^T x - 4)$$
$$h(x; v_0, v_1, v_2) = \text{sign}(v_1 a_1 + v_2 a_2 + v_0)$$

where

$$\text{sign}(x) = \begin{cases} +1 & \text{if } x > 0 \\ -1 & \text{otherwise} \end{cases}$$

(a) Draw the classifiers corresponding to a_1 and a_2 on the axes above. Label them clearly, including their normal vectors.

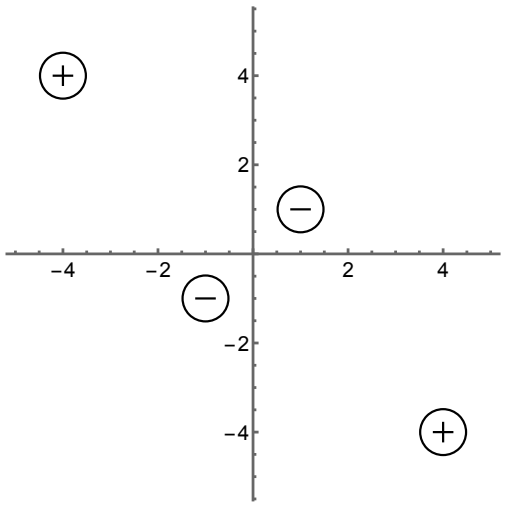
(b) Select values of the following parameters so that the nested classifier correctly predicts the values in the data set.

i. v_1 **-1**

ii. v_2 **1**

iii. v_0 **.5**

Name: _____



Approach 2: Data-driven features

We'll define a new feature transformation ϕ that maps a point $x \in \mathbb{R}^2$ into a four-dimensional vector:

$$(K(x,(-4,4)), K(x,(-1,-1)), K(x,(1,1)), K(x,(4,-4)))$$

where K is a function of two points: $K(x,x') = e^{-\|x-x'\|^2}$. Intuitively, feature i of $\phi(x)$ has value 1 if x is equal to point p_i and its value decreases as x moves away from p_i.

(c) We find a classifier in the transformed space with parameters $\theta = (1,-1,-1,1)$

$$h(x;\theta) = \text{sign}(\theta^T \phi(x))$$

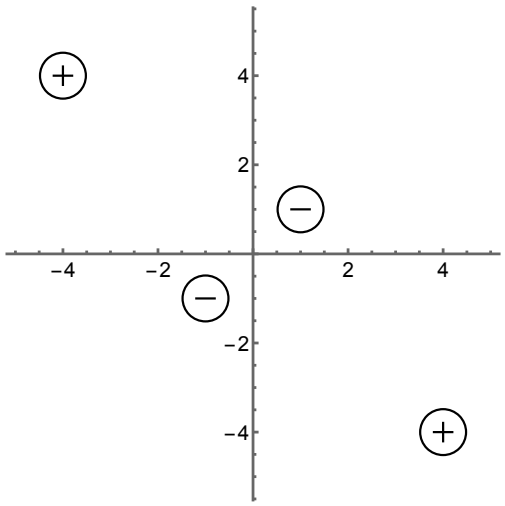
i. What fraction of the training data does this classifier predict correctly?

Solution: 100%

ii. What prediction does it make for point (0, 0)?

