CS 188 Spring 2020 Introduction to Artificial Intelligence Final

- You have approximately 170 minutes.
- The exam is closed book, closed calculator, and closed notes except your two-page crib sheet.
- Mark your answers on gradescope, and click "submit".
- For multiple choice questions,
  - ☐ means mark **all options** that apply
  - ◯ means mark a **single choice**

First name	
Last name	
SID	

Your Discussion TA(s) (fill all that apply):

☐ Ajan ☐ Albert ☐ Amitav ☐ Angela ☐ Anusha ☐ Arin
☐ Benson ☐ Carl ☐ Cathy ☐ Charles ☐ Harry (Huazhe) ☐ Jade
☐ Jasmine ☐ Jeffrey ☐ Jierui ☐ Lindsay ☐ Mesut ☐ Pravin
☐ Rachel ☐ Ryan ☐ Saagar ☐ Yanlai

For staff use only:

Q1.	Potpourri	/17
Q2.	Learn From Old Data	/6
Q3.	Value of Perfect Information	/11
Q4.	Languages	/20
Q5.	Pacman Loses Control	/13
Q6.	A Nonconvolutional Nontrivial Network	/9
Q7.	Searching for a Bayes Network	/13
Q8.	Particle Madness	/11
	Total	/100

THIS PAGE IS INTENTIONALLY LEFT BLANK

SID: _____

Q1. [17 pts] Potpourri

(a) Recall that in search algorithms, that nodes are expanded and each node corresponds to a state s and a path from the start state to s. Recall further that in graph search one can only expand a node corresponding to a state once. Also, below we will denote a heuristic for A^* by $h(\cdot)$. Assume all state transitions have a positive cost.

(i) [1 pt] Which search algorithm would typically have a bigger fringe in graph search for large search problems?

● BFS ○ DFS

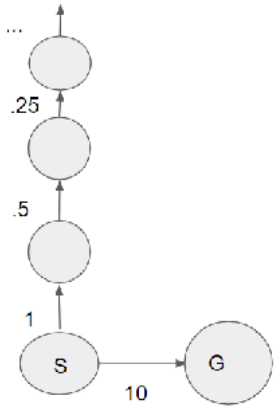
BFS scales exponentially with the size of the path while DFS scales linearly.

(ii) [1 pt] Which search algorithms are complete for tree search?

■ BFS ☐ DFS ■ A^* with any heuristic ■ A^* with a consistent heuristic. ■ Uniform Cost Search

BFS,A*, and UCS are all complete because they explore nodes in the order of increasing distance away from start node, preventing the from getting stuck in cycles. Note, even with a really bad (or even negative heuristic) the positive backwards cost for nodes in A* will eventually accumulate and it will eventually be forced to explore a path to the goal node if it exists.

Special Note, this question was intended for finite state spaces, but with infinite graphs we can get different results. In the example below, your fringe will always prefer to go upwards since the backwards cost for all of those nodes is less than 2. Thus the goal will never be explored and UCS is not complete. UCS is equivalent to A* with the trivial (h = 0) heuristic which is also admissible and consistent. Thus A* with any heuristic and A* with consistent heuristic are also not complete, only BFS is. So if one marked BFS only they were also given full credit.



(iii) [1 pt] Which search algorithms are optimal for graph search?

☐ BFS ☐ DFS ☐ A* with any heuristic ■ A* with a consistent heuristic. ■ Uniform Cost Search

Only UCS and A* even take into account edge weight distance and A* requires a heuristic to at least be admissible to guarantee optimality.

(iv) [1 pt]

If for a constant c and all states s, $h(s) = c$, A^* is equivalent to UCS (uniform cost search).

● True ○ False]

If every node has the same constant added to its backwards cost, when comparing nodes in the fringe node state will look relatively the same as if the constant wasn't added (which is UCS since A* is just UCS with no heuristic).

(v) [1 pt]

If for any pair of states, s and s', $h(s) - h(s') \leq c(s, s')$ where $c(s, s')$ is the minimum cost path from s to s', then A^* will expand a node at most once in a tree search.

● True ○ False

This is the definition of consistency which guarantees that a node will only be explored in A* if it is by the optimal path to that node.

(vi) [1 pt]

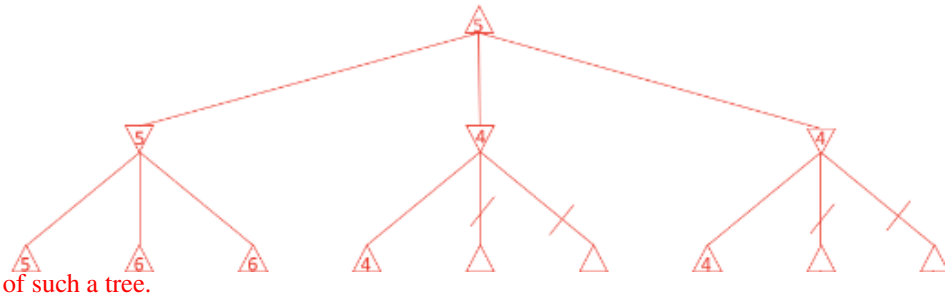
In A^* tree search with any heuristic, one may not get the optimal path.

● True ○ False

The heuristic needs to be admissible to ensure optimality.

(vii) [1 pt] In a minimax tree of depth 2 (one max layer, one min layer, and a leaf layer) with a branching factor of 3, what is the maximum number of nodes that can be pruned by alpha-beta pruning?

○ 0 ○ 2 ○ 3 ● 4 ○ 6 ○ 9 ○ None of the above



This is an example of such a tree.

(viii) [1 pt]

In full depth minimax search, with $\alpha - \beta$ pruning, the minimum number of leaves that can be explored is that one can do in a depth-D tree (the depth is the total depth considering both players) with branching factor B (both players have B options) is

○ $(B/2)^D$ ○ $D^{B/2}$ ○ B^D ○ D^B ● $B^{D/2}$

Normally there are B^D leaf nodes. In Alpha beta pruning we have to look at all the first player's moves but we only have to look at one of the second players moves. Thus we "skip" every other levels and only have to look at $B^{D/2}$.

(ix) [1 pt]

In expectimax search with two players, one max and the other chance, one can use pruning to reduce the search cost.

○ True ● False

Pruning is about exploiting your knowledge of how both players will choose actions to look at less nodes. Since in expectimax one player will be picking randomly we have to look at every one of their actions, which will affect the expected value and the value of the node above it.

(b) Suppose we are solving a CSP that has 20 variables (X_i for $i = 1, 2, ..., 20$) and all constraints are binary. X_1 is involved in binary constraints with 6 other variables. X_2 is involved in binary constraints with 9 other variables.
While running the arc consistency algorithm, we reach a point when all variables have 4 values left in their domains, and we have one last arc in the queue: $X_1 \longrightarrow X_2$. (Recall a value x for a variable X is consistent with an arc $X \longrightarrow Y$ if there is still a value y for variable Y where (x, y) is valid for the binary constraint on X and Y.)

(i) [1 pt] Now, we are processing the arc $X_1 \longrightarrow X_2$. We are able to remove a value from the domain of a variable because it was inconsistent, and add the necessary arcs into the queue.
How many arcs are in the queue now?

○ 3 ○ 4 ○ 5 ● 6 ○ 9 ○ 18
○ None of these ○ We cannot determine the exact number of arcs

We remove values of the variable on the tail, so X_1 loses a value. This means we have to add every possible arc that points into X_1. There are 6 of those, one for each constraint that involves X_1.

(ii) [1 pt] Following the previous part, we processed any arcs that may have been added to the queue. No more values were removed from any variable. As a result, we plan to assign a value to one of the variables to continue with backtracking search. Pick the statement below that is most valid.

○ We should assign a value to X_1 because it has the Least Constraining Value
● We should assign a value to X_1 because it has the Minimum Remaining Values
○ We should assign a value to X_2 because it has the Least Constraining Value
○ We should assign a value to X_2 because it has the Minimum Remaining Values
○ We should assign a value to some X_i ($i \geq 3$) that has the Least Constraining Value
○ We should assign a value to some X_i ($i \geq 3$) that has the Minimum Remaining Values

We choose variables based on MRV not LCV. X_1 has the fewest number of values remaining as it lost a value in part (i) and no other variable lost a value.

(c) We are given the traditional game of Pacman with food and 2 ghosts on an M by N grid. Pacman can move up, left, right, or down each turn and he wins by eating all the food. Whenever Pacman moves, the ghosts will move in the opposite direction.

(i) [1 pt] **State representation 1** is to keep track of Pacman's location, the location of both ghosts, and which food was eaten. Please select all of the terms below that should be multiplied together as one product to correctly quantify the

number of states in this representation.
☐ 3^{MN} ■ MN ■ 2^{MN} ■ $(MN)^2$
There are MN possible locations for pacman, $(MN)^2$ possible location combos for both ghosts, and 2^{MN} possible values of the MN booleans we need to keep track of whether each food was eaten.

(ii) [1 pt] **State representation 2** is to keep track of whether each food was eaten, and what is in each location (whether it contains Pacman, a Ghost, or nothing). Please select all of the terms below that should be multiplied together as one product to correctly quantify the number of states in this representation.
■ 3^{MN} ☐ MN ■ 2^{MN} ☐ $(MN)^2$
There are still $2^{M}N$ possible food configurations and there are $3^{M}N$ possible configurations to keep track of what is in each location(there are 3 options for each of MN locations).

(iii) [1 pt] Which state representation would have a higher branching factor?
○ **State representation 1** will have a higher branching factor ○ **State representation 2** will have a higher branching factor ● They will have the same branching factor
Pacman has 4 actions available regardless of the state representation so they will have the same branching factor.

(iv) [1 pt] If we were to run DFS on a particular starting game using both state representations which representation will expand more states?
○ **State representation 1** will expand more states ○ **State representation 2** will expand more states
● They will expand the same amount of states
Since the starting location are the same and the actions available are the same, the successors for each representation will be the same. Thus at each step in DFS we would be presented with the same options, so DFS will run identically on both representations. Note, **State representation 2** has way more possible states but most of them are unreachable by the games rule so they will not affect DFS.

