CS 188 Spring 2016 Introduction to Artificial Intelligence Final V2

- You have approximately 2 hours and 50 minutes.
- The exam is closed book, closed calculator, and closed notes except your three crib sheets.
- Mark your answers ON THE EXAM ITSELF. If you are not sure of your answer you may wish to provide a *brief* explanation or show your work.
- For multiple choice questions,
  - □ means mark **all options** that apply
  - ○ means mark a **single choice**
- There are multiple versions of the exam. For fairness, this does not impact the questions asked, only the ordering of options within a given question.

First name	
Last name	
SID	
edX username	

First and last name of student to your left	
First and last name of student to your right	

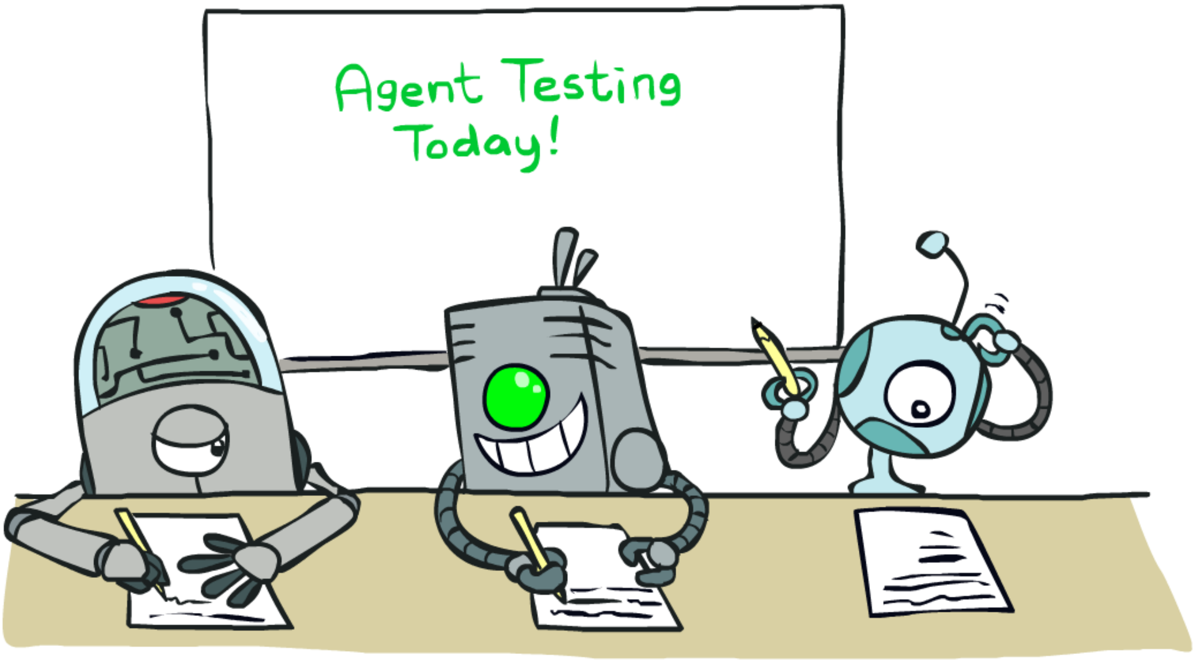
For staff use only:

Q1.	Agent Testing Today!	/1
Q2.	Potpourri	/21
Q3.	Bayes Nets and Sampling	/6
Q4.	Deep Learning	/15
Q5.	MDPs: Reward Shaping	/11
Q6.	Zero Sum MDP's	/6
Q7.	Planning ahead with HMMs	/11
Q8.	Naïve Bayes	/7
Q9.	Beyond Ordinary Pruning	/12
Q10.	Iterative Deepening Search	/10
	Total	/100

THIS PAGE IS INTENTIONALLY LEFT BLANK

Q1. [1 pt] Agent Testing Today!

It's testing time! Circle your favorite robot below. We hope you have fun with the rest of the exam!



Any answer was acceptable.

Q2. [21 pts] Potpourri

(a) **(i)** [1 pt] Suppose we have a multiclass perceptron with three classes A, B, C and with weights initially set to $w_A = [1, 2]$, $w_B = [2, 0]$, $w_C = [2, -1]$. Write out the vectors w_A, w_B, w_C of the perceptron after training on the following two dimensional training example once.

x_0	x_1	label
1	1	A

$w_A = [1, 2]$ $w_B = [2, 0]$ $w_C = [2, -1]$

The predicted label is $y' = \arg\max_{y \in \{A,B,C\}} w_y \cdot [x_0, x_1] = A$, since $w_A \cdot [x_0, x_1] = [1, 2] \cdot [1, 1] = 3$ is greater than both $w_B \cdot [x_0, x_1] = [2, 0] \cdot [1, 1] = 2$ and $w_C \cdot [x_0, x_1] = [2, -1] \cdot [1, 1] = 1$. Since the predicted label $y' = A$ is equal to the correct label $y^* = A$, the weights are not updated, so the weights are the same as the initial ones

(ii) [1 pt] Suppose we have a different multiclass perceptron with three classes A, B, C and with weights initially set to $w_A = [2, 4]$, $w_B = [-1, 0]$, $w_C = [2, -2]$. Write out the vectors w_A, w_B, w_C of the perceptron after training on the following two dimensional training example once.

x_0	x_1	label
-2	1	C

$w_A = [2, 4]$ $w_B = [1, -1]$ $w_C = [0, -1]$

The predicted label is $y' = \arg\max_{y \in \{A,B,C\}} w_y \cdot [x_0, x_1] = B$, since $w_B \cdot [x_0, x_1] = [-1, 0] \cdot [-2, 1] = 2$ is greater than both $w_A \cdot [x_0, x_1] = [2, 4] \cdot [-2, 1] = 0$ and $w_C \cdot [x_0, x_1] = [2, -2] \cdot [-2, 1] = -6$. Since the predicted label $y' = B$ is not equal to the correct label $y^* = C$, the weights are updated by subtracting the datum from the weights of the predicted label and by adding the datum to the weights of the correct label:

$$w_B \leftarrow w_B - [-2, 1] = [1, -1]$$
$$w_C \leftarrow w_C + [-2, 1] = [0, -1].$$

(iii) [3 pts] Suppose we have a different multiclass perceptron with three classes A, B, C and with weights initially set to $w_A = [1, 0]$, $w_B = [1, 1]$, $w_C = [3, 0]$. After training on the following set of training data an infinite number of times, select which of the following options must be True given no additional information. Convergence indicates that the values do not change even within a pass through the data set.

training example i	x_0	x_1	label
0	1	1	A
1	-1	1	B
2	1	-1	C
3	-1	-1	A

○ All of the weight vectors w_A, w_B, w_C converge.

○ Only two of the weight vectors w_A, w_B, w_C converge.

● Only one of the weight vectors w_A, w_B, w_C converge.

○ None of the weight vectors w_A, w_B, w_C converge.

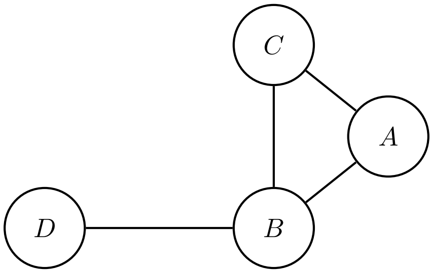
○ None of the above.

First, notice that the data is not linearly separable, so not all of the weight vectors converge (this is the case for the perceptron). Second, notice that it's impossible for only two of the weight vectors to converge since every time the prediction is wrong, two of the weights are updated with a datum (this is the case for the multiclass perceptron), all of which are non-zero for this particular data set. Therefore, there are two possibilities left: either only one of the weight vectors converge or none of them converge. To find out which one is the case, you can do a single pass through the data in order to notice a pattern emerge.

For the training example with $i = 0$, the perceptron incorrectly predicts a label of C, so the weight vectors w_A and w_C get updated to $w_A = [2, 1]$ and $w_C = [2, -1]$. Then, for the training examples with $i = 1$ and $i = 2$, the perceptron correctly predicts the labels B and C, respectively, so the weight vectors are not updated. Then, for the training example with $i = 3$, the perceptron incorrectly predicts a label of C, so the weight vectors w_A and w_C get updated to $w_A = [1, 0]$ and $w_C = [3, 0]$, which are the values that you originally started with! Therefore, as you train the perceptron on the data set an infinite number of times, the value of the weight vector w_A fluctuates between $w_A = [2, 1]$ and $w_A = [1, 0]$, while the value of the weight vector w_C fluctuates between $w_C = [2, -1]$ and $w_C = [3, 0]$. Meanwhile, the value of the weight vector w_B stays at $w_B = [1, 1]$, and therefore it is the only weight vector that converges.

