

6.036: Final, Fall 2018  
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 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: \_\_\_\_\_ MIT Email: \_\_\_\_\_

Question	Points	Score
1	6	
2	10	
3	10	
4	6	
5	13	
6	12	
7	10	
8	14	
9	11	
10	8	
Total:	100	

**Tron**

1. (6 points) In both parts of this question, we are considering perceptron *with an offset*, **not** through the origin.

(a) Consider the data set in two dimensions:

	$x$	$y$
$p_1$	(+1, 0)	-1
$p_2$	(0, +1)	+1
$p_3$	(-1, 0)	+1
$p_4$	(0, -1)	-1

Simulate the perceptron algorithm starting with  $p_1$  and going through the points in order until the process terminates with a separator. Indicate what weight updates are made at each step and the final separator.

**Solution:**

- $p_1 : ((-1, 0), -1)$
  
- $p_2 : ((-1, 1), 0)$

- (b) Assume you run the perceptron algorithm on the data set (**slightly different than the previous one**):

	$x$	$y$
$p_1$	(+1, 0)	+1
$p_2$	(0, +1)	+1
$p_3$	(-1, 0)	+1
$p_4$	(0, -1)	-1

and it makes 2 mistakes on  $p_1$ , 0 mistakes on  $p_2$ , 2 mistakes on  $p_3$  and 3 mistakes on  $p_4$ .

**You don't need to actually simulate the algorithm.**

What is the resulting hypothesis?

$$\theta = \underline{\quad (0, +3) \quad}$$

$$\theta_0 = \underline{\quad 1 \quad}$$

Does it separate the data?    **Yes**    **No**

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

## Our problems multiply

2. (10 points) We will consider a neural network with a slightly unusual structure. Let the input  $x$  be  $d \times 1$  and let the weights be represented as  $k$   $1 \times d$  vectors,  $W^{(1)}, \dots, W^{(k)}$ . Then the final output is

$$\hat{y} = \prod_{i=1}^k \sigma(W^{(i)}x) = \sigma(W^{(1)}x) \times \dots \times \sigma(W^{(k)}x) .$$

Define  $a^{(j)} = \sigma(W^{(j)}x)$ .

- (a) What is  $\partial L(\hat{y}, y) / \partial a^{(j)}$  for some  $j$ ? Since we have not specified the loss function, you can express your answer in terms of  $\partial L(\hat{y}, y) / \partial \hat{y}$ .

**Solution:**

$$\frac{\partial L(\hat{y}, y)}{\partial \hat{y}} \prod_{i \neq j} \sigma(W^{(i)}x)$$

- (b) What are the dimensions of  $\partial a^{(j)} / \partial W^{(j)}$ ?

**Solution:** Because  $a^{(j)}$  is a scalar, they are the same as for  $W^{(j)}$ , which is  $1 \times d$ .

- (c) What is  $\partial a^{(j)} / \partial W^{(j)}$ ? (Recall that  $d\sigma(v)/dv = \sigma(v)(1 - \sigma(v))$ .)

**Solution:**

$$a^{(j)}(1 - a^{(j)})x^T$$

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

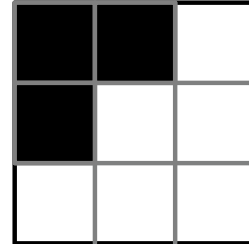
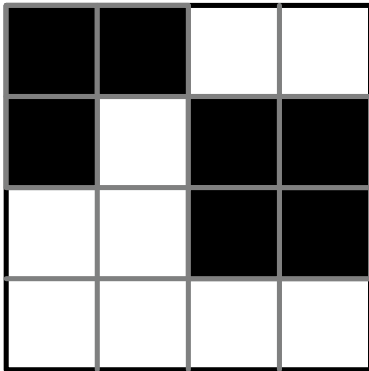
- (d) What would the form of a stochastic gradient descent update rule be for  $W^{(j)}$ ? Express your answer in terms of  $\partial L(\hat{y}, y)/\partial a^{(j)}$  and  $\partial a^{(j)}/\partial W^{(j)}$ . Use  $\eta$  for the step size.