(d) [1 pt] Which of the following are true statements about perceptrons?
☐ A larger magnitude of activation corresponds to higher certainty.
☐ The main purpose of the bias term is to restrict the activation values to fall in our desired range.
■ We can use a black-box binary perceptron to build a multi-class perceptron.
○ None of the above
A) Perceptrons are only trained to label samples not give confidence bounds. B) The bias term is used to shift the boundary so that it's not restricted to stay at the origin. C) A binary perceptron can be used for each class to build a multiclass perceptron. To classify a new training sample, we will select one of the classes that the binary perceptrons classifies it as since we don't know the exact activation.

(e) [1 pt] You are training a logistic regression model and you find that your training loss is near 0 but test loss is very high. Which of the following is expected to help to reduce test loss? Select all that apply.
■ (A) Increase the training data size.
☐ (B) Decrease the training data size.
☐ (C) Increase model complexity.
■ (D) Decrease model complexity.
■ (E) Train on a combination of your training data and your test data but you test only on your test data
☐ (F) Conclude that Machine Learning does not work
Since we are facing overfitting we can increase the training data size or decrease the model complexity to combat this. Also, if we train on our test data our test loss will definitely improve dramatically (although you should NEVER do this in practice because it defeats the purpose of testing)

Q2. [6 pts] Learn From Old Data

You encounter a generic MDP and decide to play the game, performing random actions at every timestep. You record all of your transitions in the form (s, a, s', r), where s is the orginal state, a is the action taken, s' is the state you reached, and r is the reward you earned from that transition. After collecting a large amount of these data points you decide reinforcement learning some things about this MDP.

(a) [1 pt] You want to estimate a function $Q(s, a)$ which returns the total rewards you will earn in expectation if you start in state s, take action a, and act optimally from then on. You will do this by initializing $Q(s, a)$ to 0 for every state and action pair. Then you will examine your data points one at a time, updating your approximation after every data point. Assign each of the following terms into an entry of the equation below, which indicates how to use **a collected data point** (s, a, s', r) to update your $Q(s, a)$ function. Fill in the blanks with an integer from 1 to 5 corresponding to the correct expression. Note, α is the learning rate.

1. r
2. α
3. $Q_{\text{curr}}(s', a')$
4. $1 - \alpha$
5. $\max_{a'}$

$Q_{\text{updated}}(s, a) \leftarrow$ [A= 4] $Q_{\text{curr}}(s, a) +$ [B= 2] [[C= 1] $+ \gamma$ [D= 5] [E= 3]]

Recall the formula for Q-learning:

$$Q_{updated}(s, a) \leftarrow (1 - \alpha) Q_{curr}(s, a) + \alpha \cdot sample$$
$$\leftarrow (1 - \alpha) Q_{curr}(s, a) + \alpha \cdot [r + \gamma \max_{a'} Q_{curr}(s', a')]$$

(b) After you finish examining your data in the order you collected it and updating the $Q(s, a)$ function accordingly you realize you have converged to the true optimal and accurate $Q(s, a)$ function. However, you have an accident and lose all of your memory about that $Q(s, a)$ function. Even worse, you also randomly shuffled the order of your data points and have no way of knowing what the original order was.

(i) [1 pt] Can you use your procedure from part (a) on the shuffled data to recreate the optimal $Q(s, a)$ given that you only examine each data point once?

○ Yes ● No

We are not guaranteed in Q-learning to converge to the optimal Q function after only making a single pass on our data (s, a, s', r) because of the fact way we use a max over the current approximation of $Q(s, a)$ so the order we do the updates in matters.

(ii) [1 pt] Can you use your procedure from part (a) on the shuffled data to recreate the optimal $Q(s, a)$ given that you are allowed to run the procedure as much as you want (You are allowed to examine the same data points multiple times)?

● Yes ○ No

Q learning convergence requires us to look at all the (s, a, s', r) and iterate infinitely many times over that data with sufficient exploration before we converge to the optimal Q-learning function. Since there was sufficient exploration in the old ordering there will be sufficient exploration in the data overall.

(c) Later on, you decide to try to approximate a different function, $V^*(s)$ which returns the total rewards you will earn in expectation if you start in state s and act optimally from then on. You want to do this by initializing $V^*(S)$ to 0 for every state. Then you will examine your data points one at a time, updating your approximation after every data point.

(i) [1 pt] Assign each of the following terms into an entry of the equation below, which indicates how to use **a collected data point** (s, a, s', r) to update your $V^*(s)$ function. Note, α is the learning rate.

1. r

2. α
3. $V_{\text{curr}}(s')$
4. $1-\alpha$

$V_{\text{updated}} \leftarrow$ [A= 4] $V_{\text{curr}}(s)+$ [B= 2] [[C= 1] $+\gamma$ [D= 3]]

We can use TD-learning.

$$V_{updated}(s) \leftarrow (1-\alpha)V_{curr}(s) + \alpha(sample)$$
$$\leftarrow (1-\alpha)V_{curr}(s) + \alpha[r + \gamma V_{curr}(s')]$$

(ii) [2 pts] Can you use your data points to train the true optimal and accurate $V^*(s)$ function?

○ Yes, because the r and s' in the definition is the result of taking action a from state s. This is true for r and s' from any data point (s, a, s', r) collected under any policy (i.e., even an old one)

○ Yes, because the source of the data doesn't matter in RL

○ No, because the old data is useless without the policy

○ No, because the max cannot be calculated after-the-fact

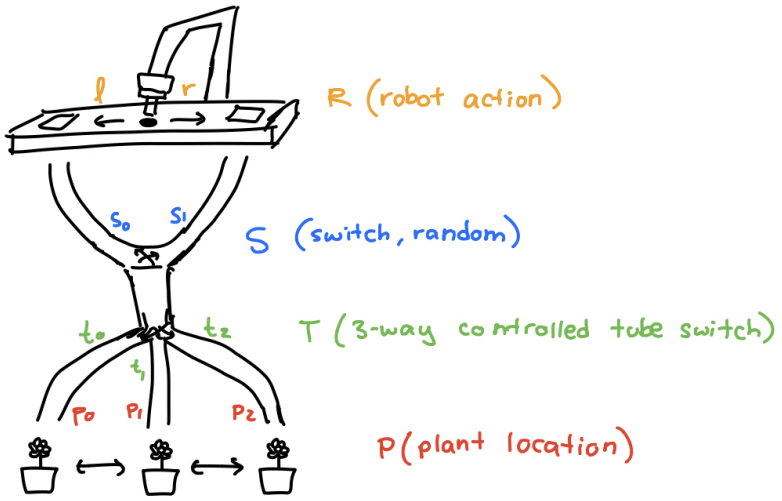
● No, because the r and s' in the definition is the result of taking the optimal action from s. This is only true for r and s' from a data point (s, a, s', r) collected under the most recent/optimal policy

By definition, TD learning only converges to the optimal V^* if we are exploring on the optimal policy π^*. Since we are not guaranteed to receive (s, a, s', r) on the optimal π^*, we cannot necessarily converge to V^*.

Q3. [11 pts] Value of Perfect Information

Consider the setup shown in the figure below, involving a robotic plant-watering system with some mysterious random forces involved. Here, there are 4 main items at play.
(1) The robot (R) can choose to move either left (l) or right (r). Its chosen action pushes a water pellet into the corresponding opening.
(2) The random switch (S) is arbitrarily in one of two possible positions $\{s_0, s_1\}$. When in position (s_0), it accepts a water pellet only from the (l) tube. When in position (s_1), it accepts a water pellet only from the (r) tube.
(3) A controllable three-way switch (T) can be chosen to be placed in one of three possible positions $\{t_0, t_1, t_2\}$.
(4) A plant (P) is arbitrarily located in one of three possible locations $\{p_0, p_1, p_2\}$. When in position p_i, it can only be successfully watered if the corresponding tube t_i has been selected **and** if the water pellet was sent in a direction that was indeed accepted by the first switch (S).

Finally, in this problem, utility (U) is 1 when the plant successfully receives the water pellet, and 0 otherwise.

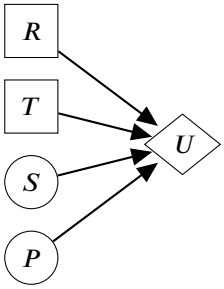


(a) Let's first set this problem up as a decision network.

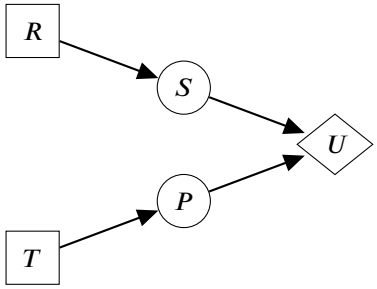
(i) [1 pt] Which of the following decision networks correctly describe the problem described above? Select all that apply. Recall the conventions from the lecture notes:

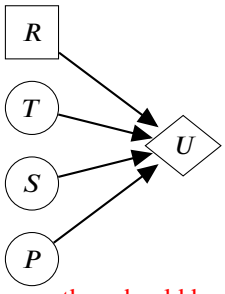
action nodes as rectangles  , chance nodes as ovals  , and utility nodes as diamonds  .

■ (A)

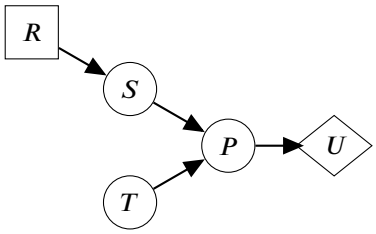


□ (B)





☐ (C)



☐ (D)

R and T are actions you can choose so they should be rectangles, S and P are random outcomes so they should be ovals. All the variables are involved in the calculation of U but they do not influence each other directly, so A has to be the answer.

(ii) [1 pt] Fill in the following probability tables, given that there is an equal chance of being at each of their possible locations.

S	$P(S)$
s_0	$\frac{1}{2}$
s_1	$\frac{1}{2}$

P	$P(P)$
p_0	$\frac{1}{3}$
p_1	$\frac{1}{3}$
p_2	$\frac{1}{3}$

(iii) [1 pt] Consider the table below.

6 rows would have $U = 1$, and 30 rows would have $U = 0$.

$1/6$ of the combinations will successfully water the plant and there are 36 total combinations. So there are 6 with $U = 1$ and 30 (the rest) with $U = 0$.

