CS 188 Fall 2021

Introduction to Artificial Intelligence

Final

- You have approximately 170 minutes.
- The exam is open book, open calculator, and open notes.
- For multiple choice questions,
  - ☐ means mark **all options** that apply
  - ◯ means mark a **single choice**

First name	
Last name	
SID	

For staff use only:

Q1.	Learning to Act	/15
Q2.	Fun with Marbles	/6
Q3.	Course Evaluations	/12
Q4.	Unsearchtainty	/5
Q5.	Phase <3	/11
Q6.	Particle Filtering on an MDP	/16
Q7.	Machine Learning	/10
Q8.	CSPs and Bayes Nets	/13
Q9.	Potpourri	/12
	Total	/100

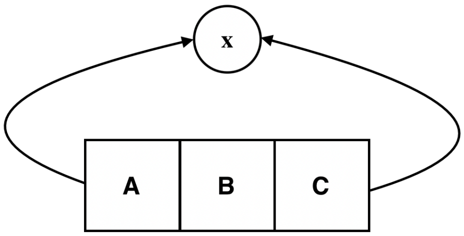
THIS PAGE IS INTENTIONALLY LEFT BLANK

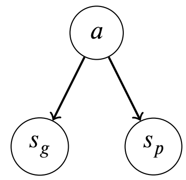
SID: _______________

Q1. [15 pts] Learning to Act

In lecture and discussion, we have mainly used the Naive Bayes algorithm to do binary classification, such as classifying whether an email is spam. However, we can also use Naive Bayes to learn how to act in an environment. This problem will explore learning good policies with Naive Bayes and comparing them to policies learned with RL.

We consider the following one-dimensional grid world environment with three squares, named A, B, and C from left to right. Pacman has two possible actions at each square: left and right. Taking the left action at square A or the right action at square C will transition to a terminal state x where no further action can be taken. At each timestep, Pacman observes his own position (s_p) as well as the ghost's position (s_g), and he uses these observations to decide on an action.





(a) In this part, Pacman has no idea about the transition probabilities of the ghost or the rewards it gets. However, Pacman has access to an expert demonstration dataset, which gives reasonably good actions to take in a number of scenarios. The dataset is divided into **training**, **validation**, and **test** datasets. The following is the **training set** of the dataset.

s_p	s_g	a
B	C	left
C	A	left
A	B	left
C	C	right
B	A	right
C	B	right

(i) [1 pt] **Q1.1** Using the standard Naive Bayes algorithm, what are the maximum likelihood estimates for the following conditional probabilities (encoded in the Bayes Net)?

$P(s_p = C \mid a = \text{left}) =$ ___1/3___

There are three left actions. Of these three, one entry has $s_p = C$.

$P(s_p = A \mid a = \text{right}) =$ ___0___

There are three right actions and no entry among these where $s_p = A$ so the probability is 0.

$P(a = \text{left}) =$ ___0.5___

There are 3 left actions out of 6 actions total.

(ii) [2 pts] **Q1.2** Using the standard Naive Bayes algorithm, which action should we choose in the following new scenarios?

$s_p = A, s_g = C$ ● Left ○ Right

Left: $P(s_p = A, s_g = C|a = left) = P(s_p = A|a = left) \cdot P(s_g = C|a = left) = \frac{1}{3} \cdot \frac{1}{3} = \frac{1}{9}$

Right: $P(s_p = A, s_g = C|a = right) = P(s_p = A|a = right) \cdot P(s_g = C|a = right) = 0 \cdot \frac{1}{3} = 0$

Left action has a higher probability so agent would choose left.

$s_p = C, s_g = B$ ○ Left ● Right

Left: $P(s_p = C, s_g = B|a = left) = P(s_p = C|a = left) \cdot P(s_g = B|a = left) = \frac{1}{3} \cdot \frac{1}{3} = \frac{1}{9}$

Right: $P(s_p = C, s_g = B|a = right) = P(s_p = C|a = right) \cdot P(s_g = B|a = right) = \frac{2}{3} \cdot \frac{1}{3} = \frac{2}{9}$

Right action has a higher probability so agent would choose right.

(iii) [2 pts] **Q1.3** Suppose we want to add Laplace smoothing with strength k in the Naive Bayes algorithm. (There is no smoothing when $k = 0$.) Which of the following are true?

☐ To find the optimal value of k, we pick the value of k which gives the highest accuracy on the training set.

■ To find the optimal value of k, we pick the value of k which gives the highest accuracy on the validation set.

☐ To find the optimal value of k, we pick the value of k which gives the highest accuracy on the test set.

■ If $k = 0$, we may observe low accuracy on the test set due to overfitting.

■ If k is a very large integer, the posterior probability for each action will be close to 0.5.

○ None of the above.

k is a hyperparameter and we should choose its optimal value based on performance on the validation set. Note that we cannot test hyperparameters on the test set and k is already being trained on the training set.
$k = 0$ means that there is no laplace smoothing so it is possible to overfit on the training data.
As k approaches ∞, the probability will approach uniform over all the possible actions which would be $1/2$.

(iv) [2 pts] **Q1.4** We can extend this approach to supervised learning algorithms other than Naive Bayes, such as logistic regression and neural networks.

Which of the following are true if we use another machine learning algorithm instead of Naive Bayes on a much larger, arbitrary training and test set, assuming that the training set contains no conflicting labels (i.e. two identical data points must have the same class label)?

■ If we use a binary perceptron, it might not terminate even after a sufficiently long time.

☐ If we use logistic regression, with some suitable hyperparameters it will classify all data points in the training set correctly.

■ If we use a large neural network, with some suitable hyperparameters it will classify all data points in the training set correctly.

☐ If we use a large neural network, with some suitable hyperparameters we should expect it to classify all the data points in the test set correctly.

○ None of the above

1. If the data is not linearly separable, binary perceptron will not terminate.
2. Logistic regression is not guaranteed to classify all the data points correctly if it is not linearly separable.
3 & 4. Since there are no conflicting labels, a large enough neural net will always be able to classify the training data correctly. However, this is not true for the test set which may contain unseen data.

(b) We now switch to using reinforcement learning to solve the problem. Specifically, we will use **model-based reinforcement learning** in this sub-part, though it is definitely possible to use model-free RL algorithms as well. We define a state as a tuple (s_p, s_g).

Assume that we have chosen a proper exploration policy, and collected 8 transition samples as follows: (Note: the data in the previous part should no longer be considered)

s	a	s'	r
(B, C)	right	(C, B)	-1
(C, B)	right	x	1
(B, C)	right	(C, C)	-5
(C, C)	right	x	0
(B, C)	right	(C, B)	-1
(C, B)	left	(B, A)	0
(B, A)	left	(A, A)	-5
(A, A)	left	x	0

(i) [1 pt] **Q1.5** What is the estimation of the following transition probabilities and rewards, based on the samples we collected?

$\hat{T}((B, C), right, (C, B)) =$ ____2/3____

There are three three entries (1, 3, 5) which contain (B, C), $right$ and of those, two arrive at successor state (C, B) (1 & 5).

$\hat{R}((B, C), right, (C, B)) =$ ____-1____

Averaging the rewards for both entries in the training data with $[(B, C), right, (C, B)]$, the reward is -1 in both cases.

(ii) [2 pts] **Q1.6** Which of the following statements are correct regarding model-based reinforcement learning vs. model-free reinforcement learning?

■ In model-based RL we explicitly learn a model for the transition probabilities and rewards, while in model-free RL we do not.

☐ In model-based RL we do exploration, while in model-free RL we do not.

☐ Model-based RL is off-policy, while model-free RL is on-policy.

☐ Model-based RL involves pure offline computation, while model-free RL requires online interaction with the environment.

1. Definition of model-based and model-free RL. See bottom of page 1 of note 5.
2. Model-based and model-free RL agents both use exploration to learn about the environment (rewards and transitions).
3. Model-free learning can be either off-policy or on-policy depending on the type of algorithm (ex. TD learning vs Q-learning).
4. Both model-based and model-free rerquire online interaction with the environment.

(iii) [2 pts] **Q1.7** We observe that a lot of the states are not covered in our small dataset of transitions. This is a general drawback of the naive model-based reinforcement learning algorithm. (e.g. When we have a larger grid, the size of the state space will be huge, and we need an even larger dataset of transitions).

Which of the following algorithmic improvements that we learned in lecture can be incorporated to alleviate this issue?

☐ Arc-consistency as an improvement of naive back-tracking

☐ Policy iteration as an improvement of value iteration

■ Approximate Q-learning as an improvement of naive Q-learning

☐ Variable elimination as an improvement of inference by enumeration

Only approximate Q-learning is relevant because it overcomes the large grid/state spaces by using a much smaller feature representations to approximate the model, allowing it to explore a small number of states and generalize to the much larger state space.

(c) Now we have seen supervised learning (part (a)) and reinforcement learning (part (b)), two potential methods to solve the decision-making problem.

For each of the following quantities, determine whether they hold for supervised learning only, RL only, both, or neither.

(i) [1 pt] **Q1.8** This method usually requires data from a near-optimal policy (e.g. a human expert) to work well.

● Supervised Learning ○ Reinforcement Learning ○ Both ○ Neither Supervised learning tries to replicate the given data, so if the data does not come from a near-optimal policy, it will also not learn a useful policy. RL algorithms are able to improve upon existing policies, hence does not require the policy to be initially near-optimal.

(ii) [1 pt] **Q1.9** This method requires us to know the transition probabilities of the MDP.

○ Supervised Learning ○ Reinforcement Learning ○ Both ● Neither Supervised learning directly learns from the data and does not care about transition probabilities. Model-free RL also does not care about transition probabilities. Model-based RL tries to estimate the transition probabilities from data, but we never actually acquire the ground-truth transition probabilities of the MDP.

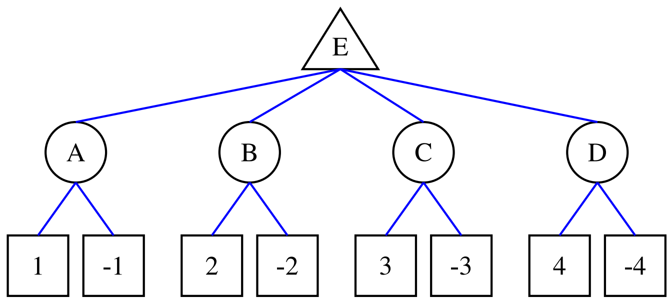
(iii) [1 pt] **Q1.10** This method requires us to design a reward function.

○ Supervised Learning ● Reinforcement Learning ○ Both ○ Neither Supervised learning does not need a reward function. RL algorithms need a reward function to work.

