UC Berkeley – Computer Science
CS188: Introduction to Artificial Intelligence
Josh Hug and Adam Janin

Final, Fall 2016

This test has **10** questions worth a total of **100** points, to be completed in 170 minutes. The exam is closed book, except that you are allowed to use three two-sided hand written cheat sheets. No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided.

Write the statement out below in the blank provided and sign. You may do this before the exam begins. Any plagiarism, no matter how minor, will result in an F.

"I have neither given nor received any assistance in the taking of this exam."

____

____

Signature: ______________________________

Name: ________________________ ________________________

SID: ________________________ **left:**________________________

Exam Room: ________________________ **right:** ________________________

Primary TA: ________________________

Tips:

- ○ indicates that only one circle should be filled in.
- ▢ indicates that more than one box may be filled in.
- Be sure to fill in the ○ and ▢ boxes completely and erase fully if you change your answer.
- There may be partial credit for incomplete answers. Write as much of the solution as you can, but bear in mind that we may deduct points if your answers are much more complicated than necessary.
- There are a lot of problems on this exam. Work through the ones with which you are comfortable first. **Do not get overly captivated by interesting problems or complex corner cases you're not sure about.**
- Not all information provided in a problem may be useful.
- **There are some problems on this exam with a slow brute force approach and a faster, clever approach. Think before you start calculating with lots of numbers!**
- Write the last four digits of your SID on each page in case pages get shuffled during scanning.

Problem	1	2	3	4	5	6	7	8	9	10
Points	7	6	8.5	8.5	17	14	9	9	10.5	10.5

Optional. Mark along the line to show your feelings
on the spectrum between :(and ☺.

Before exam: [:(____________________ ☺].
After exam: [:(____________________ ☺].

1. (7 pts) Naive Bayes - Josh Detector

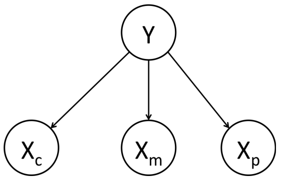
Suppose we want to find if Josh is in this office ($Y = 1$) or not ($Y = 0$) by analyzing the measurements of the following three sensors deployed in his office:

- a chair pressure sensor informs you if someone is on his chair ($X_c = 1$) or not ($X_c = 0$)
- a motion detector tells you if someone moves in the room ($X_m = 1$) or not ($X_m = 0$)
- a power meter indicates if someone consumes electricity in the room ($X_p = 1$) or not ($X_p = 0$)

Each sensor above only reveals partial information about Josh's presence. For instance, if Josh is in his office but not sitting on the chair, then the chair pressure sensor can not reliably indicate Josh's presence. Suppose we have some historical data logs of sensor measurements and Josh's presence status:

Day	1	2	3	4	5	6	7	8	9	10
X_c	0	0	0	0	0	1	0	1	0	1
X_m	0	0	0	0	0	0	0	0	1	1
X_p	1	1	0	0	1	1	1	1	0	0
Y	0	0	0	0	1	1	1	1	1	1

i. (2 pts) Let us use Naive Bayes to model the sensor measurements and Josh's presence status as shown in the Bayes Net model below. Fill in the maximum likelihood estimate (MLE) of each entry in the probability tables.



(B1)

Y	$P(Y)$
0	
1	

Y	X_c	$P(X_c\|Y)$
0	0	
0	1	
1	0	
1	1	

Y	X_m	$P(X_m\|Y)$
0	0	
0	1	
1	0	
1	1	

Y	X_p	$P(X_p\|Y)$
0	0	
0	1	
1	0	
1	1	

ii. (1 pt) Suppose we get a new set of sensor observations: $X_c = 0,\ X_m = 0,\ X_p = 1$. According to the probability tables' maximum likelihood estimate, is Josh more likely to be present or absent?

○ Present ○ Absent ○ Equally Likely ○ Not Enough Information

iii. (1 pt) Suppose that instead of maximum likelihood, we now use Laplace Smoothing to estimate each entry in the following probability tables. Assume the prior strength $k = 1$.

Y	$P(Y)$
0	
1	

Y	X_c	$P(X_c \mid Y)$
0	0	
0	1	
1	0	
1	1	

iv. (1 pt) What happens to our CPTs as we increase k, i.e. what sort of distribution do we get in the large k limit?

v. (1 pt) Suppose we want to generate exactly two samples from the distribution P(Y, X_c, X_m, X_p) given by the CPTs you computed from MLE estimates in part i. Suppose we run Gibbs sampling for a long time so that the samples are indeed drawn from the correct distribution, and get the resulting sample GS1: (Y = 1, X_c = 0, X_m = 1, X_p = 0). Suppose we then generate a new sample GS2 by resampling from P(X_p | Y = 1, X_c = 0, X_m = 1). What are the diffe[illegible]ssible values for GS2, and how likely is each?