R	S	T	P	U
⋮	⋮	⋮	⋮	⋮

(b) Before selecting your actions, suppose that someone could tell you the value of either S or P. Follow the steps below to calculate the **maximum expected utility (MEU)** when knowing S, or when knowing P. Then, decide which one you would prefer to be told.

(i) [1 pt] What is $MEU(S)$?

○ 0 ○ $\frac{1}{9}$ ○ $\frac{1}{6}$ ○ $\frac{1}{4}$ ● $\frac{1}{3}$ ○ $\frac{1}{2}$
○ $\frac{2}{3}$ ○ $\frac{3}{4}$ ○ $\frac{5}{6}$ ○ 1 ○ None of the above

Note: can definitely answer this with intuition (and no math).

$= \frac{1}{2} MEU(S = s_0) + \frac{1}{2} MEU(S = s_1)$

$= \frac{1}{2}(max_t(EU(S = s_0, T = t_0), EU(S = s_0, T = t_1), EU(S = s_0, T = t_2)))$

$+ \frac{1}{2}(max_t(EU(S = s_1, T = t_0), EU(S = s_1, T = t_1), EU(S = s_1, T = t_2)))$

$= \frac{1}{2}(max(1/3, 1/3, 1/3)) + \frac{1}{2}(max(1/3, 1/3, 1/3))$

$= \frac{1}{3}$

(ii) [1 pt] What is $MEU(P)$?

○ 0 ○ $\frac{1}{9}$ ○ $\frac{1}{6}$ ○ $\frac{1}{4}$ ○ $\frac{1}{3}$ ● $\frac{1}{2}$
○ $\frac{2}{3}$ ○ $\frac{3}{4}$ ○ $\frac{5}{6}$ ○ 1 ○ None of the above

Note: can definitely answer this with intuition (and no math).

$= \frac{1}{3} MEU(P = p_0) + \frac{1}{3} MEU(P = p_1) + \frac{1}{3} MEU(P = p_2)$

$= \frac{1}{3}(max_R(EU(P = p_0, R = l), EU(P = p_0, R = r))) +$

$\frac{1}{3}(max_R(EU(P = p_1, R = l), EU(P = p_1, R = r))) +$

$\frac{1}{3}(max_R(EU(P = p_2, R = l), EU(P = p_2, R = r)))$

$= \frac{1}{3}(max(1/2, 1/2)) +$

$\frac{1}{3}(max(1/2, 1/2)) +$

$\frac{1}{3}(max(1/2, 1/2))$

$= \frac{1}{2}$

(iii) [1 pt] Would you prefer to be told S or P? ○ S ● P

P since it has a higher MEU (and therefore a higher VPI).

(c) **(i)** [1 pt] What is $MEU(S, P)$?

○ 0 ○ $\frac{1}{9}$ ○ $\frac{1}{6}$ ○ $\frac{1}{4}$ ○ $\frac{1}{3}$ ○ $\frac{1}{2}$
○ $\frac{2}{3}$ ○ $\frac{3}{4}$ ○ $\frac{5}{6}$ ● 1 ○ None of the above

1, because you have enough information to definitely get the water pellet to the plant.

(ii) [1 pt] In this problem, does $VPI(S, P) = VPI(S) + VPI(P)$? ○ Yes ● No

$VPI(S, P) = MEU(S, P) - MEU(none)$

$VPI(S) = MEU(S) - MEU(none)$

$VPI(P) = MEU(P) - MEU(none)$
Answer is NO, because $MEU(S, P) = 1$, $MEU(S) = \frac{1}{3}$, and $MEU(P) = \frac{1}{2}$.

(iii) [1 pt] In general, does $VPI(a, b) = VPI(a) + VPI(b)$? Select all of the statements below which are true.

- ◯ Yes, because of the additive property.
- ◯ Yes, because the order in which we observe the variables does not matter.
- ◯ Yes, but the reason is not listed.
- ● No, because the value of knowing each variable can be dependent on whether or not we know the other one.
- ◯ No, because the order in which we observe the variables matters.
- ◯ No, but the reason is not listed.

(d) For each of the following new variables introduced to this problem, what would the corresponding VPI of that variable be?

(i) [1 pt] A new variable X indicates the weather outside, which affects the overall health of the plant.

◯ VPI(X)<0 ● VPI(X)=0 ◯ VPI(X)>0

The health of the plant does not affect the utility so the VPI is 0.

(ii) [1 pt] A new variable X indicates the weather outside, which affects the metal of switch S such that when it's hot outside, the switch is most likely to remain in position s_0 with probability 0.9 (and goes to s_1 with probability 0.1).

◯ VPI(X)<0 ◯ VPI(X)=0 ● VPI(X)>0

This will allow us to predict which direction to move the robot with more accuracy so the VPI is greater than 0.

Q4. [20 pts] Languages

To discuss a strategy to play against Pacman, the ghosts send each other encrypted messages. Pacman knows that they are using one of English, Romanian, French, German and Swedish. He intercepted the ghosts' messages, but only characters "a", "o", "u", "ä", "ö", "ü" and "â" are decrypted correctly and everything else were lost.
Pacman would like to know which language the ghosts are using. He gathered information about the 5 languages, as below. Assume that characters that are not checked in the table will never appear in texts in that language.

Characters occurrence table:

	a	o	u	ä	ö	ü	â
English	✓	✓	✓				
Romanian	✓	✓	✓				✓
French	✓	✓	✓			✓	✓
German	✓	✓	✓	✓	✓	✓	
Swedish	✓	✓	✓	✓	✓		

Frequency table:

	a	o	u
English	0.4	0.4	0.2
Romanian	0.6	0.1	0.25
French	0.4	0.25	0.3
German	0.4	0.2	0.3
Swedish	0.5	0.2	0.1

(a) Probability Warm Up

(i) [1 pt] If Pacman does not see any ü, select the languages that are possible:

■ English ■ Romanian ■ French ■ German ■ Swedish ○ None of the above

Text from any language has a non-zero probability of not having ü, although for some languages it happens with probability 1 and for German and French, the probability is smaller.

(ii) [2 pts]

English	Romanian	French	German	Swedish
0 %	0 %	0 %	34 %	66 %

Pacman assumes that the ghosts have chosen their language uniformly randomly, and he did research on the character frequencies and normalized them among the 7 characters in each language, as listed in the frequency table above. Unfortunately Pacman lost the old decryption data and can only recall from memory that he has seen ä. **For languages that are not possible, please write down 0 in the table above.**

The ghosts sent a new message just now, and Pacman has 4 characters successfully decrypted, which are 3 occurrences of a and 1 occurrence of o. Assuming the occurrences of characters are mutually independent, what is Pacman's best estimation of the probabilities of each language, in percentages? **Write down the nearest integers in the table above.**

The two language that have ä are German and Swedish.
$P(\text{Swedish}) = \frac{0.5^3 0.2^1}{0.5^3 0.2^1 + 0.4^3 0.2^1} = 0.66$, and $P(\text{German}) = \frac{0.4^3 0.2^1}{0.5^3 0.2^1 + 0.4^3 0.2^1} = 0.34$. They sum to 1.
Note that we can omit the $\binom{3+1}{1}$ since it's a common factor.

Pacman thinks it's a good idea to train neural networks to classify the text based on the decrypted characters.

(b) **(i)** [1 pt] Given an arbitrary function from x to y, with enough training time and appropriate hyper-parameters, a neural net with 2 hidden layers that have sufficient number of parameters can gain a training accuracy arbitrarily close to 100% for an arbitrarily large training set. ● True ○ False

Universal function approximation theorem.

(ii) [1 pt] Given any dataset and enough training time, with appropriate hyper-parameters, a sufficiently large neural net can gain a training accuracy arbitrarily close to 100%. ○ True ● False

It's possible to have exactly the same features but different labels, and the training accuracy would never reach 100% in this case. In general, we just don't get 100% training accuracy even heavily overfitted.

(iii) [1 pt] Given any dataset and enough training time, with appropriate hyper-parameters, a sufficiently large neural net can gain a test accuracy arbitrarily close to 100%. ○ True ● False

We cannot guarantee 100% test accuracy.

(c) [7 pts]

Given the following concepts in neural nets, match them to the ...

(a) Back Propagation.
(b) Output layer

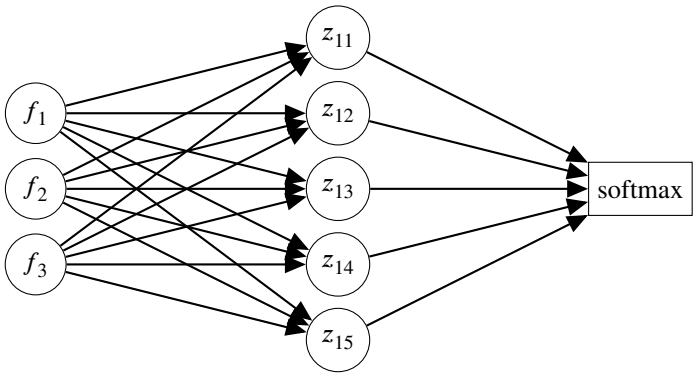
(c) Hidden unit
(d) Stochastic Gradient Descent
(e) Batch Gradient Descent
(f) Activation function.
(g) Gradient

Match them to the most related concept or procedure in the list.

(1) tanh can be used as the (f)
(2) weight vectors are used in a (c)
(3) a single data point is used in (d) to compute a(n) (g) using (a)
(4) many data points are used in (e) to compute a(n) (g) using (a)
(5) (b) can compute a probability distribution over classes

But Pacman decided to start simple.

- f_1, f_2, f_3 are the (normalized) frequencies of "a", "o", "u", respectively
- $z_{1i} = w_{1i} f + b_{1i}$, where $f = [f_1\ f_2\ f_3]^\top$ and each w_{1j} is a 1×3 vector



He built Neural Net A as above.

(d) [1 pt] Pacman used the set up for Project 5 to implement neural network A. He trained it for 1 epoch and got a training accuracy of 50%, but he forgot to save the model. The TA suggests that Pacman can train neural network A from the start again for 1 epoch, passing in the same data in the same order, and Pacman will have the same weights and the 50% training accuracy.
This is not guaranteed. If you tried and remember, you don't always get the same thing by rerunning the Project 5 codes. Basically, there's randomness in initialization and Stochastic GD (if you are using it).