Solution: -1

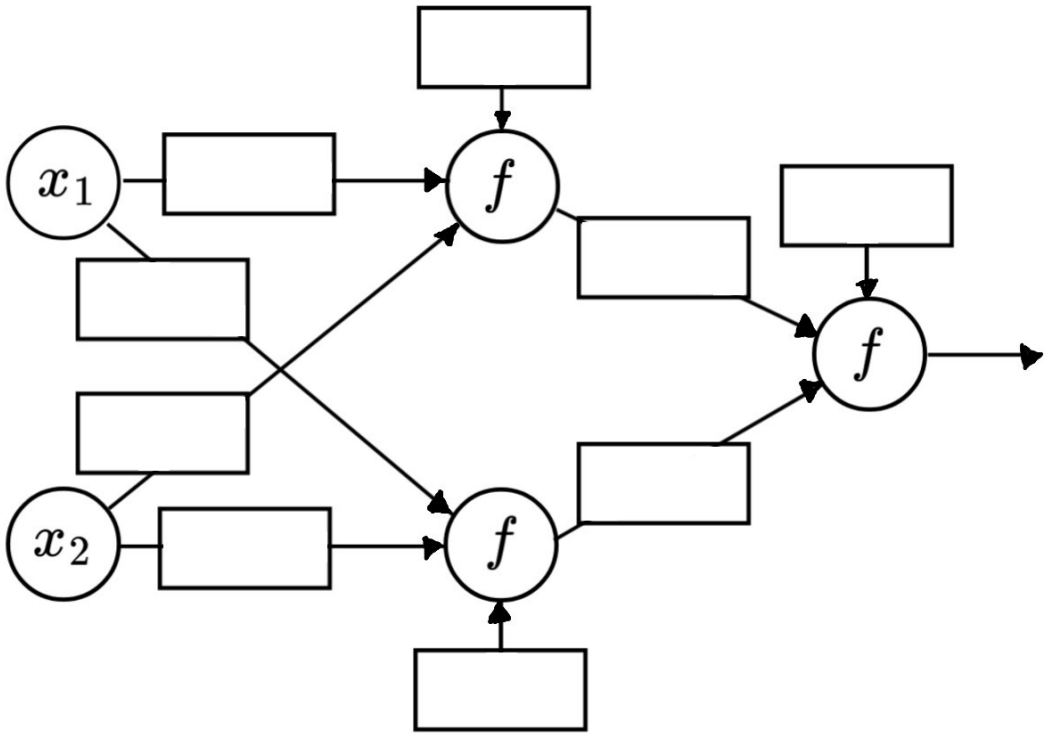
Name: _____



Approach 3: Neural nets

Suppose we use a neural network with one hidden layer to classify the points.

(d) We can classify the points correctly if f (in both layers) is sigmoid. Provide the weights so this network will correctly classify the given points.



- w_{11} **-10**

Name: _____

- w_{21} **10**
- w_{01} **-40**
- w_{12} **10**
- w_{22} **-10**
- w_{02} **-40**
- v_1 **1**
- v_2 **1**
- v_0 **-.5**

4 Regression residuals

4. (18 points) We're given a data set $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$, where $x^{(i)} \in R^d$ and $y^{(i)} \in R$. Let X be a $d \times n$ matrix in which the $x^{(i)}$ are the columns and let Y be a $1 \times n$ vector containing the values of $y^{(i)}$. Using the analytical regression (ordinary least-squares) formula, we can compute

$$W_{ols} = (XX^T)^{-1}XY^T$$

Using ridge regression, we can compute

$$W_{ridge} = (XX^T + \lambda I)^{-1}XY^T$$

We decide to try to use these methods to initialize a single-unit "neural network" with a linear activation function and no offset:

$$h(x; W) = W^T x \ .$$

Assume that XX^T is invertible and not equal to the identity matrix, and that neither W_{ols} nor W_{ridge} is equal to $(0, 0, \ldots, 0)$. Note also that we are not using an explicit offset/bias term.

(a) If we initialized our unit with W_{ols} and did batch gradient descent (summing the error over all the data points) with squared loss, no regularization, and a fixed small step size, which of the following would most typically happen:

- ◯ The weights would change substantially at the beginning, but then converge back to the values we initialized it with.
- √ **The weights would not change.**
- ◯ The weights would make small oscillations around the initial weights.
- ◯ The weights would converge to a different value.
- ◯ Something else would happen.

(b) Explain why.

> **Solution:** These weights are an optimum of the objective and the gradient will be (nearly) zero.

(c) If we initialized our unit with W_{ols} and did stochastic gradient descent (one data point at a time) with squared loss, no regularization, and a fixed small step size, many different things could happen. For each of the results below, explain briefly the circumstances in which it might occur, or explain why it is unlikely.

i. The weights would not change.

> **Solution:** If the OLS solution had 0 error on all training examples, then SGD will not result in any changes.

ii. The weights would make small oscillations around the initial weights.

> **Solution:** If there was error, and the gradients are not too big, then in expectation the steps should be small motions around the optimum.

iii. The weights would converge to a different value.

> **Solution:** It's possible that it will bounce out of the current optimum and end up in another one.

(d) Consider a neural-network unit initialized with W_{ridge}. Provide an objective function $J(W)$ that depends on the data, such that batch gradient descent to minimize J will have no effect on the weights, or argue that one does not exist.