Possible Value #1 for GS2: Y = X_c = X_m = X_p = probability:

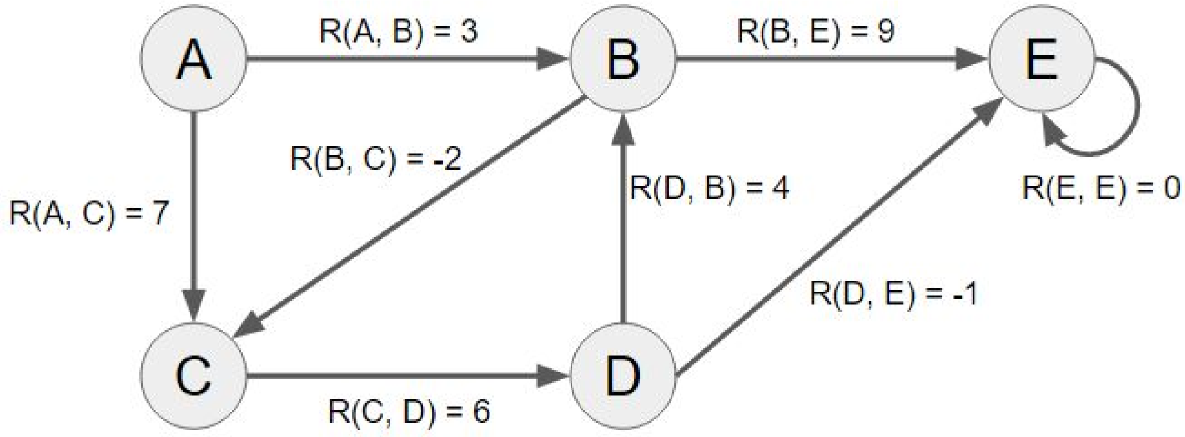
Possible Value #2 for GS2: Y = X_c = X_m = X_p = probability:

vi. (1 pt) Suppose we instead use prior sampling to generate samples PS1 and PS2. Which samples are of higher quality, the two samples generated by Gibbs Sampling or the two samples generated by Prior Sampling? We have intentionally left "higher quality" undefined. Use your best judgment. For partial credit in the event of a wrong answer, very briefly explain your answer below.

○ GS1 and GS2 are higher quality ○ PS1 and PS2 are higher quality ○ They are the same quality

2. MDPs [6 pts]

Consider the following Markov Decision Process. Unlike most MDP models where actions have many potential outcomes of varying probability, assume that the transitions are **deterministic**, i.e. occur with T(s, a, s')=100% probability as shown in the graph below. The reward for each transition is displayed adjacent to each edge:



i. (3 pts) Compute the following quantities, assuming a discount factor of 1. Recall that $V_k(s)$ represents the expected utility of the state s under the optimal policy assuming the game ends in k more time steps, and V*(s) is the expected utility starting in state s if we play optimally forever.

$V_2(A) =$ $V_5(A) =$

$V^*(A) =$

$V_2(B) =$ $V_5(B) =$

$V^*(B) =$

ii. (1 pt) Consider using feature-based Q-learning to learn what the optimal policy should be. Suppose we use the feature $f_1(s, a)$ equal to the smallest number of edges from s to E, and feature $f_2(s, a)$ equal to the smallest number of edges from a's target to E. For example $f_1(A, \rightarrow B)$ is 2, and $f_2(C, \rightarrow D)$ is 1.

Let the current weight for these features be [3, 2]. What is the current prediction for Q(C, →D) given these weights for these two features?

Q(C, →D) =

iii. (2 pts) Suppose we update w based on a single observation of a transition from C to D, assuming $\alpha = 0.5$. What is w after updating based on this observation?

w =

3. Probability (8.5 pts)

Consider the following CPT, **to be used for part i only!**

A	B	C	P(A, B, C)
+a	+b	+c	0.15
+a	+b	-c	0.2
+a	-b	+c	0.3
+a	-b	-c	0.1
-a	+b	+c	0.15
-a	+b	-c	0
-a	-b	+c	0.05
-a	-b	-c	0.05

i. (2 pts) Compute the following expressions for the table above. Write each answer as a single numerical value. It is ok to leave your answer as a fraction of decimal numbers.

- P(+a | -c) =
- P(+b | -a, -c) =
- P(-a | +b, +c) =
- P(-b, +c | -a) =

ii. (1 pt) Let A, B, and C be binary random variables. Select all of the following CPTs that are guaranteed to have 4 rows **for ANY distribution**, not the table above.