(b) You are given a constraint graph for a Constraint Satisfaction Problem as follows. The domains of all variables are indicated in the table, and the binary constraints are as follows:

- A> B
- A $\neq$ C
- C > B
- D < B



A	0	1	2	3
B	0	1	2	3
C	0	1	2	3
D	0	1	2	3

(i) [3 pts] Enforce arc consistency on this graph and indicate what the domains of all the variables are after arc consistency is enforced, in the table below by crossing out eliminated values from the domains.

A			2	3
B		1	2	
C			2	3
D	0	1		

(ii) [2 pts] Now suppose you are given a different CSP with variables still being A, B, C, D, but you are not given the constraints. The domains of variables remaining after enforcing arc consistency for this CSP are given to you below. Select *all* of the following options which can be inferred given just this information.

A			2	3
B			2	3
C	0	1	2	
D			2	3

■ The CSP may have no solution.
□ The CSP must have a solution.
□ The CSP must have exactly one solution.
■ The CSP may have more than one solution.
□ The CSP must have more than one solution.
□ None of the above.

An example CSP that may have more than one solution with these domains is: A >C, A $\leq$ B, B$\neq$ C, D=B. You get solutions with A=2 and B=3, as well as solutions with A=3 and B=2.
A CSP with no solution is with these domains is: A$\neq$ B, B$\neq$D, D$\neq$ A. There is a cycle between A, B, D and each arc is consistent but there is no overall consistent solution.

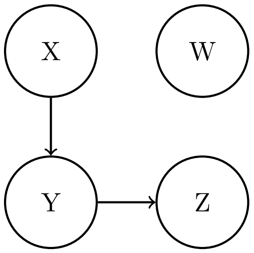
(c) [3 pts] Your assistant gives you the probability distributions for 4 mysterious binary variables: W, X, Y, and Z. Circle the Bayes net(s) amongst those given, that can represent a distribution that is consistent with the tables below using the fewest edges. If there is more than one such minimal net, circle all of them.

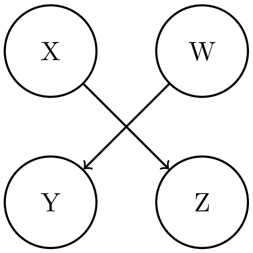
X	$P(X)$
0	0.75
1	0.25

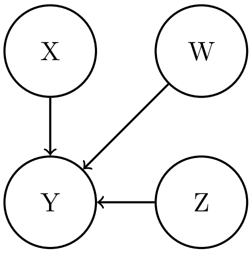
X	W	$P(W\|X)$
0	0	0.4
0	1	0.6
1	0	0.4
1	1	0.6

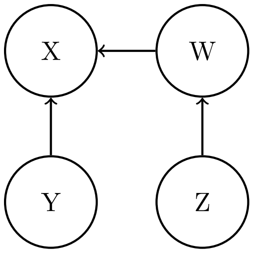
X	Y	$P(Y\|X)$
0	0	0.3
0	1	0.7
1	0	0.1
1	1	0.9

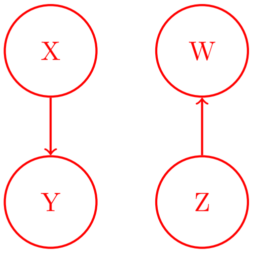
Z	W	$P(W\|Z)$
0	0	0.2
0	1	0.8
1	0	0.8
1	1	0.2

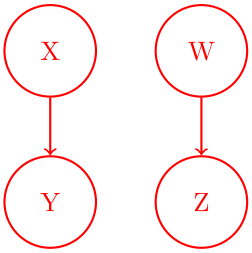








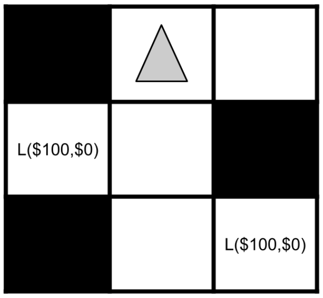




The correct answers are the last two options in the second row. Notice that in the table for $P(W|X)$, the probability of W does not change when X changes. This means that W is independent of X. You can use the values to calculate $P(Y, Z)$ vs $P(Y)P(Z)$, and they are independent. The same holds for X and Z. The minimal bayes net is the one with the fewest arrows (that is, the most enforced independencies). The last two in the bottom row encode the information necessary for the given distribution: the dependence betwen X and Y, and the dependence between Z and W.

(d) Triangle is a rational agent in the world below, where it gains or loses Utility from moving and picking up money. Triangle can move deterministically Up, Down, Left or Right or Stay still. Black squares indicate that the Triangle cannot traverse them. The squares marked with $L(\$100, \$0)$ indicate lotteries of [0.5, \$100; 0.5 \$0]. Taking a step onto a blank square gives Triangle no utility, but stepping onto a lottery square gives it the utility of the lottery, and the lottery disappears.

Additionally, taking a step in any direction has a probability p of giving Triangle pain in addition to whatever money it might earn upon landing on a spot. If Triangle chooses to stay still, it will not feel pain. The utilities are not discounted in this problem so $\gamma = 1$.



In both of the problems below, Triangle's starting position is as shown in the figure above.

(i) [1 pt] For this part, Triangle's utility is as follows(where $k > 0$):

$$U(\text{pain}) = -k; \quad U(\$m) = m$$

What is the expected utility of going to the closest lottery and staying in that spot forever? Express your answer in terms of numerical constants, p, k.

There are 2 moves required to go to the closest lottery, so the expected utility is $-2kp + 0.5 * 100 + 0.5 * 0 = 50 - 2pk$

(ii) [2 pts] Now, triangle's utility function is as follows:

$$U(\text{pain}) = -k; \quad U(\$m) = \sqrt{m}$$

For what range of k (where $k > 0$) will triangle always go to both lotteries. Express your answer in terms of numerical constants and p. If no such range exists, write None in the blank below.

There are 3 possible best options for triangle: staying put, going to the first money, and going to both moneys. $U(\text{both}) = -5pk + 10 * 0.5 + 10 * 0.5$, $U(\text{first}) = -2pk + 10 * 0.5, U(\text{Neither}) = 0$. For both to the optimal strategy, $U(\text{both})$ must be greater than all other options, therefore:

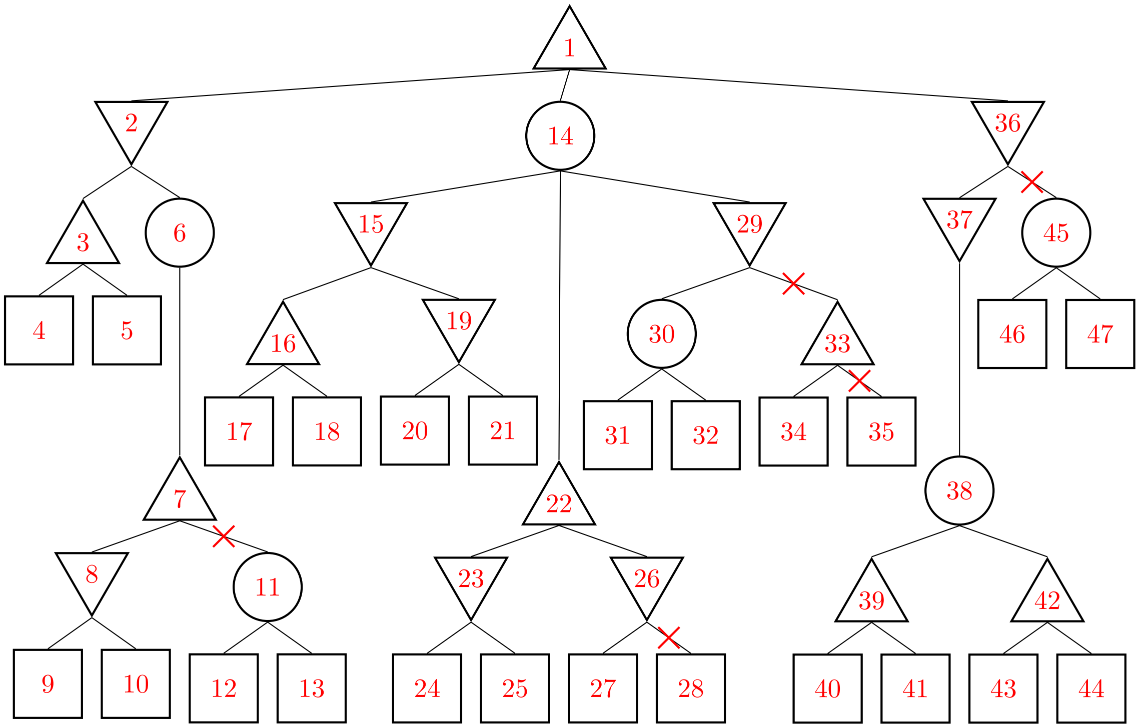
$$-5pk + 10 > -2pk + 5 \text{ and } -5pk + 10 > 0$$

$$k < \frac{5}{3p} \text{ and } k < \frac{10}{5p}$$

$\frac{2}{p} > \frac{5}{3p}$, so the range for k is $k \in (0, \frac{5}{3p})$

(e) [5 pts] For each of the branches in the game tree below, put an 'X' on the **branches** if there exists an assignment of values to leaf nodes, for which that branch could be pruned. The max nodes are upward pointing triangles (△), the min nodes are downward pointing triangles (▽), and the chance nodes are circles (○). Assume that the children of a node are visited in left-to-right order.