Q2. [6 pts] Fun with Marbles

Alice is playing a friendly, simplified game of marbles against Bob where each player starts with 10 marbles. Every turn, Alice chooses up to 4 of her marbles to bet, and Bob guesses whether the amount Alice is betting is even (2 or 4) or odd (1 or 3). If Bob guesses correctly, he gets to keep all the marbles that Alice bet and add them to his total count; else, Bob must give Alice the same amount of marbles that was bet. The game ends when one player has all 20 marbles.
Note that Alice and Bob never switch roles: Alice is always choosing marbles and Bob is always guessing at every turn.

(a) [2 pts] **Q2.1** Beginning the game, Alice wants to decide how much she should bet. She assumes that Bob has a 60% chance of picking even and a 40% chance of picking odd, and constructs the following depth 1 expectimax tree based on this first turn, modeling this as a zero-sum game where Alice is the maximizer. For the leaf nodes, she uses an evaluation function that outputs the **difference in marbles** she gets after Bob guesses:



Determine the value of all the nodes in the expectimax tree.

A: 0.2

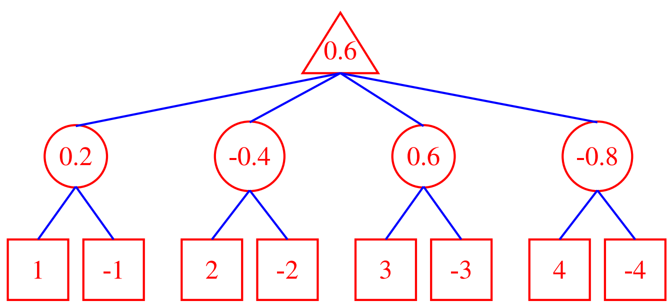
B: -0.4

C: 0.6

D: -0.8

E: 0.6

Branches correspond to $1, 2, 3, 4$ respectively. A leaf node's utility is positive when Bob guesses incorrectly, and negative when Bob guess correctly.



(b) [4 pts] Alice's friend Eve wants to make things more interesting, and offers a large cash prize to the winner of the game. However, for every turn that passes, Eve will reduce the prize by multiplying the amount remaining by some $\gamma \in (0, 1)$.

To model this, Alice incorporates γ into the tree, multiplying the values propagated up at each depth by γ.

For each letter **(A), (B), (C), (D), (E), (F), (G), (H)** select a single entry for the term corresponding to the correct equation to calculate the value of Alice's node represented as $V_A(s)$, where s represents the current Alice node. Denote the set of Alice's possible actions as a and the set of Bob's possible actions as b. Further, let $T_A(s, a, s')$ represent Alice's transition probability for taking an action a from state s to s' and let $T_B(s, b, s')$ represent Bob's transition probability of action b from some state s to s'.

$$V_A(s) = \textbf{(A)} \sum_{s'} \textbf{(B)}\ \textbf{(C)}\ \textbf{(D)}\ \textbf{(E)}\ \left[\textbf{(F)} + \textbf{(G)}\ V_A(\textbf{(H)})\right]$$

$$V_A(s) = \max_a \sum_{s'} T_A(s,a,s') \sum_{b,s''} P(b|s') T_B(s',b,s'') \gamma V_A(s'')$$

Q2.2 (A) ● $\max_a$ ○ $\max_b$ ○ $\max_{s'}$ ○ $\max_{s''}$ ○ $\sum_a$ ○ $\sum_b$ ○ $\sum_{s'}$ ○ $\sum_{b,s''}$ ○ 1

Q2.3 (B) ● $T_A(s,a,s')$ ○ $T_A(s',a,s'')$ ○ $T_B(s,b,s')$ ○ $T_B(s',b,s'')$ ○ 1

Q2.4 (C) ○ $\max_a$ ○ $\max_b$ ○ $\max_{s'}$ ○ $\max_{b,s''}$ ○ $\sum_a$ ○ $\sum_b$ ○ $\sum_{s'}$ ● $\sum_{b,s''}$ ○ 1

Q2.5 (D) ○ $P(a|s')$ ● $P(b|s')$ ○ $P(a,b|s')$ ○ $P(a,b|s'')$ ○ 1

Q2.6 (E) ○ $T_A(s,a,s')$ ○ $T_A(s',a,s'')$ ○ $T_B(s,b,s')$ ● $T_B(s',b,s'')$ ○ 1

Q2.7 (F) ○ γ ○ $T_A(s,a,s')$ ○ $T_A(s',a,s'')$ ○ $T_B(s,b,s')$ ○ $T_B(s',b,s'')$ ● 0 ○ 1

Q2.8 (G) ● γ ○ $T_A(s,a,s')$ ○ $T_A(s',a,s'')$ ○ $T_B(s,b,s')$ ○ $T_B(s',b,s'')$ ○ 0 ○ 1

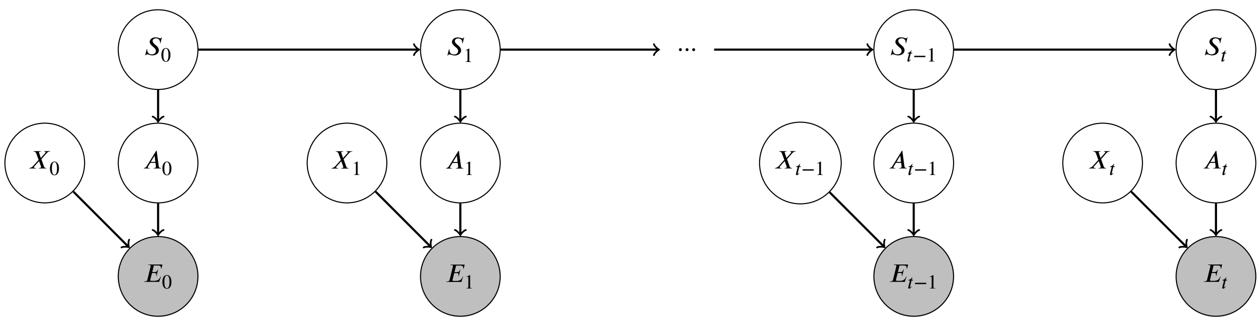
Q2.9 (H) ○ s ○ s' ● s'' ○ 1

Q3. [12 pts] Course Evaluations

Every semester we try to make CS 188 a little better. Let S_t represent the quality of the CS 188 offering at semester t, where $S_t \in \{1, 2, 3, 4, 5\}$. Changes to the course are incremental so between semester t and $t+1$, the value of S can only change by at most 1. Each possible transition occurs with equal probability. As examples: if $S_t = 1$, then $S_{t+1} \in \{1, 2\}$ each with probability $1/2$. If $S_t = 2$, then $S_{t+1} \in \{1, 2, 3\}$ each with probability $1/3$.

Let $E_t \in \{+e, -e\}$ represent the feedback we receive from student evaluations for the semester t, where $+e$ is generally positive feedback and $-e$ is negative feedback. Student feedback is dependent on whether released assignments were helpful for the given semester ($A_t \in \{+a, -a\}$) which is dependent on the quality of the semester's course offering (S_t). Additionally, student evaluations are also dependent on events external to the class ($X_t = \{+x, -x\}$).

The following HMM depicts the described scenario:



(a) Consider the above dynamic bayes net which ends at some finite timestep t. In this problem, we are trying to approximate the most likely value of S_t given all the evidence variables up to and including t. For each of the following subparts, first decide whether the given method can be used to solve this problem. Then, if yes, select all CPTs which must be known to run the algorithm.

(i) [1 pt] **Q3.1** Variable elimination

○ No ● Yes: ■ $P(S_0)$, $P(S_t|S_{t-1}), t > 0$ ■ $P(E_t|X_t, A_t)\forall t$ ■ $P(A_t|S_t)\forall t$ ■ $P(X_t)\forall t$

(ii) [1 pt] **Q3.2** Value iteration

● No ○ Yes: □ $P(S_0)$, $P(S_t|S_{t-1}), t > 0$ □ $P(E_t|X_t, A_t)\forall t$ □ $P(A_t|S_t)\forall t$ □ $P(X_t)\forall t$

(iii) [1 pt] **Q3.3** Gibbs sampling

○ No ● Yes: ■ $P(S_0)$, $P(S_t|S_{t-1}), t > 0$ ■ $P(E_t|X_t, A_t)\forall t$ ■ $P(A_t|S_t)\forall t$ ■ $P(X_t)\forall t$

(iv) [1 pt] **Q3.4** Prior sampling

○ No ● Yes: ■ $P(S_0)$, $P(S_t|S_{t-1}), t > 0$ ■ $P(E_t|X_t, A_t)\forall t$ ■ $P(A_t|S_t)\forall t$ ■ $P(X_t)\forall t$

(v) [1 pt] **Q3.5** Particle Filtering

○ No ● Yes: ■ $P(S_0)$, $P(S_t|S_{t-1}), t > 0$ ■ $P(E_t|X_t, A_t)\forall t$ ■ $P(A_t|S_t)\forall t$ ■ $P(X_t)\forall t$

First option is true because you can use variable elimination to solve for $P(S_t|E_{0:t})$ and then take the argmax over all possible values of S_t.
Second option is false because you are not solving for the max of S_t given the reward, actions, and previous value. Value iteration also does not consider evidence variables.
Third option is true because you can approximate the probability using gibbs sampling by sampling each variable over many iterations and using the final probability distribution as the result.
Fourth option is true because you can sample from all the CPTs to get $P(S_t|E_{0:t})$ and then take the argmax.
Fifth option is true because you can use particle filtering to approximate HMMs and determine the most likely S_t from the value with the most particles.

All methods which are valid will require using all of the CPTs in order to calculate the probability since the CPTs of $E_{0:t}$ and S_t depend on all the other variables in the DBN.

(b) For the HMM shown above, determine the correct recursive formula for the belief distribution update from $B(S_{t-1})$ to $B(S_t)$. Recall that the belief distribution $B(S_t)$ represents the probability $P(S_t|E_{0:t})$ and involves two steps: **(i)** Time elapse and **(ii)** Observation update.

$$B(S_t) \propto \underline{\quad\textbf{(ii)}\quad} \cdot \underline{\quad\textbf{(i)}\quad}$$

(i) [1 pt] **Q3.6** Time elapse

● $\sum_{S_{t-1}} P(S_t|S_{t-1})B(S_{t-1})$

○ $\sum_{S_{t-1}} \sum_{A_{t-1}} P(S_t|S_{t-1})P(A_{t-1}|S_{t-1})B(S_{t-1})$

○ $\sum_{S_{t-1}} P(S_t|S_{t-1})P(A_t|S_t)B(S_{t-1})$

○ $\sum_{S_{t-1}} \sum_{A_{t-1}} P(S_t|S_{t-1})P(A_{t-1}|S_{t-1})P(A_t|S_t)B(S_{t-1})$

This is the standard time elapse equation from the notes (see bottom of page 5 in note 8). Note that the extra variables do not affect that transition from S_t to its next state S_{t+1} so the time elapse expression is the same.