> **Solution:** $J(W) = (W^T X - Y)^2 + \lambda \|W\|^2$

(e) Rory has solved many problems from this particular domain before and the solution has typically been close to $W^* = (1, ..., 1)^T$. Define an objective function $J(W)$ that we could minimize in order to obtain good estimates for Rory's next problem, even with very little data.

> **Solution:** $J(W) = (W^T X - Y)^2 + \lambda \|W - \mathbf{1}\|^2$

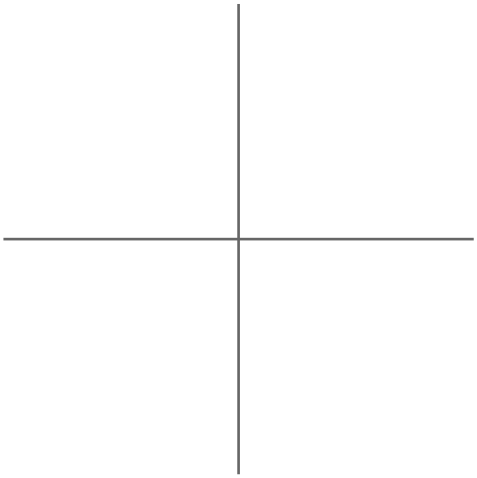
Name: _____

(f) Ryo thinks they can get a better hypothesis by using knowledge about neural networks, and considers the hypothesis class

$$g = w_a \sigma(w_b x + w_c) + w_d$$

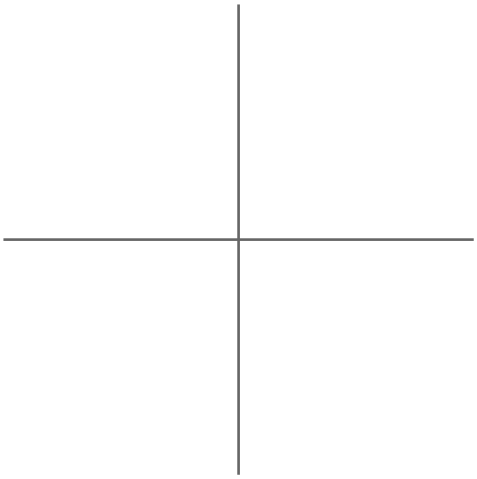
Assume that the inputs x are 1-dimensional and recall $\sigma(z) = 1/(1 + e^{-z})$.

Provide a data set with 3 points for which Ryo's hypothesis class can reach a lower MSE than the original OLS solution or argue that one does not exist.



Solution: $(0, 0), (1, 1), (2, 1)$

(g) Provide a data set with 3 points for which the original OLS hypothesis class can reach a **substantially** lower MSE than Ryo's hypothesis class or argue that one does not exist.



Solution: Does not exist: You can stretch out the sigmoid so that the linear part of it is pretty linear and goes wherever you want it to.

(h) Ryu thinks getting initial parameters from Ryo's hypothesis might be a good way to initialize a two-layer neural network. Consider the case where we have a simple neural network with

- Two units in the hidden layer
- Sigmoid activation function in the hidden layer
- Linear activation function on the output unit

So, the hypothesis is:

$$g = w_a\sigma(w_b x + w_c) + w_e\sigma(w_f x + w_g) + w_d$$

If Ryu first trained Ryo's hypothesis to reach a local optimum using gradient descent to obtain w_a, w_b, w_c, w_d, then initialized the more complex network above using $w_e = w_a$, $w_f = w_b$, and $w_g = w_c$, with w_d as before, and did batch gradient descent with squared loss and a fixed small step size, which of the following would most typically happen:

◯ The weights would change substantially at the beginning, but then converge back to the original values.

◯ The weights would not change.

◯ The weights would make small oscillations around the initial weights.

◯ The weights would converge to a different value, with a lower loss than we could achieve with Ryo's hypothesis class.

√ **The weights would converge to a different value, with the same loss as we would obtain with Ryo's hypothesis class.**

(i) Explain why.

Solution: Because the two units are initialized exactly the same, the gradients for both of them will be the same. So, it is as if we had a single linear unit, ran it through a sigmoid, and then added an offset.