□ P(+a | B, C) □ P(B, C | +a) □ P(B | A) □ P(+a, C | +b)

iii. (2 pts) Select all of the following that are guaranteed to equal 1 for ANY distribution, not the table above.

□ P(+a) + P(-a)

□ P(+a | -c)P(-c) / (P(+a, -c, +b) + P(+a, -c, -b))

□ P(+a | +b)P(+b) + P(-a | -b)P(-b)

□ P(-c | -a)P(-a) + P(-c | +a)P(+a)

For parts iv - vi: It’s winter break, and you’re staying at Sherdil’s Casino. Every morning, you receive (as a gift) a random ***odd*** number of gambling chips, C, in the range 1 to k, where k is some odd integer constant, according to the distribution:

$$P(C = c) = \begin{cases} 2/(k+1), & c \text{ is odd and } c \in [1, k] \\ 0, & \text{otherwise} \end{cases}$$

Assume the number of chips you receive on a particular day is independent of the number of chips you received on any and all prior days.

iv. (1 pt) Let the number of chips you receive on days 1, 2, and 3 be C1, C2, and C3 respectively. Is C3 conditionally independent of C1 given C2? (Answer yes or no. No explanation is needed.)

○ Yes ○ No

v. (1 pt) Suppose *k* is 5. On average, how many chips can you expect to receive per day?

E[# chips] =

vi. (1.5 pts) Suppose you stay for five days and receive the following chip amounts: {5, 11, 1, 9, 9}. Based on your five samples, compute the maximum likelihood estimate for *k*.

MLE of k =

2016 Most Valuable Goat Celebration Zone

4. CSPs (8.5 pts)

In the game of Hidato, we start with a partially filled grid with N blanks, and our goal is to fill in all the blanks with integers 1 through N such that every number is unique, and every number q (except N) appears adjacent to its successor q + 1, with diagonal moves allowed. For example, if we start from the figure on the left, we can get the solution on the right. Black squares act as obstacles and never receive numbers.

■		3	
14	1		
		■	
		9	

■	2	3	4
14	1	6	5
13	12	■	7
11	10	9	8

We can formulate this as a CSP, where the domain of the variables is {1, …, N}, where N is the number of non-black (i.e. initially blank) squares.

i. (2.5 pts) If we start from the figure below with 4 variables assigned (to 1, 3, 9, and 14), and run AC-3 (our arc-consistency algorithm), what values remain in the domain of the variable marked with an X? Assume that all 10 blank squares (including X and Y) are unassigned.

■	X	3	
14	1	Y	
		■	
		9	

ii. (2.5 pts) If we assign X=2, and re-run AC-3, what will Y's domain be after AC-3 completes? Assume that all 9 blank squares (including Y) other than X are unassigned.

iii. (1 pt) If we ensure arc-consistency after every assignment in any Hidato puzzle, we'll never backtrack.

○ True ○ False

iv. (2.5 pts) Repeat part i, but suppose we run an algorithm that ensures 3-consistency instead of arc-consistency (recall that arc-consistency is just 2-consistency). In this case, what is the domain of X?

Domain of X:

5. (17 pts) Potpourri

i. (2.5 pts) You are given a choice between the following:

- A random lottery in which you receive either \$A or \$B uniformly at random.
- An event in which you receive \$*k*, where *k* is a positive real-valued constant.

Your utility function for money is $U(d) = d^2$ where $U(d)$ is the utility corresponding to d dollars. For what values of *k* should you take the lottery? Give your answer as an interval or a union of intervals as appropriate, e.g., $[12,\ \infty]$ or $[5,\ 9] \cup [7,\ \infty)$. You may handle ties however you wish. Assume that A, B, and k are non-negative.

$k \in$

For parts ii and iii: Let X, Y, and Z be independent random variables with the following distributions:

X	$P(X)$		Y	$P(Y)$		Z	$P(Z)$
3	2/3		2	1/2		-2	1/4
-6	1/3		10	1/2		6	3/4

You can take either action 1 or action 2.

- Action 1 has utility $U_1(X, Y, Z) = 10 * X - 3 * Y * Z$
- Action 2 has utility $U_2(X, Y, Z) = 10 * X + 5 * Y - 5 * Y * Z$

ii. (3.5 pts) Calculate the maximum expected utility. Hint, to avoid tedious calculations, recall from your prior coursework that E[A*B] = E[A]*E[B], so long as A and B are independent variables, and that E[X+Y] = E[X] + E[Y] for any random variables X and Y.