(ii) [1 pt] **Q3.7** Observation update

○ $P(E_t|X_t, A_t)$

○ $P(E_t|X_t, A_t)P(X_t)P(A_t|S_t)$

○ $\sum_{x\in X_t} \sum_{a\in A_t} P(E_t|x, a)$

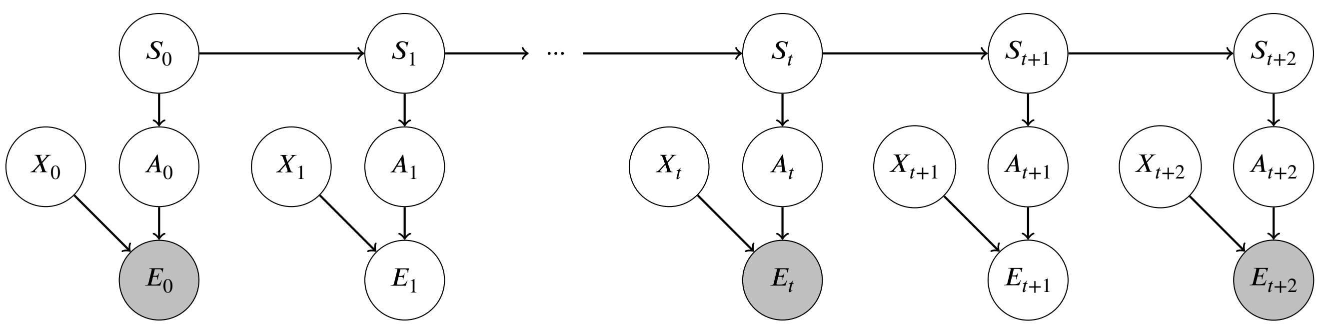
● $\sum_{x\in X_t} \sum_{a\in A_t} P(E_t|x, a)P(x)P(a|S_t)$

○ $\prod_{x\in X_t} \prod_{a\in A_t} P(E_t|x, a)$

○ $\prod_{x\in X_t} \prod_{a\in A_t} P(E_t|x, a)P(x)P(a|S_t)$

The observation update is the probability of evidence (E_t) given the state (S_t). Since there are extra hidden variables X_t and A_t at each timestep, these need to be summed out and eliminated to get the correct expression for $P(E_t|S_t)$ at each timestep.

Due to the differences between CS188 offerings in the fall and spring semesters, we realize that only student evaluations from past fall semesters are accurate enough to be incorporated into our model. Assume that a fall semester occurs during an even timestep and that t is even in the diagram below. The new HMM can be repesented as follows:



(c) [2 pts] **Q3.8** In this question, we are trying to derive a recursive formula for the two-step belief distribution update from $B(S_t)$ to $B(S_{t+2})$ for the new problem described above. Which of the following steps represent the **correct and most efficient** method of performing HMM updates to get the belief distribution at S_{t+2} from the current belief at S_t?
For the following notation, let $B(S_t) = P(S_t|E_{0:t:2})$ and $B'(S_t) = P(S_t|E_{0:t-1:2})$ where $E_{0:i:2}$ represents the set of all evidence variables at even timesteps up to i. Further, let $O(E_t)$ represent the value of the observation update expression from the previous part **(ii)**. (Note that $O(E_{t+1})$ and $O(E_{t+2})$ would represent the appropriate observation update expressions for timestep $t+1$ and $t+2$ respectively.)

● $B'(S_{t+1}) = \sum_{S_t} P(S_{t+1}|S_t)B(S_t)$
$B'(S_{t+2}) = \sum_{S_{t+1}} P(S_{t+2}|S_{t+1})B'(S_{t+1})$
$B(S_{t+2}) \propto O(E_{t+2})B'(S_{t+2})$

○ $B'(S_{t+2}) = \sum_{S_{t+1}} \sum_{S_t} P(S_{t+2}|S_{t+1})P(S_{t+1}|S_t)B(S_t)$
$B(S_{t+2}) \propto O(E_{t+2})B'(S_{t+2})$

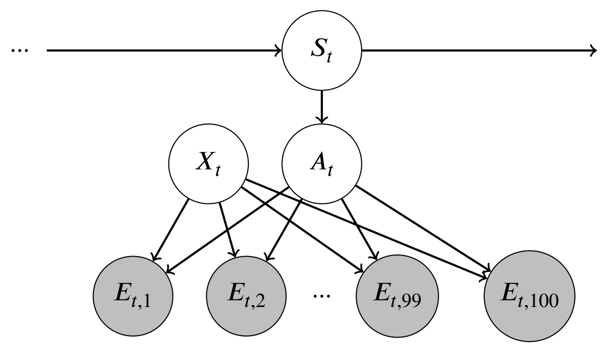
○ $B'(S_{t+1}) = \sum_{S_t} P(S_{t+1}|S_t)B(S_t)$
$B(S_{t+1}) = \sum_{E_{t+1}} O(E_{t+1})B'(S_{t+1})$
$B'(S_{t+2}) = \sum_{S_{t+1}} P(S_{t+2}|S_{t+1})B(S_{t+1})$
$B(S_{t+2}) \propto O(E_{t+2})B'(S_{t+2})$

○ $B'(S_{t+1}) = \sum_{S_t} P(S_{t+1}|S_t)B(S_t)$
$B(S_{t+1}) \propto O(E_{t+1})B'(S_{t+1})$
$B'(S_{t+2}) = \sum_{S_{t+1}} P(S_{t+2}|S_{t+1})B(S_{t+1})$
$B(S_{t+2}) \propto O(E_{t+2})B'(S_{t+2})$

○ None of the above.

Top left: $B'(S_{t+1})$ is a normal time elapse update. Since there is no evidence at timestep $t+1$, we do not perform an observation update and directly use the belief from $B'(S_{t+1})$ to calculate the new belief at $B'(S_{t+2})$. Then given evidence at timestep $t+2$, we update our belief at $B'(S_{t+2})$ with evidence to get $B(S_{t+2})$ in a normal observation update step.
Top right: This is the same as the first choice except the two time elapse steps are done in one equation to directly get $B'(S_{t+2})$. This is less efficient because you have to recompute $\sum_{S_t} P(S_{t+1}|S_t)B(S_t)$ at every timestep.
Bottom left: This is also equivalent to the first choice because $B(S_{t+2}) = B'(S_{t+2})$. Since we are summing over all possible values of evidence at E_{t+1}, this term becomes 1 and we are only left with $B'(S_{t+2})$. This is inefficient because extra calculation is done to sum out the unknown variables at odd timesteps.
Bottom right: This is the normal HMM update process, but since we don't have evidence for E_{t+1}, it doesn't make sense to incorporate evidence to get $B(S_{t+1})$ since the value of E_{t+1} is unknown.

(d) Now consider a scenario where instead of getting one general student feedback as evidence (E_t), we instead get individual student feedback from 100 students. Let the variable $E_{t,n}$ represent the evidence from student n at timestep t. Assume that the new evidence variables ($E_{t,n}$) can each take on the value of $+e$ or $-e$ with the same probability distribution as the single variable case (E_t).



(i) [2 pts] **Q3.9** Which of the following statements are true regarding this new setup?

☐ The evidence variables **within the same timestep** are independent of each other ($E_{t,j} \perp\!\!\!\perp E_{t,k}\ \forall j \neq k$).

☐ The evidence variables **between any two different timesteps** are independent of each other ($E_{t_1,j} \perp\!\!\!\perp E_{t_2,k}\ \forall t_1 \neq t_2$).

■ The expression to calculate the time elapse step from S_t to S_{t+1} for this new setup will be the same as the time elapse expression in the case of one evidence from Q3.6.

○ None of the above.

The first option is false because the variables are dependent through any connecting path (E-X-E, E-A-E, etc.).
The second option is false because evidence variables are still connected in an active path by common cause.
The third option is true because the change in evidence variables will only affect the observation update step.

(ii) [1 pt] **Q3.10** In the observation update at timestep t, we receive as evidence 60 positive evaluations ($+e$) and 40 negative evaluations ($-e$). Let x be the observation update probability at timestep t of observing one positive evaluation ($x = O(E_t = +e)$). Now, let $f(x)$ represent the new observation update expression for the case with the observed 100 evidence variables. Which of the following functions f gives the correct observation update for the new scenario? For this part only, regardless of your previous answer, please assume that each students' feedback is **independent** of each other.

○ $f(x) = x^{100}$

○ $f(x) = 60x \cdot 40(1-x)$

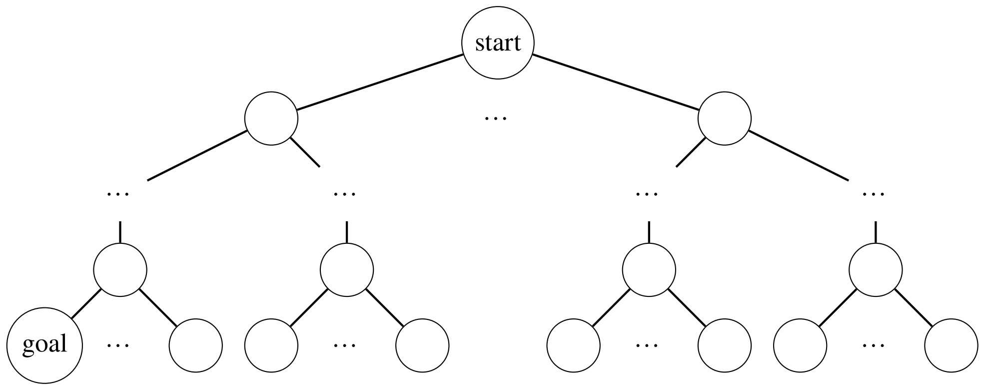
● $f(x) = x^{60} \cdot (1-x)^{40}$

○ None of the above.

The new observation update would be $P(E_1, ..., E_{100}|S)$. Since the evidence is independent, $P(E_1, ..., E_{100}|S) = \prod_t P(E_t|S) = (P(+e|S))^{60} \cdot (P(-e|S))^{40}$. Replacing $P(+e|S)$ with x since that was the probability solved before for one evidence variable, we would get $x^{60} \cdot (1-x)^{40}$.

Q4. [5 pts] Unsearchtainty

(a) You have a tree-structured undirected graph with branching factor b and depth d as shown below (where we consider the root to be at depth 0). You want to run A* **graph** search on this graph where your start state is the root node and there is exactly one goal node, which is at a leaf. All edges have cost 1 to traverse. You can traverse each edge either way: from the parent to its child, or from the child to its parent.



For the following two subparts, determine how many nodes would be expanded if you used the given heuristics. Recall that $h^*(n)$ represents the true cost of reaching the goal from a node n.