**Solution:**

$$W^{(j)} = W^{(j)} - \eta \frac{\partial L(\hat{y}, y)}{\partial a^{(j)}} \frac{\partial a^{(j)}}{\partial W^{(j)}}$$

## Convolutional News Network

3. (10 points) (a) Consider the following image (on the left) and filter (on the right):



Consider what results from filtering this image with this filter, assuming that the input image is padded with zeros, and using a stride of 1. To compute the output value of a particular pixel  $(i, j)$ , apply the filter with its center on pixel  $(i, j)$  of the input image.

**Assume dark pixels have a value of 1 and light pixels have a value of -1.**

- i. What is the output value for the top-left image pixel (that is, the pixel with indices  $(1, 1)$  in one-based indexing)?

\_\_\_\_\_ **-2** \_\_\_\_\_

- ii. What element of the output image will have the highest value? (Assume the rows and columns of the image are numbered starting with 1.)

\_\_\_\_\_ **3, 1** \_\_\_\_\_

- (b) If we used 5 different filters with size  $3 \times 3$  and stride 1 on this image, what would the dimensions of the resulting output be?

**Solution:**  $4 \times 4 \times 5$

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

(c) What would be the result of applying max-pooling with size  $k = 2$  and stride 2 on the original, unfiltered image above?

i. What are the dimensions of the resulting image?

\_\_\_\_\_  $2 \times 2$  \_\_\_\_\_

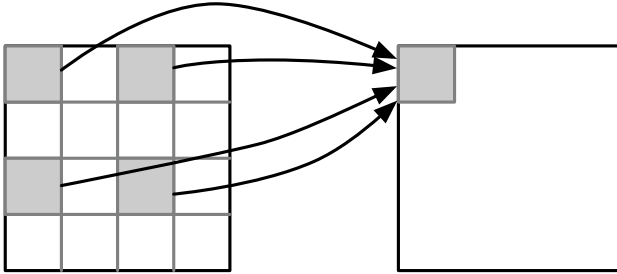
ii. Draw the actual image with numerical values for each pixel in the space below.

**Solution:**

1 1  
-1 1

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

- (d) Dana has an idea for a new kind of network called a ModConv NN. If the network is  $n \times n$ , we will use a filter of size  $n/k$  (assume  $k$  evenly divides  $n$ ). To compute entry  $(a, b)$  of the resulting image, we apply this filter to the “subimage” of pixels  $(i, j)$  from the original image, where  $i \bmod k = a$  and  $j \bmod k = b$ .



- i. Could we train the weights of a ModConvNN using gradient descent? Explain why or why not.

**Solution:** Sure. Just another parametric model.

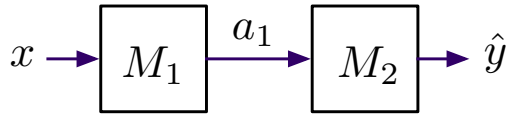
- ii. What underlying assumption about patterns in images is built into a regular convolutional network, but not this one?

**Solution:** This one does not encode the fact that nearby groups of pixels work together to encode information (that there is spatial locality of useful patterns in an image).

## Mix and match

4. (6 points) You are working on a new system that will replace Keras for building neural networks. It is founded on the ideas of series and parallel combination. For simplicity, in this problem, we will assume all of our modules have input and output dimension  $n$ .

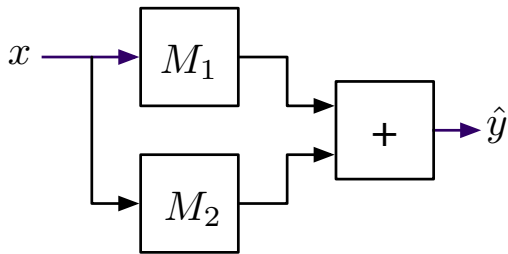
A **series** combination of two modules looks like this:



If you think of each module as a function, then the final output

$$\hat{y} = M_2(M_1(x; W_1); W_2) .$$

A **parallel** combination of two modules looks like this (we added the outputs of the two modules to keep the input and output dimensions equal).