(j) Does your answer to the previous question change if we set w_e, w_f, and w_g randomly instead? Why or why not?

Solution: Yes! Now the second unit has the freedom to change its weights and we might expect a solution with lower training loss.

Name: _____

5 A Concerted Effort

5. (18 points) You are hired by an event-planning company to build some machine-learning predictors. One of them, for predicting the number of attendees of a concert, takes as input x, a vector of aspects of the event (day of the week, start time, city size, music genre (classical, folk, rock, pop, rap), number of attendees at last concert, if known).

   (a) What would be a good encoding strategy for each attribute of concert? Select a choice, and explain your choice very briefly. If you think more than one are reasonable it's fine to pick them and explain. By *discretized numeric* we mean dividing a numeric range into bins and using one-hot encoding on the bins.

   - music genre
     ◯ numeric (standardized) √ **one-hot** ◯ discretized numeric ◯ other
     Explain briefly.

     Solution: The different genres are not ordered in a useful way so it's just a discrete choice.

   - number of attendees at last concert, if known
     √ **numeric (standardized)** ◯ one-hot ◯ discretized numeric ◯ other
     Explain briefly.

     Solution: This is sensible to interpret as a continuous value.

   - start time
     ◯ numeric (standardized) ◯ one-hot √ **discretized numeric** ◯ other
     Explain briefly.

     Solution: Numeric or discretized numeric could be okay. Because time wraps around awkwardly, discretizing into bins might be good.

(b) If you didn't know anything more about this problem, what would be a reasonable loss function to use?

> **Solution:** Squared loss

It turns out that the losses in this problem might be asymmetric, in that it might be worse to over-estimate the crowd size than to under-estimate the crowd size, or vice versa. Cody decides to use a loss function of the following form

$$\mathcal{L}_{\text{cody}}(g, a) = \begin{cases} \lambda(g-a)^2 & \text{if } g > a \\ (g-a)^2 & \text{otherwise} \end{cases}$$

where g is the guessed value and a is the actual value, and λ is an adjustable parameter.

(c) Jody thinks this loss function is not sufficiently general and suggests that we should use a version with two parameters:

$$\mathcal{L}_{\text{jody}}(g, a) = \begin{cases} \lambda_1(g-a)^2 & \text{if } g > a \\ \lambda_2(g-a)^2 & \text{otherwise} \end{cases}$$

Assuming that λ, λ_1 and λ_2 are all constrained to be strictly greater than 0, is Jody's loss function actually able to capture a larger class of losses? **Either**

1. Show that for any λ_1 and λ_2 there is a value of λ such that choosing g to minimize $\mathcal{L}_{\text{jody}}$ with λ_1, λ_2 yields the same result as choosing g to minimize $\mathcal{L}_{\text{cody}}$ with λ.
2. Give a counterexample: provide values of λ_1, λ_2 such that there is no value of λ such that choosing g to minimize $\mathcal{L}_{\text{jody}}$ with λ_1, λ_2 is equivalent to choosing g to minimize $\mathcal{L}_{\text{cody}}$ with λ.

> **Solution:** Setting $\lambda = \lambda_1/\lambda_2$ lets Cody do anything Jody could do. If we multiply by λ_2 this becomes Jody's objective, and multiplying by a constant doesn't change the g value that optimizes it.

We talk to Bodie, who really knows the concert business and says a better model for the total loss is:

$$\mathcal{L}_{\text{bodie}}(g, a) = \alpha g + \beta \max(0, a - g) \quad .$$

This loss has two terms. The first part of the loss comes from the cost of renting a venue: if we guess that there will be g attendees, we have to rent a place with g seats, and we assume that such a rental costs α per seat. The second part of the loss comes from the loss of potential ticket sales: if a people really wanted to attend but we can only seat g, then we lose β for each of the $g - a$ people we have to turn away. Note that it is safe to assume that $\alpha < \beta$ (otherwise, we should not bother holding the concert!).

(d) We would like to train a linear regression model to minimize this loss. Let $g = \theta^T x$ be the prediction given input x and parameters θ, and let y be the target training value for that x. Provide an expression for $\partial \mathcal{L}_{\text{bodie}}(g, y)/\partial \theta$.

Note that this loss is not everywhere differentiable, which we have seen before with ReLU units. Don't worry about the what the value should be at that one point.