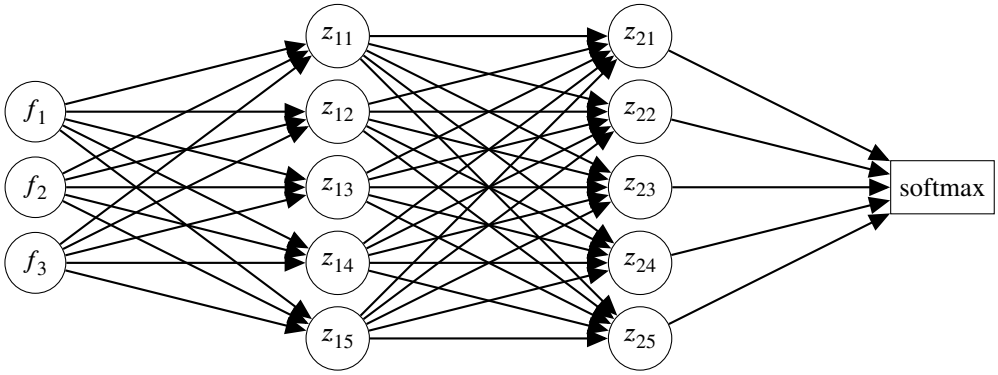
(e) Pacman trained Neural Net A with some data. He tried to classify some new data with the trained model, but the test accuracy was low. What can he do to improve the performance?

(i) [1 pt] Pacman can replace the softmax layer with sigmoid
☐ if the training accuracy is low ☐ if the training accuracy is high ● None of the above
Sigmoid is an activation function.

(ii) [1 pt] Pacman can add more nodes ($z_{16}, z_{17}, \ldots$) to the layer
☐ if the training accuracy is low ☐ if the training accuracy is high ● None of the above
There are 5 labels, so we should have 5 nodes in the layer before softmax.

(iii) [1 pt] Pacman can add more training data
☐ if the training accuracy is low ■ if the training accuracy is high ○ None of the above
With this simple architecture, it's hard to overfit unless we have very few/biased training data. Adding more training data would hopefully solve the overfitting issue.

Neural Net B:



(f) [1 pt] Pacman would like to try adding a layer on Neural Net A to get Neural Net B. Which of the following would lead you to expect Neural Net B to have better training accuracy than Neural Net A?

☐ $z_{2i} = w_{2i} \cdot z_1 + b_{2i}$, where $z_1 = [z_{11}, z_{12}, z_{13}, z_{14}, z_{15}]^\top$

☐ $z_{2i} = w_{2i} \cdot z_1 + b_{2i}$, where $z_1 = [z_{11}, z_{12}, z_{13}, z_{14}, z_{15}]^\top$, and change $z_{1i} = w_{1i} f + b_{1i}$ to $z_{1i} = \text{ReLU}\left(w_{1i} f\right) + b_{1i}$

■ $z_{2i} = w_{2i} \cdot \text{ReLU}\left(z_1\right) + b_{2i}$, where $z_1 = [z_{11}, z_{12}, z_{13}, z_{14}, z_{15}]^\top$

○ None of the above

2 layers of linear combination without non-linearity in between is in no way more powerful than 1 layer of linear combination. This is learned in lecture and reinforced in WHW 4.

ReLU $\left(w_{1i} f\right)$ seems to include non-linearity, but without the bias term included in the parenthesis, it's really a scaled feature or a 0, which is nothing more powerful than the original z_1 layer.

(g) [2 pts] Suppose the activation function for z_{1i} is g, then we can represent the NN in the graph as

☐ softmax $\left(g\left(w_2(w_1 f + b_1) + b_2\right)\right)$,

■ softmax $\left(w_2 \cdot g\left(w_1 f + b_1\right) + b_2\right)$,

○ both are incorrect

with dimensions: w_1: 5 × 3, b_1: 5 × 1, w_2: 5 × 5, b_2: 5 × 1.

SID: _____

Q5. [13 pts] Pacman Loses Control

Pacman finds himself inside the grid world Markov Decision Process (MDP) depicted below. Each rectangle represents a possible state. Pacman has two possible actions, left or right. However, these movement actions only work with probability p. With probability q Pacman moves in the opposite direction. Otherwise Pacman stays in the same state. I.E. $T(B, right, C) = p$, $T(B, right, A) = q$ and $T(B, right, B) = 1 - q - p$. If Pacman physically moves right from state D (with probability p if he chooses right from D, or with probability q if he chooses left from D) he earns a reward of 3000 and enters the terminal state where he can no longer perform actions. Similarly if Pacman physically moves left from A he earns a reward of 0 and enters the terminal state. Note γ is the discount factor.

(a) Assume $p = .5$, $q = 0$, and $\gamma = \frac{2}{3}$ for part (a)

(i) [1 pt] What is the value of $V^*(D)$ (the expected value of total discounted rewards pacman can get from state D)

$V^*(D) =$ 2250

$V^*(D) = .5 * 3000 + .5 * \frac{2}{3} * V^*(D)$
solve for $V^*(D)$
.5 chance of moving and getting a reward 3000, .5 chance of staying and getting a reward of $V^*(D) * \gamma$

(ii) [1 pt] What is the value of $V^*(C)$ (the expected value of total discounted rewards pacman can get from state C)

$V^*(C) =$ 1125

$V^*(C) = .5 * \frac{2}{3} * V^*(D) + .5 * \frac{2}{3} * V^*(C)$
solve for $V^*(C)$ using $V^*(D) = 2250$ part the previous part
.5 chance of moving and getting a reward $V^*(D) * \gamma$, .5 chance of staying and getting a reward of $V^*(C) * \gamma$

(b) For each subpart in part (b) you will be given two sets of parameters. Select which set of parameters results in a greater value of $V^*(D)$ (or whether they are the same).

(i) [1 pt] Set I: $\{p = .5, q = 0, \gamma = .5\}$ Set II: $\{p = .6, q = 0, \gamma = .5\}$

○ Set I's $V^*(D)$ is greater ● Set II's $V^*(D)$ is greater ○ They are equal
Since p is higher in Set II it will reach the reward faster and incur less discounting.

(ii) [1 pt] Set I: $\{p = .5, q = 0, \gamma = 1\}$ Set II: $\{p = .6, q = 0, \gamma = 1\}$

○ Set I's $V^*(D)$ is greater ○ Set II's $V^*(D)$ is greater ● They are equal
Even though p is higher in Set II so it will reach the reward faster there is no discounting so they will both get the max reward anyway.

(iii) [1 pt] Set I: $\{p = .6, q = .1, \gamma = .5\}$ Set II: $\{p = .1, q = .6, \gamma = .5\}$

○ Set I's $V^*(D)$ is greater ○ Set II's $V^*(D)$ is greater ● They are equal
Since p and q are swapped, for whatever policy is optimal for Set I we can choose the opposite actions and get the same performance in Set II.

(iv) [1 pt] Set I: $\{p = .1, q = 0, \gamma = 1\}$ Set II: $\{p = .8, q = .1, \gamma = 1\}$

● Set I's $V^*(D)$ is greater ○ Set II's $V^*(D)$ is greater ○ They are equal
In Set I there is no discounting so you will eventually reach and earn the max reward of 3000. In Set II there is a nonzero probability of accidentally taking the reward of 0 so we will have an expectation less than 3000.

(v) [1 pt] Set I: $\{p = .5, q = 0, \gamma = .5\}$ Set II: $\{p = .5, q = .1, \gamma = .5\}$

● Set I's $V^*(D)$ is greater ○ Set II's $V^*(D)$ is greater ○ They are equal

In Set II, the positive Q Value causes you to accidentally go in the opposite direction and earn less than 3000 in expectation.

(c) Assume $p = 0$, $q = .5$, and $\gamma = .5$ for part (c). Pacman decides to use policy iteration to figure out the optimal policy for this MDP. He starts with this policy:

$\pi(A)$	$\pi(B)$	$\pi(C)$	$\pi(D)$
right	left	left	left

(i) [1 pt] Pacman uses policy evaluation to evaluate his policy and then he uses policy improvement to update his policy. Select which states (if any) have a different action according to the policy after improvement is over.

■ A ☐ B ☐ C ☐ D ○ None changed

Since $q = .5$ and $p = 0$ it is optimal to choose left at every state and try to accidentally move towards the 3000 reward. Thus the only state whose action needs to change to be optimal is state A.

(ii) [1 pt] After part (i) we run policy evaluation and policy improvement infinitely more times. How many more times will the policy change (not including a potential change in part (i))?

● 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 16 ○ ∞

Since the policy is optimal after state A changes to left there will be no more changes.

(d) After some practice runs, Pacman realizes what he thought about $T(s, a, s')$ and $R(s, a, s')$ might be wrong. A policy π is **strictly greedy** with respect to a set of Q-values as long as $\forall s\ \forall a \neq \pi(s)\ Q(s, \pi(s)) > Q(s, a)$ I.E. the Q value for the action chosen by the policy must be strictly greater than the Q value for all other actions. Pacman decides to use approximate q-learning to come up with a strictly greedy policy.

Pacman has 4 feature functions available.

$$f_1(s,a) = \begin{cases} 1 & \text{if s = C} \\ 0 & else \end{cases} \qquad f_2(s,a) = \begin{cases} 1 & \text{if (a = right)} \wedge \text{(s = A)} \\ 0 & else \end{cases}$$

$$f_3(s,a) = \begin{cases} 1 & \text{if a = left} \\ 0 & else \end{cases} \qquad f_4(s,a) = \begin{cases} 1 & \text{if (a = left)} \wedge \text{((s = A)} \vee \text{(s = B))} \\ 0 & else \end{cases}$$

He wants to choose weights w_1, w_2, w_3, and w_4 that he can use to calculate $Q(s, a)$ by multiplying each weight by its respective feature and adding the products together. For each policy below select which weights **must be non-zero** (weights can be negative or positive) for the calculated Q-values to generate the policy as a strictly greedy policy. If it is not possible to generate the strictly greedy policy with the given features mark "Not Possible".

(i) [1 pt]

$\pi(A)$	$\pi(B)$	$\pi(C)$	$\pi(D)$
right	right	right	right

☐ w_1 ☐ w_2 ■ w_3 ☐ w_4 ○ Not Possible

If we choose w_3 to be negative $Q(s, Left)$ will be less than $Q(s, Right)$ for every state.

