CS 188 Fall 2023 — Introduction to Artificial Intelligence — Final

- You have 170 minutes.
- The exam is closed book, no calculator, and closed notes, other than four double-sided cheat sheets that you may reference.
- For multiple choice questions,
  - ☐ means mark **all options** that apply
  - ◯ means mark a **single choice**

First name	
Last name	
SID	
Name of person to the right	
Name of person to the left	
Discussion TAs (or None)	

Honor code: "As a member of the UC Berkeley community, I act with honesty, integrity, and respect for others."

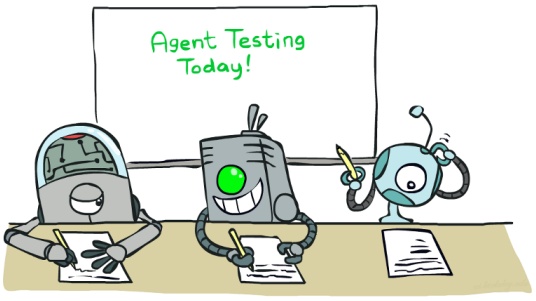
By signing below, I affirm that all work on this exam is my own work, and honestly reflects my own understanding of the course material. I have not referenced any outside materials (other than my cheat sheets), nor collaborated with any other human being on this exam. I understand that if the exam proctor catches me cheating on the exam, that I may face the penalty of an automatic "F" grade in this class and a referral to the Center for Student Conduct.

Signature: _____

Point Distribution

Q1.	Search: Squirrel Shenanigans	11
Q2.	CSPs and BNs: Bayes for Days	11
Q3.	MDPs and BNs: Coin Blackjack	10
Q4.	RL: Rest and ReLaxation	14
Q5.	Bayes' Nets: Pac Pong Performance	16
Q6.	HMMs and VPI: Markov Menagerie	13
Q7.	Games and ML: BeatBlue	12
Q8.	Potpourri	13
	Total	100

It's testing time for our CS188 robots!
Circle your favorite robot below.
(ungraded, just for fun)



Q1. [11 pts] Search: Squirrel Shenanigans

A squirrel is moving around on a grid representing the UC Berkeley campus, picking up and dropping off nuts, in order to store nuts in its nest for the upcoming winter.

At the start of the problem, each square on the $M \times N$ grid contains between 0 and A nuts, inclusive. (A is some constant that is fixed for the entire problem.) There cannot be more than A nuts on any square at any time.

At the start of the problem, the squirrel is carrying 0 nuts. The squirrel can only carry between 0 and A nuts, inclusive, at any time.

The squirrel has the following actions available, and every action costs 1:

- Pick up any number of nuts from the current square (as long as the total number of nuts carried does not exceed A).
- Drop off any number of carried nuts onto the current square (as long as the total number of nuts on the square does not exceed A).
- When the squirrel is carrying C nuts, the squirrel can move 1 to $A - C + 1$ squares in any of the four cardinal directions (up, down, left, right).

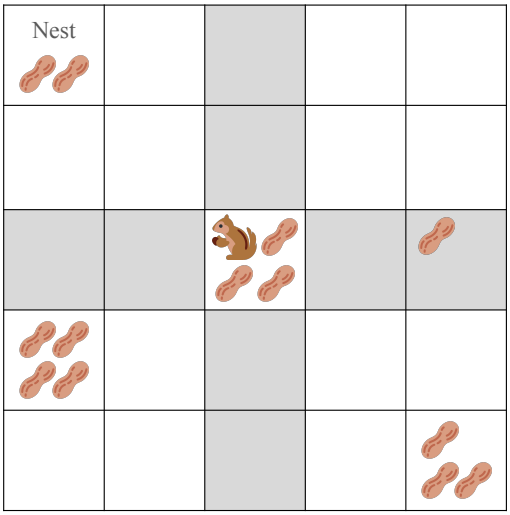
The squirrel's goal is to reach a state where the nest square contains A nuts. The nest square is always fixed to be the top-left corner of the grid.

Here is an example, though your answers should work for any arbitrary problem, not just the example shown.

In this example only, assume $A = 4$, the squirrel is currently carrying $C = 2$ nuts, and there are 3 additional nuts on the square the squirrel is on.