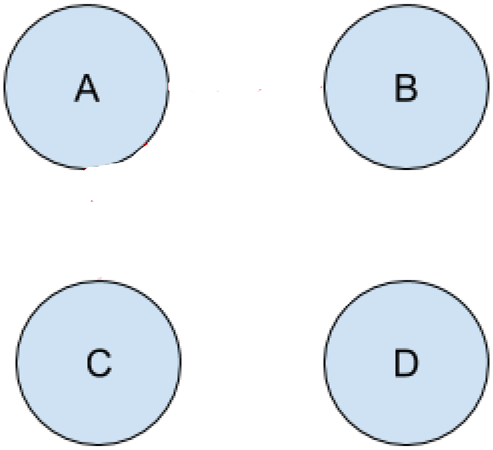
$MEU(\{\}) =$

iii. (2 pts) Which variable has the lowest VPI? Justify your answer either in words or numerically. Please restrict any answers to fewer than 20 words.

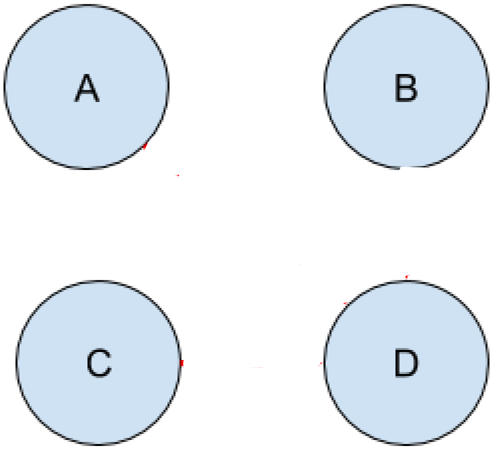
○ X ○ Y ○ Z

iv. (3 pts) You are given a Bayes Net with the four nodes below. The "number of rows" in a Bayes Net is the total number of rows in all CPTs (including the two-row tables for variables with no incoming edges). Assume all variables are binary.

Draw **3** edges such that the Bayes Net is valid, and the number of rows in the Bayes Net is **smaller** than the number of rows in the full joint table for all four variables.



Draw **3** edges such that the Bayes Net is valid, and the number of rows in the Bayes Net is **larger** than the full joint table.



v. (2 pts) Suppose you're trying to choose a move in a two-player adversarial game by using **minimax**. Suppose we use an evaluation function *f(s)* to evaluate the quality of a state during depth limited search. From the list of possibilities for *g(s)* below, which are guaranteed to result in the **exact same action** as if we used *f(s)* instead?

You may assume that all else is equal (including any tie-breaking scheme) except the evaluation functions. You may assume that *f(s)* is always positive.

□ $g(s) = f(s) + 10$

□ $g(s) = 5f(s)$

□ $g(s) = f(s)^2$

□ $g(s) = \log f(s)$

vi. (2 pts) Now suppose you're using **expectimax** instead of minimax to choose a move. From the list of possibilities for *g(s)* below, which are guaranteed to result in the **exact same action** as if we used *f(s)* instead?

Again, you may assume everything except the evaluation functions is the same, and that *f(s)* returns a positive value.

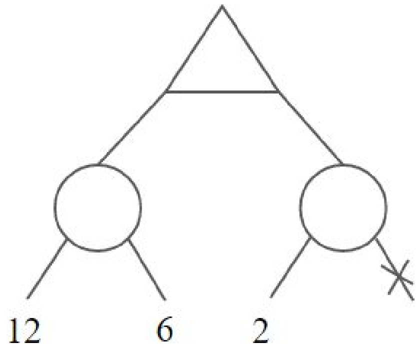
□ $g(s) = f(s) + 10$

□ $g(s) = 5f(s)$

□ $g(s) = f(s)^2$

□ $g(s) = \log f(s)$

vii. (2 pts) Suppose we are running depth 1 expectimax, and know that our evaluation function always returns a value between B and T, where B ≤ T. For what values of B and T can we perform the pruning step as shown below? Assume that all chance outcomes are equally likely.



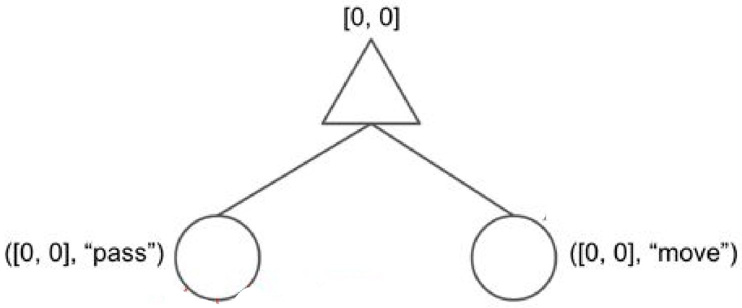
6. MDPs and RL for Solving Adversarial Games (14 pts)

Consider the problem of using an MDP to develop a strategy to maximize a player's reward in a game called **The Downward Spiral.** The game starts with both players having an "elevation" of zero. A game consists of N rounds, where each round consists of a turn by player 1, followed by a turn by player 2. After N rounds, the game is over. During each turn:

- A player may "pass" or "move".
- If a player "passes", their elevation is unchanged.
- If a player "moves", then their elevation ***increases*** by 1 with probability 0.25, and ***decreases*** by 1 with probability 0.75. Moving only changes the elevation of the active player.