If you think of each module as a function, then the final output

$$\hat{y} = M_1(x; W_1) + M_2(x; W_2) .$$

We won't assume that we know anything about the modules, except that they are feed-forward, have some collection of parameters  $W_i$ , which we will treat as a single vector, and that we can compute

$$M_i(v; W_i) , \quad \frac{\partial M_i(v; W_i)}{\partial W_i} \quad \text{and} \quad \frac{\partial M_i(v; W_i)}{\partial v}$$

for each module, where  $v$  is the input to *that module*. Assume that our loss function is squared loss, so

$$L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2 .$$



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

- (a) What is  $\partial L(\hat{y}, y)/\partial W_1$  for a series combination of  $M_1$  and  $M_2$ ? Write your answer in terms of input  $x$ , target output  $y$ , and weights  $W_1$  and  $W_2$ , using the given forward and gradient functions.

**Solution:**

$$\left( \frac{\partial M_2(a_1; W_2)}{\partial a_1} \frac{\partial M_1(x; W_1)}{\partial W_1} \right)^T (M_2(M_1(x; W_1)) - y)$$

where  $a_1 = M_1(x; W_1)$ .

- (b) What is  $\partial L/\partial W_1$  for a parallel combination of  $M_1$  and  $M_2$ ? Write your answer in terms of input  $x$ , target output  $y$ , and weights  $W_1$  and  $W_2$ , using the given forward and gradient functions.

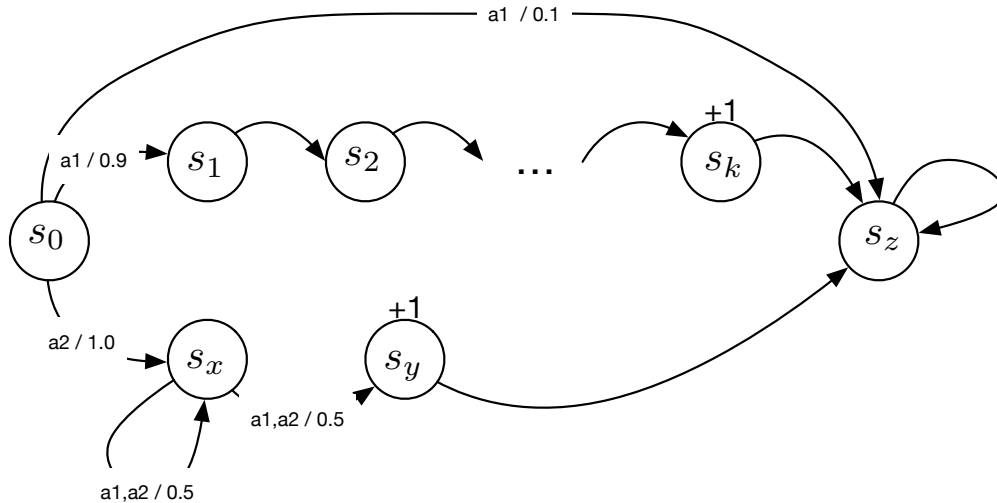
**Solution:**

$$\left( \frac{\partial M_1(x; W_1)}{W_1} \right)^T (M_1(x; W_1) + M_2(x; W_2) - y)$$

## Making Decisions Practical

5. (13 points) Consider the following MDP with  $k + 4$  states. There are two actions,  $a_1$  and  $a_2$ . Arrows with no labels represent a transition for both actions with probability 1. Arrows labeled  $a/p$  make the transition on action  $a$  with probability  $p$ . States with no label have reward 0. Two states have reward  $+1$ , obtained when taking an action in that state. There are  $k - 2$  states between  $s_1$  and  $s_k$ , with a deterministic transition on any action (so that once you are in  $s_1$  you are guaranteed to end up in  $s_k$  in  $k - 1$  steps).