Explicitly write down "Not possible" below if no branches can be pruned, in which case any 'X' marks above will be ignored. Any 'X' on the nodes and leaves will be ignored.



When there is an adversarial min-max set of nodes, then bounds can be imposed on how good or bad a node can be which allows us to prune certain branches. However, we always have to look at the left most branch in order to determine whether pruning is ever going to be possible. Branches immediately under a chance node however can never be pruned because they all need to be looked at in order to compute an expectation.

The numbers inside the nodes have been added to the solution for the sole purpose of referencing them in the explanations that follow. There are 5 branches that can be pruned (from left to right):

The first branch (the one above node 11) can be pruned if the minimizer node 8 has a value that is greater than the value that the minimizer node 2 has so far (i.e. the value of node 3). This is what typically happens in a standard minimax tree since the chance node 6 with a single child can be thought of as not doing anything.

The second branch (the one above leave 28) can be pruned if the leave node 27 has a value that is less than the value that the maximizer node 22 has so far (i.e. the value of the node 23).

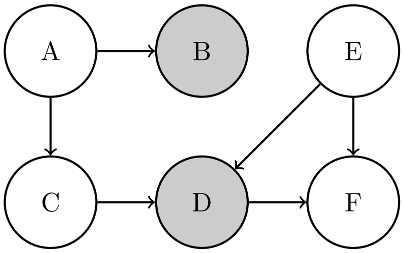
The third branch can be pruned (the one above node 33) if the chance node 14 has a value that is, so far, less than the value that the root maximizer node 1 has so far. This is because once the value of the chance node 30 is known, the value of the chance node 14 can't increase.

The fourth branch (the one above leave 35) can be pruned if the leave 34 has a value that is greater than the value of the chance node 30. Even though the third branch can be pruned, this fourth one should also be marked because there might be cases in which the third branch can't be pruned but the fourth branch can. However, the rubric doesn't penalize for not marking the fourth branch *if* the third branch is marked. This is because the question didn't explictly specify to mark branches that are underneath branches that can be pruned.

The fifth branch (the one above the chance node 45) can be pruned if the minimizer node 37 has a value that is less than the value that the root maximizer node 1 has so far.

Q3. [6 pts] Bayes Nets and Sampling

You are given a bayes net with the following probability tables:



E	D	F	$P(F\|E, D)$
0	0	0	0.6
0	0	1	0.4
0	1	0	0.7
0	1	1	0.3
1	0	0	0.2
1	0	1	0.8
1	1	0	0.7
1	1	1	0.3

A	$P(A)$
0	0.75
1	0.25

A	B	$P(B\|A)$
0	0	0.1
0	1	0.9
1	0	0.5
1	1	0.5

A	C	$P(C\|A)$
0	0	0.3
0	1	0.7
1	0	0.7
1	1	0.3

E	$P(E)$
0	0.1
1	0.9

E	C	D	$P(D\|E, C)$
0	0	0	0.5
0	0	1	0.5
0	1	0	0.2
0	1	1	0.8
1	0	0	0.5
1	0	1	0.5
1	1	0	0.2
1	1	1	0.8

You want to know $P(C = 0|B = 1, D = 0)$ and decide to use sampling to approximate it.

(a) [2 pts] With prior sampling, what would be the likelihood of obtaining the sample [A=1, B=0, C=0, D=0, E=1, F=0]?

○ 0.25*0.1*0.3*0.9*0.8*0.7

○ 0.75*0.1*0.3*0.9*0.5*0.8

○ 0.25*0.9*0.7*0.1*0.5*0.6

● 0.25*0.5*0.7*0.5*0.9*0.2

○ 0.25*0.5*0.3*0.2*0.9*0.2

○ 0.75*0.1*0.3*0.9*0.5*0.2 + 0.25*0.5*0.7*0.5*0.9*0.2

○ Other _________ Prior sampling samples without taking the evidence into account, so the probability of the sample is P(A)P(B—A)P(C—A)P(D—C,E)P(E)P(F—E,D)

(b) [2 pts] Assume you obtained the sample [A = 1, B=1, C=0, D=0, E=1, F=1] through likelihood weighting. What is its weight?

○ 0.25*0.5*0.7*0.5*0.9*0.8

○ 0.25*0.7*0.9*0.8 + 0.75*0.3*0.9*0.8

○ 0.25*0.5*0.7*0.5*0.8

○ 0

● 0.5*0.5

○ 0.9*0.5 + 0.1*0.5

○ Other _________ The weight of a sample in gibbs sampling is the probability of the evidence given their parents: P(D=0—E=1,C=0)*P(B=1—A=1)

(c) [2 pts] You decide to use Gibb's sampling instead. Starting with the initialization [A = 1, B=1, C=0, D=0, E=0, F=0], suppose you resample F first, what is the probability that the next sample drawn is [A = 1, B=1, C=0, D=0, E=0, F=1]?

● 0.4

○ 0.6*0.1*0.5

○ 0.25*0.5*0.7*0.5*0.1*0.3

○ 0.6

○ 0

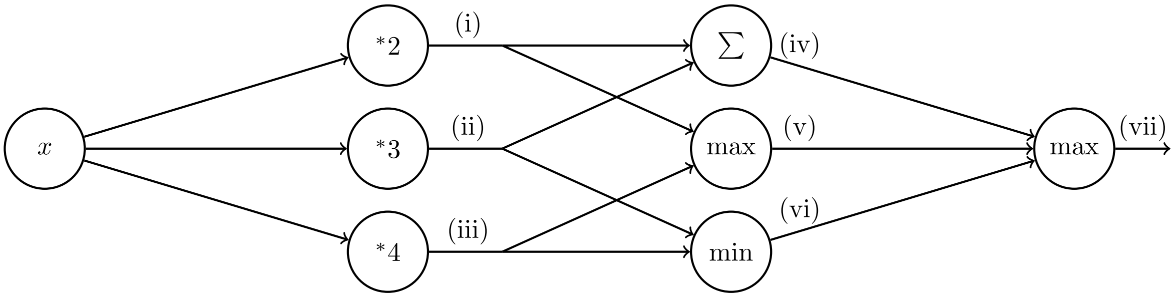
○ 0.9*0.5 + 0.1*0.5

○ Other ________ In Gibb's sampling, you resample individual vairables conditioned on the rest of the sample. The distribution of F given the rest of the sample is 0.4 for F=1 and 0.6 for F=0.

Q4. [15 pts] Deep Learning

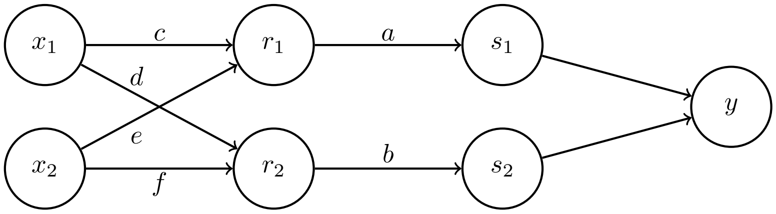
(a) [3 pts] Perform forward propagation on the neural network below for $x = 1$ by filling in the values in the table. Note that (i), ..., (vii) are outputs after performing the appropriate operation as indicated in the node.

(i)	(ii)	(iii)	(iv)	(v)	(vi)	(vii)
2	3	4	5	4	3	5



(b) [6 pts] Below is a neural network with weights a, b, c, d, e, f. The inputs are x_1 and x_2.
The first hidden layer computes $r_1 = \max(c \cdot x_1 + e \cdot x_2, 0)$ and $r_2 = \max(d \cdot x_1 + f \cdot x_2, 0)$.
The second hidden layer computes $s_1 = \frac{1}{1+\exp(-a \cdot r_1)}$ and $s_2 = \frac{1}{1+\exp(-b \cdot r_2)}$.
The output layer computes $y = s_1 + s_2$. Note that the weights a, b, c, d, e, f are indicated along the edges of the neural network here.