Part 1: Suppose that player 1's utility is equal to the number of times that they end a round with a strictly higher elevation than player 2. One approach is to model the game's state as a pair of values representing the elevation of each player, i.e. S_0 is always equal to [0, 0]. After one round, suppose player 1 passes, and player 2 moves and goes down, then the new state is S_X = [0, -1]. To account for player 1's goal, our model has a reward function such that player 1 earns a reward of 1 point for ending the round with a higher elevation. So in the example above $R(S_0, \text{pass}, S_X) = 1$, since the round ended with player 1 having a higher elevation. Throughout the problem, let the discount factor be 1.

i. (3 pts) Player 1 will use an MDP model to decide what actions to take to maximize their rewards. Fill in the **MDP search tree** below for the game assuming that the chance of heads is 0.25, that the model predicts player 2 will randomly pick pass/flip with p=0.5 and the game lasts **<u>only one round (N = 1)</u>**. **<u>Label transitions with their reward and probability</u>**. Label states with the appropriate pair of values [x, y]. The starting state and Q-states are drawn for you. Only the edges and states below the left q-state ([0, 0], "pass") will be graded.



ii. (1 pt) In the **entire** MDP search tree **from part i** (including the side of the tree that you were not required to draw), what is the exact value of the smallest transition probability?

iii. (2 pts) **Let N = 2 rounds**. Let Z be the number of transitions in the entire MDP. How many total transitions are there? Give an exact answer. By transition, we mean an edge from a Q-state to a state.

Z =

iv. (1 pt) Suppose we want to use **model-based reinforcement learning** to approximate player 2's strategy rather than assuming that they choose randomly. Assume that they have some sort of consistent, but not necessarily deterministic strategy. Assuming there are Z transitions, how many T(s, a, s') values do we need to learn? Again assume that **N = 2**. You can do this problem even if you didn't do part 1-ii. If the problem seems ambiguous, state your assumptions.

Number of T(s, a, s') values to learn =

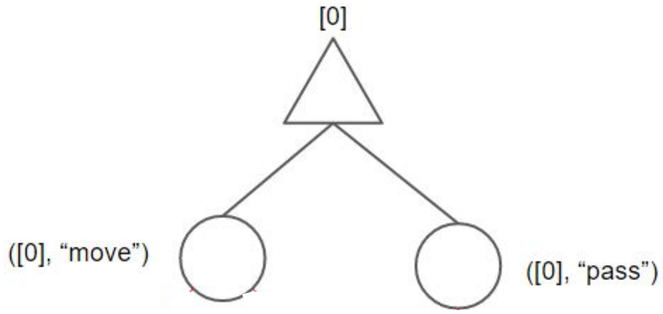
v. (1 pt) Same as part iii, but how many R(s, a, s’) values do we need to learn? If ambiguous, state assumptions.

Number of R(s, a, s’) values to learn =

Part 2) Suppose we keep the rules of The Downward Spiral the same, but change player 1’s utility to **be equal to their elevation at the end of round N**, with no regard to player 2’s elevation. This means adjusting our state space as well as reward function. Note: **You can do this problem without doing part 1.**

vi. (1 pt) Give a minimal representation of the state needed to achieve this new goal, and provide S_0. Hint: Player 2’s actions no longer matter.

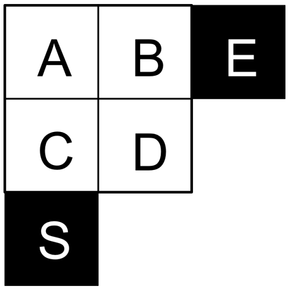
vii. (3 pts) Draw the entire MDP search tree for the game assuming N = 1. Make sure to label the probabilities and rewards for each transition for round 0, as well as the possible states for round 0 and round 1.



viii. (2 pt) If **N = 10,** give the expected utility $V^*(S_0)$ achieved under the optimal policy for this new MDP. Describe the optimal policy.