We are interested in the infinite-horizon discounted values of some states in this MDP.



- (a) What is  $V(s_1)$  as a function of  $k$  when  $\gamma = 0$ ?     **0**
- (b) What is  $V(s_1)$  as a function of  $k$  when  $\gamma = 1$ ?     **1**
- (c) What is  $V(s_1)$  as a function of  $k$  when  $0 < \gamma < 1$ ?      $\gamma^{k-1}$
- (d) What is  $V(s_x)$  when  $\gamma = 0$ ?     **0**
- (e) What is  $V(s_x)$  when  $\gamma = 1$ ?     **1**
- (f) What is  $V(s_x)$  when  $0 < \gamma < 1$ ?      $\gamma/(2-\gamma)$

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

- (g) Under what conditions on  $k$  and  $\gamma$  would we prefer to take action  $a_1$  in state  $s_0$ ? Write down a specific mathematical relationship.

**Solution:** When  $(9/10)\gamma^{k-1} > \gamma/(2 - \gamma)$ .

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

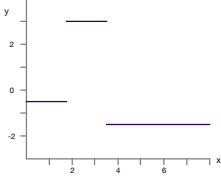
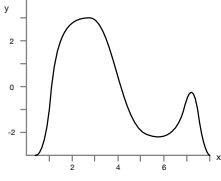
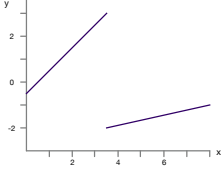
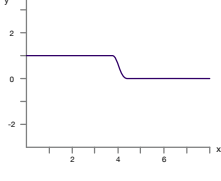
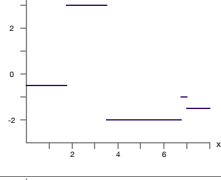
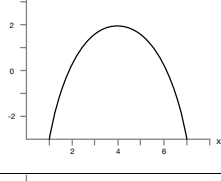
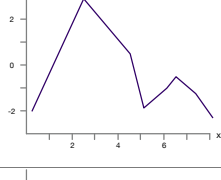
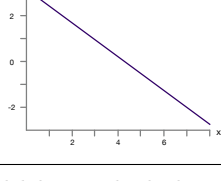
## Regression

6. (12 points) We generated a data set with 5 data-points, with  $x$  and  $y$  values in  $\mathbb{R}$  and applied several regression methods to it.

For each figure below, specify (a) which regression methods *could possibly* have generated the hypothesis on some data set and (b) given that each hypothesis was actually generated by **exactly** one of these methods, match each hypothesis to a single method.

- A 1-Nearest neighbor
- B Regression tree (with constants in the leaves)
- C Regression tree (with linear regressors in the leaves)
- D Linear regression with no feature transformation
- E Linear regression with second-order polynomial features
- F Linear regression with fifth-order polynomial features
- G Neural network with no hidden layer and sigmoid output non-linearity
- H Neural network with one ReLU hidden layer and no output non-linearity

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

	Plot	All possible methods	Best match method
1.		B, C, A*	B
2.		F	F
3.		C	C
4.		G	G
5.		A, B, C	A
6.		E, F	E
7.		H, C	H
8.		D, C, E, F, H	D

\* Could be included or not. Only works if two of the points in the data set are repeated.