Suppose the network has inputs $x_1 = 1, x_2 = -1$.
The weight values are $a = 1, b = 1, c = 4, d = 1, e = 2, f = 2$.
Forward propagation then computes $r_1 = 2, r_2 = 0, s_1 = 0.9, s_2 = 0.5, y = 1.4$. Note: some values are rounded.



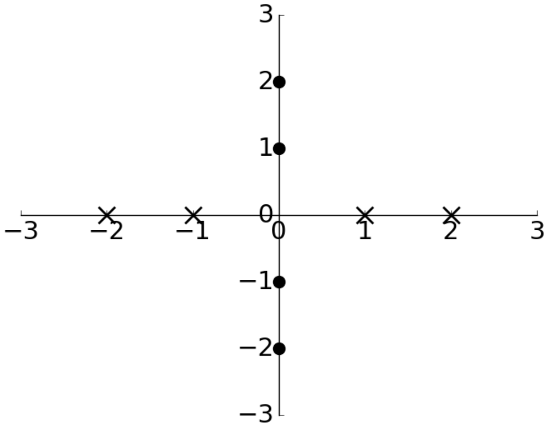
Using the values computed from forward propagation, use backpropagation to numerically calculate the following partial derivatives. Write your answers as a single number (not an expression). You do not need a calculator. Use scratch paper if needed.
Hint: For $g(z) = \frac{1}{1+\exp(-z)}$, the derivative is $\frac{\partial g}{\partial z} = g(z)(1 - g(z))$.

$\frac{\partial y}{\partial a}$	$\frac{\partial y}{\partial b}$	$\frac{\partial y}{\partial c}$	$\frac{\partial y}{\partial d}$	$\frac{\partial y}{\partial e}$	$\frac{\partial y}{\partial f}$
0.18	0	0.09	0	-0.09	0

$$\begin{aligned}
\frac{\partial y}{\partial a} &= \frac{\partial y}{\partial s_1}\frac{\partial s_1}{\partial a} \\
&= 1 \cdot \frac{\partial g(a \cdot r_1)}{\partial a} \\
&= r_1 \cdot g(a \cdot r_1)(1 - g(a \cdot r_1)) \\
&= r_1 \cdot s_1(1 - s_1) \\
&= 2 \cdot 0.9 \cdot (1 - 0.9) \\
&= 0.18 \\
\frac{\partial y}{\partial b} &= \frac{\partial y}{\partial s_2}\frac{\partial s_2}{\partial b} \\
&= 1 \cdot \frac{\partial g(b \cdot r_2)}{\partial b} \\
&= r_2 \cdot g(b \cdot r_2)(1 - g(b \cdot r_2)) \\
&= r_2 \cdot s_2(1 - s_2) \\
&= 0 \cdot 0.5(1 - 0.5) \\
&= 0 \\
\frac{\partial y}{\partial c} &= \frac{\partial y}{\partial s_1}\frac{\partial s_1}{\partial r_1}\frac{\partial r_1}{\partial c} \\
&= 1 \cdot [a \cdot g(a \cdot r_1)(1 - g(a \cdot r_1))] \cdot x_1 \\
&= [a \cdot s_1(1 - s_1)] \cdot x_1 \\
&= [1 \cdot 0.9(1 - 0.9)] \cdot 1 \\
&= 0.09 \\
\frac{\partial y}{\partial d} &= \frac{\partial y}{\partial s_2}\frac{\partial s_2}{\partial r_2}\frac{\partial r_2}{\partial d} \\
&= \frac{\partial y}{\partial s_2}\frac{\partial s_2}{\partial r_2} \cdot 0 \\
&= 0 \\
\frac{\partial y}{\partial e} &= \frac{\partial y}{\partial s_1}\frac{\partial s_1}{\partial r_1}\frac{\partial r_1}{\partial e} \\
&= 1 \cdot [a \cdot g(a \cdot r_1)(1 - g(a \cdot r_1))] \cdot x_2 \\
&= [a \cdot s_1(1 - s_1)] \cdot x_2 \\
&= [1 \cdot 0.9(1 - 0.9)] \cdot -1 \\
&= -0.09 \\
\frac{\partial y}{\partial f} &= \frac{\partial y}{\partial s_2}\frac{\partial s_2}{\partial r_2}\frac{\partial r_2}{\partial f} \\
&= \frac{\partial y}{\partial s_2}\frac{\partial s_2}{\partial r_2} \cdot 0 \\
&= 0
\end{aligned}$$

(c) [6 pts] Below are two plots with horizontal axis x_1 and vertical axis x_2 containing data labelled $\times$ and $\bullet$. For each plot, we wish to find a function $f(x_1, x_2)$ such that $f(x_1, x_2) \geq 0$ for all data labelled $\times$ and $f(x_1, x_2) < 0$ for all data labelled $\bullet$.

Below each plot is the function $f(x_1, x_2)$ for that specific plot. Complete the expressions such that all the data is labelled correctly. If not possible, mark "No valid combination".

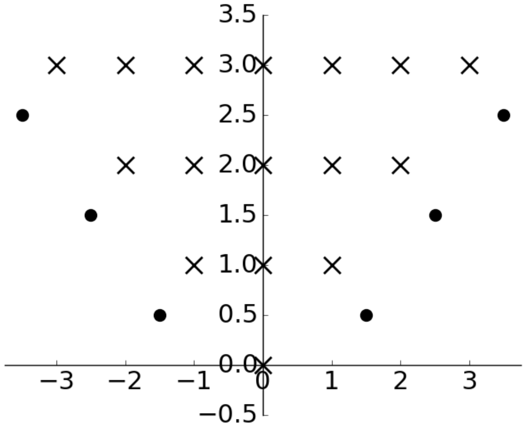


$f(x_1, x_2) = \max(\underline{\textbf{(i)}} + \underline{\textbf{(ii)}}, \underline{\textbf{(iii)}} + \underline{\textbf{(iv)}}) + \underline{\textbf{(v)}}$

(i) ● x_1 ○ $-x_1$ ○ 0
(ii) ○ x_2 ○ $-x_2$ ● 0
(iii) ○ x_1 ● $-x_1$ ○ 0
(iv) ○ x_2 ○ $-x_2$ ● 0
(v) ○ 1 ● -1 ○ 0
○ No valid combination

There are two possible solutions:

$$f(x_1, x_2) = \max(x_1, -x_1) - 1$$
$$f(x_1, x_2) = \max(-x_1, x_1) - 1$$



$f(x_1, x_2) = \underline{\textbf{(vi)}} - \max(\underline{\textbf{(vii)}} + \underline{\textbf{(viii)}}, \underline{\textbf{(ix)}} + \underline{\textbf{(x)}})$

(vi) ● x_2 ○ $-x_2$ ○ 0
(vii) ● x_1 ○ $-x_1$ ○ 0
(viii) ○ x_2 ○ $-x_2$ ● 0
(ix) ○ x_1 ● $-x_1$ ○ 0
(x) ○ x_2 ○ $-x_2$ ● 0
○ No valid combination

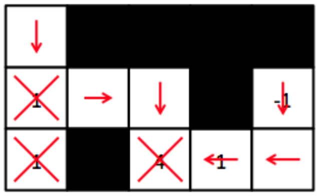
There are four possible solutions:

$$f(x_1, x_2) = x_2 - \max(x_1, -x_1)$$
$$f(x_1, x_2) = x_2 - \max(-x_1, x_1)$$
$$f(x_1, x_2) = -\max(x_1 - x_2, -x_1 - x_2)$$
$$f(x_2, x_2) = -\max(-x_1 - x_2, x_1 - x_2)$$

Q5. [11 pts] MDPs: Reward Shaping

PacBot is in a Gridworld-like environment E. It moves deterministically Up, Down, Right, or Left except that it cannot move onto squares which are blackened. PacBot must move at every step or exit. The reward for any of these actions is always zero. Additionally, from a numbered square, PacBot can choose to exit to a terminal state and collect reward equal to the number on the square. **PacBot is not required to exit on a numbered square; it can also move in any direction off that square.**

(a) [3 pts] Draw an arrow in **each** square (including numbered squares) in the following board on the right to indicate the optimal policy PacBot will calculate with the discount factor $\gamma = 0.5$ in the board on the left. (For example, if PacBot would move Down from the square in the middle on the left board, draw a down arrow in that square on the right board.) If PacBot's policy would be to exit from a particular square, draw an X instead of an arrow in that square.