7. Where is Question Goat? (9 pts)

Question Goat is wandering around between positions A, B, C and D below. Question Goat's location at time t is X_t. At each time step, Question Goat moves clockwise [e.g. A → B, B → D, etc.] with probability 0.8, moves counterclockwise with probability 0.1, and stays at the same position with probability 0.1. To track down Question Goat's position, we deploy two sensors in the East (E) and South (S). The measurements of sensors E and S at time t are Y_E and Y_S, respectively. The distribution of sensor measurements are determined by the Manhattan distance between QG and the sensor. QG's Manhattan distance to sensor S is given by d_S and QG's Manhattan distance to sensor E is given by d_E, e.g. QG at B yields $d_S = 3$, $d_E = 1$. The measurement model of the sensors is given in the tables below. Essentially, as QG gets closer to a sensor, its chance of being on increases.



Y_E	$P(Y_E \mid d_E)$
0	$0.2\ d_E$
1	$1 - 0.2\ d_E$

Y_S	$P(Y_S \mid d_S)$
0	$0.3\ d_S$
1	$1 - 0.3\ d_S$

i. (2 pts) Let us find Question Goat's location by particle filtering with 4 particles. Suppose at *t = 33*, the particles we have are $X^{(1)}$=A, $X^{(2)}$=B, $X^{(3)}$=C, $X^{(4)}$=D. What is the probability that the particles are $X^{(1)}$=B, $X^{(2)}$=A, $X^{(3)}$=D, $X^{(4)}$=D after completion of the time elapsing step in particle filtering?

Probability:

ii. (2 pts) Assume the time elapse step yields $X^{(1)}$=B, $X^{(2)}$=A, $X^{(3)}$=D, $X^{(4)}$=D. Assume the sensor measurements at *t = 34* are $Y_E = 0$ and $Y_S = 1$. What are the **unnormalized** particle weights given these observations?

Particle	Weights
$X^{(1)}$=B	$w^{(1)}$=
$X^{(2)}$=A	$w^{(2)}$=
$X^{(3)}$=D	$w^{(3)}$=
$X^{(4)}$=D	$w^{(4)}$=

iii. (3 pts) Given the weights in part (b), what is the probability that the particles are resampled as $X^{(1)}$=**A**, $X^{(2)}$=**D**, $X^{(3)}$=**B**, and $X^{(4)}$=**B** at time t=34? You do not need to calculate the probability as a number. Instead, represent your solution in terms of $w^{(1)}$, $w^{(2)}$, $w^{(3)}$, $w^{(4)}$.

iv. (2 pts) If the sensors were malfunctioning at *t = 34*, i.e., no observations are produced, what is the probability that the original set of particles $X^{(1)}$=**A**, $X^{(2)}$=**B**, $X^{(3)}$=**C**, $X^{(4)}$=**D** are *resampled* as $X^{(1)}$=**B**, $X^{(2)}$=**A**, $X^{(3)}$=**D**, $X^{(4)}$=**D**?
Probability:

8. Perceptrons and Neurons (9 pts, 1 pt each)

After a single weight update on a single incorrectly classified sample, a binary perceptron correctly classifies that sample: ○ Always ○ Sometimes ○ Never

After a single weight update, the accuracy of a perceptron over the entire training set (check all that apply):

□ Can get better □ Can get worse □ Can stay the same

In a binary perceptron, if a feature vector for a sample is exactly equal to the weight vector, the sample will be classified as part of the positive class. Assume the feature vector is not all zeros.

○ Always ○ Sometimes ○ Never

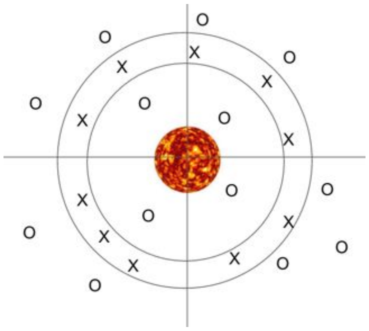
In a multiclass perceptron, if a feature vector for a sample is exactly equal to the weight vector for some class, the sample will be classified as part of that class. Assume the feature vector is not all zeros.

○ Always ○ Sometimes ○ Never

The decision boundary for a binary perceptron is linear in the feature space.

○ Always ○ Sometimes ○ Never

Suppose we are trying to classify the "habitable" zone around a star in a 2D universe. The data is shown below, where X denotes belonging to the habitable zone, and O is not part of the habitable zone.



A binary perceptron with the right weights would correctly classify all planets shown. Assume you can have any features you want.

[✓ True] ○ False

On Project 6, you built a neural network of only a single two-class neuron with a logistic activation function that classified a sample as belonging to the positive class if its activation was greater than 0.5. The decision boundary for this neuron was linear in the feature space.

[✓ Always] ○ Sometimes ○ Never

A single two-class neuron as implemented in Project 6 (and described above) would be able to correctly classify the planetary data from above. Assume you can have any features you want.