Name: _____

Please write your answer in terms of x, y, α, β, and θ.

Solution:

$$\frac{\partial \mathcal{L}_{\text{bodie}}(g, y)}{\partial g} \frac{\partial g}{\partial \theta}$$

$$\left(\alpha + \beta \begin{cases} 0 & \text{if } y < \theta^T x \\ -1 & \text{otherwise} \end{cases} \right) x$$

But Bodie isn't really sure how to set the α and β parameters, so we still have a problem! However, they are able to find a set of data of the form (s, p, l) where s describes the number of seats in the venue rented, p describes the actual number of people who attempted to attend (including the number of people who were turned away) and l describes the actual loss value.

(e) Bodie wants to use this data to estimate α and β in $\mathcal{L}_{\text{bodie}}$ by finding values of these parameters that predict the loss the most accurately in the mean-squared-error sense. Describe how to use the (s, p, l) data to formulate a **linear** regression problem that will recover good estimates of α and β.

i. What are the inputs, x?

Solution: Vectors of $(s, \max(0, p - s))$

ii. What are the target outputs, y?

Solution: Values l

Name: _____

Roadie decides Bodie's strategy is too complicated.

Roadie has a very complicated economic simulator that can be used to evaluate the quality of a prediction, but it is too complicated to differentiate and use directly as a loss in a gradient optimization approach.

Instead, Roadie wants to adopt Cody's original simple loss function

$$\mathcal{L}_{\text{cody}}(g, a) = \begin{cases} \lambda(g-a)^2 & \text{if } g > a \\ (g-a)^2 & \text{otherwise} \end{cases}$$

which captures asymmetric loss and has free parameter λ.

(f) Given a true loss function $\mathcal{L}_{\text{true}}$, which is not differentiable, how could you use it to find a good value of λ so that you can use $\mathcal{L}_{\text{cody}}$ with that λ to construct a good predictive hypothesis? Assume you have a dataset $\mathcal{D}$ and that you are given a set `lambdas` of plausible values for λ. Let's write out a strategy in very abstract pseudo-code, using the following basic procedures:

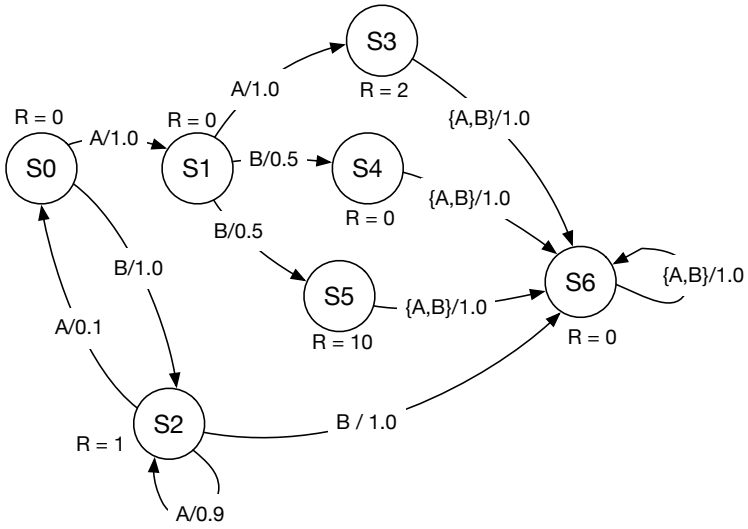
- `train(data, lossfn)` : trains a regression model to minimize `lossfn` on `data`, returns parameters `theta`
- `subpart(data, j, K)` : divides data into `K` equal parts and returns the `j`th subpart
- `allbutsubpart(data, j, K)` : divides data into `K` equal parts and returns all except the `j`th subpart
- `eval(theta, data, lossfn)` : returns average loss of hypothesis with weights `theta` on `data` according to `lossfn`
- `L_true` : the true loss function that maps a guess and an actual value into a cost
- `L_cody(lambda)` : returns $\mathcal{L}_{\text{cody}}$ for this value of `lambda`, which is itself a loss function that maps a guess and an actual value into a cost

Fill in the blanks in the code below, for a process in which we perform 10-fold cross-validation to find the best lambda value.

```
best_lambda = None; best_loss = None
for lambda in lambdas do

   for k in range(_____10_____) do

      hypoth = train(allbutsubpart(data, k, 10), _____L_cody(lambda)_____)

      loss = eval(_____hypoth_____, __subpart(data, k, 10)__,

                                          _____L_true_____)
      if best_lambda is None or loss < best_loss then
         best_lambda = lambda
         best_loss = loss
return best_lambda
```

Name: _____

6 ABBA!



6. (10 points)

Consider the MDP shown above. It has states $S_0, \ldots, S_6$ and actions A, B. Each arrow is labeled with one or more actions, and a probability value: this means that if any of those actions is chosen from the state at the start of the arrow, then it will make a transition to the state at the end of the arrow with the associated probability.