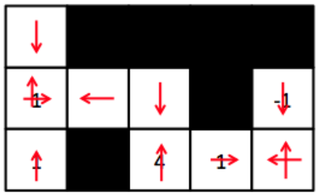
The decision between actions in this MDP at each state is to either take the number on the board or take half the reward that can be attained from a neighboring square. On the square showing 4, the value for moving toward it from one square away is 2. Since that's greater than 1, we move toward the four instead of exiting from the 1 in the bottom right. From two squares away, we can get reward 1 by moving toward the 4. This means that from the cell in the second row, second column, we can get reward 1 by moving right and only reward $\frac{1}{2}$ by moving left. Finally, at distance more then 2 from the 4, we get reward less than 1, so it is optimal to exit from the squares showing 1 on the left side rather than go toward the 4.

PacBot now operates in a new environment E' with an additional reward function $F(s, a, s')$, which is added to the original reward function $R(s, a, s')$ for every (s, a, s') triplet.

(b) [4 pts] Consider an additional reward F_1 that favors moving toward numbered squares. Let $d(s)$ be defined as the Manhattan distance from s to the nearest numbered square. If s is numbered, $d(s) = 0$.

$$F_1(s, a, s') = \begin{cases} 0 & s' \text{ is a terminal state,} \\ 10 & d(s') < d(s) \text{ i.e. } s' \text{ is closer to a numbered square than } s \text{ is,} \\ 0 & d(s') \geq d(s). \end{cases}$$

Fill in the diagram on the right as in (a) to indicate the optimal policy PacBot will calculate with the discount factor $\gamma = 0.5$ and the modified reward function $R_1'(s, a, s') = R(s, a, s') + F_1(s, a, s')$ in the board on the left.



Here, from all exit squares, we can step off of the numbered squares and back on to get a reward of 5. This is 5 and not 2.5 because the value at a numbered square is (let s' be the state resulting from taking the optimal action from s)

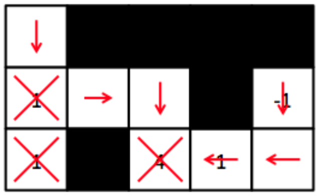
$$\begin{aligned} V(s) &= \max_a (R_1'(s,a,s') + \gamma V(s')) \\ &= 0 + \gamma \max_{a'} (R_1'(s',a',s'') + \gamma V(s'')) && \text{the best choice from } s \text{ is to move to a non-numbered square} \\ &= 10\gamma + \gamma^2 V(s'') && \text{the best choice from } s' \text{ is to move onto a numbered square} \\ &= 5 + \gamma^2 V(s''). \end{aligned}$$

So from all numbered states we move off the numbers onto blank states, and from non-numbered states we move onto numbered states to collect the reward of 10. Note that the reward is only achieved for *strictly* decreasing the distance to the nearest numbered square, so moving back and forth between numbered squares (or squares with $d(s) = 1$) does not get the reward and is thus not optimal.

(c) [4 pts] Consider a different artificial reward that also favors moving toward numbered squares in a slightly different way:

$$F_2(s,a,s') = \begin{cases} 0 & s' \text{ is a terminal state,} \\ 10\left(d(s) - \frac{1}{2}d(s')\right) & \text{otherwise.} \end{cases}$$

Fill in the diagram on the right as in (a) to indicate the optimal policy PacBot will calculate with the discount factor $\gamma = 0.5$ and the modified reward function $R_2'(s,a,s') = R(s,a,s') + F_2(s,a,s')$ in the board on the left.



Here we will never get into a cycle because for every reward we could gain by stepping in one direction, that reward is lost when we step back the other way. It can be shown that the sum of F_2 rewards along a path from a numbered square to another numbered square all cancel to 0. In fact, along any (multi-step) path from state s to state s', the sum of F_2 rewards along that path will sum to $10\left(d(s) - \frac{1}{2}d(s')\right)$. This means that the F_2 rewards do not favor one path over another as long as they both lead to exiting, and they favor exiting eventually over never exiting. Thus we will have a policy that never gets stuck in a cycle and chooses which exit to go to in exactly the same way as in (a). Explicitly computing the Q-values by hand will also show that the optimal policy is the same as that in (a).

Q6. [6 pts] Zero Sum MDP's

Consider a Markov Decision Process where it is not just Pacman in the environment, but there is also a ghost. Pacman plays one turn, then the ghost plays one turn and they continue alternating, each of their actions transitioning the state forward using the same transition function T. At any one time step, only one of Pacman and ghost can play a turn. Let A be Pacman's action set can take and B be the ghost's action set. The game is infinite horizon, with discount factor γ applied at every turn no matter which agent is taking the turn. $|A|$ is the size of A's action set and $|B|$ is the size of B's action set. R indicates the utility received by Pacman.

(a) [2 pts] Let us first consider the situation where Pacman tries to maximize his expected utility, while the ghost tries to minimize Pacman's utility, thus playing adversarially. Both Pacman and the ghost try to play optimally and they are aware of this. Given the standard notation for an MDP, choose which of the following updates is the correct one for Q-Value Iteration under this formulation, given that Q^*_{pac} is the infinite horizon Q-function for Pacman.

○ $Q^*_{pac}(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \sum_{b\in B} \sum_{s''}(T(s',b,s'')[R(s',b,s'') + \gamma \max_{a'\in A} Q^*_{pac}(s'',a')])]$
○ $Q^*_{pac}(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_{a'\in A} Q^*_{pac}(s',a')]$
○ $Q^*_{pac}(s,a) = \sum_{s'} R(s,a,s') + \gamma \max_{a'\in A} Q^*_{pac}(s',a')$
○ $Q^*_{pac}(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \sum_{b\in B} \sum_{s''}(T(s',b,s'')[R(s',b,s'') + \gamma \frac{1}{|B|} \max_{a'\in A} Q^*_{pac}(s'',a')])]$
● $Q^*_{pac}(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \min_{b\in B} \sum_{s''}(T(s',b,s'')[R(s',b,s'') + \gamma \max_{a'\in A} Q^*_{pac}(s'',a')])]$
○ $Q^*_{pac}(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_{b\in B} \sum_{s''}(T(s',b,s'')[R(s',b,s'') + \gamma \max_{a'\in A} Q^*_{pac}(s'',a')])]$
○ None of the above.

You can project this game by imagining that the ghost takes the next move, and instead of maximizing utility, the ghost is minimizing utility. The ghost's optimal value is again decided by imagining you playing optimally, which leads to the inner maximization of the Q function.

(b) [2 pts] For this part, let us suppose that instead of having a ghost which is adversarial, the ghost is a friendly ghost who is also trying to maximize Pacman's utility. Both Pacman and the ghost know this arrangement, and are aware of the others knowledge. Given the standard notation for an MDP, choose which of the following updates is the correct one for Q-Value Iteration under this formulation, given that Q^*_{pac} is the Q-function for Pacman.

○ $Q^*_{pac}(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \min_{b\in B} Q^*_{pac}(s',b)]$
○ $Q^*_{pac}(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \sum_{b\in B} \sum_{s''}(T(s',b,s'')[R(s',b,s'') + \gamma \max_{a'\in A} Q^*_{pac}(s'',a')])]$
○ $Q^*_{pac}(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \frac{1}{|B|} \sum_{b\in B} \sum_{s''}(T(s',b,s'')[R(s',b,s'') + \gamma \max_{a'\in A} Q^*_{pac}(s'',a')])]$
● $Q^*_{pac}(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_{b\in B} \sum_{s''}(T(s',b,s'')[R(s',b,s'') + \gamma \max_{a'\in A} Q^*_{pac}(s'',a')])]$
○ $Q^*_{pac}(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \frac{1}{|B|} \max_{b\in B} Q^*_{pac}(s',b)]$
○ $Q^*_{pac}(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \min_{b\in B} \sum_{s''}(T(s',b,s'')[R(s',b,s'') + \gamma \max_{a'\in A} Q^*_{pac}(s'',a')])]$
○ None of the above.

You can project this game by imagining that the ghost takes the next move, and the ghost is maximizing your utility. The ghost's optimal value is again decided by imagining you playing optimally, which leads to the inner maximization of the Q function, along with the outer maximization where the ghost is trying to help you.

(c) [2 pts] For this part let us suppose that instead of having a ghost which is friendly, the ghost is a confused ghost who takes random actions, with uniform probability in the environment. Given the standard notation for an MDP, choose which of the following updates is the correct one for Q-Value Iteration under this formulation, given that Q_{pac} is the Q-function for Pacman.