(ii) [1 pt]

$\pi(A)$	$\pi(B)$	$\pi(C)$	$\pi(D)$
right	left	left	left

☐ w_1 ■ w_2 ■ w_3 ☐ w_4 ○ Not Possible

If we choose w_3 to be positive $Q(s, Left)$ will be greater than $Q(s, Right)$ for every state. If we choose w_2 to be even more positive then $Q(A, Right)$ will be greater than $Q(A, Left)$.

(iii) [1 pt]

$\pi(A)$	$\pi(B)$	$\pi(C)$	$\pi(D)$
left	right	left	right

☐ w_1 ☐ w_2 ☐ w_3 ☐ w_4 ● Not Possible

There is no way to distinguish better actions for C and D since none of the features consider those states and an action at the same time.

(iv) [1 pt]

$\pi(A)$	$\pi(B)$	$\pi(C)$	$\pi(D)$
right	left	right	right

☐ w_1 ■ w_2 ■ w_3 ■ w_4 ○ Not Possible

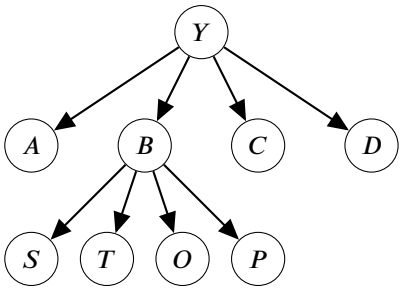
If we choose w_3 to be negative $Q(s, Left)$ will be less than $Q(s, Right)$ for every state. If we choose w_4 to be even more negative, states A and B will prefer left. If we choose w_2 to be even more positive than those, state A will prefer right

SID: _____

Q6. [9 pts] A Nonconvolutional Nontrivial Network

You have a robotic friend MesutBot who has trouble passing Recaptchas (and Turing tests in general). MesutBot got a 99.99% on the last midterm because he could not determine which squares in the image contained stop signs. To help him ace the final, you decide to design a few classifiers using the below features.

- $A = 1$ if the image contains an octagon, else 0.
- $B = 1$ if the image contains the word STOP, else 0.
  - $S = 1$ if the image contains the letter S, else 0.
  - $T = 1$ if the image contains the letter T, else 0.
  - $O = 1$ if the image contains the letter O, else 0.
  - $P = 1$ if the image contains the letter P, else 0.
- $C = 1$ if the image is more than 50% red in color, else 0.
- $D = 1$ if the image contains a post, else 0.



(a) First, we use a Naive Bayes-inspired approach to determine which images have stop signs based on the features and Bayes Net above. We use the following features to predict $Y = 1$ if the image has a stop sign anywhere, or $Y = 0$ if it doesn't.

(i) [1 pt] Using the independence assumptions encoded in the Bayes Net, which of the following are true?

■ If we know whether the picture has the word "STOP" (B), the appearance of the letter "S" is independent from the appearance of the letter "T" in the image.

☐ If we know whether the picture has a STOP sign (Y), the appearance of the letter "S" is independent from the appearance of the letter "T" in the image.

■ $S \perp\!\!\!\perp D|Y$

☐ $A \perp\!\!\!\perp B|\{S, T, O, P\}$

☐ The 7 features (A, S, T, O, P, C, D) satisfy the Naive Bayes independence assumptions.

○ None

- $S \perp\!\!\!\perp T|B$ due to inactive triple (blocked common cause S-T-B).
- S is not $\perp\!\!\!\perp T|Y$ due to active triple (common cause S-T-B).
- $S \perp\!\!\!\perp D|Y$ due to inactive triple (blocked common cause B-Y-D).
- $A \perp\!\!\!\perp B|\{S, T, O, P\}$ is not true due to active triple (common cause A-Y-B).
- This Bayes net doesn't satisfy the Naive Bayes independence assumptions because the S,T,O,P features are dependent given Y.

(ii) [1 pt] Which expressions would a Naive Bayes model use to predict the label for B if given the values for features $S = s, T = t, O = o, P = p$? Choose all valid expressions.

■ $b = \arg\max_b P(b)P(s|b)P(t|b)P(o|b)P(p|b)$

☐ $b = \arg\max_b P(s|b)P(t|b)P(o|b)P(p|b)$

■ $b = \arg\max_b P(b|s, t, o, p)$

■ $b = \arg\max_b P(b, s, t, o, p)$

☐ $b = \arg\max_b P(s, t, o, p|b)$

○ None

Note $\arg\max_b P(b)P(s|b)P(t|b)P(o|b)P(p|b) = \arg\max_b P(b, s, t, o, p)$, which are both correct. The conditional probability assumptions from the Bayes Net enable us to write this equality.

Note $P(s|b)P(t|b)P(o|b)P(p|b) = P(s, t, o, p|b)$. This can be read off of the Bayes Net as well, because all the features are independent given the label $B = b$.

Finally note $\arg\max_b P(b|s,t,o,p) = \arg\max_b \frac{P(b,s,t,o,p)}{P(s,t,o,p)} = \arg\max_b P(b,s,t,o,p)$ because $P(s,t,o,p)$ has all four of its values already given, and does not depend on our optimization variable b in any way.

(iii) [1 pt] Which expressions would we use to predict the label for Y with our Bayes Net above? Assume we are given all features except B. So $A = a, S = s, T = t$, etc. For the below choices, the underscore means we are dropping the value of that variable. So $y, \_\_ = (0, 1)$ would mean $y = 0$.

☐ $y, \_\_ = \arg\max_{y,b} P(y)P(a|y)P(b|y)P(c|y)P(d|y)P(s|b)P(t|b)P(o|b)P(p|b)$

☐ $y, \_\_ = \arg\max_{y,b} P(s)P(t)P(o)P(p)P(a)P(b|s,t,o,p)P(c)P(d)P(y|a,b,c,d)$

☐ First compute $b' = \arg\max_b$ of the formula chosen in part (ii).
Then compute $y = \arg\max_y P(y)P(a|y)P(b'|y)P(c|y)P(d|y)$

☐ First compute $b' = \arg\max_b$ of the formula chosen in part (ii).
Then compute $y = \arg\max_y P(y|a,b',c,d)$

■ $y = \arg\max_y \sum_{b'} P(y)P(a|y)P(b'|y)P(c|y)P(d|y)P(s|b')P(t|b')P(o|b')P(p|b')$

○ None

Sum out possibilities for b given the features S, T, O, P

(iv) [1 pt] One day MesutBot got allergic from eating too many cashews. The incident broke his letter S detector, so that he no longer gets reliable S features. Now what expressions would we use to predict the label for Y? Assume all features except B, S are given. So $A = a, T = t, O = o$, etc.

☐ $y = \arg\max_y P(y)P(a|y)P(c|y)P(d|y)$

☐ $y, \_\_, \_\_ = \arg\max_{y,b,s} P(y)P(a|y)P(b|y)P(c|y)P(d|y)P(s|b)P(t|b)P(o|b)P(p|b)$

☐ $y, \_\_ = \arg\max_{y,s} P(y)P(a|y)P(b|y)P(c|y)P(d|y)P(s|b)P(t|b)P(o|b)P(p|b)$

☐ $y, \_\_ = \arg\max_{y,b} P(y)P(a|y)P(b|y)P(c|y)P(d|y)P(t|b)P(o|b)P(p|b)$

☐ $y, \_\_ = \arg\max_{y,b} P(y)P(a|y)P(b|y)P(c|y)P(d|y)P(s|b)P(t|b)P(o|b)P(p|b)$

☐ $y, \_\_ = \arg\max_{y,b} P(y|a,b,c,d)$

■ $y = \arg\max_y P(y)P(a|y)P(c|y)P(d|y) \sum_{b',s'} P(b'|y)P(s'|b')P(t|b')P(o|b')P(p|b')$

○ None

Use variable elimination on s and b (because b cannot be accurately calculated without s).

(b) [1 pt] You decide to try to output a probability $P(Y|features)$ of a stop sign being in the picture instead of a discrete ± 1 prediction. We denote this probability as $P(Y|\vec{f}(x))$. Which of the following functions return a **valid** probability distribution for $P(Y = y|\vec{f}(x))$? Recall that $y \in \{-1, 1\}$.

■ $\frac{e^{y \cdot \vec{w}^T \vec{f}(x)}}{e^{-y \cdot \vec{w}^T \vec{f}(x)} + e^{y \cdot \vec{w}^T \vec{f}(x)}}$