Possible actions from this state (each costing 1):

- Pick up 1 or 2 nuts from the current square. (The squirrel can't pick up 3 nuts, because it can't hold more than 4 nuts.)
- Drop off 1 nut on the current square. (The squirrel can't drop off 2 nuts, because the square can't hold more than 4 nuts.)
- Move to any of the shaded squares ($A - C + 1 = 2$ squares in each of the cardinal directions).



(a) [2 pts] For this subpart only, assume $N > A$ and $M > A$ (the board is very large).

What is the maximum branching factor for this search problem?

◯ 1	◯ $A + 4$	◯ $5A + 4$	◯ $MN + A$
◯ 4	◯ $4A + 4$	◯ MN	◯ AMN

(b) [1 pt] True or false: The squirrel's location needs to be part of the state space.

- ◯ True, because it's needed for the successor function.
- ◯ True, because it's needed for the goal test.
- ◯ False, because it's not needed for the successor function.
- ◯ False, because it's not needed for the goal test.

The actions available (each costing 1), repeated for your convenience:

- Pick up any number of nuts from the current square (as long as the total number of nuts carried does not exceed A).
- Drop off any number of carried nuts onto the current square (as long as the total number of nuts on the square does not exceed A).
- When the squirrel is carrying C nuts, the squirrel can move 1 to $A - C + 1$ squares in any of the four cardinal directions (up, down, left, right).

(c) [2 pts] Give a reasonably tight upper-bound on the size of the minimal state space.

Your answer should be in the form $p \cdot q^r$, where p, q, r are expressions that possibly include M, N, A, constants, and arithmetic operators (add, subtract, multiply, divide).

$p =$ $q =$ $r =$

(d) [3 pts] Select all of the admissible heuristics.

- ☐ A, minus the number of nuts on the nest square.
- ☐ 0 if the nest has A nuts. Otherwise, Manhattan distance to the closest non-nest square containing nuts.
- ☐ 0 if the nest has A nuts. Otherwise, Manhattan distance to the furthest non-nest square containing nuts.
- ○ None of the above

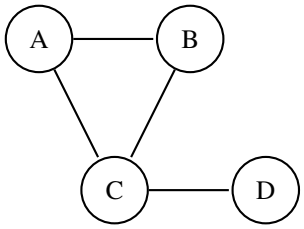
(e) [3 pts] For this subpart only, suppose we change the search problem: The squirrel is now only able to drop off or pick up a single nut in one time step.

Select all algorithms that can be used to find an optimal solution to this modified problem.

- ☐ Leave the successor function unchanged from the original problem, and run UCS.
- ☐ Modify the successor function so that the squirrel can only pick up or drop off one nut at a time (costing 1 each time). Then, run BFS with the modified successor function.
- ☐ Modify the costs in the successor function so that picking up or dropping off k nuts costs k, instead of 1. Then, run UCS with the modified successor function.
- ○ None of the above

Q2. [11 pts] CSPs and BNs: Bayes for Days

Consider the following incomplete Bayes' net, with four random variables, and four edges that do not have directions yet:

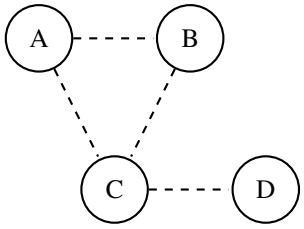


(a) [1 pt] How many undirected paths are there between A and D?

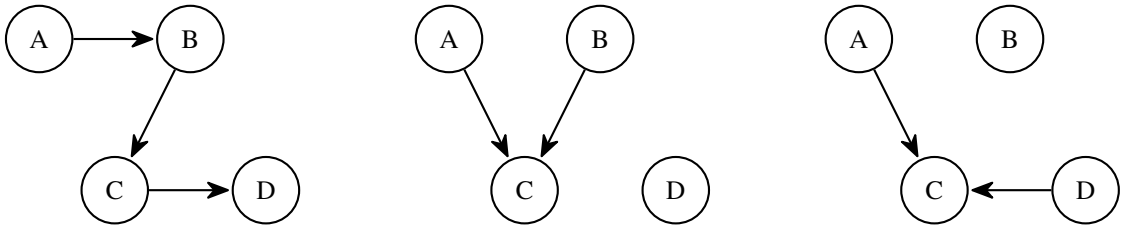
◯ 1 ◯ 2 ◯ 3 ◯ 4 ◯ > 4

(b) [2 pts] How many possible ways are there to assign directions to the arrows, such that A and D are independent?