(i) [1 pt] **Q4.1** $h(n) = 0$

○ $O(d)$	○ $O(b^{d/2})$	○ $O(b \log d)$
○ $O(bd)$	○ $O(b^{d/3})$	○ $O((\log b)^d)$
● $O(b^d)$	○ $O(b^{d/4})$	○ $O(b^{\log d})$

With the constant zero heuristic, you are performing breadth-first search, and will expand all the nodes in the tree, of which there are $O(b^d)$.

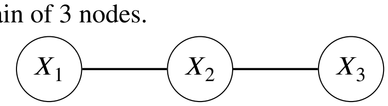
(ii) [1 pt] **Q4.2** $h(n) = h^*(n)$

● $O(d)$	○ $O(b^{d/2})$	○ $O(b \log d)$
○ $O(bd)$	○ $O(b^{d/3})$	○ $O((\log b)^d)$
○ $O(b^d)$	○ $O(b^{d/4})$	○ $O(b^{\log d})$

Let $f(n) = g(n)+h(n)$. With the true heuristic, the f value of each node is its distance from the root plus its distance to the goal. Therefore, for nodes on the shortest path to the goal, the f value will be d, while any nodes off the path to the goal will have a higher f value, taking into account the cost of deviating from the path. So, the only nodes to be expanded will be those along the path to the true goal, and there are $d + 1$ such nodes.

(b) This subpart is completely unrelated to the previous one. In this subpart, you are given a chain of nodes where the edge directions are unknown. For each of the following cases, assume that the direction of each edge is uniformly random and that the direction of any two edges is independent of each other.

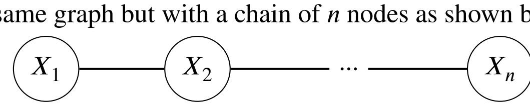
(i) [1 pt] **Q4.3** Consider the following chain of 3 nodes.



What is the probability that X_1 and X_3 are guaranteed to be independent? $\underline{\quad \frac{1}{4} \quad}$

With no given nodes, you must have a common effect for the triple to be inactive. So $X_1 \rightarrow X_2 \leftarrow X_3$. There are 2^2 possibilities for arrow direction combinations, so the probability of $X_1 \perp\!\!\!\perp X_n$ being guaranteed is $\frac{1}{2^2}$

(ii) [2 pts] **Q4.4** Now consider the same graph but with a chain of n nodes as shown below.



How many directed graphs can be formed where X_1 and X_n are **not guaranteed** to be independent? (You may use the variable n in your answer.)

n ____________

X_1 and X_N are not gauranteed to be independent when all the triples are a mix of causal chain or common cause. We can actually get all of the possible bayes nets where X_1 and X_n are not guaranteed to be independent by “sliding" a common cause triple across the entire bayes net, like so.

Graph 1:

$$X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow \cdots \rightarrow X_{n-2} \rightarrow X_{n-1} \rightarrow X_n$$

Graph 2:

$$X_1 \leftarrow X_2 \rightarrow X_3 \rightarrow \cdots \rightarrow X_{n-2} \rightarrow X_{n-1} \rightarrow X_n$$

Graph 3:

$$X_1 \leftarrow X_2 \leftarrow X_3 \rightarrow \cdots \rightarrow X_{n-2} \rightarrow X_{n-1} \rightarrow X_n$$

Graph $n-1$:

$$X_1 \leftarrow X_2 \leftarrow X_3 \leftarrow \cdots \leftarrow X_{n-2} \leftarrow X_{n-1} \rightarrow X_n$$

Graph n:

$$X_1 \leftarrow X_2 \leftarrow X_3 \leftarrow \cdots \leftarrow X_{n-2} \leftarrow X_{n-1} \leftarrow X_n$$

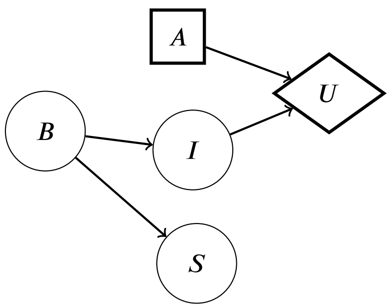
Thus, there are n different possible bayes net configurations where X_1 and X_n are guaranteed independent.

Q5. [11 pts] Phase <3

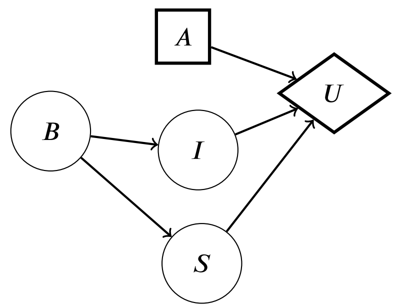
The EECS department decides to offer an exciting new course, CS1888, next semester. You and your friends are deciding if you want to take it or not, and decide to draw inspiration from CS188 and model this using decision networks. For all of the following parts, A represents the action of taking CS1888 or not, and U represents the utility function of a specific student.

(a) [1 pt] **Q5.1** Your friend Lexy will only take CS1888 if a specific instructor will be teaching it (I), and doesn't care about the course size or curriculum. However, due to uncertainty in instructor hiring practices in the university, the instructor chosen to teach is affected by the outcome of instructor collective bargaining efforts (B). The collective bargaining outcome will also affect the size of the course (S).

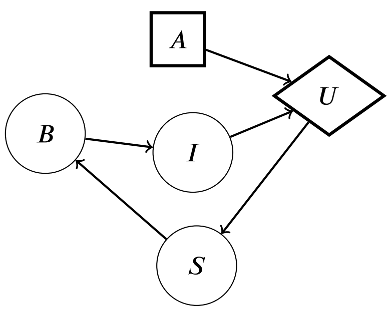
Select all of the decision networks which can represent Lexy's decision.



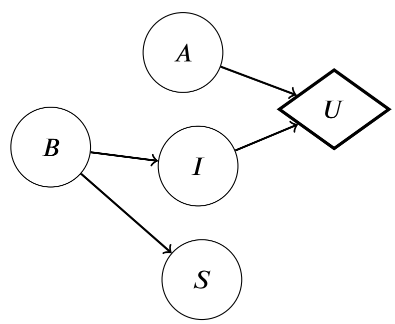
(1)



(2)



(3)



(4)

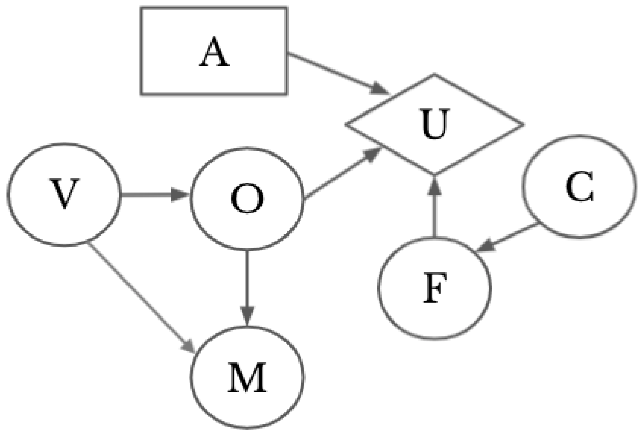
- [x] **(1)**
- [x] **(2)**
- [ ] **(3)**
- [ ] **(4)**
- ○ None of the above

The first option is correct since it contains edges encoding each dependence between variables in the problem statement. The second option is correct because it contains all the correct edges but with an additional edge between S and U. This edge does not guarantee independence so the Bayes net can still be used to represent Lexy's decision network.
The third option is false because there is a cycle among B, I, U, and S. This makes the Bayes net invalid.
The fourth option is false because the action is a chance node. However, in decision networks, the action is deterministic since the agent (in this case Lexy) has full control over the action she chooses.

Your friend Varun is also making the same decision. His decision is represented by the following decision network, where the chance nodes correspond to the following random variables:

- O: if classes will be online next semester
- V: the current virus situation

- M: the chancellor's message about how optimistic the university is about the virus situation
- F: whether Varun's friend takes CS1888
- C: whether Varun's friend's crush takes CS1888



Each variable has a binary domain. The conditional probability tables of variables are known, but not represented here.

(b) [2 pts] **Q5.2** We want $P(F,O|c)$, where c is a value that C can take on. Which of the following algorithms could be used for calculating this?

- [ ] Forward Algorithm
- [x] Prior Sampling
- [ ] Laplace Smoothing
- [x] Inference by Enumeration
- [ ] Particle Filtering
- ○ None of the Above

1. Forward algorithm is false since there is no notion in the decision network of time-based sequences with a transition or evidence model.
2. Prior sampling can be used to sample all variables and then use counting to determine the probability.
3. Laplace smoothing is used to prevent overfitting but cannot be used to calculate a probability.
4. Like option 1, particle filtering is false since there is no notion in the decision network of time-based sequences with a transition or evidence model.

(c) Varun wants to reason about the VPI of observing different evidence variables. For the following statements, select if they are always, sometimes, or never true.

(i) [1 pt] **Q5.3** $VPI(F) \geq VPI(C)$

- ● Always true
- ○ Sometimes true
- ○ Never true

Since C only affects the utility through F, the VPI of knowing F directly will always be higher than or equal to the VPI of knowing C.

(ii) [1 pt] **Q5.4** $VPI(O|V) + VPI(V|O) > VPI(V,O)$

- ○ Always true
- ○ Sometimes true
- ● Never true

VPI(V,O) = VPI(O|V) + VPI(V) = VPI(V|O) + VPI(O). We can see that VPI(V|O) will always be 0, while VPI(V) is not necessarily 0. So it is possible for the left side to be less than or equal to the right, but not greater

(d) [2 pts] **Q5.5** Varun peeks at his friend's crush's CalCentral and sees whether she is taking CS1888 ($C = c'$). Which of the following formulas represents the highest utility he has now?

● $\max_a \sum_f \sum_o P(f, o|c')U(f, o, a)$

○ $\max_a \sum_f P(f|c')U(f, a)$

○ $\sum_c P(c)[\max_a \sum_f \sum_o P(f, o|c')U(f, o, a)]$

○ $\sum_c P(c)[\max_a \sum_f P(f|c')U(f a)]$

We have to use the formula for MEU(e, e') instead of that for MEU(e, E'). The new evidence is no longer a random variable because Varun has observed C. Therefore, the last two options are incorrect. There is no such thing as a utility function without O, so the second option is incorrect.

(e) In the following question, we will investigate the idea of sampling applied in both Particle Filtering and Likelihood weighting applications.

(i) [2 pts] **Q5.6** Which of the following statements are true? Recall that in the particle filtering algorithm, we resample each particle at every timestep according to its weight.

■ HMMs require keeping track of probabilities over time, and particle filtering provides a more efficient method than the forward algorithm to approximate the true probability distribution.

☐ HMMs have a **constant** transition and sensor model across all timesteps, which is necessary to perform resampling at every iteration of particle filtering.

■ Resampling is necessary in particle filtering to ensure particles start each iteration with weight 1 while still representing a correct observation-updated distribution.