## Delay lines

7. (10 points) Recall the specification of a standard recurrent neural network (RNN): input  $x_t$  of dimension  $\ell \times 1$ , state  $s_t$  of dimension  $m \times 1$ , and output  $y_t$  of dimension  $v \times 1$ . The weights in the network, then, are

$$\begin{aligned} W^{sx} &: m \times \ell \\ W^{ss} &: m \times m \\ W^O &: v \times m \end{aligned}$$

with activation functions  $f_1$  and  $f_2$ . **Throughout this problem, for simplicity, we will treat all offsets as equal to 0.** Finally, the operation of the RNN is described by

$$\begin{aligned} s_t &= f_1(W^{sx}x_t + W^{ss}s_{t-1}) \\ y_t &= f_2(W^O s_t) \quad . \end{aligned}$$

- (a) Consider an RNN defined by  $\ell = 1$ ,  $m = 2$ ,  $v = 1$ ,  $f_1 = f_2 =$  the identity function, and

$$W^{sx} = \begin{bmatrix} 5 \\ 6 \end{bmatrix} \quad W^{ss} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad W^O = [-3 \quad -2]$$

Assuming the initial state is all 0, and the input sequence is  $[[1], [-1]]$ , what is the output sequence?

**Solution:**

$$\begin{aligned} s_1 &= [5, 6]^T \\ y_1 &= -15 - 12 = -27 \\ s_2 &= [-5, -6]^T + [5 + 12, 15 + 24]^T = [12, 33]^T \\ y_2 &= -36 - 66 = -102 \end{aligned}$$

So answer is  $[-27], [-102]$ . Don't worry about transpose.

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

(b) We can think of the RNN as mapping input sequences to output sequences. Jody thinks that if we remove  $f_1$  and  $f_2$  then the mapping from input sequence to output sequence can be achieved by a hypothesis of the form  $Y = WX$ . In the case of a length 3 sequence, assuming inputs and outputs are 1-dimensional,  $s_0 = [0]$ ,  $X = [x_1, x_2, x_3]^T$ ,  $Y = [y_1, y_2, y_3]^T$ , and  $W$  is  $3 \times 3$ .

i. Is Jody right?  **Yes**    No

ii. If Jody is right, provide a definition for  $W$  in Jody's model in terms of  $W^{sx}$ ,  $W^{ss}$ , and  $W^O$  of the original RNN that makes them equivalent. If Jody is wrong, explain why.

**Solution:**

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

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

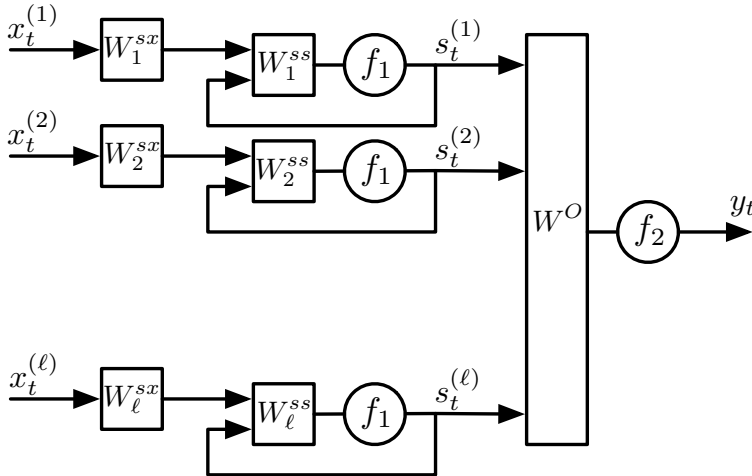
(c) Pat thinks a different RNN model would be good. Its operation is defined by

$$s_t^{(i)} = f_1 \left( W_i^{sx} x_t^{(i)} + W_i^{ss} s_{t-1}^{(i)} \right)$$

$$y_t = f_2 \left( W^O s_t \right) .$$

where the dimension of the state,  $m = k \cdot \ell$ , so there are  $k$  state dimensions for each input dimension,  $s^{(i)}$  is the  $i$ th group of  $k$  dimensions in the state vector,  $x^{(i)}$  is the  $i$ th entry in the input vector,  $W_i^{sx}$  is  $k \times 1$  and  $W_i^{ss}$  is  $k \times k$ .

Here is a diagram.