[✓ True] ○ False

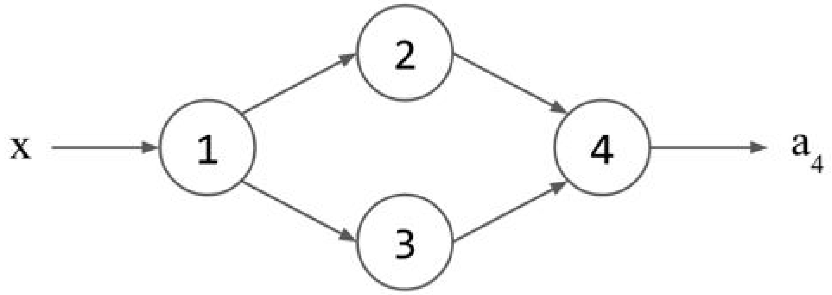
The training data for Project 6 featured handwritten digits that were all about the same size. However, if you hand-wrote a single much smaller "5" and fed it to the classify method (using the same training data as in the project, and assuming the handwritten digit was appropriately converted into a 28x28 pixel format)), it would still probably be classified correctly.

○ True [✓ False]

9. Deep Learning (10.5 pts)

Consider the neural network below. Note that:

- **x**, a scalar, is the input. a_4, also a scalar, is the output.
- Each $\mathbf{a_i}$ value is the final output for neuron **i** in the network.
- Each $\mathbf{z_i}$ value is the *pre-activation* value for neuron **i** in the network (i.e. the dot product).
- $\mathbf{w_1}$ is the weight used by neuron 1.
- Let $\mathbf{w_{ij}}$ be the weight from neuron **i** to neuron **j**.
- The activation function for each neuron is the function $g(x) = e^x$ **.**
- ***L(y, a₄)*** is the loss function. It is $(y - a_4)^2$, where y is the training label.



i. (2.5 pts) Use the Chain Rule and the equations given to calculate $\partial L/\partial z_2$. You may use *x* and *y*, along with all of the *w*, *a*, and *z* values. Note that the picture above only shows the neural network, i.e. does not include the loss function L.

ii. (1.5 pts) Use the Chain Rule and the equations given to calculate $\partial L/\partial z_3$. You may use *x* and *y*, along with all of the *w*, *a*, and *z* values. (*Hint***:** If you're comfortable with the Chain Rule, you should be able to do this quickly by looking at your solution to Part A.)

iii. (2.5 pts) Use the Chain Rule and the equations given to calculate $\partial L/\partial w_1$. You may use *x* and *y*, along with all of the *w*, *a*, and *z* values. ***NOTE*:** You can use the symbols $\partial L/\partial z_2$ and $\partial L/\partial z_3$ in your answer, since you calculated their values above. This is *highly recommended*.

iv. (1 pt) Suppose we're using stochastic gradient descent. What values will be updated based on the partial you computed in part iii, i.e. $\partial L/\partial w_1$? Check all that apply.

□ w_1 □ w_{12} □ w_{13} □ w_{24} □ w_{34} □ x

v. (1 pt) Suppose we're using mini-batching instead of stochastic gradient descent. What values will be updated based on the computation of $\partial L/\partial w_1$? Check all that apply.

□ w_1 □ w_{12} □ w_{13} □ w_{24} □ w_{34} □ x

vi. (1 pt) What is the main difference between mini-batching and stochastic gradient descent? Describe in 10 words or less. **Answers with more than 10 words will receive no credit.**

vii. (1 pt) We can think of backpropagation as a graph traversal over a computational graph. To ensure that all values are available at the time they are needed, only some traversals are accurate. There are some orderings of nodes/neurons that we must use for maximum efficiency. ***For this particular network***, select **all** such orderings that apply.

□ BFS ordering, starting from a_1
□ DFS ordering, starting from a_1
□ UCS ordering, starting from a_1 and all edge weights are 188.
□M BFS ordering, starting from a_4 and flipping all edge directions
□ DFS ordering, starting from a_4 and flipping all edge directions
□M UCS ordering, starting from a_4, flipping all edges, and all edge weights are 188.

10. Pacmanian Speech Recognition (10.5 pts)[1]

As Adam taught during his lecture, one common approach for converting sound files to words (speech recognition) is to first featurize the sound file by converting each 10 millisecond segment into 13 binned "spectral energies", which are essentially how strong the low frequency, medium frequency, etc. parts of the signal are for each 10 millisecond segment.

For this problem, the notation for our features is $F_t(i)$ where t ranges from 0 to the number of frames in the audio file and i ranges from 0 to 12. For example, if we are processing a 1 second clip, we'd have $F_0(i)$ through $F_{99}(i)$, for a total of 1300 features. If $F_5(0) = 2.3$, that means the very low frequency component of the sound clip starting at 50ms and ending at 60ms had an energy of 2.3 in some arbitrary unit.