☐ Solving an HMM using particle filtering without the resampling step (instead storing the product of a particle's weight across all timesteps as the probability of that particle's state) would be equivalent to running the forward algorithm on an HMM.

○ None of the above.

1. Particle filtering and the forward algorithm are both methods to solve an HMM. Particle filtering is more efficient since instead of calculating the absolute probability for every state, it uses particles to approximate the probabilities and we only need to record p particle values at each time step rather than calculating the entire probability table.
2. A constant transition and observation model is not necessary for HMMs. It is also not necessary for resampling (you can just use the new transition models to calculate the weights and transitions for the appropriate timesteps).
3. This is true because after weighting each particle, we want to update the particle positions according to the more likely positions and reset the particles to weight 1 for the next iteration.
4. This is false bc particle filtering is always an approximation whereas the forward algorithm is exact.

(ii) [2 pts] **Q5.7** Consider the following modification of likelihood weighting: after getting an initial n samples and corresponding weights, resample a new set of n samples from the normalized weight distribution of the original samples. For the resulting n resampled samples, consider the following options:

- Method 1: Reweight each resampled sample with weight 1.
- Method 2: Keep the original weights of each resampled sample (ie. each sample still has weight equal to the probability of evidence given parents)

Select all the statements below that are true regarding either resampling method applied to likelihood weighting.

■ Method 1 would give a sampling method which is **consistent** with the underlying distribution.

☐ Method 2 would give a sampling method which is **consistent** with the underlying distribution.

☐ Using resampling reduces the potentially negative effect of downstream evidence on likelihood weighting calculations.

○ None of the above.

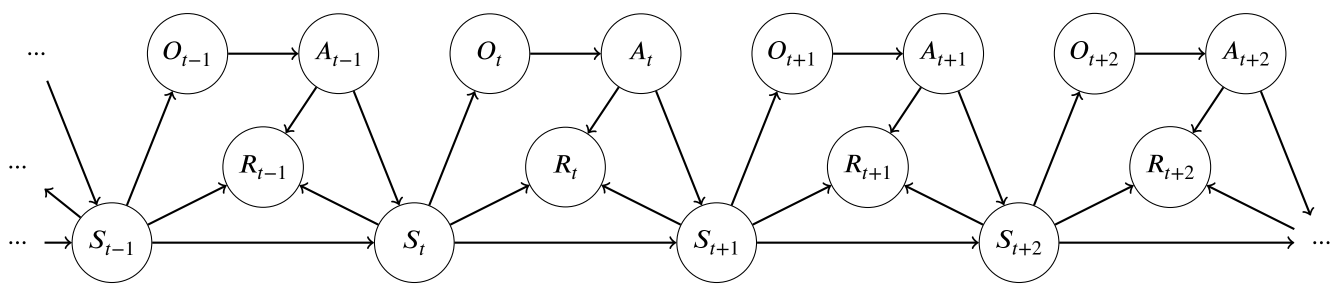
Method 1 is consistent because when we resample based on the normalized weight distribution, we would expect the resampled n samples, the probability of the sample being chosen is its normalized weight. Since this probability is already encoded in the resulting samples, we do not need to additionally weight each particle by its weight. In fact, doing so (As in method 2) would cause the weight probability to be applied twice (ie. it would be squared).
The final option is false because resampling does not affect how we generated the samples the first time (before resampling). Downstream evidence variables are not taken into consideration when creating the initial n samples and resampling is only sampling from this first pool of samples that were given, so it is still possible to have very low weights due to downstream evidence.

SID: ______________

Q6. [16 pts] Particle Filtering on an MDP

An agent is acting in an environment where they get noisy observations about their underlying state. The agent makes actions in a fixed, stochastic way based on their observation in the given timestep. The reward depends on the current state, the action taken, and the next state; however, unlike the MDPs we've seen in lecture, the reward function is **non-deterministic**.

We can represent this scenario by the Dynamic Bayes Net below:



(a) **(i)** [3 pts] **Q6.1** Which of the following independence assumptions are guaranteed to be true?

- [x] $S_{t-1} \perp\!\!\!\perp S_{t+1} \mid S_t$
- [ ] $O_{t-1} \perp\!\!\!\perp O_{t+1} \mid O_t$
- [ ] $A_{t-1} \perp\!\!\!\perp A_{t+1} \mid A_t$
- [ ] $R_{t-1} \perp\!\!\!\perp R_{t+1} \mid R_t$
- [ ] $A_{t-1} \perp\!\!\!\perp A_{t+1} \mid A_t, O_{0:t}$
- [x] $A_{t-1} \perp\!\!\!\perp A_{t+1} \mid A_t, O_{0:t+1}$
- [ ] $A_{t-1} \perp\!\!\!\perp A_{t+1} \mid A_t, O_{0:\infty}$
- [x] $R_{t-1} \perp\!\!\!\perp R_{t+1} \mid S_t$

Note that $A_{t-1} \perp\!\!\!\perp A_{t+1} \mid A_t, O_{0:\infty}$ is not true due to common effect. A_{t-1} and A_{t+1} share a common child S_{t+2} and knowing O_{t+2} as evidence is a downstream variable of S_{t+2} which causes the triple to become active.

(b) The agent wants to create a recursive algorithm to estimate its underlying state S_t. The agent knows what the observations, previous actions, and previous rewards are, but doesn't know the underlying states.

Remember, at a given timestep, the agent uses the observation for that timestep and (probabilistically) chooses the action to perform. It then observes the reward.

(i) [3 pts] **Q6.2** Determine an expression for the recursive relationship of $P(S_t \mid o_{1:t}, a_{1:t}, r_{1:t-1})$ from the previous timestep $P(s_{t-1} \mid o_{1:t-1}, a_{1:t-1}, r_{1:t-2})$. For the following equation, select the **minimum number** of answer choices that should be multiplied together to fill in the blank. Assume $t > 2$.

$$P(S_t \mid o_{1:t}, a_{1:t}, r_{1:t-1}) \propto \sum_{s_{t-1}} P(s_{t-1} \mid o_{1:t-1}, a_{1:t-1}, r_{1:t-2}) \cdot P(S_t \mid s_{t-1}, a_{t-1}) P(r_{t-1} \mid s_{t-1}, a_{t-1}, S_t) P(o_t \mid S_t) P(a_t \mid o_t)$$

$$\propto P(o_t \mid S_t) \sum_{s_{t-1}} P(S_t \mid s_{t-1}, a_{t-1}) P(r_{t-1} \mid s_{t-1}, a_t, s_t) P(s_{t-1} \mid o_{1:t-1}, a_{1:t-1}, r_{1:t-2})$$

- [ ] $P(S_t \mid s_{t-1})$
- [x] $P(S_t \mid s_{t-1}, a_{t-1})$
- [ ] $P(S_t \mid s_{t-1}, a_{t-1}, r_{t-1}, r_{t+1})$
- [ ] $P(r_{t-2} \mid s_{t-2}, a_{t-2}, S_{t-1})$
- [x] $P(r_{t-1} \mid s_{t-1}, a_{t-1}, S_t)$
- [ ] $P(r_t \mid s_t, a_t, S_{t+1})$
- [ ] $P(r_{t-1} \mid r_{t-2})$
- [x] $P(o_t \mid S_t)$
- [ ] $P(o_{t-1} \mid S_{t-1})$
- [ ] $P(a_t \mid o_t)$
- [ ] $P(S_{t+1} \mid s_t)$
- [ ] $P(S_{t+1} \mid s_t, a_t)$

Note that $P(a_t \mid o_t)$ is unnecessary because it is a constant factor applied too all the probabilities for varying states S_t so it is not included in the minimum representation since we are asking for a proportional expression.

(ii) [2 pts] We adapt particle filtering to approximate inference on this Dynamic Bayes Net. Below is pseudocode for the body of our loop. Fill in the blanks!

- **Q6.3** Elapse time: For each particle s_t, sample successor s_{t+1} from $P(S_{t+1} \mid s_t, a_t)$

  ○ $P(S_{t+1} \mid s_t)$ ● $P(S_{t+1} \mid s_t, a_t)$ ○ $P(S_{t+1} \mid s_t, a_t, r_t)$ ○ $P(S_{t+1} \mid s_t, a_t, r_t, r_{t+1})$

- **Q6.4** Incorporate Evidence: For each particle s_{t+1}, weight the sample by $P(o_{t+1} \mid s_{t+1})P(r_t \mid s_t, a_t, s_{t+1})$

  ○ $P(o_{t+1} \mid s_{t+1})$ ○ $P(r_t \mid s_t, a_t, s_{t+1})$ ○ $P(a_{t+1} \mid o_{t+1})$
  ● $P(o_{t+1} \mid s_{t+1})P(r_t \mid s_t, a_t, s_{t+1})$ ○ $P(o_{t+1} \mid s_{t+1})P(a_{t+1} \mid o_{t+1})$ ○ None of these

- Resample particles from the normalized weighted particle distribution distribution.

(c) Now, we are the agent's Swiss banker. We can only see the rewards (which the agent deposits into the bank) and can't see the agent's actions or observations.

(i) [3 pts] For each letter **(A), (B), (C), (D), (E), (F)** select a single entry for the term corresponding to the **correct and most efficient** recursive expression to determine $P(S_t, O_t, A_t \mid r_{1:t-1})$ from $P(s_{t-1}, o_{t-1}, a_{t-1} \mid r_{1:t-2})$.

$$P(S_t, O_t, A_t \mid r_{1:t-1}) \propto \textbf{(A) (B) (C) (D) (E) (F)}\ P(s_{t-1}, o_{t-1}, a_{t-1} \mid r_{1:t-2})$$

Q6.5 (A): ○ 1 ● $P(O_t \mid S_t)$ ○ $P(O_t \mid O_{t-1})$ ○ $P(r_{t-1} \mid S_{t-1}, a_{t-1}, S_t)$ ○ $P(r_{t-1} \mid r_{t-2})$

Q6.6 (B): ○ 1 ● $P(A_t \mid O_t)$ ○ $P(A_t \mid A_{t-1})$ ○ $P(S_t \mid S_{t-1}, A_{t-1})$ ○ $P(S_t \mid S_{t-1})$

Q6.7 (C): ○ 1 ● $\sum_{s_{t-1}, a_{t-1}}$ ○ $\sum_{s_{t-1}, a_{t-1}, o_{t-1}}$ ○ $\sum_{a_{t-1}}$ ○ $\sum_{s_{t-1}}$ ○ $\sum_{o_{t-1}}$

Q6.8 (D): ○ 1 ○ $P(a_t \mid O_t)$ ○ $P(a_t \mid a_{t-1})$ ● $P(S_t \mid s_{t-1}, a_{t-1})$ ○ $P(S_t \mid s_{t-1})$

Q6.9 (E): ○ 1 ○ $P(O_t \mid S_t)$ ○ $P(O_t \mid O_{t-1})$ ● $P(r_{t-1} \mid s_{t-1}, a_{t-1}, S_t)$ ○ $P(r_{t-1} \mid r_{t-2})$

Q6.10 (F): ○ 1 ○ $\sum_{s_{t-1}}$ ● $\sum_{o_{t-1}}$ ○ $\sum_{a_{t-1}}$ ○ $\sum_{r_{t-1}}$

$$P(S_t, O_t, A_t \mid r_{1:t-1}) \propto P(O_t \mid S_t)P(A_t \mid O_t) \sum_{s_{t-1}} \sum_{a_{t-1}} P(S_t \mid s_{t-1}, a_{t-1})P(r_{t-1} \mid s_{t-1}, a_{t-1}, S_t) \sum_{o_{t-1}} P(s_{t-1}, o_{t-1}, a_{t-1} \mid r_{1:t-2})$$

(ii) [4 pts] We again want to adapt particle filtering to approximate inference. Below is the pseudocode for the body of our loop. Each particle contains a state s_t, observation o_t, and action a_t.