For the rest of the question, consider the following incomplete Bayes' net:

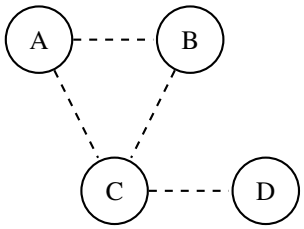


To complete this Bayes' net, each pair of variables connected by a dashed line can be filled in with an edge (in either direction), or no edge. Here are some examples of completed Bayes' nets:



Our goal is to complete this Bayes' net, such that $A \perp\!\!\!\perp B$ and $A \perp\!\!\!\perp D$.

The Bayes' net, repeated for your convenience:



We would like to use a CSP to find a way to complete this Bayes' net.

(c) [1 pt] What are the variables in this CSP?

- ◯ The 4 dashed edges
- ◯ The 4 random variables (A, B, C, D)
- ◯ The conditional independence assumptions

(d) [1 pt] How many possible values can each variable take on in this CSP?

◯ 1 ◯ 2 ◯ 3 ◯ 4 ◯ > 4

For each of the following statements, select all type(s) of constraints that are needed to represent that statement in the CSP. Each statement is independent.

(e) [2 pts] The graph must be acyclic.

☐ Unary constraint ☐ Binary constraint ☐ Three-way constraint ☐ Four-way constraint ◯ None of the above

(f) [2 pts] A and B are independent.

☐ Unary constraint ☐ Binary constraint ☐ Three-way constraint ☐ Four-way constraint ◯ None of the above

(g) [2 pts] A and D are independent.

☐ Unary constraint ☐ Binary constraint ☐ Three-way constraint ☐ Four-way constraint ◯ None of the above

Q3. [10 pts] MDPs and BNs: Coin Blackjack

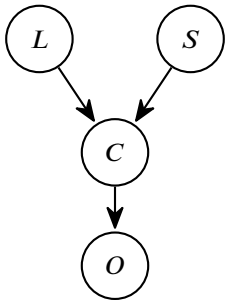
Consider the following game (similar to blackjack):

- You begin the game with a score of 0 points.
- At every time step, you can choose to flip a coin, or end the game with reward equal to your current score.
- When you choose to flip: if the coin comes up heads, then 1 point is added to your score. If the coin comes up tails, then 2 points are added to your score.
- If your score ever reaches 4 or more points, you bust and end the game with 0 reward.

There are two coins: a fair coin that comes up heads exactly half of the time, and a biased coin that comes up heads more than half of the time.

Each time you choose to flip a coin, the dealer will first randomly select one of the two coins, and then flip the chosen coin for you. The probability that the dealer selects the fair or biased coin depends on two things: the coin used on the previous flip, and your current score (before flipping).

The randomness of the coin flip can be modeled by the following Bayes' net:



- L: The coin that was last used on the previous flip (fair or biased).
- S: The current score (0, 1, 2, or 3).
- C: The selected coin (fair or biased).
- O: The outcome of the coin flip (heads or tails).

(a) [1 pt] From the Bayes' net, what is the most efficient way to learn the probability that the biased coin comes up heads? (CPT = conditional probability table)

- ◯ Read the value from the CPT in the C node.
- ◯ Read the value from the CPT in the O node.
- ◯ Perform variable elimination with C unobserved.
- ◯ Perform variable elimination with C observed as evidence.
- ◯ This probability is not in the Bayes' net.

We choose to model this game as an MDP. For each transition, select the corresponding transition probability.

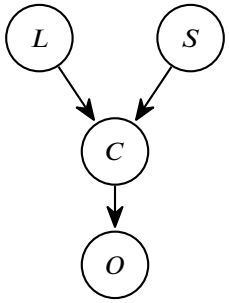
(b) [2 pts] State: The biased coin was last used. The current score is 1.

Action: Flip.

Successor state: The fair coin was last used. The current score is 3.

- ◯ 0
- ◯ $P(C = \text{fair} | L = \text{biased}, S = 1) \cdot P(O = \text{tails} | C = \text{fair})$
- ◯ $\sum_c P(C = c | L = \text{biased}, S = 1) \cdot P(O = \text{tails} | C = c)$
- ◯ $\sum_l P(C = \text{fair} | L = l, S = 1) \cdot P(O = \text{tails} | C = \text{fair})$
- ◯ 1

The Bayes' net, repeated for your convenience:



- L: The coin that was last used on the previous flip (fair or biased).
- S: The current score (0, 1, 2, or 3).
- C: The selected coin (fair or biased).
- O: The outcome of the coin flip (heads or tails).