Rewards are associated with states, and independent, in this example, from the action that is taken in that state. Remember that with horizon $H = 1$, the agent can collect the reward associated with the state it is in, and then terminates.

(a) Consider the policy π that takes action B in S_0 and action A in S_2. If the system starts in S_0 or S_2, then under that policy, only those two states (S_0 and S_2) are reachable.

Recall that, for a fixed policy π,

$$V_\pi(s) = R(s, \pi(s)) + \gamma \sum_{s'} P(S_{t+1} = s' \mid S_t = s, A_t = \pi(s)) V_\pi(s') \ .$$

Assuming the discount factor $\gamma = 0.8$, what are the infinite-horizon values $V_\pi(S_0)$ and $V_\pi(S_2)$? It is sufficient to write out a small system of linear equations involving just those two variables; you do not have to take the time to solve them numerically.

Solution:

$$\begin{aligned} V_\pi(S_0) &= 0 + 0.8 \cdot V_\pi(S_2) \\ V_\pi(S_2) &= 1 + 0.8 \cdot (0.9 V_\pi(S_2) + 0.1 V_\pi(S_0)) \end{aligned}$$

(b) What is the **optimal** value $V_{h=1}(s) = \max_a Q_{h=1}(s, a)$ for each state for horizon $H = 1$ with no discounting?

i. $V_{h=1}(S_0)$ **0**

ii. $V_{h=1}(S_1)$ **0**

iii. $V_{h=1}(S_2)$ **1**

(c) What is the optimal action and value $V_{h=2}(s)$ for each state for horizon $H = 2$ with no discounting? (If the actions are tied in value, list both).

i. S_0 : A: **B** $V_{h=2}$: **1**

ii. S_1 : A: **B** $V_{h=2}$: **5**

iii. S_2 : A: **A** $V_{h=2}$: **1 + .9**

(d) What is the optimal action and value $V(s)$ for each state for horizon $H = 3$ with no discounting? (If the actions are tied in value, list both).

i. S_0 : A: **A** $V_{h=3}$: **5**

ii. S_1 : A: **B** $V_{h=3}$: **5**

iii. S_2 : A: **A** $V_{h=3}$: $1 + .9 \cdot 1.9 + .1 \cdot 1$

(e) If we increase the horizon beyond 3, will the optimal action in state S_0 ever change? Explain.

Solution: Yes. With a longer horizon, it's worth taking action B in S_0 and going around and around that loop.

Name: _____

7 ABCs of RL

7. (8 points) Consider an MDP with four states, called A, B, C, and D, and with two actions called **Move** and **Stay**, but you don't know anything about what these actions really do. The discount factor $\gamma = 0.9$.

   Here is a reminder of the Q-learning update formula, based on experience tuple (s_t, a_t, r_t, s_{t+1}):

   $$Q(s_t, a_t) := (1 - \alpha)Q(s_t, a_t) + \alpha \left(r_t + \gamma \max_{a'} Q(s_{t+1}, a') \right) \quad .$$

   In the following, let $\alpha = 1$.

   Assume we see the following state-action-reward sequence:
   A, Move, 0
   B, Move, 0
   C, Move, 1
   A, Move, 0
   B, Move, 0
   C

   With Q-values all starting at 0, we run the Q-learning algorithm on that state-action sequence.