- **Q6.11** Elapse Time: For each particle (s_t, o_t, a_t), sample the successor state s_{t+1} from $P(S_{t+1} \mid s_t, a_t)$

  ○ $P(S_{t+1} \mid s_t)$ ● $P(S_{t+1} \mid s_t, a_t)$ ○ $P(S_{t+1} \mid s_t, a_t, r_t)$ ○ $P(S_{t+1} \mid s_t, a_t, r_t, r_{t+1})$

- **Q6.12** Incorporate evidence: For each new s_{t+1}, assign weight $P(r_t \mid s_t, a_t, s_{t+1})$

  ○ $P(o_{t+1} \mid s_{t+1})$ ● $P(r_t \mid s_t, a_t, s_{t+1})$ ○ $P(a_{t+1} \mid o_{t+1})$
  ○ $P(o_{t+1} \mid s_{t+1})P(r_t \mid s_t, a_t, s_{t+1})$ ○ $P(o_{t+1} \mid s_{t+1})P(a_{t+1} \mid o_{t+1})$ ○ None of these

- Resample the s_{t+1} according to the normalized weighted distribution on s_{t+1}.
- **Q6.13** Finally, sample the following successor values:

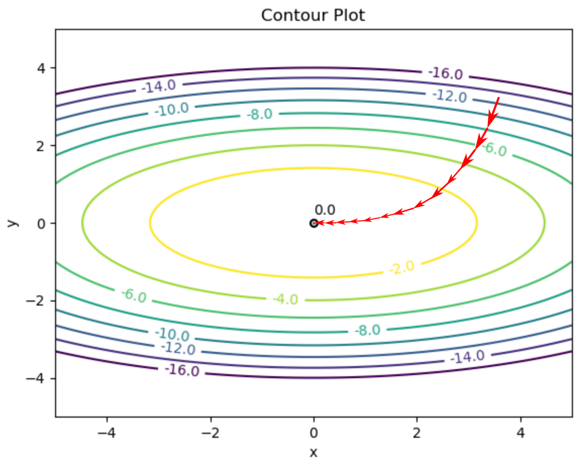
  ■ Sample o_{t+1} from $P(O_{t+1} \mid S_{t+1})$
  ■ Sample a_{t+1} from $P(A_{t+1} \mid O_{t+1})$
  □ Sample s_{t+1} from $P(S_{t+1} \mid S_t, A_t)$
  □ Sample r_{t+1} from $P(R_{t+1} \mid S_{t+1}, A_{t+1}, S_{t+2})$

(iii) [1 pt] **Q6.14** Alternatively, we could sample the successor values (thus complete our particle) before incorporating evidence. Then, when resampling, we would resample completed particles from the weighted particle distribution. Which method would you expect to be more accurate? ● Original method ○ Alternate method

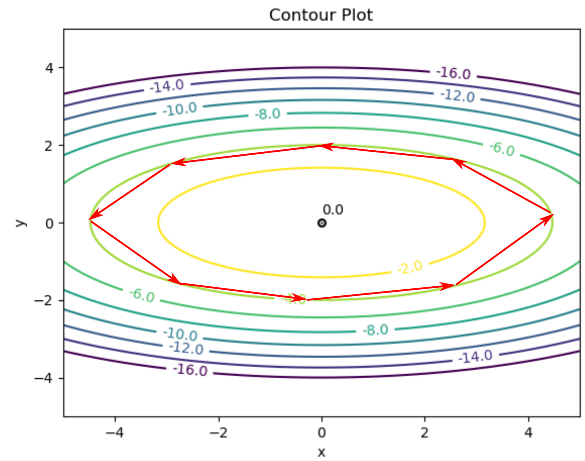
The "original method" of particle filtering is better since we're sampling o_{t+1} and a_{t+1} based on the new timestep. Otherwise, all the resampled particles that came from the same s_t would have the same a_{t+1} and o_{t+1}. The "original method" samples the action and observation for each new particle separately, so it's not as likely to get swayed by the way the random chance played out for the original sample.

Q7. [10 pts] Machine Learning

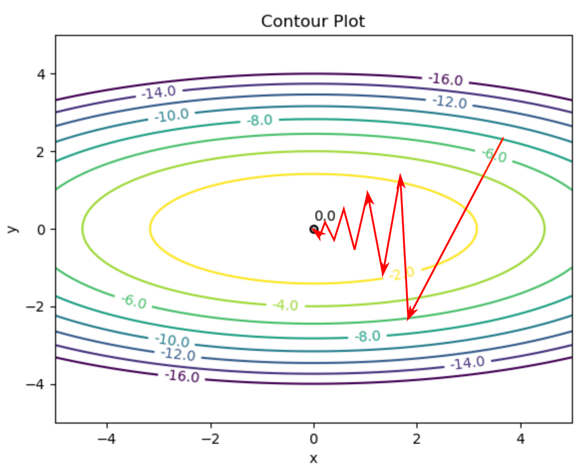
(a) **(i)** [2 pts] **Q7.1** The following four figures contain the steps of possible gradient ascent implementations each for some learning rate. Each arrow designates a step of gradient ascent. The numbers in the contour lines denote the value of the function we are maximizing on that contour. Which of the following figures, if any, contain paths that could be generated by gradient ascent?



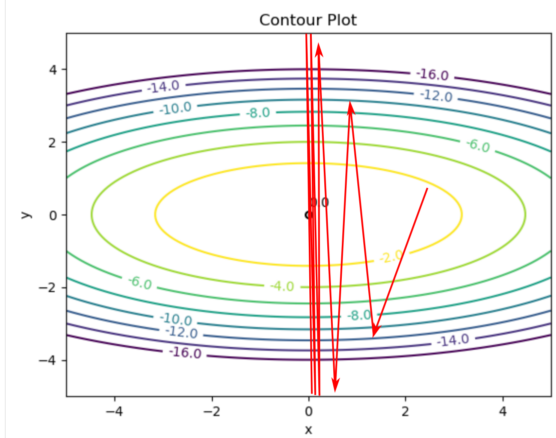
A.



B.



C.



D.

■ A. ☐ B. ■ C. ■ D.

A and C show gradient ascent steps in which C has a larger learning rate. D also shows gradient ascent steps with a learning rate large enough to make the method diverge. In B on the other hand the gradient directions are not perpendicular to the contour lines as they should be, since gradients point towards the steepest ascent-descent direction.

(ii) [2 pts] **Q7.2** Assume that we are trying to maximize a function that has many local maxima using gradient ascent. Which of the following variations of gradient ascent are likely to help obtain a better solution than what would be achieved with basic gradient ascent that uses a fixed step size α?

■ Once gradient ascent has converged, store the solution, randomly perturb the solution, and run gradient ascent again until convergence. Repeat this process K times and return the best solution.

■ Run N independent gradient ascent methods starting from N different initial solutions. Out of the ones that converge choose the best solution.

☐ For the chosen magnitude of the learning rate $|\alpha|$, at each step choose its sign with equal probability.
☐ After each iteration, increase the learning rate linearly.

Gradient ascent is not guaranteed to converge to the global minimum when the function has many local maxima. If we store the solution after convergence, perturb the solution and then rerun gradient ascent from the perturbed solution it is possible that the "new" gradient ascent will return a better solution. It is quite likely that at least one of these "new" gradient descents will return a solution better than the first one.
Running N independent gradient ascent methods and choosing the best solution is clearly beneficial as compared to a single gradient ascent implementation.
Randomly choosing the sign of the learning rate would switch between performing gradient descent and gradient ascent updates which is not what we want given that we are maximizing a function.
Finally, increasing the learning rate at each iteration is obviously not likely to help as in that case the method would most likely diverge after some point.

(b) **(i)** [1 pt] **Q7.3** Assume that we observe some data features $x_1^{(i)} \in \mathbb{R},\ i = 1, \ldots, N$ and outputs $y^{(i)} \in \mathbb{R},\ i = 1, \ldots, N$. We will use a linear model of the form $\hat{y}^{(i)} = a + bx_1^{(i)}$ to predict the output y. If we use the squared euclidean norm as an objective, i.e. $loss = \frac{1}{2}\sum_{i=1}^N \left(\hat{y}^{(i)} - y^{(i)}\right)^2$ and a learning rate α, which of the following is the correct gradient descent rule for finding the optimal value of b?

○ $b \leftarrow b + \alpha \sum_{i=1}^N (a + bx_1^{(i)} - y^{(i)})x_1^{(i)}$
● $b \leftarrow b - \alpha \sum_{i=1}^N (a + bx_1^{(i)} - y^{(i)})x_1^{(i)}$
○ $b \leftarrow b + \alpha \sum_{i=1}^N (a + bx_1^{(i)} - y^{(i)})$
○ $b \leftarrow b - \alpha \sum_{i=1}^N y^{(i)}x_1^{(i)}$

The gradient of the objective function with respect to b is $\frac{\partial loss}{\partial b} = \sum_{i=1}^N (a + bx_1^{(i)} - y^{(i)})\frac{\partial a + bx_1^{(i)}}{\partial b} = \sum_{i=1}^N (a + bx_1^{(i)} - y^{(i)})x_1^{(i)}$. Given that we perform gradient descent we need a negative sign for the gradient.

Now we observe the following dataset in which the outputs y are binary. Furthermore, we are given three features x_1, x_2, and x_3. The training dataset is as follows:

y	1	1	0	0
x_1	1	0	1	0
x_2	0	1	1	0
x_3	0	0	0	0

(ii) [1 pt] **Q7.4** We decide to use logistic regression for this classification task. What is the shape of the decision boundary we obtain from logistic regression?

● Linear
○ Nonlinear in general
○ Sigmoid
○ None of the above

The decision boundary we obtain from logistic regression is linear.