Suppose throughout this problem that Pacmanian has 20 different sounds, a.k.a. "phonemes" that occur with exactly equal probability. Each word in Pacmanian is composed of a sequence of one or more phonemes. For example, the word "waka" is composed of the phonemes `/w/`, `/a/`, `/k/`, `/a/`.

Part 1: The first step of our speech processing algorithm is to build a model for how likely each phoneme is, given a sound segment. For example, our model might predict that $F_5(i)$, the sound segment between 50ms and 60ms, had a 40% percent chance of being a `/w/`, a 20% percent chance of being a `/y/`, etc.

To do this, we can build a neural network that takes in all 13 binned energies for a particular time segment, and which has 20 outputs that use a softmax activation function.

The notation for the outputs of the neural network is $N_t(i)$, where t is the frame number and i ranges from 0 to 19. $N_{15}(0)$ is for `/a/` at time 15, $N_{31}(1)$ is for `/b/` at time 31, etc.

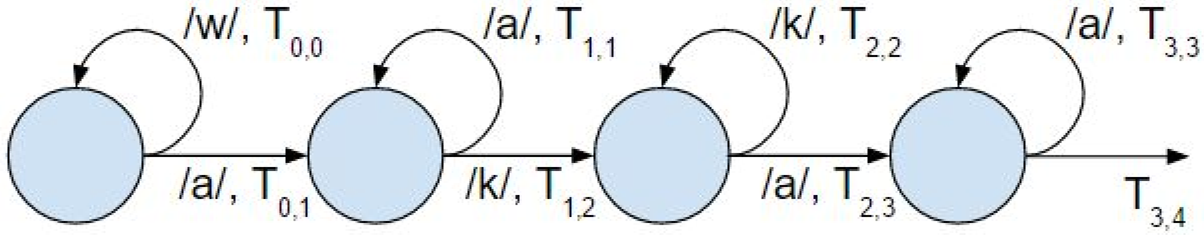
In other words we build a classifier that takes a sound clip (13 energy values) and outputs 20 values (one for each possible phoneme), where $N_t(i)$ represents how likely the sound at timestep t is actually phoneme number i.

i. (2 pts) For each of the following expressions, indicate if it is Always True, Sometimes True, or Never True for all values of t, i, and k. Mark "Sometimes" even if the chance is very small.

	Always	Sometimes	Never	Reason
$0 <= N_t(i) <= 1.0$	[✓ ○]	○	○	
$\sum_i N_t(i) = 1.0$	[✓ ○]	○	○	
$N_t(0) > N_t(1)$	○	[✓ ○]	○	
$F_t(i) = N_t(i)$	○	[✓ ○]	○	

[1] I know there were some complaints about wordy problems, but I felt this problem was too hard without reexplaining at least some of Adam's lecture. With a strong enough 188 foundation, you can do this problem without having deeply understood Adam's lecture.

Part 2: One way to try to detect words is to build an HMM for each possible word. The state transition diagram for the HMM is shown below, where state i is the ith phoneme of the word, and $T_{i,j}$ is the probability of moving from state i to state j. For example, for the word waka, we might assume that there is a 20% chance that that a speaker saying the `/a/` sound keeps making that sound 10 ms later, and an 80% chance that they move on to the `/k/` sound. In that case, $T_{1,1}$ would be 0.2 and $T_{1,2}$ would be 0.8. $T_{3,4}$ represents the speaker finishes the word. If you'd like, you can think of the terminal state is the speaker being silent.



ii. (2.5 pts) Fill in the table, similar to what you did in part i. Mark "Sometimes" even if the chance is very small.

	Always	Sometimes	Never	Reason
$0 <= T_{i,k} <= 1.0$	[✓ ○]	○	○	
$T_{0,1} + T_{0,0} = 1.0$	[✓ ○]	○	○	
$T_{0,1} + T_{1,2} + T_{2,3} = 1.0$	○	[✓ ○]	○	
$T_{1,1} > T_{2,2}$	○	[✓ ○]	○	
$T_{1,1} > T_{3,3}$	○	○	[✓ ○]	

Part 3: Suppose we have a sound file containing a single word. Suppose this sound file is S samples long, has 13 bins of spectral energy, and that Pacmanian has 20 phonemes. **For iii through vi, suppose that we want to use our HMM from above to determine whether the word is "waka".**

iii. (2 pts) How many **hidden variables** must our HMM have? Recall that an HMM is just a Bayes Net with a special structure.

iv. (2 pts) If q_5 is our 5th hidden variable, and q_6 is our 6th hidden variable, how many rows does $P(q_6 \mid q_5)$ have? If the problem seems ambiguous, make sure to state your assumptions.

v. (2 pt) For each hidden variable, how many scalar evidence values are there?