(c) [2 pts] State: The fair coin was last used. The current score is 2.

Action: Flip.

Successor state: Bust.

- ◯ 0
- ◯ $P(C = \text{fair} | L = \text{fair}, S = 2) \cdot P(O = \text{tails} | C = \text{fair})$
- ◯ $\sum_c P(C = c | L = \text{fair}, S = 2) \cdot P(O = \text{tails} | C = c)$
- ◯ $\sum_l P(C = \text{fair} | L = l, S = 2) \cdot P(O = \text{tails} | C = \text{fair})$
- ◯ 1

(d) [2 pts] State: The fair coin was last used. The current score is 3.

Action: Flip.

Successor state: Bust.

- ◯ 0
- ◯ $P(C = \text{fair} | L = \text{fair}, S = 3) \cdot P(O = \text{tails} | C = \text{fair})$
- ◯ $\sum_c P(C = c | L = \text{fair}, S = 3) \cdot P(O = \text{tails} | C = c)$
- ◯ $\sum_l P(C = \text{fair} | L = l, S = 3) \cdot P(O = \text{tails} | C = \text{fair})$
- ◯ 1

Lastly, we need to deal with the first coin flip at the start of the game, when we don't know what coin was used on the previous flip. We will represent this as being uncertain about the state of the MDP: we could either be in a state where the fair coin was last used, or the biased coin was last used.

(Hint: The next subpart can be solved independently of the rest of the question.)

(e) [3 pts] Consider an MDP where you are uncertain about your current state. The probability that you are in state s is $P(s)$. The actions available from every state are the same.

Write the Bellman expression that represents the optimal action in this situation.

(i) (ii) (iii) $\sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V(s') \right]$

(i)	◯ 1	◯ $\sum_a$	◯ $\arg\max_a$	◯ $\arg\min_a$
(ii)	◯ 1	◯ $\sum_s$	◯ $\max_s$	◯ $\min_s$
(iii)	◯ 1	◯ $P(s)$	◯ $T(s, a, s')$	◯ $R(s, a, s')$

Q4. [14 pts] RL: Rest and ReLaxation

Consider the grid world MDP below, with unknown transition and reward functions.

A	B	C
D	E	F
G	H	I

The agent observes the following samples in this grid world:

s	a	s'	$R(s,a,s')$
E	East	F	-1
E	East	H	-1
E	South	H	-1
E	South	H	-1
E	South	D	-1

Reminder: In grid world, each non-exit action succeeds with some probability. If an action (e.g. North) fails, the agent moves in one of the cardinally adjacent directions (e.g. East or West) with equal probability, but will not move in the opposite direction (e.g. South).

Let p denote the probability that an action succeeds.

In this question, we will consider 3 strategies for estimating the transition function in this MDP.

Strategy 1: The agent does not know the rules of grid world, and runs model-based learning to directly estimate the transition function.

(a) [1 pt] From the samples provided, what is $\hat{T}$(E, South, H)?

◯ 0 ◯ 1/5 ◯ 2/5 ◯ 3/5 ◯ 4/5 ◯ 1/3 ◯ 2/3 ◯ 1/2 ◯ 1 ◯ Not enough information

(b) [1 pt] From the samples provided, what is $\hat{T}$(E, West, D)?

◯ 0 ◯ 1/5 ◯ 2/5 ◯ 3/5 ◯ 4/5 ◯ 1/3 ◯ 2/3 ◯ 1/2 ◯ 1 ◯ Not enough information

Strategy 2: The agent knows the rules of grid world, and runs model-based learning to estimate p. Then, the agent uses the estimated $\hat{p}$ to estimate the transition function.

(c) [1 pt] From the samples provided, what is $\hat{p}$, the estimated probability of an action succeeding?

◯ 0 ◯ 1/5 ◯ 2/5 ◯ 3/5 ◯ 4/5 ◯ 1/3 ◯ 2/3 ◯ 1/2 ◯ 1 ◯ Not enough information

(d) [1 pt] Based on $\hat{p}$, what is $\hat{T}$(E, West, D)?