(iii) [2 pts] **Q7.5** We decide to train a neural network to perform classification on the whole dataset shown above. We propose using three hidden layers each of dimension 10 with a ReLu activation function except for the output layer which uses a sigmoid activation function. We will test the classifier on new unseen test data. Which of the following scenarios are likely to occur?

■ The network will achieve zero classification error on the training set.
■ The network will overfit the data.
☐ The network will underfit the data.
☐ The network will achieve zero classification error on the test set.

The network has many parameters and our training dataset is very small. This would lead to overfitting the data and achieving most likely zero training error while it will most likely perform poorly on the test set.

(iv) [2 pts] **Q7.6** This is a more general question independent of the aforementioned datasets. Assume that we have trained a logistic regression classifier on the last dataset obtaining the weight vector $\mathbf{w} = [w_1, w_2, w_3]$ and we observe a new feature vector $\mathbf{x}' = [x_1', x_2', x_3']$. Let $P(y = 1|\mathbf{x}; \mathbf{w}) = \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{x}}}$, with $\cdot$ denoting the inner product. Which of the following formulas describe the correct classification rule for the label y' of this data point?

■ $y' = \underset{y' \in \{-1,1\}}{\text{argmax}}\, P(y'|\mathbf{x}'; \mathbf{w})$

□ $y' = \begin{cases} 1, & \text{if } P(y' = 1|\mathbf{x}'; \mathbf{w}) \geq 0 \\ -1, & \text{otherwise} \end{cases}$

■ $y' = \begin{cases} 1, & \text{if } P(y' = 1|\mathbf{x}'; \mathbf{w}) \geq 0.5 \\ -1, & \text{otherwise} \end{cases}$

□ $y' = \begin{cases} 1, & \text{if } P(y' = 1|\mathbf{x}'; \mathbf{w})(1 - P(y' = 1|\mathbf{x}'; \mathbf{w})) \geq 0 \\ -1, & \text{otherwise} \end{cases}$

In logistic regression the logistic function $P(y|\mathbf{x}; \mathbf{w}) = \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{x}}}$ is used to model the probability of of $y = 1$. Hence we use 0.5 as a cutoff value and any feature $\mathbf{x}$ with that value being higher than 0.5 is classified as $y' = 1$ and for any value less than 0.5 we classify it as $y = 0$.

Q8. [13 pts] CSPs and Bayes Nets

(a) **(i)** [1 pt] **Q8.1** *True/False*: When enforcing arc consistency in a CSP, the set of values which remain when the algorithm terminates does not depend on the order in which arcs are processed from the queue.

● True ○ False

(ii) [1 pt] **Q8.2** In a general CSP with n variables, each taking d possible values, what is the maximum number of times a backtracking search algorithm might have to backtrack (i.e. the number of the times it generates an assignment, partial or complete, that violates the constraints) before finding a solution or concluding that none exists?

○ 0 ○ $O(1)$ ○ $O(nd^2)$ ○ $O(n^2d^3)$ ● $O(d^n)$ ○ ∞

In general, the search might have to examine all possible assignments.

(iii) [1 pt] **Q8.3** What is the maximum number of times a backtracking search algorithm might have to backtrack in a general CSP, if it is running arc consistency and applying the MRV and LCV heuristics?

○ 0 ○ $O(1)$ ○ $O(nd^2)$ ○ $O(n^2d^3)$ ● $O(d^n)$ ○ ∞

The MRV and LCV heuristics are often helpful to guide the search, but are not guaranteed to reduce backtracking with arc consistency in the worst case.

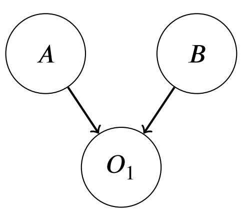
In fact, CSP solving is NP-complete.

(b) We consider how to solve CSPs by converting them to Bayes nets.

First, assume that you are given a CSP over two binary variables, A and B, with **only one constraint:** $A = B$. Note that $A = B$ is satisfied when both are positive ($A = +a$ and $B = +b$) or both are negative ($A = -a$ and $B = -b$).

As shown below, the Bayes net is composed of nodes A and B, and an additional node for a binary random variable O_1 that represents the constraint. $O_1 = +o_1$ when A and B satisfy the constraint, and $O_1 = -o_1$ when they do not. The way you will solve for values of A and B that satisfy the CSP is by running inference for the query $P(A, B \mid +o_1)$. The setting(s) of A, B with the highest probability will satisfy the constraints.

(i) [2 pts] **Q8.4 Determine the values** in the rightmost CPT that will allow you to perform the inference query $P(A, B \mid +o_1)$ with correct results.



A	$P(A)$
$-a$	0.5
$+a$	0.5

B	$P(B)$
$-b$	0.5
$+b$	0.5

A	B	O_1	$P(O_1 \mid A, B)$
$-a$	$-b$	$-o_1$	**(1)** 0
$-a$	$-b$	$+o_1$	**(2)** 1
$-a$	$+b$	$-o_1$	**(3)** 1
$-a$	$+b$	$+o_1$	**(4)** 0
$+a$	$-b$	$-o_1$	**(5)** 1
$+a$	$-b$	$+o_1$	**(6)** 0
$+a$	$+b$	$-o_1$	**(7)** 0
$+a$	$+b$	$+o_1$	**(8)** 1

$+o_1$ is an indicator for $a = b$ and $-o_1$ is an indicator for the complement $a \neq b$.

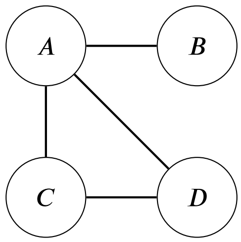
(ii) [2 pts] **Q8.5 Compute the Posterior Distribution:** Now you can find the solution(s) using probabilistic inference. Fill in the values below. Refer to the table in pt. (i).

O_1	A	B	$P(A, B \mid +o_1)$
$+o_1$	$-a$	$-b$	**(1)** $1/2$
$+o_1$	$-a$	$+b$	**(2)** 0
$+o_1$	$+a$	$-b$	**(3)** 0
$+o_1$	$+a$	$+b$	**(4)** $1/2$

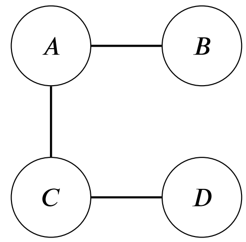
(c) Consider the following CSP:
Variables: $\{A, B, C, D\}$
Domains: $\{1, 2, 3, 4\}$ for each variable
Constraints:
$C_1 : A < 3$
$C_2 : B > 2$
$C_3 : C < D$

C_4 : $D < B$
C_5: all variables take on different values

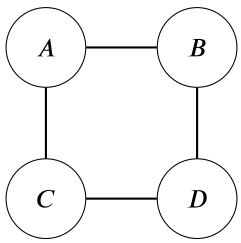
(i) [1 pt] **Q8.6** Choose which of the following indicates the constraint graph of this CSP:



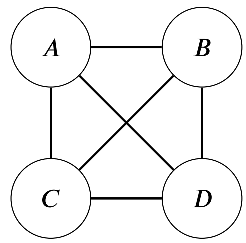
(1)



(2)



(3)



(4)

○ **(1)**
○ **(2)**
○ **(3)**
● **(4)**
○ None of the above

(ii) [1 pt] Enforce all unary constraints. For parts 8.7 to 8.10, mark all the values that **are pruned** from the domains of each node.

Variable	Domain			
A	1	2	3	4
B	1	2	3	4
C	1	2	3	4
D	1	2	3	4

Q8.7 *A*: ☐ 1 ☐ 2 ■ 3 ■ 4 ○ Nothing is pruned when enforcing unary constraints
Q8.8 *B*: ■ 1 ■ 2 ☐ 3 ☐ 4 ○ Nothing is pruned when enforcing unary constraints
Q8.9 *C*: ☐ 1 ☐ 2 ☐ 3 ☐ 4 ● Nothing is pruned when enforcing unary constraints
Q8.10 *D*: ☐ 1 ☐ 2 ☐ 3 ☐ 4 ● Nothing is pruned when enforcing unary constraints

Variable	Domain			
A	1	2	~~3~~	~~4~~
B	~~1~~	~~2~~	3	4
C	1	2	3	4
D	1	2	3	4

(iii) [2 pts] Run arc consistency. For parts 8.11 to 8.14, mark all the additional values that are pruned from each domain. **(Don't mark values that were already pruned from unary constraints).**i

Variable	Domain			
A	1	2	3	4
B	1	2	3	4
C	1	2	3	4
D	1	2	3	4

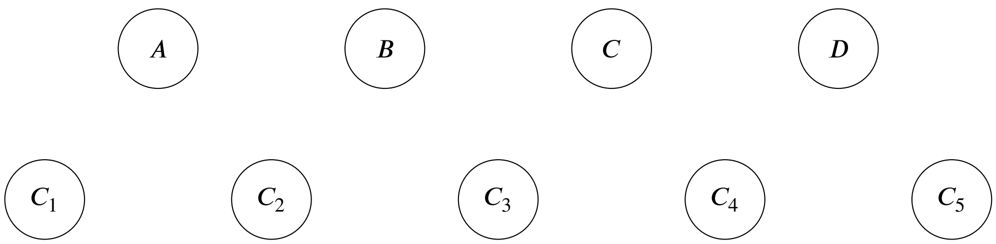
Q8.11 *A*: ☐ 1 ☐ 2 ☐ 3 ☐ 4 ● Nothing additional is pruned
Q8.12 *B*: ☐ 1 ☐ 2 ☐ 3 ☐ 4 ● Nothing additional is pruned
Q8.13 *C*: ☐ 1 ☐ 2 ■ 3 ■ 4 ○ Nothing additional is pruned

Q8.14 D: ■ 1 □ 2 □ 3 ■ 4 ○ Nothing additional is pruned

Variable	Domain			
A	1	2	~~3~~	~~4~~
B	~~1~~	~~2~~	3	4
C	1	2	~~3~~	~~4~~
D	~~1~~	2	3	4

(d) [2 pts] Now, in order to formulate the CSP from part (c) as a Bayes net we create 5 nodes for each of the constraints ($C_1, ..., C_5$).

We wish to construct the Bayes net below such that, when queried for the posterior $P(A, B, C, D \mid +c_1, +c_2, +c_3, +c_4, +c_5)$, will assign the highest probability to the setting(s) of values that satisfy all constraints. In the following graph, determine all the edges that need to be added to the Bayes Net.