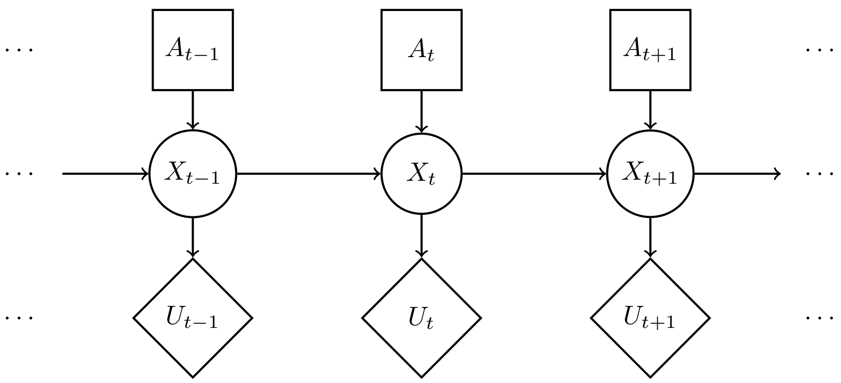
○ $Q^*_{pac}(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \frac{1}{|B|} \max_{b\in B} \sum_{s''}(T(s',b,s'')[R(s',b,s'') + \gamma \max_{a'\in A} Q^*_{pac}(s'',a')])]$
○ $Q^*_{pac}(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \sum_{b\in B} \sum_{s''}(T(s',b,s'')[R(s',b,s'') + \gamma \max_{a'\in A} Q^*_{pac}(s'',a')])]$
○ $Q^*_{pac}(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_{b\in B} Q^*_{pac}(s',b)]$
● $Q^*_{pac}(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \frac{1}{|B|} \sum_{b\in B} \sum_{s''}(T(s',b,s'')[R(s',b,s'') + \gamma \max_{a'\in A} Q^*_{pac}(s'',a')])]$
○ $Q^*_{pac}(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \frac{1}{|B|} \min_{b\in B} \sum_{s''}(T(s',b,s'')[R(s',b,s'') + \gamma \max_{a'\in A} Q^*_{pac}(s'',a')])]$

○ $Q^*_{pac}(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_{a' \in A} Q^*_{pac}(s',a')]$
○ None of the above.

You can project this game by imagining that the ghost takes the next move, and the ghost is playing randomly, which means that you can take an expectation over all possible actions that the ghost could play. The ghost's optimal value is decided by imagining you playing optimally, which leads to the inner maximization of the Q function, along with the outer expectation where we average over all random actions that the ghost could take.

Q7. [11 pts] Planning ahead with HMMs

Pacman is tired of using HMMs to estimate the location of ghosts. He wants to use HMMs to plan what actions to take in order to maximize his utility. Pacman uses the HMM (drawn to the right) of length T to model the planning problem. In the HMM, $X_{1:T}$ is the sequence of hidden states of Pacman's world, $A_{1:T}$ are actions Pacman can take, and U_t is the utility Pacman receives at the particular hidden state X_t. Notice that there are no evidence variables, and utilities are not discounted.



(a) The belief at time t is defined as $B_t(X_t) = p(X_t|a_{1:t})$. The forward algorithm update has the following form:

$$B_t(X_t) = \underline{\quad\textbf{(i)}\quad} \; \underline{\quad\textbf{(ii)}\quad} \; B_{t-1}(x_{t-1}).$$

Complete the expression by choosing the option that fills in each blank.

(i) [1 pt] ● $\sum_{x_{t-1}}$ ○ $\max_{x_{t-1}}$ ○ $\max_{x_t}$ ○ $\sum_{x_t}$ ○ 1

(ii) [1 pt] ○ $p(X_t|x_{t-1})p(X_t|a_t)$ ○ $p(X_t|x_{t-1})$ ○ $p(X_t)$ ● $p(X_t|x_{t-1}, a_t)$ ○ 1

○ None of the above combinations is correct

$$\begin{aligned} B_t(X_t) &= p(X_t|a_{1:t}) \\ &= \sum_{x_{t-1}} p(X_t|x_{t-1}, a_t)p(x_{t-1}|a_{1:t-1}) \\ &= \sum_{x_{t-1}} p(X_t|x_{t-1}, a_t)B_{t-1}(x_{t-1}) \end{aligned}$$

(b) Pacman would like to take actions $A_{1:T}$ that maximizes the expected sum of utilities, which has the following form:

$$\text{MEU}_{1:T} = \underline{\;\textbf{(i)}\;} \; \underline{\;\textbf{(ii)}\;} \; \underline{\;\textbf{(iii)}\;} \; \underline{\;\textbf{(iv)}\;} \; \underline{\;\textbf{(v)}\;}$$

Complete the expression by choosing the option that fills in each blank.

(i) [1 pt] ○ $\max_{a_T}$ ● $\max_{a_{1:T}}$ ○ $\sum_{a_{1:T}}$ ○ $\sum_{a_T}$ ○ 1

(ii) [1 pt] ○ $\prod_{t=1}^{T}$ ○ $\max_t$ ○ $\min_t$ ● $\sum_{t=1}^{T}$ ○ 1

(iii) [1 pt] ● $\sum_{x_t}$ ○ $\sum_{x_t, a_t}$ ○ $\sum_{a_t}$ ○ $\sum_{x_T}$ ○ 1

(iv) [1 pt] ○ $p(x_t)$ ○ $p(x_t|x_{t-1}, a_t)$ ○ $B_T(x_T)$ ● $B_t(x_t)$ ○ 1

(v) [1 pt] ○ $\frac{1}{U_t}$ ○ U_T ● U_t ○ $\frac{1}{U_T}$ ○ 1

○ None of the above combinations is correct

$$\text{MEU}_{1:T} = \max_{a_{1:T}} \sum_{t=1}^{T} \sum_{x_t} B_t(x_t)U_t(x_t)$$

(c) [2 pts] A greedy ghost now offers to tell Pacman the values of some of the hidden states. Pacman needs your help to figure out if the ghost's information is useful. Assume that the transition function $p(x_t|x_{t-1}, a_t)$ is not deterministic. **With respect to the utility** U_t, mark all that can be True:

■ VPI($X_{t-1}|X_{t-2}$) > 0 □ VPI($X_{t-2}|X_{t-1}$) > 0 ■ VPI($X_{t-1}|X_{t-2}$) $= 0$ ■ VPI($X_{t-2}|X_{t-1}$) $= 0$
□ None of the above

It is always possible that VPI $= 0$. Can guarantee VPI($E|e$) is not greater than 0 if E is independent of parents(U) given e.

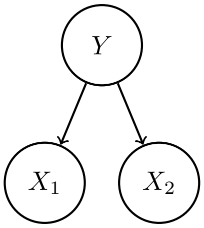
(d) [2 pts] Pacman notices that calculating the beliefs under this model is very slow using exact inference. He therefore decides to try out various particle filter methods to speed up inference. Order the following methods by how accurate their estimate of $B_T(X_T)$ is? If different methods give an equivalently accurate estimate, mark them as the same number.

	Most accurate			Least accurate
Exact inference	● 1	○ 2	○ 3	○ 4
Particle filtering with no resampling	○ 1	● 2	○ 3	○ 4
Particle filtering with resampling before every time elapse	○ 1	○ 2	○ 3	● 4
Particle filtering with resampling before every other time elapse	○ 1	○ 2	● 3	○ 4

Exact inference will always be more accurate than using a particle filter. When comparing the particle filter resampling approaches, notice that because there are no observations, each particle will have weight 1. Therefore resampling when particle weights are 1 could lead to particles being lost and hence prove bad.

Q8. [7 pts] Naïve Bayes

You are given a naïve bayes model, shown below, with label Y and features X_1 and X_2. The conditional probabilities for the model are parametrized by p_1, p_2 and q.



X_1	Y	$P(X_1\|Y)$
0	0	p_1
1	0	$1-p_1$
0	1	$1-p_1$
1	1	p_1

X_2	Y	$P(X_2\|Y)$
0	0	p_2
1	0	$1-p_2$
0	1	$1-p_2$
1	1	p_2

Y	$P(Y)$
0	$1-q$
1	q

Note that some of the parameters are shared (e.g. $P(X_1 = 0|Y = 0) = P(X_1 = 1|Y = 1) = p_1$).

(a) [2 pts] Given a new data point with $X_1 = 1$ and $X_2 = 1$, what is the probability that this point has label $Y = 1$? Express your answer in terms of the parameters p_1, p_2 and q (you might not need all of them).