■ $\frac{1}{2}$

☐ $\frac{0.5}{1+e^{-\vec{w}^T \vec{f}(x)}}$

☐ $\frac{-1}{1+e^{\vec{w}^T \vec{f}(x)}} + 1$

○ None

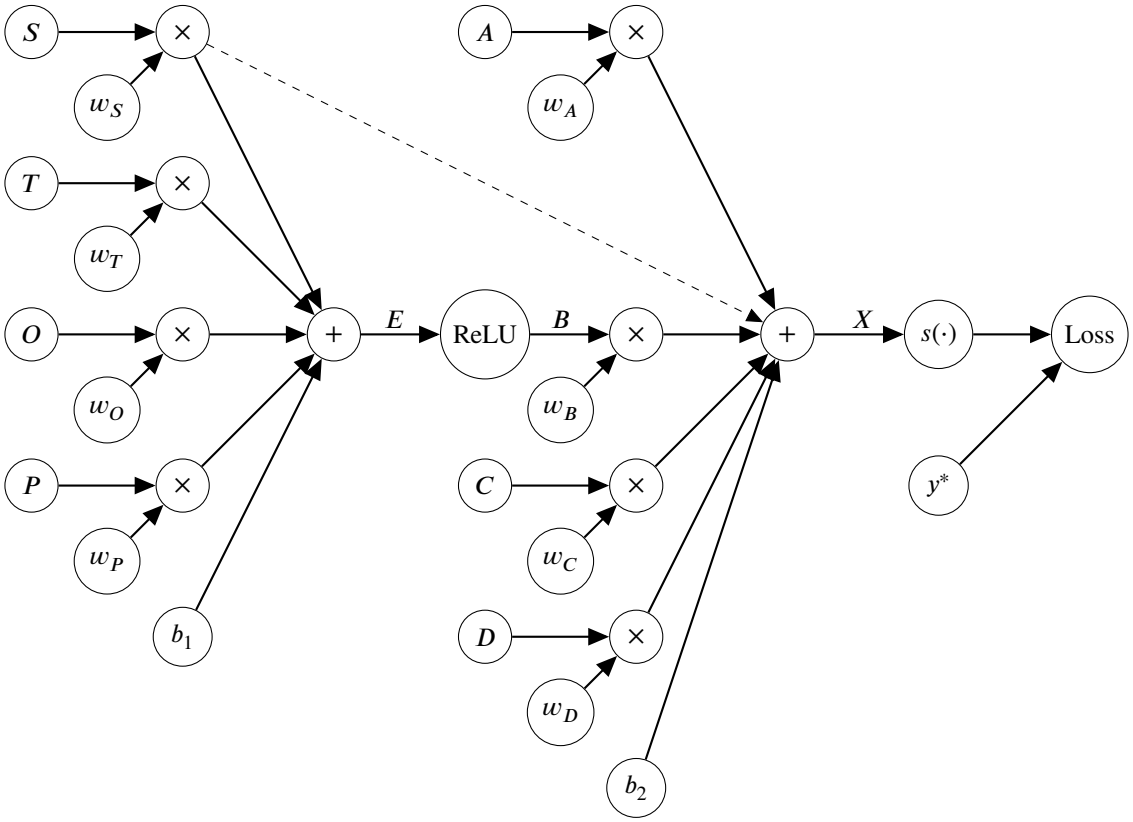
Valid probability distribution means that the probabilities over all possible values of y must sum to 1.

$P(Y = y|\vec{f}(x)) = \frac{e^{y \cdot \vec{w}^T \vec{f}(x)}}{e^{-y \cdot \vec{w}^T \vec{f}(x)} + e^{y \cdot \vec{w}^T \vec{f}(x)}}$ works because $P(Y = 1|\vec{f}(x)) + P(Y = -1|\vec{f}(x)) = 1$ (it is the softmax function).

$\frac{1}{2}$ works because we just need $P(Y = 1|\vec{f}(x)) + P(Y = -1|\vec{f}(x)) = \frac{1}{2} + \frac{1}{2} = 1$, so it is valid.

$\frac{0.5}{1+e^{-\vec{w}^T \vec{f}(x)}}$ and $\frac{-1}{1+e^{\vec{w}^T \vec{f}(x)}} + 1$ don't depend on y so we can't guarantee the sum of the two probabilities adds to 1, and thus cannot guarantee that those two expressions are a valid probability distribution.

Unimpressed by the perceptron, you note that features are inputs into a neural network and the output is a label, so you modify the Bayes Net from above into a Neural Network computation graph. Recall the logistic function $s(x) = \frac{1}{1+e^{-x}}$ has derivative $\frac{\partial s(x)}{\partial x} = s(x)[1 - s(x)]$



(c) For this part, ignore the dashed edge when calculating the below.

(i) [1 pt] What is $\frac{\partial Loss}{\partial w_A}$?

- (●) $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot A$
- ( ) $2(s(X) - y^*) \cdot [s(X) \cdot (1 - s(X))] \cdot A$
- ( ) $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot 2A + 1$
- ( ) $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot 2A$
- ( ) $2(s(X) - y^*) \cdot [s(X) \cdot (1 - s(X))] \cdot A + 1$
- ( ) $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot A + 1$
- ( ) None

$$\frac{\partial Loss}{\partial w_A} = \frac{\partial Loss}{\partial s(X)} \cdot \frac{\partial s(X)}{\partial X} \cdot \frac{\partial X}{\partial A w_A} \cdot \frac{\partial A w_A}{\partial w_A}$$
$$= \frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot 1 \cdot A$$

(ii) [1 pt] What is $\frac{\partial Loss}{\partial w_S}$? Keep in mind we are still ignoring the dotted edge in this subpart.

- ● $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases} \right) \cdot S$
- ○ $2(s(X) - y^*) \cdot [s(X) \cdot (1 - s(X))] \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases} \right) \cdot S$
- ○ $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases} \right) \cdot 2S + S$
- ○ $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases} \right) \cdot 2S$
- ○ $2(s(X) - y^*) \cdot [s(X) \cdot (1 - s(X))] \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases} \right) \cdot S + S$
- ○ $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases} \right) \cdot S + S$
- ○ None

$$\frac{\partial Loss}{\partial w_S} = \frac{\partial Loss}{\partial s(X)} \cdot \frac{\partial s(X)}{\partial X} \cdot \frac{\partial X}{\partial Bw_B} \cdot \frac{\partial Bw_B}{\partial ReLU(E)} \cdot \frac{\partial ReLU(E)}{\partial E} \cdot \frac{\partial E}{\partial Sw_S} \cdot \frac{\partial Sw_S}{\partial w_S}$$
$$= \frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot 1 \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases} \right) \cdot 1 \cdot S$$

(d) MesutBot is having trouble paying attention to the S feature because sometimes it gets zeroed out by the ReLU, so we connect it directly to the input of $s(\cdot)$ via the dotted edge. For the below, treat the dotted edge as a regular edge in the neural net.

(i) [1 pt] Which of the following is equivalent to $\frac{\partial Loss}{\partial w_A}$?

- ● $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot A$
- ○ $2(s(X) - y^*) \cdot [s(X) \cdot (1 - s(X))] \cdot A$
- ○ $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot 2A + A$
- ○ $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot 2A$
- ○ $2(s(X) - y^*) \cdot [s(X) \cdot (1 - s(X))] \cdot A + A$
- ○ $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot A + A$
- ○ None

This doesn't change because the added edge is further upstream from w_A and doesn't affect gradient flows between w_A and $Loss$. From above, we copy:

$$\frac{\partial Loss}{\partial w_A} = \frac{\partial Loss}{\partial s(X)} \cdot \frac{\partial s(X)}{\partial X} \cdot \frac{\partial X}{\partial Aw_A} \cdot \frac{\partial Aw_A}{\partial A}$$
$$= \frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot 1 \cdot w_A$$

(ii) [1 pt] Which of the following is equivalent to $\frac{\partial Loss}{\partial w_S}$? Keep in mind we are still treating the dotted edge as a regular edge.

- ○ $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases} \right) \cdot S$
- ○ $2(s(X) - y^*) \cdot [s(X) \cdot (1 - s(X))] \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases} \right) \cdot S$
- ○ $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases} \right) \cdot 2S + S$
- ○ $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases} \right) \cdot 2S$
- ○ $2(s(X) - y^*) \cdot [s(X) \cdot (1 - s(X))] \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases} \right) \cdot S + S$
- ○ $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases} \right) \cdot S + S$
- ● None

Due to the new dotted edge, there are now two paths along the neural network that lead from output to w_S.

$$\frac{\partial Loss}{\partial w_S} = \frac{\partial Loss}{\partial s(X)} \cdot \frac{\partial s(X)}{\partial X} \cdot \left(\frac{\partial X}{\partial Bw_B} \cdot \frac{\partial Bw_B}{\partial ReLU(E)} \cdot \frac{\partial ReLU(E)}{\partial E} \cdot \frac{\partial E}{\partial Sw_S} + \frac{\partial X}{\partial Sw_S} \right) \cdot \frac{\partial Sw_S}{\partial w_S}$$

$$= \frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot \left(1 \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases} \right) \cdot 1 + 1 \right) \cdot S$$

$$= \frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot \left(\begin{cases} w_B + 1 & E \geq 0 \\ 1 & E < 0 \end{cases} \right) \cdot S$$

Q7. [13 pts] Searching for a Bayes Network

Your friend gives you a joint distribution $P(X_1, X_2, ..., X_n)$ and wants you to find the structure of a Bayes Network that best represents this data. This can be formulated as a search problem.

Initial state:

Legal actions: Add an edge between any pair of nodes $X_i \rightarrow X_j$ so long as it does not violate the structure of a Bayes Network (remember, no cycles!).

Goal state: We don't know this but our friend does and can tell us if a given state is the goal state or not.

Answer the following questions regarding the search tree for this problem setup.

(a) [1 pt] Considering the set of legal actions, what is the branching factor of the search tree at depth 1? That is, the branching factor of the node representing the initial state. Write your answer in terms of the number of nodes, N.

$N \times (N-1)$

We are allowed to add an edge between $X_i \rightarrow X_j$ so long as it doesn't violate the structure of a Bayes Net. When choosing where to draw our edge, there are N possible ways to select the X_i node and $(N-1)$ possible ways to select the X_j node (can't draw an edge from a node to itself!)

(b) [1 pt] Will any states be repeated in our search tree? ● Yes ○ No

(c) Assume infinite computational resources, and that the goal state exists somewhere in the search tree.

(i) [1 pt] DFS is guaranteed to return the solution

■ if $N = 10000$ ☐ if $N = 10000$ and deleting edges are also legal actions ○ None of the above

Note that in the original search setup, we cannot form cycles in the search tree, so we will eventually reach the solution. However, if deleting edges are also legal actions, then we can form cycles. DFS is not complete and therefore is not guaranteed to find a solution if it exists.

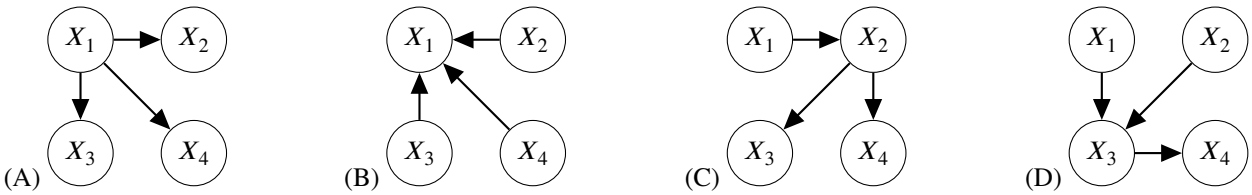
(ii) [1 pt] BFS is guaranteed to return the solution

■ if $N = 10000$ ■ if $N = 10000$ and deleting edges are also legal actions ○ None of the above

BFS is complete so if a solution exists, BFS will find it given infinite computational resources.