- i. Can this model represent the same set of state machines as a regular RNN?  
 Yes     No
- ii. If yes, explain how to convert the weights of a regular RNN into weights for Pat's model.

If no, describe a concrete input/output relationship (for example, the output  $y_t$  is the sum of all the inputs  $x_t^{(1)}, \dots, x_t^{(l)}$ ) that **can** be represented by a regular RNN but cannot be represented by Pat's model, for any value of  $k$ .

**Solution:** Output a 1 if and only if  $x^{(1)}$  and  $x^{(2)}$  were simultaneously non-zero.



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

## Rotten tomatoes

8. (14 points) (a) Alex finds a solution to a collaborative filtering problem with 4 users, 4 movies, and  $k = 2$ :

$$U = \begin{bmatrix} 1 & 2 \\ 3 & 1 \\ 2 & 5 \\ 1 & 3 \end{bmatrix} \quad V^T = \begin{bmatrix} 2 & 1 & 5 & 3 \\ 3 & 4 & 2 & 1 \end{bmatrix}$$

Assume  $b_U^{(a)} = 0$  and  $b_V^{(i)} = 0$  for all  $a$  and  $i$ .

Assuming that the first entry has index 1, what is the predicted value for user 3's rating of movie 1?

**Solution:** 19

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

Now, we are going to combine collaborative filtering with content-based filtering. Assume we have  $\ell$  features provided for each movie (encoding things like their genre, average rating, and director). So  $\Phi$  is an  $m \times \ell$  matrix of values specifying each feature for each movie, and let  $\Phi^{(i)}$  be the row of  $\Phi$  corresponding to movie  $i$ .

We consider two different models for using  $\Phi$ :

$$M_1 : X_{ai} = U^{(a)} \cdot V^{(i)} + W^{(a)} \cdot \Phi^{(i)}$$

$$M_2 : X_{ai} = U^{(a)} \cdot V^{(i)} + W \cdot \Phi^{(i)}$$

In  $M_1$ ,  $W^{(a)}$  is a  $1 \times \ell$  vector of weights. In  $M_2$ ,  $W$  is a  $1 \times \ell$  vector of weights.

(b) Assume that the first feature for each movie is its average review score. Jesse is a contrarian and tends to like movies that get bad reviews.

i. Can model  $M_1$  encode this information about Jesse?

**Yes**    **No**

ii. If it cannot, explain why not. If it can, explain which parts of  $U$ ,  $V$  or  $W$  in the model would encode Jesse's preference.

**Solution:** M1: Jesse would have a high negative value for  $W_1^{(a)}$ .

iii. Can model  $M_2$  encode this information about Jesse?

**Yes**    **No**

iv. If it cannot, explain why not. If it can, explain which parts of  $U$ ,  $V$  or  $W$  in the model would have particularly low or high values to encode Jesse's preference.

**Solution:**  $W$  is shared across all users

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

Let  $\mathcal{D}_W$  be the whole training data set of  $(a, i, r)$  tuples,  $\mathcal{D}_a$  be the set of  $(i, r)$  values in  $\mathcal{D}$  for user  $a$  and let  $\mathcal{D}_i$  be the set of  $(a, r)$  values in  $\mathcal{D}$  for movie  $i$ .

Assume we have a linear regression algorithm **regress**( $D$ ) which takes a data set of pairs and returns solution  $\theta$ . For simplicity, we will ignore offsets throughout this question.

The standard ALS algorithm loops, alternating between two steps:

1. For all  $a$ , compute  $U^{(a)} = \text{regress}((V^{(i)}, r)$  for  $(i, r) \in \mathcal{D}_a$ )
2. For all  $i$ , compute  $V^{(i)} = \text{regress}((U^{(a)}, r)$  for  $(a, r) \in \mathcal{D}_i$ )

Now, we would like to find versions of ALS algorithms for finding  $U$ ,  $V$ , and  $W$  in models  $M_1$  and  $M_2$ . Fill in the blanks in the algorithms below for computing the parameters of each of these models. You can use  $\mathcal{D}_W, \mathcal{D}_a, \mathcal{D}_i, \Phi^{(i)}$ . If necessary, you may:

- make more than one call to **regress** in each case;
- concatenate vectors to construct the input of a regression and treat the output as supplying a concatenation of more than one weight vector;
- use all or a subset of parts (i), (ii), and (iii) below.