◯ 0 ◯ 1/5 ◯ 2/5 ◯ 3/5 ◯ 4/5 ◯ 1/3 ◯ 2/3 ◯ 1/2 ◯ 1 ◯ Not enough information

(e) [3 pts] Select all true statements about comparing Strategy 1 and Strategy 2.

☐ Strategy 1 will usually require fewer samples to estimate the transition function to the same accuracy threshold.
☐ There are fewer unknown parameters to learn in Strategy 1.
☐ Strategy 1 is more prone to overfitting on samples.
◯ None of the above

The grid world and samples, repeated for your convenience:

A	B	C
D	E	F
G	H	I

s	a	s'	$R(s, a, s')$
E	East	F	−1
E	East	H	−1
E	South	H	−1
E	South	H	−1
E	South	D	−1

Strategy 3: The agent knows the rules of grid world, and uses an exponential moving average to estimate p. Then, the agent uses the estimated $\hat{p}$ to estimate the transition function.

(f) [2 pts] Consider this update equation: $\hat{p} \leftarrow (1 - \alpha)\hat{p} + (\alpha)x$

Given a sample (s, a, s'), what value of x should be used in the corresponding update?

- ◯ $R(s, a, s')$
- ◯ 1.0 if the action succeeded, and 0.0 otherwise
- ◯ 1.0 if the action failed, and 0.0 otherwise
- ◯ $V(s)$
- ◯ $V(s')$

(g) [3 pts] Select all true statements about comparing Strategy 2 and Strategy 3.

- ☐ Strategy 2 gives a more accurate estimate, because it is the maximum likelihood estimate.
- ☐ Strategy 3 gives a more accurate estimate, because it gives more weight to more recent samples.
- ☐ Strategy 3 can be run with samples streaming in one at a time.
- ◯ None of the above

The rest of the question is independent from the previous subparts.

Suppose the agent runs Q-learning in this grid world, with learning rate $0 < \alpha < 1$, and discount factor $\gamma = 1$.

(h) [1 pt] After iterating through the samples once, how many learned Q-values will be nonzero?

◯ 0 ◯ 1 ◯ 2 ◯ 3 ◯ 4 ◯ > 4

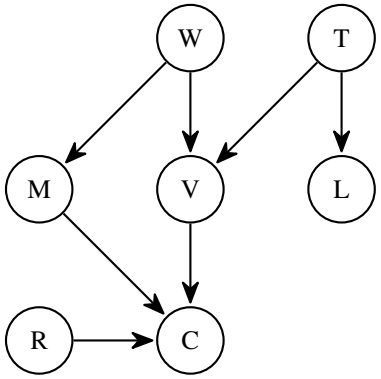
(i) [1 pt] After iterating through the samples repeatedly until convergence, how many learned Q-values will be nonzero?

◯ 0 ◯ 1 ◯ 2 ◯ 3 ◯ 4 ◯ > 4

Q5. [16 pts] Bayes' Nets: Pac Pong Performance

Pacman is going out to play a Pong tournament, and he wants to model all the uncertainty in the game in order to ensure victory.

Consider the Bayes' net below. All random variables are binary.



Pacman is trying to compute $P(\mathrm{C}|+\mathrm{w},-\mathrm{t})$.

First, Pacman uses the standard variable elimination algorithm (as seen in lecture) to eliminate L and M.

(a) [2 pts] Select the factor(s) needed to eliminate L and M.

- ☐ $P(+\mathrm{w})$
- ☐ $P(-\mathrm{t})$
- ☐ $P(\mathrm{M}|+\mathrm{w})$
- ☐ $P(\mathrm{C}|+\mathrm{w})$
- ☐ $P(\mathrm{C}|\mathrm{R},+\mathrm{w},\mathrm{V})$
- ☐ $P(\mathrm{L}|-\mathrm{t})$
- ☐ $P(\mathrm{C}|\mathrm{R},\mathrm{M},\mathrm{V})$
- ☐ $P(\mathrm{R})$

(b) [2 pts] Select the factor(s) that remain after eliminating L and M.