Now your friend gives you the joint distribution $P(X_1, X_2, X_3, X_4)$ and wants you to run A* search to find the right structure. Recall that A* search expands states with the lowest estimated total cost, where total cost is equal to the sum of backward cost (sum of edge weights in the path to the state) and estimated forward cost (heuristic value).

(d) For the following heuristics, select the state (or multiple states, if there are ties) that would be expanded by A* search if the backward cost for all of the states are the same. **Assume that X_1, X_2, X_3 and X_4 are binary variables.**



(i) [1 pt] Number of nodes with no ancestors.

■ A ☐ B ■ C ☐ D

(ii) [1 pt] Number of parameters in Bayes Net representation.

■ A ☐ B ■ C ☐ D

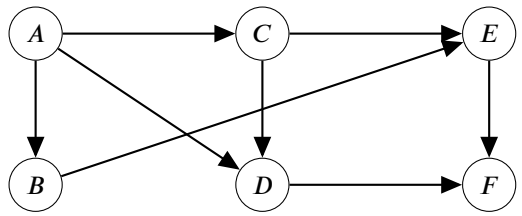
To solve this we need to count the number of parameters in each individual CPT and add them together. We are told that the variables are binary.
(A) Node X_1 has 2 parameters and nodes X_2, X_3 and X_4 have 2^2 parameters each so there are a total of $2 + 2^2 + 2^2 + 2^2 = 14$ parameters.
(B) Node X_1 has 2^4 parameters and nodes X_2, X_3 and X_4 have 2 parameters each so there are a total of $2^4+2+2+2 = 22$ parameters.
(C) Node X_1 has 2 parameters and nodes X_2, X_3 and X_4 have 2^2 parameters each so there are a total of $2 + 2^2 + 2^2 + 2^2 = 14$ parameters.
(D) Nodes X_1 and X_2 have 2 parameters each, node X_3 has 2^3 parameters and node X_4 has 2^2 parameters so there are a total of $2 + 2 + 2^3 + 2^2 = 16$ parameters.

(iii) [1 pt] Number of pairs of nodes that are not independent. Note that in each pair, the 2 nodes are unordered, i.e., the set of all pairs of nodes = $\{X_1X_2, X_1X_3, X_1X_4, X_2X_3, X_2X_4, X_3X_4\}$).

☐ A ■ B ☐ C ☐ D

To solve this we should count the number of pairs of nodes that are not guaranteed to be independent, running d-separation where needed.
(A) 6 pairs. X_1X_2, X_1X_3, and X_1X_4 are not guaranteed to be independent because they share an edge. Pair X_2X_3 **is not** guaranteed to be independent because the only path between them is an active triple $X_3 \leftarrow A \rightarrow X_2$. X_2X_4 **is not** guaranteed to be independent because the only path between them is an active triple $X_4 \leftarrow A \rightarrow X_2$. X_3X_4 **is not** guaranteed to be independent because the only path between them is an active triple $X_3 \leftarrow A \rightarrow X_4$.
(B) 3 pairs. X_1X_2, X_1X_3, and X_1X_4 are not guaranteed to be independent because they share an edge. Pair X_2X_3 **is** guaranteed to be independent because the only path between them is an inactive triple $X_3 \rightarrow A \leftarrow X_2$. X_2X_4 **is** guaranteed to be independent because the only path between them is an inactive triple $X_4 \rightarrow A \leftarrow X_2$. X_3X_4 **is** guaranteed to be independent because the only path between them is an inactive triple $X_3 \rightarrow A \leftarrow X_4$.
(C) 6 pairs. X_1X_2, X_2X_3, and X_2X_4 are not guaranteed to be independent because they share an edge. Pair X_1X_3 **is not** guaranteed to be independent because the only path between them is an active triple $X_1 \rightarrow B \rightarrow X_3$. X_1X_4 **is not** guaranteed to be independent because the only path between them is an active triple $X_1 \rightarrow B \rightarrow X_4$. X_3X_4 **is not** guaranteed to be independent because the only path between them is an active triple $X_3 \leftarrow B \rightarrow X_4$.
(D) 5 pairs. X_1X_3, X_2X_3, and X_3X_4 are not guaranteed to be independent because they share an edge. Pair X_1X_2 **is** guaranteed to be independent because the only path between them is an inactive triple $X_1 \rightarrow C \leftarrow X_2$. X_1X_4 **is not** guaranteed to be independent because the only path between them is an active triple $X_1 \rightarrow C \rightarrow X_4$. X_2X_4 **is not** guaranteed to be independent because the only path between them is an active triple $X_2 \rightarrow C \rightarrow X_4$.

You found a Bayes Net that represents a joint distribution $P(A, B, C, D, E, F)$, as below.



A	P(A)
a_1	0.2
a_2	0.3
a_3	0.5

B	A	P(B|A)
b_1	a_1	0
b_2	a_1	1
b_1	a_2	0.4
b_2	a_2	0.6
b_1	a_3	0.9
b_2	a_3	0.1

C	A	P(C|A)
c_1	a_1	0.2
c_2	a_1	0.8
c_1	a_2	0.4
c_2	a_2	0.6
c_1	a_3	0.5
c_2	a_3	0.5

D	A	C	P(D|A, C)
d_1	a_1	c_1	0.7
d_2	a_1	c_1	0.3
d_1	a_2	c_1	0.4
d_2	a_2	c_1	0.6
d_1	a_3	c_1	0.8
d_2	a_3	c_1	0.2
d_1	a_1	c_2	0.6
d_2	a_1	c_2	0.4
d_1	a_2	c_2	0.5
d_2	a_2	c_2	0.5
d_1	a_3	c_2	0.9
d_2	a_3	c_2	0.1

E	B	C	P(E|B, C)
e_1	b_1	c_1	0.3
e_2	b_1	c_1	0.4
e_3	b_1	c_1	0.3
e_1	b_2	c_1	0.1
e_2	b_2	c_1	0.8
e_3	b_2	c_1	0.1
e_1	b_1	c_2	0.6
e_2	b_1	c_2	0.3
e_3	b_1	c_2	0.1
e_1	b_2	c_2	0.5
e_2	b_2	c_2	0.2
e_3	b_2	c_2	0.3

F	D	E	P(F|D, E)
f_1	d_1	e_1	0.5
f_2	d_1	e_1	0.5
f_1	d_2	e_1	0.8
f_2	d_2	e_1	0.2
f_1	d_1	e_2	0.6
f_2	d_1	e_2	0.4
f_1	d_2	e_2	0.4
f_2	d_2	e_2	0.6
f_1	d_1	e_3	0.1
f_2	d_1	e_3	0.9
f_1	d_2	e_3	0.3
f_2	d_2	e_3	0.7

(e) [1 pt] Starting with the original 6 CPTs, you eliminated D and got a factor. What is the size of the factor?

36

We have $P(A)$, $P(B|A)$, $P(C|A)$, $P(D|A, C)$, $P(E|B, C)$, $P(F|D, E)$. To eliminating D, we look at P(D|A,C) and P(F|D,E), and get a factor $f(F|A, C, E)$. The domain sizes of A, C, E, F are 3, 2, 3, 2, respectively. So the size of the factor is 36.

(f) You would like to find $P(c_1|e_1, f_1)$. For each ordering i of variable elimination, we denote $S(i)$ to be the size of the largest factor that gets generated during the variable elimination process following the ordering i.

(i) [1 pt] Among the 4 orderings below, select the ordering(s) i with the largest $S(i)$.

■ A, B, D ■ A, D, B □ B, A, D □ B, D, A

(ii) [1 pt] Among the 4 orderings below, select the ordering(s) i with the smallest $S(i)$.

□ A, B, D □ A, D, B ■ B, A, D ■ B, D, A

The original 6 CPTs: $P(A)$, $P(B|A)$, $P(C|A)$, $P(D|A, C)$, $P(E|B, C)$, $P(F|D, E)$. ~~Domain sizes: A: 3, B: 2, C: 2, D: 2, E: 3, F: 2.~~

Domain sizes: A: 3, B: 2, C: 2, D: 2, e_1: 1, f_1: 1.

A, B, D: 8

Eliminate A: $f(B, C, D)$, $P(e_1|B, C)$, $P(f_1|D, e_1)$. Factor size is 8.

Eliminate B: $f(C, D, e_1)$, $P(f_1|D, e_1)$. Factor size is 4.

Then eliminate D and get $f(C, e_1, f_1)$. Factor size is 2.

A, D, B: 8

Eliminate A: $f(B, C, D)$, $P(e_1|B, C)$, $P(f_1|D, e_1)$. Factor size is 8.

Eliminate D: $f(B, C, f_1|e_1)$, $P(e_1|B, C)$. Factor size is 4.

Then eliminate B and get $f(C, e_1, f_1)$. Factor size is 2.

B, A, D: 6

Eliminate B: $f(e_1|A, C)$, $P(A)$, $P(C|A)$, $P(D|A, C)$, $P(f_1|D, e_1)$. Factor size is 6.

Eliminate A: $f(C, D, e_1)$, $P(f_1|D, e_1)$. Factor size is 4.

Then eliminate D and get $f(C, e_1, f_1)$. Factor size is 2.

B, D, A: 6

Eliminate B: $f(e_1|A, C)$, $P(A)$, $P(C|A)$, $P(D|A, C)$, $P(f_1|D, e_1)$. Factor size is 6.

Eliminate D: $f(e_1|A, C)$, $P(A)$, $P(C|A)$, $f(f_1|A, C, e_1)$. Factor size is 6.

Then eliminate A and get $f(C, e_1, f_1)$. Factor size is 2.

(g) You'd like to try out sampling to find $P(c_1|e_1, f_1)$. For each question below, select the sampling method(s) that may generate the specified data point.

(i) [1 pt] $a_2, b_1, c_1, d_2, e_1, f_1$

■ prior sampling ■ rejection sampling ■ likelihood sampling ■ gibbs sampling ○ None

The evidences match, and the joint distribution of this observation would be non-zero. All sampling methods can have this sample.

(ii) [1 pt] $a_1, b_2, c_2, d_2, e_3, f_1$

■ prior sampling ☐ rejection sampling ☐ likelihood sampling ☐ gibbs sampling ○ None

Not all the evidences match, so it can only be generated in prior sampling.