$P(Y = 1|X_1 = 1, X_2 = 1) =$ $\frac{p_1p_2q}{p_1p_2q+(1-p_1)(1-p_2)(1-q)}$

$$\begin{aligned}
P(Y=1, X_1=1, X_2=1) &= P(X_1=1|Y=1)P(X_2=1|Y=1)P(Y=1)\\
&= p_1p_2q\\
P(Y=0, X_1=1, X_2=1) &= P(X_1=1|Y=0)P(X_2=1|Y=0)P(Y=0)\\
&= (1-p_1)(1-p_2)(1-q)\\
P(Y=1|X_1=1, X_2=1) &= \frac{P(Y=1, X_1=1, X_2=1)}{P(X_1=1, X_2=1)}\\
&= \frac{P(Y=1, X_1=1, X_2=1)}{P(Y=1, X_1=1, X_2=1)+P(Y=0, X_1=1, X_2=1)}\\
&= \frac{p_1p_2q}{p_1p_2q+(1-p_1)(1-p_2)(1-q)}
\end{aligned}$$

The model is trained with the following data:

sample number	1	2	3	4	5	6	7	8	9	10
X_1	0	0	1	0	1	0	1	0	1	1
X_2	0	0	0	0	0	0	0	1	0	0
Y	0	0	0	0	0	0	0	1	1	1

(b) [5 pts] What are the maximum likelihood estimates for p_1, p_2 and q?

$p_1 =$ $\frac{3}{5}$ $\qquad p_2 =$ $\frac{4}{5}$ $\qquad q =$ $\frac{3}{10}$

The maximum likelihood estimate of p_1 is the fraction of counts of samples in which $X_1 = Y$. In the given training data, samples $1, 2, 4$ and 6 have $X_1 = Y = 0$ and samples 9 and 10 have $X_1 = Y = 1$, so 6 out of the 10 samples have $X_1 = Y$ and thus $p_1 = \frac{6}{10} = \frac{3}{5}$. Analogously, 8 out of the 10 samples have $X_2 = Y$ and thus $p_2 = \frac{8}{10} = \frac{4}{5}$. The maximum likelihood estimate of q is the fraction of counts of samples in which $Y = 1$, thus $q = \frac{3}{10}$.

You can find what these parameters are equal to by maximizing the likelihood of the data with respect to the parameters. First, notice that the probabilities can be written as

$$\begin{aligned} P(X_1 = x_1|Y = y) &= (p_1)^{\mathbb{1}[x_1=y]}(1-p_1)^{(1-\mathbb{1}[x_1=y])} \\ P(X_2 = x_2|Y = y) &= (p_2)^{\mathbb{1}[x_2=y]}(1-p_2)^{(1-\mathbb{1}[x_2=y])} \\ P(Y = y) &= (q)^{y}(1-q)^{(1-y)}, \end{aligned}$$

where $\mathbb{1}[x = y]$ is an indicator function that evaluates to 1 when x is equal to y, and 0 otherwise. Let X be all the data: $x_1^{(i)}, x_2^{(i)}, y^{(i)}, \forall i = 1, \ldots, 10$, where the superscripts i denote the sample number. Then, the likelihood of the data given the parameters is

$$\begin{aligned} l(X|p_1, p_2, q) &= P(X|p_1, p_2, q) \\ &= \prod_{i=1}^{10} P(x_1^{(i)}, x_2^{(i)}, y^{(i)}) \\ &= \prod_{i=1}^{10} P(x_1^{(i)}|y^{(i)})P(x_2^{(i)}|y^{(i)})P(y^{(i)}) \\ &= \prod_{i=1}^{10} (p_1)^{\mathbb{1}[x_1^{(i)}=y^{(i)}]}(1-p_1)^{(1-\mathbb{1}[x_1^{(i)}=y^{(i)}])}(p_2)^{\mathbb{1}[x_2^{(i)}=y^{(i)}]}(1-p_2)^{(1-\mathbb{1}[x_2^{(i)}=y^{(i)}])}(q)^{y^{(i)}}(1-q)^{(1-y^{(i)})}. \end{aligned}$$

We optimize for the parameters by maximizing the likelihood $l(X|p_1, p_2, q)$, which is equivalent to maximizing the log likelihood $ll(X|p_1, p_2, q)$,

$$\begin{aligned} p_1, p_2, q &= \arg\max_{p_1,p_2,q} l(X|p_1, p_2, q) \\ &= \arg\max_{p_1,p_2,q} ll(X|p_1, p_2, q), \end{aligned}$$

where

$$\begin{aligned} ll(X|p_1, p_2, q) &= \log l(X|p_1, p_2, q) \\ &= \sum_{i=1}^{10} \mathbb{1}[x_1^{(i)} = y^{(i)}]\log(p_1) + (1 - \mathbb{1}[x_1^{(i)} = y^{(i)}])\log(1-p_1) \\ &\quad + \mathbb{1}[x_2^{(i)} = y^{(i)}]\log(p_2) + (1 - \mathbb{1}[x_2^{(i)} = y^{(i)}])\log(1-p_2) \\ &\quad + y^{(i)}\log(q) + (1-y^{(i)})\log(1-q). \end{aligned}$$

We can find the values that obtain the maximum by setting the partial derivates of the log likelihood to zero and solving for the parameters

$$\frac{\partial ll}{\partial p_1} = \sum_{i=1}^{10} \frac{1}{p_1}\mathbb{1}[x_1^{(i)} = y^{(i)}] - \frac{1}{1-p_1}(1 - \mathbb{1}[x_1^{(i)} = y^{(i)}]) = 0 \qquad \Rightarrow p_1 = \frac{\sum_{i=1}^{10} \mathbb{1}[x_1^{(i)} = y^{(i)}]}{10}$$

$$\frac{\partial ll}{\partial p_2} = \sum_{i=1}^{10} \frac{1}{p_2}\mathbb{1}[x_2^{(i)} = y^{(i)}] - \frac{1}{1-p_2}(1 - \mathbb{1}[x_2^{(i)} = y^{(i)}]) = 0 \qquad \Rightarrow p_2 = \frac{\sum_{i=1}^{10} \mathbb{1}[x_2^{(i)} = y^{(i)}]}{10}$$

$$\frac{\partial ll}{\partial q} = \sum_{i=1}^{10} \frac{1}{q}y^{(i)} - \frac{1}{1-q}(1-y^{(i)}) = 0 \qquad \Rightarrow q = \frac{\sum_{i=1}^{10} y^{(i)}}{10}.$$

Q9. [12 pts] Beyond Ordinary Pruning

Important: For all following parts, assume that the children of a node are visited in left-to-right order. You should **not** prune on equality (This also applies to any bound on utilities, if any. For example, given all utilities are less than or equal to 10, you should not prune after seeing a node with utility of 10.)

(a) [3 pts] Consider a two-player game in which both players alternate moves and each player seeks to maximize its own utility. At a leaf node s, utilities are represented as a tuple $U(s) = \big(U_1(s), U_2(s)\big)$, with the i-th component corresponding to the utility of the i-th player.

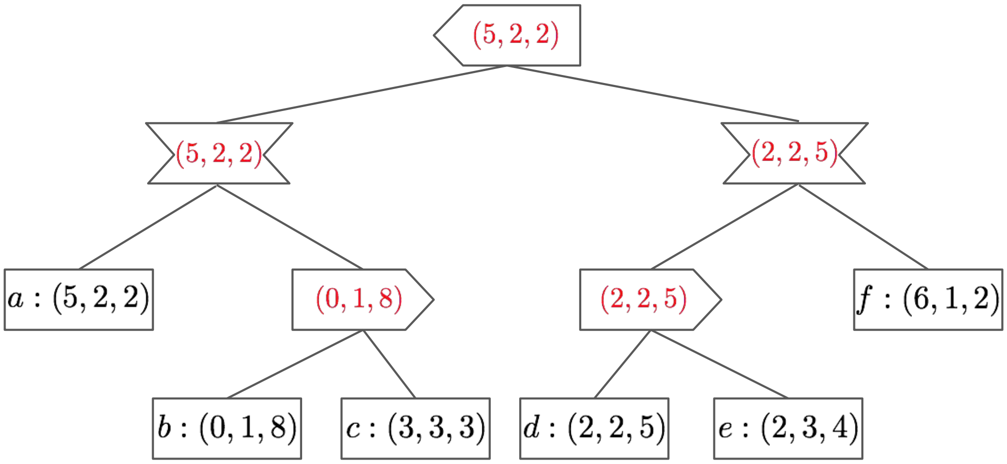
For the following special cases of two-player games, select **all** of the following in which pruning is **never** possible, given **just** this information about the relationship between utilities U_1 and U_2. Select "None of the above" if none of the options apply.

■ $0 < U_1(s), U_2(s) < M$ for all terminal states s, where M is a positive constant
□ $U_1(s) + U_2(s) = M$ for all terminal states s, where $M \neq 0$ is a constant
■ $U_1(s) = U_2(s)$ for all terminal states s
□ $U_1(s) + U_2(s) = 0$ for all terminal states s
□ None of the above

The first option is an interleaving of two independent searches; without the relationship between the two players, no pruning is possible.
The second option is minimax in which all utilities are shifted by M; alpha-beta pruning still applies.
The third option is a search problem since both players share the same objective; since the maximum utility can occur at any terminal node, no pruning is possible.
The fourth option is minimax; alpha-beta pruning applies.