- ☐ $f(\mathrm{R},+\mathrm{w},\mathrm{V},\mathrm{C})$
- ☐ $f(\mathrm{R},+\mathrm{w},-\mathrm{t},\mathrm{V},\mathrm{C})$
- ☐ $f(-\mathrm{t})$
- ☐ $f(+\mathrm{w},\mathrm{V},\mathrm{C})$
- ☐ $f(\mathrm{V},+\mathrm{w},-\mathrm{t})$
- ☐ $f(+\mathrm{w})$

(c) [2 pts] Which statement best explains why variable elimination (VE) is better than inference by enumeration (IBE)?

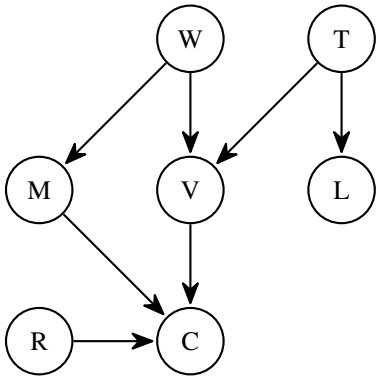
- ◯ VE can use larger factors, so we can read off more values and use less computation.
- ◯ VE can use smaller factors, so we can use less computation.
- ◯ While VE is usually better, sometimes IBE creates smaller factors.
- ◯ VE is always strictly better, because it always uses strictly smaller factors, using less computation.

(d) [2 pts] Pac-Man claims that he could have skipped eliminating L and just removed it from the graph for his query of $P(\mathrm{C}|+\mathrm{w},-\mathrm{t})$. This is:

- ◯ Correct, because L is not directly connected with C.
- ◯ Correct, because there is no downstream path from C to L.
- ◯ Correct, because L is conditionally independent of C, given w and t.
- ◯ Incorrect, because every variable can have relevant information to every query.
- ◯ Incorrect, because there exists a path from L to C.

Reminder: Pacman is trying to compute $P(\mathrm{C}|+\mathrm{w},-\mathrm{t})$.

The Bayes' net, repeated for your convenience:



Pac-Man is tired of variable elimination, so he takes a short break. He decides to approximate the query using various sampling methods.

In each of the next subparts, select all sampling algorithm(s) that could have generated the two **consecutive** samples shown.

Note: If Pacman rejects a sample while generating it, the rest of the sample will say "rejected."

(e) [1 pt]

Sample 1:	+r	+w	+t	−v	+m	+l	+c
Sample 2:	−r	−w	−t	+v	−m	+l	+c

☐ Prior Sampling ☐ Likelihood Weighting ◯ None of the above
☐ Rejection Sampling ☐ Gibbs Sampling

(f) [1 pt]

Sample 1:	−r	+w	−t	−v	+m	+l	−c
Sample 2:	−r	+w	−t	+v	+m	+l	−c

☐ Prior Sampling ☐ Likelihood Weighting ◯ None of the above
☐ Rejection Sampling ☐ Gibbs Sampling

(g) [1 pt]

Sample 1:	−r	+w	+t	rejected	rejected	rejected	rejected
Sample 2:	−r	+w	−t	rejected	rejected	rejected	rejected

☐ Prior Sampling ☐ Likelihood Weighting ◯ None of the above
☐ Rejection Sampling ☐ Gibbs Sampling

(h) [1 pt]

Sample 1:	+r	+w	−t	+v	−m	+l	−c
Sample 2:	−r	+w	−t	−v	−m	+l	−c

☐ Prior Sampling ☐ Likelihood Weighting ◯ None of the above
☐ Rejection Sampling ☐ Gibbs Sampling

(i) [4 pts] Pacman finishes his variable elimination, in the order: L, M, R, V.

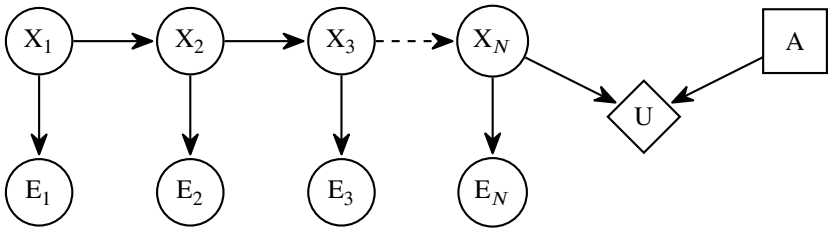
Pacman discovers that the evidence he used was wrong! The evidence was +t, not −t.

Select all steps that Pacman would need to redo. Note that the step of collecting samples does **not** involve computing the probability from those samples.