(c) Provide the steps inside a loop for computing all the parameters of model  $M_1$ .

- i. For all  $a$ , compute

$$\underline{\hspace{10em} U^{(a)}; W^{(a)} = \text{regress}((V^{(i)}; \Phi^{(i)}, r) \text{ for } (i, r) \in \mathcal{D}_a) \hspace{10em}}_\underline{\hspace{10em}}$$

- ii. For all  $i$ , compute

$$\underline{\hspace{10em} V^{(i)} = \text{regress}((U^{(a)}, r) \text{ for } (a, r) \in \mathcal{D}_i) \hspace{10em}}_\underline{\hspace{10em}}$$

- iii. Using all the data, compute

\_\_\_\_\_

(d) Provide the steps inside a loop for computing all the parameters of model  $M_2$ .

- i. For all  $a$ , compute

$$\underline{\hspace{10em} U^{(a)} = \text{regress}((V^{(i)}, r - W\Phi^{(i)}) \text{ for } (i, r) \in \mathcal{D}_a) \hspace{10em}}_\underline{\hspace{10em}}$$

- ii. For all  $i$ , compute

$$\underline{\hspace{10em} V^{(i)} = \text{regress}((U^{(a)}, r - W\Phi^{(i)}) \text{ for } (a, r) \in \mathcal{D}_i) \hspace{10em}}_\underline{\hspace{10em}}$$

- iii. Using all the data, compute

$$\underline{\hspace{10em} W = \text{regress}((\Phi^{(i)}, r - U^{(a)} \cdot V^{(i)}) \text{ for } (a, i, r) \in \mathcal{D}) \hspace{10em}}_\underline{\hspace{10em}}$$

## Adult supervision

9. (11 points) Sometimes we can make robust reinforcement-learning algorithms by reducing the problem to supervised learning. Assume:

- The state space is  $\mathbb{R}^d$ , so in general the same state  $s$  may not occur more than once in our data set.
- The action space is  $\{0, 1\}$ .
- The space of possible rewards is  $\{0, 1\}$ .
- There is a discount factor  $\gamma$ .

You are given a data set  $\mathcal{D}$  of experience interacting with a domain. It contains  $n$  tuples, each of the form  $(s, a, r, s')$ . Let  $\mathcal{D}_0$  be the subset of the data tuples where  $a = 0$ , and similarly  $\mathcal{D}_1$  be the subset of the data tuples where  $a = 1$ .

Assume you have supervised classification and regression algorithms available to you, so that you can call **classify**( $X, Y$ ) or **regress**( $X, Y$ ) where  $X$  is a matrix of input values and  $Y$  is a vector of output values, and get out a hypothesis.

In each of the following questions, we will ask you to construct a call to one of these procedures to produce a  $Q$ ,  $V$ , or  $\pi$  function. In each case, we will ask you to specify:

- Whether it is a regression or classification problem.
- The subset of  $\mathcal{D}$  you will use.
- How you will construct a training example  $(x, y)$  from an original tuple  $(s, a, r, s')$ .

For example, if you wanted to train a neural network to take in a state  $s$  and predict the expected next state given that you take action 1, then you might do a regression problem using data  $\mathcal{D}_1$ , by setting  $x = s$  and  $y = s'$ .