(b) Now we consider a three-player game similarly defined as in part (a). Then at a leaf node s, utilities are represented as a 3-tuple $U(s) = \big(U_1(s), U_2(s), U_3(s)\big)$, where the player going first (at the top of the tree) maximizes U_1, the player going second maximizes U_2, and the player going last maximizes U_3.

(i) [2 pts] Fill in the values at all nodes. Note that all players maximize their own respective utilities.



(ii) [3 pts] Without any further information, select **all** terminal nodes that can be pruned. Or check "None" if no node can be pruned.
Reminder: A node can be pruned only if the node's utilities can have no effect on the utilities at the root, irrespective of the node's utilities, and the utilities of nodes not yet visited by the left-to-right depth-first traversal.

□ a □ b □ c □ d □ e □ f ■ None

Note that without any further assumption, particularly we don't know whether any utility value is bounded. Before visiting any leaf node, we don't know whether that node has $U_3(\cdot) = \infty$ (no pruning on equality). Therefore, no node can be pruned.

(iii) [4 pts] Now we are given that for all terminal states s the following holds true:

- $U_i(s) \geq 0 \quad \forall i = 1, 2, 3$
- $\sum_{i=1}^{3} U_i(s) \leq 9$

Select **all** terminal nodes that can be pruned. Or check "None" if no node can be pruned.

□ a □ b ■ c □ d □ e □ f □ None

Nodes a, b and d cannot be pruned because each is the first child of the corresponding player and before visiting that child, the player does not yet have a bound.

Node c can be pruned. After visiting node a, the second player will only choose the right branch when $\max(U_2(b), U_2(c)) \geq 2$. After visiting node b, the third player will only choose node c when $U_3(c) \geq 8$ and $U_2(c) \geq 2$. But this is not possible given the constraints, so node c will not be chosen no matter what utilities it subsumes and hence can be pruned.

Node e cannot be pruned. To see this, assume node e subsumes a different utility tuple, for example, $(3, 0, 6)$. Now, on the right branch of the root, the third player will choose node e and subsumes $(3, 0, 6)$, the second player will choose node f and subsumes $(6, 1, 2)$, and the first player (the root) will choose the right branch and subsume $(6, 1, 2)$. Since the utilities at node e may affect the utilities at the root, node e cannot be pruned.

Node f cannot be pruned. Assume node f subsumes utility tuple $(6, 3, 0)$. Then the first player (the root) will subsume $(6, 3, 0)$ after the propagation.

Q10. [10 pts] Iterative Deepening Search

Pacman is performing search in a maze again! The search graph has a branching factor of b, a solution of depth d, a maximum depth of m, and edge costs that may not be integers. Although he knows breadth first search returns the solution with the smallest depth, it takes up too much space, so he decides to try using iterative deepening. As a reminder, in standard depth-first iterative deepening we start by performing a depth first search terminated at a maximum depth of one. If no solution is found, we start over and perform a depth first search to depth two and so on. This way we obtain the shallowest solution, but use only O(bd) space.

But Pacman decides to use a variant of iterative deepening called **iterative deepening A***, where instead of limiting the depth-first search by depth as in standard iterative deepening search, we can limit the depth-first search by the f value as defined in A* search. As a reminder $f[node] = g[node] + h[node]$ where $g[node]$ is the cost of the path from the start state and $h[node]$ is a heuristic value estimating the cost to the closest goal state.

In this question, all searches are tree searches and **not** graph searches.

(a) [7 pts] Complete the pseudocode outlining how to perform iterative deepening A* by choosing the option from the next page that fills in each of these blanks. Iterative deepening A* should return the solution with the lowest cost when given a consistent heuristic. Note that cutoff is a boolean and new-limit is a number.

```
function Iterative-Deepening-Tree-Search(problem)
    start-node ← Make-Node(Initial-State[problem])
    limit ← f[start-node]
    loop
        fringe ← Make-Stack(start-node)
        new-limit ← $\boxed{\textbf{(i)}}$
        cutoff ← $\boxed{\textbf{(ii)}}$
        while fringe is not empty do
            node ← Remove-Front(fringe)
            if Goal-Test(problem, State[node]) then
                return node
            end if
            for child-node in Expand(State[node], problem) do
                if f[child-node] ≤ limit then
                    fringe ← Insert(child-node, fringe)
                    new-limit ← $\boxed{\textbf{(iii)}}$
                    cutoff ← $\boxed{\textbf{(iv)}}$
                else
                    new-limit ← $\boxed{\textbf{(v)}}$
                    cutoff ← $\boxed{\textbf{(vi)}}$
                end if
            end for
        end while
        if not cutoff then
            return failure
        end if
        limit ← $\boxed{\textbf{(vii)}}$
    end loop
end function
```

$\mathbf{A_1}$	$-\infty$	$\mathbf{A_2}$	0	$\mathbf{A_3}$	∞	$\mathbf{A_4}$	*limit*
$\mathbf{B_1}$	True	$\mathbf{B_2}$	False	$\mathbf{B_3}$	*cutoff*	$\mathbf{B_4}$	not *cutoff*
$\mathbf{C_1}$	*new-limit*	$\mathbf{C_2}$	*new-limit* + 1	$\mathbf{C_3}$	*new-limit* + f[*node*]	$\mathbf{C_4}$	*new-limit* + f[*child-node*]
$\mathbf{C_5}$	MIN(*new-limit*, f[*node*])	$\mathbf{C_6}$	MIN(*new-limit*, f[*child-node*])	$\mathbf{C_7}$	MAX(*new-limit*, f[*node*])	$\mathbf{C_8}$	MAX(*new-limit*, f[*child-node*])

(i) [1 pt] ○ $\mathbf{A_1}$ ○ $\mathbf{A_2}$ ● $\mathbf{A_3}$ ○ $\mathbf{A_4}$

(ii) [1 pt] ○ $\mathbf{B_1}$ ● $\mathbf{B_2}$ ○ $\mathbf{B_3}$ ○ $\mathbf{B_4}$

(iii) [1 pt] ● $\mathbf{C_1}$ ○ $\mathbf{C_2}$ ○ $\mathbf{C_3}$ ○ $\mathbf{C_4}$
○ $\mathbf{C_5}$ ○ $\mathbf{C_6}$ ● $\mathbf{C_7}$ ● $\mathbf{C_8}$

(iv) [1 pt] ○ $\mathbf{B_1}$ ○ $\mathbf{B_2}$ ● $\mathbf{B_3}$ ○ $\mathbf{B_4}$

(v) [1 pt] ○ $\mathbf{C_1}$ ○ $\mathbf{C_2}$ ○ $\mathbf{C_3}$ ○ $\mathbf{C_4}$
○ $\mathbf{C_5}$ ● $\mathbf{C_6}$ ○ $\mathbf{C_7}$ ○ $\mathbf{C_8}$

(vi) [1 pt] ● $\mathbf{B_1}$ ○ $\mathbf{B_2}$ ○ $\mathbf{B_3}$ ○ $\mathbf{B_4}$

(vii) [1 pt] ● $\mathbf{C_1}$ ○ $\mathbf{C_2}$ ○ $\mathbf{C_3}$ ○ $\mathbf{C_4}$
○ $\mathbf{C_5}$ ○ $\mathbf{C_6}$ ○ $\mathbf{C_7}$ ○ $\mathbf{C_8}$

The cutoff variable keeps track of whether there are items that aren't being explored because of the limit. If cutoff is false and the algorithm has exited the while, no nodes were cutoff (not added to the fringe because of the limit). This scenario suggests that there is no solution.

In order to ensure that iterative deepening A* obtains the lowest cost solution efficiently, we want to increase the limit as much as we can while guaranteeing optimality. Setting new-limit to the smallest f cost of nodes that were cutoff achieves this. When nodes aren't cutoff (part iii), the new-limit should not change. Hence C1, C7, C8, or a combination of the three were accepted as answers.

(b) [3 pts] Assuming there are no ties in f value between nodes, which of the following statements about the number of nodes that iterative deepening A* expands is True? If the same node is expanded multiple times, count all of the times that it is expanded. If none of the options are correct, mark None of the above.

● The number of times that iterative deepening A* expands a node is greater than or equal to the number of times A* will expand a node.

○ The number of times that iterative deepening A* expands a node is less than or equal to the number of times A* will expand a node.

○ We don't know if the number of times iterative deepening A* expands a node is more or less than the number of times A* will expand a node.

○ None of the above

Iterative deepening A* runs depth first search multiples at different limit values. This causes iterative deepening A* to expand certain nodes multiple times.

THIS PAGE IS INTENTIONALLY LEFT BLANK