☐ Joining and eliminating M.
☐ Joining and eliminating R.
☐ Joining and eliminating V.
☐ Collecting samples for prior sampling.
☐ Collecting samples for rejection sampling.
☐ Collecting samples for likelihood weighting.
☐ Collecting samples for Gibbs sampling.
◯ No steps need to be redone.

Q6. [13 pts] HMMs and VPI: Markov Menagerie

Consider the following hidden Markov model (HMM). All random variables are binary.



We would like to compute VPI(E_N) = MEU(E_N) − MEU(∅).

(a) [1 pt] Which of these distributions is needed to compute MEU(∅)?

◯ $P(X_N)$ ◯ $P(X_N|X_{N-1})$ ◯ $P(X_N|E_1,\ldots,E_N)$ ◯ $P(X_N|E_N)$

(b) [2 pts] Which of these computations can be used to derive the distribution in part **(a)**?

◯ Run the forward algorithm with no modifications.
◯ Run the forward algorithm, skipping all time elapse updates.
◯ Run the forward algorithm, skipping all observation updates.
◯ Read the conditional probability table under X_N.

(c) [3 pts] Write an expression that can be used to compute MEU(E_N).

$$\text{MEU}(E_N) = \sum_{e_N} \textbf{(i)} \left[\max_a \sum_{x_N} \textbf{(ii)} \;\; \textbf{(iii)} \right]$$

(i) ◯ $P(E_N)$ ◯ $P(E_N|E_{N-1})$ ◯ $P(E_N|X_N)$ ◯ $P(E_N|X_1,\ldots,X_N)$

(ii) ◯ $P(X_N|X_{N-1})$ ◯ $P(X_N|X_1,\ldots,X_{N-1})$ ◯ $P(X_N|E_N)$ ◯ $P(X_N|E_1,\ldots,E_N)$

(iii) ◯ 1 ◯ $U(x_N)$ ◯ $U(a)$ ◯ $U(x_N,a)$

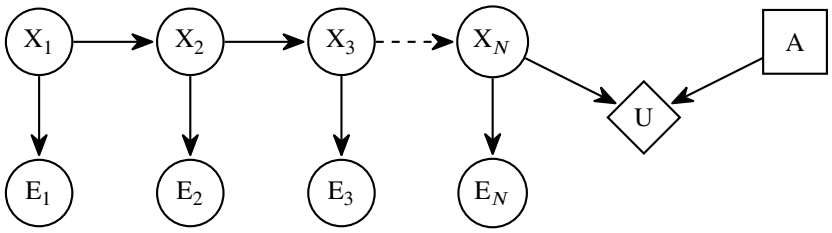
(d) [2 pts] Consider the distribution in part **(a)**, which you computed in part **(b)**.
From the distribution in **(a)**, which of the following additional computations results in the distribution in blank **(ii)**?

◯ Apply an additional observation update.
◯ Apply an additional time elapse update.
◯ Apply an additional time elapse and observation update.
◯ Re-run the forward algorithm from the beginning, with no modifications.

(e) [2 pts] Which of the following computations can be used to derive the distribution in blank **(i)**?

◯ $P(x_N)P(E_N|x_N)$ ◯ $\sum_{e_N} P(x_N)P(e_N|x_N)$
◯ $\sum_{x_N} P(x_N)P(E_N|x_N)$ ◯ $\sum_{x_N} P(E_N)P(x_N|E_N)$

The diagram, repeated for your convenience:



For the rest of the question, suppose we would like to compute VPI($E_1, \ldots, E_N$) = MEU($E_1, \ldots, E_N$) − MEU($\emptyset$).

(f) [2 pts] To compute MEU($E_1, \ldots, E_N$), we'll need one or more distribution(s) over X_N.
Which of these computations will generate the necessary distribution(s) over X_N?

- ◯ Run the forward algorithm once, with no modifications.
- ◯ Run the forward algorithm 2^N times, with no modifications.
- ◯ Run the forward algorithm once, skipping all observation updates.
- ◯ Run the forward algorithm 2^N times, skipping all observation updates.

(g) [1 pt] To compute MEU($E_1, \ldots, E_N$), which other distribution do we need?

◯ $P(E_1)$ ◯ $P(E_N)$ ◯ $P(E_1, \ldots, E_N)$ ◯ $P(E_1, \ldots, E_N | X_N)$

Q7. [12 pts] Games and ML: BeatBlue

Pacman wants to design an agent that can play chess and beat his older (more successful) brother, DeepBlue.