(iii) [1 pt] $a_1, b_1, c_2, d_2, e_1, f_1$

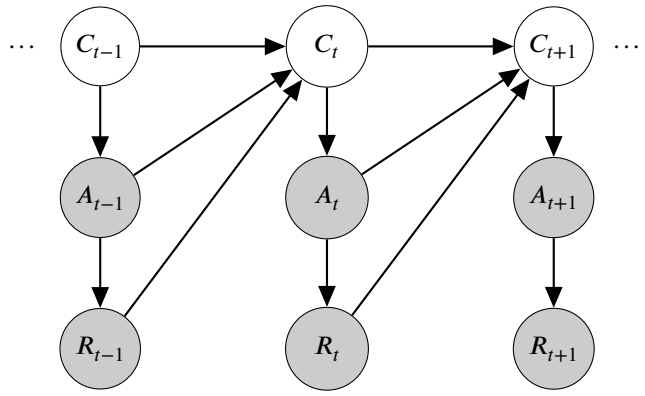
☐ prior sampling ☐ rejection sampling ☐ likelihood sampling ■ gibbs sampling ○ None

The evidences match, but the joint distribution of this observation is 0. Only Gibbs Sampling can have this sample since variables that are not the evidence are randomly assigned at first.

Q8. [11 pts] Particle Madness

Humans are finicky and have beliefs, and robots often get annoyed at dealing with us. Ideally, the human will have fully rational conscious beliefs that drive their actions.

(a) C represents the human's conscious beliefs, A represents human actions, and R represents reward distribution. Let's consider this model that makes the robots happy, in which actions and rewards directly influence conscious beliefs, and conscious beliefs influence actions.



(i) [1 pt] Select all variables that exhibit the Markov (memoryless) property in this model

■ C ☐ A ☐ R ○ None

A and R are not independent of the past, given the present. For example, using d-separation, there is an active path from A_{t-1} to A_{t+1}

All variables are binary and have their probabilities specified in the table below. **Note the first table doesn't contain all the rows.**

C_{t+1}	A_t	R_t	C_t	$P(C_{t+1}\|C_t, A_t, R_t)$
+c	+a	+r	+c	.25
-c	+a	+r	+c	.75
+c	+a	+r	-c	.2
-c	+a	+r	-c	.8
+c	+a	-r	+c	.4
+c	-a	-r	-c	.9

A_t	C_t	$P(A_t\|C_t)$
+a	+c	.8
-a	+c	.2
+a	-c	.3
-a	-c	.7

R_t	A_t	$P(R_t\|A_t)$
+r	+a	.5
-r	+a	.5
+r	-a	.4
-r	-a	.6

We are trying to run the forward algorithm to obtain a belief distribution at time i, $B(C_i) = P(C_i|a_1, .., a_i, r_1, ..., r_i)$. We have $B(C_{i-1} = +c) = .4$ and $B(C_{i-1} = -c) = .6$ and we know that $A_i = +a$ and $R_i = +r$.

(ii) [1 pt]

If possible, calculate $B'(C_i = +c)$ where $B'(C_i) = P(C_i|a_1, .., a_{i-1}, r_1, ..., r_{i-1})$, the belief distribution after the time elapse update but before including the new evidence.

$B'(C_i = +c) =$ [ ] ● Not Possible with the given information

In the time elapse update you sum over the old belief state multiplied by corresponding entries in $P(C_{t+1}|C_t, A_t, R_t)$. Since we didn't give you A_{i-1} and R_{i-1} you do not have enough information to do this.

(iii) [1 pt] Now it is time to calculate $B(C_i = +c)$ by performing the observation update. If possible, fill in the blank with a number that will be multiplied by $B'(C_i = +c)$ to create an expression that is equal to $B(C_i = +c)$ after normalization.

$$B(C_i = +c) = \frac{B'(C_i=+c) * [\ .8\ ]}{B(C_i=+c)+B(C_i=+c)}$$ ○ Not Possible with the given information

In the observation update you weight based on probability of the evidence given the state and then normalize. So

the weight is $P(+a|+r) = .8$. Note we do not consider R because R is independent of C given A so A has all the weighting information we need.

(iv) [1 pt] Your friend claims that using particle filtering with 100 particles to estimate the belief distribution for this model will result in more accurate results than using the forward algorithm. Is your friend correct?

○ Yes ● No

Particle filtering essentially is trying to approximate the belief distribution, so by definition it can't be more accurate than computing the actual belief distribution itself

(v) [1 pt] Your friend also claims that using particle filtering with 100 particles to estimate the belief distribution for this model will require less computation and time than using the forward algorithm. Is your friend correct?

○ Yes ● No

For the forward algorithm on 100 particles, it requires processing/computing 100 different numbers whereas for the forward algorithm, we only need to keep track of 2 numbers $B(+c)$ and $B(-c)$. In general you should not use particle filtering unless you have an extremely large amount of states.

You decide to listen to your friend and run particle filtering. At time i you have a particle that is in state $+c$. You also know that $A_i = +a$, $R_i = -r$, $A_{i+1} = -a$, and $R_{i+1} = -r$.

(vi) [1 pt] If possible, what is the probability that that particle is sampled into state $-c$ after the time elapse update?

.6 ○ Not possible with the given information

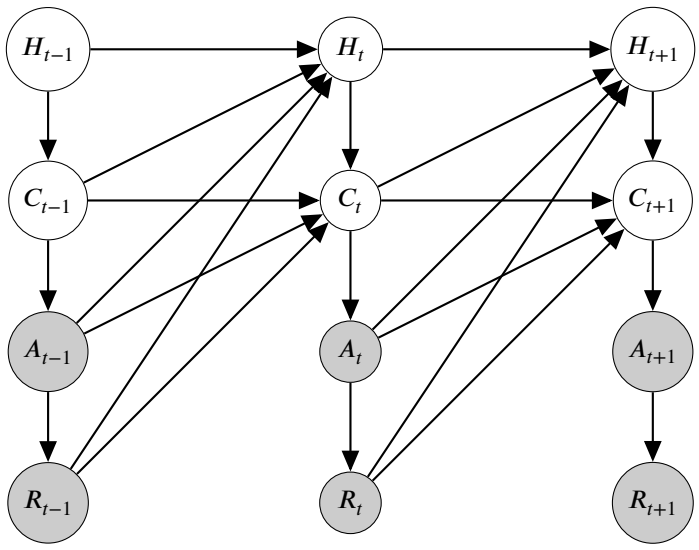
In the time update you sample from $P(C_{t+1}|C_t, A_t, R_t)$ so you are looking for $P(-c|+c,+a,-r)$. However we only gave you $P(+c|+c,+a,-r) = .4$, but since there are only 2 values of C, $P(-c|+c,+a,-r) = 1-P(+c|+c,+a,-r) = .6$

(vii) [1 pt] If possible, assume that particle did get sampled into state $-c$. What is its weight during the observation update?

.7 ○ Not possible with the given information

In particle filtering you weight by the probability of evidence given the actual states. Thus the weight of this particle is just $P(-a|-c) = 0.7$. Note we ignore R again because R is independent of C given A.

(b) Humans have unconscious preferences that can influence their beliefs. We incorporate another hidden Markov layer, where H_t represents unconscious preferences that influence C_t. Our revised model is shown below.



(i) [1 pt] From the model above, indicate which independence relations hold true.

■ $R_t \perp\!\!\!\perp C_t | A_t$

■ $C_{t+1} \perp\!\!\!\perp C_{t-1} | C_t, H_t$

☐ $C_{t+1} \perp\!\!\!\perp C_{t-1} | C_t$
■ $C_{t+1} \perp\!\!\!\perp C_{t-1} | C_t, H_t, H_{t+1}$
■ $C_{t+1} \perp\!\!\!\perp C_{t-1} | C_t, H_t, A_t$
■ $C_{t+1} \perp\!\!\!\perp C_{t-1} | C_t, H_t, A_t, R_t$

All of the answers above except $C_{t+1} \perp\!\!\!\perp C_{t-1} | C_t$ have no active paths between the selected nodes, and are thus independent.
$R_t \perp\!\!\!\perp C_t | A_t$: Independent, because the path from $C_t \rightarrow A_t \rightarrow R_t$ is blocked since we are given A_t, and the path $C_t \rightarrow H_{t+1} \leftarrow R_t$ is blocked as a common effect triple.

$C_{t+1} \perp\!\!\!\perp C_{t-1} | C_t, H_t$: Having C_t and H_t given blocks the causal chains from C_{t-1} to C_{t+1}.

$C_{t+1} \perp\!\!\!\perp C_{t-1} | C_t$: The causal chain through H_t is active, since its no longer given.

$C_{t+1} \perp\!\!\!\perp C_{t-1} | C_t, H_t, H_{t+1}$: From the second choice, since knowing C_t and H_t is sufficient for independence between the two nodes, adding additional variables will still mean that they are independent.

$C_{t+1} \perp\!\!\!\perp C_{t-1} | C_t, H_t, A_t$: Same as the previous one.

$C_{t+1} \perp\!\!\!\perp C_{t-1} | C_t, H_t, A_t, R_t$: Same as the previous one.

The computation is difficult, so we use particle filtering with n particles. Particles can be defined in a few different ways. For the next three subparts, indicate whether the proposed particle definition is valid, which means it could be used to simulate a true distribution of the hidden variables while also incorporating observed A and R values.

(ii) [1 pt] $n/2$ of the particles code for H, and $n/2$ of the particles code for C.
○ Yes ● No
Having half of the particles code for H and half code for C can't represent the true distribution because they're dependent on each other. We need to make sure particles encode the proper dependencies between the random variables.

(iii) [1 pt] All n particles code for an (H, C) pair.
● Yes ○ No
Yes, since each particle properly encodes the relationship between H and C. It is as if you combined H and C into a single joint variable.

(iv) [1 pt] All n particles code for H, and each particle generates an additional particle sampled only from $P(C_t | H_t, C_{t-1}, A_{t-1}, R_{t-1})$ to represent C only when needed
● Yes ○ No
Yes, since sampling from $P(C_t | H_t, C_{t-1}, A_{t-1}, R_{t-1})$ for each particle coded for H also properly encodes the relationship between H and C.