   (a) Specify *each of the five updates to the Q table* by saying which entry or entries are updated, and with which values. Please include an update even if the value is 0.

   i. $Q($**A**, **Move**$) =$ **0**

   ii. $Q($**B**, **Move**$) =$ **0**

   iii. $Q($**C**, **Move**$) =$ **1**

   iv. $Q($**A**, **Move**$) =$ **0**

   v. $Q($**B**, **Move**$) =$ **0.9**

(b) This example illustrates a key weakness of Q-learning that would be even worse if the problem had 103 states, A, B_1, ..., B_{100}, C, D, and the state-action-reward sequence were

$$(A, \text{Move}, 0), (B_1, \text{Move}, 0), (B_2, \text{Move}, 0), \ldots, (B_{100}, \text{Move}, 0), (C, \text{Move}, 1), (A, \text{Move}, 0) \ .$$

Very briefly describe the weakness and a strategy for overcoming this weakness.

Solution: It doesn't propagate the value all the way back the chain. Do the updates backward along the trajectory; or save your experience and replay it.

(c) Suppose, for the original 4-state (A, B, C, D), that we continue acting (using the Q function we arrived at, at the end of the part (a), and with $\alpha = 1$) and receive the following additional state-action-reward sequence:
A, Move, 0
B, Move, 0
D

What additional updates would occur?

i. $Q($**A**, **Move**$) =$ **.81**

ii. $Q($**B**, **Move**$) =$ **0**

(d) What problem with our choice of parameters for the learning method is revealed by this example? Very briefly explain a small change that will solve this problem.

Solution: Use a smaller learning rate.

Name: _____

8 The deep end of the pool

8. (10 points) We are going to consider two different simple convolutional networks over one dimensional (vector) inputs. Each network has a single convolutional layer with a single filter of size 3 and stride 1. Let $(z_1, \ldots, z_d)$ be the output of this convolutional layer, so that $z_j = w^T[x_{j-1}; x_j; x_{j+1}]$ where w are the filter parameters. Our two networks differ in terms of how the feature map values are pooled to a single output value. Assume zero-padding of size 1 at each end of the image.

Network A has a single *max-pooling* layer with input size d, so that the output of the network $g = \sigma(\max(z_1, \ldots, z_d))$ where $\sigma(\cdot)$ is the sigmoid function.

Network B has a single *min-pooling* layer with input size d, so that the output of the network $g = \sigma(\min(z_1, \ldots, z_d))$.

When the filter's output value is high it represents a positive detection of some pattern of interest.

(a) For which network does a high output value correspond, qualitatively, to "every location in x corresponds to an instance of the desired pattern"?

◯ A ✓ **B** ◯ None

(b) For which network does a high output value correspond, qualitatively, to "at least half of the locations in x correspond to an instance of the desired pattern"?

◯ A ◯ B ✓ **None**

(c) For which network does a high output value correspond, qualitatively, to "there is at least one instance of the desired pattern in this image"?

✓ **A** ◯ B ◯ None

(d) Assume for simplicity that all $z_1, \ldots, z_d$ have distinct values (we can ignore the corner cases where some of the values are equal). What is $\partial g / \partial z_i$ for network A? Feel free to make use of the fact that $\partial \sigma(z)/\partial z = \sigma(z)(1 - \sigma(z))$.

Solution: $\sigma(z_i)(1 - \sigma(z_i))$ if $z_i = \max(z_1, \ldots, z_d)$, and 0 otherwise.

(e) Now, suppose we are just given **a single training pair** (x, y) where the target y is binary 0/1. The loss that we are minimizing is again just

$$\mathcal{L}_{\text{nll}}(g, y) = -y \log g - (1 - y) \log(1 - g)$$

which is minimized when g matches the target y. We are interested in understanding qualitatively how the filter parameters w get updated in the two networks if we use simple gradient descent to minimize $\mathcal{L}_{\text{nll}}(y, g)$. For each of the four qualitative behaviors of this process described below, specify the situation in which it would occur:

- Which network (A, B, or it doesn't matter)
- Target y (1, 0, or it doesn't matter)

The behaviors are:

1. After each step of gradient descent, the filter weights w change so that their dot product with the values of one particular sub-region $[x_{j-1}; x_j; x_{j+1}]$ of the image *increases*.
   i. Network: √ **A** ◯ B ◯ irrelevant
   ii. Target y: √ **1** ◯ 0 ◯ irrelevant

2. After each step of gradient descent, the filter weights w change so that their dot product with the values of one particular sub-region $[x_{j-1}; x_j; x_{j+1}]$ of the image *decreases*.
   i. Network: ◯ A √ **B** ◯ irrelevant
   ii. Target y: ◯ 1 √ **0** ◯ irrelevant