For each subpart, select all nodes that should be a parent of the given constraint node:

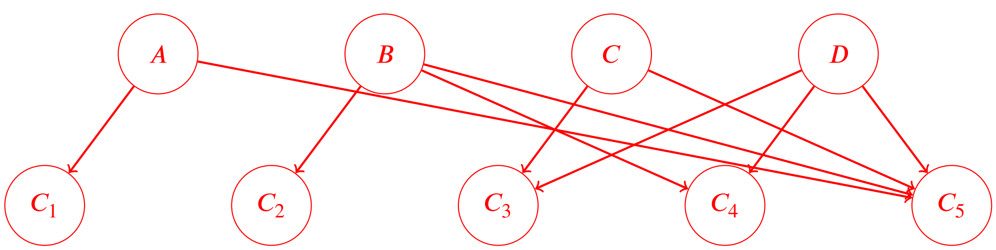
Q8.15 C_1: ■ A □ B □ C □ D □ C_1 □ C_2 □ C_3 □ C_4 □ C_5

Q8.16 C_2: □ A ■ B □ C □ D □ C_1 □ C_2 □ C_3 □ C_4 □ C_5

Q8.17 C_3: □ A □ B ■ C ■ D □ C_1 □ C_2 □ C_3 □ C_4 □ C_5

Q8.18 C_4: □ A ■ B □ C ■ D □ C_1 □ C_2 □ C_3 □ C_4 □ C_5

Q8.19 C_5: ■ A ■ B ■ C ■ D □ C_1 □ C_2 □ C_3 □ C_4 □ C_5



Q9. [12 pts] Potpourri

(a) **(i)** [2 pts] **Q9.1** Consider two heuristics h_1 and h_2. We are given that one of them is always equal to the true distance h^* and the other one is consistent, but not which one. Select all of the following heuristics h which are guaranteed to be consistent:

- ☐ $h = h_1 + h_2$
- ■ $h = \max(h_1, h_2)$
- ■ $h = \min(h_1, h_2)$
- ■ $h = p \cdot h_1 + (1 - p) \cdot h_2 \quad \forall\, p$ where $0 < p < 1$
- ☐ At each state $s \in S$, we randomly choose $h(s)$ to be either $h_1(s)$ or $h_2(s)$ with equal probability.
- ○ None of the above.

The first option is false because it overestimates.
The second option is true because the max of both heuristics is guaranteed to be the true optimal value.
The third option is true because the min of both heuristics is guaranteed to be the consistent heuristic.
The fourth option is true because the weighted sum of two consistent heuristics is consistent.
The fifth option is false. Consider the following scenario: A –(1)– B –(1)–C. If we chose the true distance heuristic at A and then the consistent zero heuristic at B, we would have $h(A) = 2$ and $h(B) = 0$ but this overestimates the edge (A, B).

(ii) [2 pts] **Q9.2** Consider 4 new heuristics h_1, h_2, h_3, and h_4 which are all **nonnegative**. We are given that $h_3 = h_1 + h_2$ and h_3 is admissible. Additionally, we know that h_4 is consistent. Select all of the following statements which are guaranteed to be true given the above information:
Recall that a heuristic h_i dominates h_j if the estimated goal distance for h_i is greater than or equal to the estimated goal distance for h_j at every node in the state space graph.

- ■ Both h_1 and h_2 are guaranteed to be admissible
- ■ $\frac{1}{2}\left(h_1 + h_2\right)$ is admissible
- ☐ h_4 dominates h_3
- ☐ $\max(h_3, h_4)$ is consistent
- ☐ $\min(h_3, h_4)$ is consistent
- ☐ $\frac{1}{2}\left(h_3 + h_4\right)$ is consistent
- ○ None of the above.

The first option is true because both h_1 and h_2 are nonnegative and their sum is admissible, so each heuristic on its own must be admissible.
The second option is true because we know that $h_1 + h_2$ is admissible so half their value must also be admissible.
The third option is false because we are not given any information about h_4 in relation to h_3. As an example, h_4 could be the zero heuristic which could never dominate any other admissible heuristic.
The fourth option is false because the max of both heuristics could be the admissible heuristic which would not guarantee consistency. (Ex: h_4 is the zero heuristic)
The fifth option is false because the min of both heuristics could be the admissible heuristic which would not guarantee consistency. (Ex: h_4 is the true distance and h_3 is admissible but inconsistent)
The sixth option is false. Consider a case where h_4 is the zero heuristic. Then this problem is reduced to asking if half of an admissible heuristic is consistent. This is not guaranteed.

(b) [2 pts] **Q9.3** Select all of the following statements about MDP and RL that are true.

- ■ Let π^* be the optimal policy. Then value iteration starting from random values will converge to $V^{\pi^*}(s)$ for all states.
- ☐ Approximate Q-learning is guaranteed to return the optimal policy upon convergence.
- ☐ In environments with deterministic transitions, no exploration is required for Q-learning to converge to the optimal policy.
- ■ A large discount factor γ (approaching 1) on an MDP means that the agent emphasizes long-term rewards.
- ■ Unlike ϵ-Greedy which requires randomness, using an exploration function provides a deterministic method to explore new states.

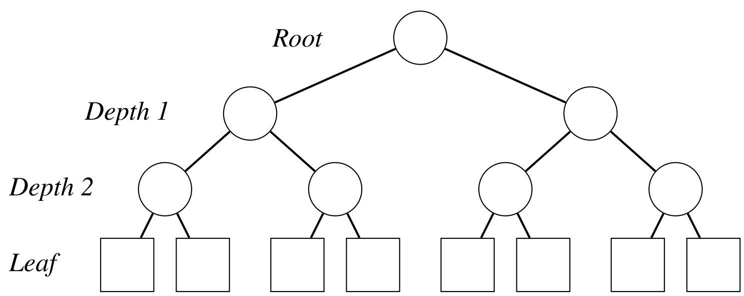
1. Under the optimal policy, value iteration will converge to the optimal value.
2. The linear function approximation (or any other approximation) is not guaranteed to model the Q-values sufficiently and thus nothing can be said about optimality.
3. A deterministic transition function means that from a specific state and for a given action the next state is fixed. Exploration though is still needed to learn optimal policies.
4. $\gamma \to 1$ puts more weight on future rewards than lower values of γ.
5. Exploration functions are deterministic since they assign an exploration bonus to states using a deterministic function (generally dependent on how often the state was visited).

(c) [2 pts] **Q9.4** Select all of the following statements about Variable Elimination that are true.

- ☐ To find an optimal elimination ordering, we always want to choose the variable involved in the least amount of edge connections.
- ☐ In variable elimination, we continue eliminating until every single node in the bayes net has been eliminated.
- ■ Variable elimination is at least as efficient as inference by enumeration.
- ☐ Variable elimination runs in polynomial time
- ○ None of the above

1. The optimal elimination ordering is the one that creates the smallest resulting factor. It is possible for a variable to be connected by one edge to a variable that has a very large domain vs another that is connected by two edges to binary variables. Also, if the variable is a query or evidence variable, we do not want to eliminate it regardless of its edges.
2. We don't want to eliminate our query and evidence variables.
3. This is true because variable elimination sums out to eliminate each variable as its factors are multiplied whereas inference by enumeration first multiplies all CPTs to make the full joint and then sums out variables.
4. Variable elimination is exponential in the number of variables.

(d) [2 pts] **Q9.5** Consider the following game tree which contains only expectation nodes and **leaf values which are strictly positive**.



For the following options, select the option if there exists a set of leaf values in the positive domain that would allow you to prune. Consider each option independently.

- ☐ It is possible to prune the tree in its current form.
- ☐ Replace the root node with a maximizer.
- ☐ Replace both nodes at depth 1 with maximizers.
- ☐ Replace all 4 nodes at depth 2 with maximizers.
- ■ Replace the root node with a minimizer.
- ■ Replace both nodes at depth 1 with minimizers.
- ☐ Replace all 4 nodes at depth 2 with minimizers.
- ○ None of the above

Option 1 is false because if all the nodes are expectation nodes, then we cannot prune because there is no upper bound to the value of any node.
For options 2 through 4 (maximizers): The leaf values are lower bounded which means that we cannot prune on an alpha value. There is no limit to the upper bound of the leaf values so a maximizer will never prune since it is possible for the expectation node values to increase infinitely with a new leaf value.

For the minimizer options, is possible to prune from the root or depth 1.
Root: Consider all 4 leaf nodes in left subranch are 0.1 and the fifth and sixth leaf nodes each have value 10. Then, we can prune the remaining subtree (leaves 7 and 8) because we know the value of the depth 1 right expectimax node is greater than 5 and the root minimizer is guaranteed to choose left (0.1).
Depth 1: Consider first 2 leaf values are 0.1 and third leaf is 10. Then we can prune the fourth leaf based on the left depth 1 minimizer.
Depth 2 is not a solution because the minimizers are directly above the leaf values and the higher depths do not have any bounds on them since they are all expectation nodes.

(e) [2 pts] **Q9.6** In class we saw a formulation for approximate Q-learning that uses a linear function on the features. Consider using the sigmoid function $s(x) = \frac{1}{1+e^{-x}}$ as our approximation function, which we could potentially do if we wanted to bound the values of the Q functions. Let α denote the learning rate and $w_i, f_i, i = 1.\dots, n$ denote the weights and features respectively. If we use $Q(s,a) = \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{f}}}$ as our Q-function approximation, where $\mathbf{w}\cdot\mathbf{f} = \sum_{i=1}^{n} w_i f_i(s,a)$, and use the squared norm metric for the approximation, which of the following is the correct update for parameter w_j?
Hint: The derivative of the sigmoid function is $s'(x) = s(x)(1-s(x))$.

- ○ $w_j' = w_j + \alpha\left(r + \gamma \max_{a'} Q_w(s',a') - Q_w(s,a)\right)\frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{f}}}\left(1 - \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{f}}}\right) f_j$
- ○ $w_j' = w_j + \alpha\left(r + \gamma \max_{a'} Q_w(s',a') - Q_w(s,a)\right) f_j$
- ● $w_j' = w_j - \alpha\left(r + \gamma \max_{a'} Q_w(s',a') - Q_w(s,a)\right)\frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{f}}}\left(1 - \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{f}}}\right) f_j$
- ○ $w_j' = w_j - \alpha\left(r + \gamma \max_{a'} Q_w(s',a') - Q_w(s,a)\right)\frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{f}}} f_j$

The derivative of the sigmoid function is $s'(x) = s(x)(1-s(x))$, hence the second and fourth options are wrong. The correct signs in the update rule are:

$$\begin{aligned}
w_j' &= w_j + \alpha\left(r + \gamma \max_{a'} Q_w(s',a') - \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{f}}}\right)\left(\frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{f}}}\right)' \\
&= w_j + \alpha\left(r + \gamma \max_{a'} Q_w(s',a') - \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{f}}}\right)\left(\frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{f}}}\right)\left(1 - \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{f}}}\right)\frac{\partial}{\partial w_j}(-\mathbf{w}\cdot\mathbf{f}) \\
&= w_j - \alpha\left(r + \gamma \max_{a'} Q_w(s',a') - \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{f}}}\right)\left(\frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{f}}}\right)\left(1 - \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{f}}}\right) f_j
\end{aligned}$$