(a) Assume horizon  $h = 1$ . Construct a supervised learning problem to find  $Q^1(s, 0)$ , that is, the horizon-1  $Q$  value for action 0, as a function of state  $s$ .

i.  Classification     **Regression**

ii.   $\mathcal{D}$       $\mathcal{D}_0$       $\mathcal{D}_1$

iii.  $x$ : \_\_\_\_\_  $s$  \_\_\_\_\_

iv.  $y$ : \_\_\_\_\_  $r$  \_\_\_\_\_

(b) Still assuming horizon  $h = 1$ , construct a supervised learning problem to find the optimal policy  $\pi^1$ . Recall that the space of possible rewards is  $\{0, 1\}$ .

i.  **Classification**     Regression

ii.   $\mathcal{D}$       $\mathcal{D}_0$       $\mathcal{D}_1$

iii.  $x$ : \_\_\_\_\_  $s$  \_\_\_\_\_

iv.  $y$ :  $a$  if  $r = 1$  else  $1 - a$

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

- (c) Now, assume that we have already learned  $V^3(s)$ , that is, a function that maps a state  $s$  into the optimal horizon-three value.

Construct a supervised learning problem to find the optimal horizon 4 Q function for action 0,  $Q^4(s, 0)$ . You can make calls to  $V^3$ .

i.  Classification     **Regression**

ii.   $\mathcal{D}$       $\mathcal{D}_0$       $\mathcal{D}_1$

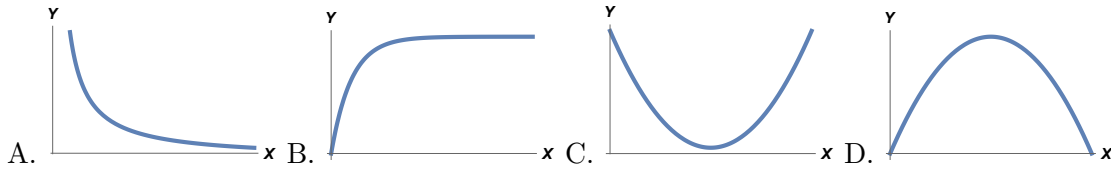
iii.  $x$ : \_\_\_\_\_  $s$  \_\_\_\_\_

iv.  $y$ :  $r + \gamma V^3(s')$

- (d) Because the state space is continuous, it is difficult to train  $V^4$  without first estimating  $Q^4$ , given only our data set and  $V^3$ . Explain briefly why.

**Solution:** For any given  $s$  we only know what happens when we take one of the actions, but not the other, since they don't line up, we don't have a way to take the max over actions.

### What goes up?



10. (8 points) We can do machine learning experiments in which we hold all parameters constant, vary a single parameter, take the hypotheses we have learned at each point, and plot its error on the training and test sets. For each experiment below, select a plot from above that indicates the generally expected shape of the curve for training and for test error. If the experiment doesn't make any sense, choose "not sensible" rather than a curve above.

Don't worry about the fact that we would generally expect curves to bounce around a bit and not be as smooth as these. Also, don't necessarily interpret the plotted  $x$  axis as being for the value  $y = 0$ .

**Provide a one-sentence justification for each answer.**

- (a) X axis: size of training set

train error:       **B**      

      **It's easier to get low training error on small dataset.**      

test error:       **A**      

      **As we get more training data, we generalize better to new test data.**      

- (b) X axis: number of iterations of gradient descent

train error:       **A**      

      **Training error is usually our objective, and generally decreases with iterations.**      

test error:       **C**      

      **Early, we have not fit well enough; later we may overfit**

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

(c) X axis: gradient-descent step size  $\eta$

train error: \_\_\_\_\_ **C** \_\_\_\_\_

Small step size is slow to converge; big step size may diverge.

test error: \_\_\_\_\_ **C** \_\_\_\_\_

Test error is likely to suffer in the same way as training error.

(d) X axis: regularization parameter  $\lambda$

train error: \_\_\_\_\_ **B** \_\_\_\_\_

With bigger  $\lambda$  we quit caring about training error.

test error: \_\_\_\_\_ **C** \_\_\_\_\_

With small  $\lambda$  we may overfit; with big  $\lambda$  we may not fit well enough.