3. After each step of gradient descent, the filter weights w change so that their dot product with the values of some image sub-region *increases*, but the specific region may change from one step to another.
   i. Network: ◯ A √ **B** ◯ irrelevant
   ii. Target y: √ **1** ◯ 0 ◯ irrelevant

4. After each step of gradient descent, the filter weights w change so that their dot product with the values of some image sub-region *decreases*, but the specific region may change from one step to another.
   i. Network: √ **A** ◯ B ◯ irrelevant
   ii. Target y: ◯ 1 √ **0** ◯ irrelevant

Name: _____

9 RNN

9. (10 points) Consider three RNN variants:

   1. The **basic** RNN architecture we studied was

   $$s_t = f(W^{ss} s_{t-1} + W^{sx} x_t)$$
   $$y_t = W^o s_t$$

   where W^{ss} is $m \times m$, W^{sx} is $m \times d$, and f is an activation function to be specified later. We omit the offset parameters for simplicity (set them to zero).

   2. **Nora** thinks the basic RNN is representationally weak, and it would be better not to decompose the state update in this way. Nora's proposal is to instead

   $$s_t = f(W^{ssx} \mathit{concat}(s_{t-1}, x_t))$$
   $$y_t = W^o s_t$$

   where $\mathit{concat}(s_{t-1}, x_t)$ is a vector of length $m + d$ obtained by concatenating s_{t-1} and x_t, so W^{ssx} has dimensions $m \times (m + d)$.

   3. **Rina** wants to try yet another model, of the form:

   $$s_t = f(W^{ss} s_{t-1}) + f(W^{sx} x_t)$$
   $$y_t = W^o s_t$$

   Lets try to understand these models a bit better, and how they might relate.

   (a) Select the correct claim and answer the associated question.

   (1) Claim: The three models are all equivalent when $f(z) = z$. In this case, define W^{ssx} in terms of W^{ss} and W^{sx} so that the equivalence holds.

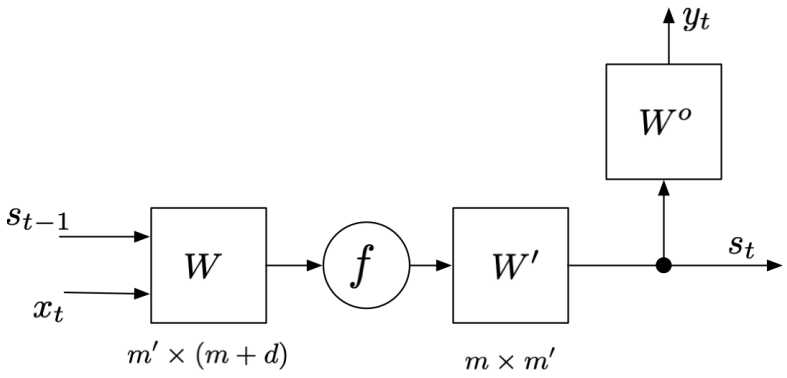
   (2) Claim: The three models are not all equivalent when $f(z) = z$. In this case, assume $m = d = 1$ and provide one setting of W^{ssx} in Nora's model such that W^{ss} and W^{sx} cannot be chosen to make the basic model and Rina's model the same as Nora's.

   Solution: Claim 1 $W^{ssx} = \mathit{hstack}(W^{ss}, W^{sx})$

(b) Here is Rina's model again:

$$s_t = f(W^{ss} s_{t-1}) + f(W^{sx} x_t)$$
$$y_t = W^o s_t$$

Something interesting might happen with this model when $f(z)$ is not the identity. Specifically, it supposedly corresponds to the architecture shown in the figure below, which includes an additional hidden layer. Specify what W, W', and m' are so that this architecture indeed corresponds to Rina's model. Specify your answers in terms of m, W^{ss}, W^{sx}, and W^O.



i. m'

Solution: $2m$

ii. W

Solution: A block-diagonal matrix of the form

$$\begin{bmatrix} W^{ss} & 0 \\ 0 & W^{sx} \end{bmatrix}$$

iii. W'

Solution: $hstack(I(m); I(m))$

Name: _____

Scratch paper

Name: _____

Scratch paper