First, Pacman would like to design a machine learning algorithm that takes in a board state and outputs a real number between 0.00 (for states where Pacman is losing) and 1.00 (for states where Pacman is winning).

(a) [2 pts] Pacman starts by asking experts to manually assign values to board states. Select all true statements.

- ☐ We can use the manually-assigned values as our training dataset.
- ☐ We can use the manually-assigned values as our testing dataset.
- ☐ Since we have values assigned by experts, the machine learning algorithm provides no additional benefit.
- ◯ None of the above

(b) [1 pt] Pacman suggests using a perceptron for this problem. Is it reasonable to use a perceptron for this problem?

- ◯ Yes, because this is a classification problem.
- ◯ Yes, because the training data is linearly separable.
- ◯ No, because this is a regression problem, and perceptrons output discrete classes, not continuous numbers.
- ◯ No, because perceptrons should never be used when the data is not linearly separable.

Pacman decides to represent the chess board state as a 64-dimensional vector. Pacman designs a fully-connected, feed-forward neural network with the following architecture:

$$h = \text{ReLU}(x \cdot W_1 + b_1)$$
$$\hat{y} = \text{ReLU}(h \cdot W_2 + b_2)$$

This network takes in a 1×64 vector, x, and outputs a scalar real number, $\hat{y}$.

Pacman sets the hidden layer size to be 128. In other words, h has dimensions 1×128.

(c) [1 pt] What are the dimensions of W_1?

- ◯ 1×1
- ◯ 128×1
- ◯ 8192×1
- ◯ 128×128
- ◯ 64×1
- ◯ 192×1
- ◯ 64×64
- ◯ 64×128

(d) [1 pt] What are the dimensions of W_2?

- ◯ 1×1
- ◯ 128×1
- ◯ 8192×1
- ◯ 128×128
- ◯ 64×1
- ◯ 192×1
- ◯ 64×64
- ◯ 64×128

(e) [1 pt] Pacman considers changing the second layer of the neural network. Which of these proposed changes is best for this problem?

Reminders: $\text{ReLU}(x) = \max(x, 0)$ $\quad \sigma(x) = \frac{1}{1+e^{-x}}$ $\quad \text{sgn}(x) = \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases}$

- ◯ $\hat{y} = \text{ReLU}(h \cdot W_2 + b_2)$
- ◯ $\hat{y} = \text{sgn}(h \cdot W_2 + b_2)$
- ◯ $\hat{y} = \sigma(h \cdot W_2 + b_2)$
- ◯ $\hat{y} = h \cdot W_2 + b_2$

Pacman now has a neural network $\hat{y} = f(x)$ that takes in board states (x) and outputs values ($\hat{y}$), and would like to use the network to select actions in the chess game.

(f) [1 pt] Let $s' = g(s, a)$ represent the successor function that takes in a state s and action a, and outputs a successor state s'.

Which of the following expressions represents a reflex agent's optimal action from state s, based on the values outputted by the neural network?

- ◯ $\arg\max_s f(s)$
- ◯ $\arg\max_a f(g(s, a))$
- ◯ $\max_a g(s, a)$
- ◯ $\sum_a f(g(s, a))$

Finally, Pacman decides to run depth-limited minimax search and use the neural network as an evaluation function.

(g) [1 pt] Is it reasonable to use the neural network as an evaluation function?

- ◯ Yes, because the neural network maps states to real-numbered values.
- ◯ Yes, because the network is guaranteed to correctly identify terminal states where the game is over.
- ◯ No, because it would be more efficient and accurate to expand the entire minimax tree to the terminal states.
- ◯ No, because evaluation functions should map states to actions.

(h) [1 pt] Is it possible to use alpha-beta pruning in this problem?

- ◯ Yes, pruning works even if the neural network output was unbounded.
- ◯ Yes, but only because the neural network only outputs numbers between 0 and 1.
- ◯ No, because pruning requires you to know the values at all leaf nodes in advance.
- ◯ No, because the values outputted by the neural network are not guaranteed to be correct.

(i) [1 pt] Is it better to spend more time on training the neural network, or expanding more layers of the game tree?

- ◯ Training the neural network
- ◯ Expanding the game tree
- ◯ Not enough information

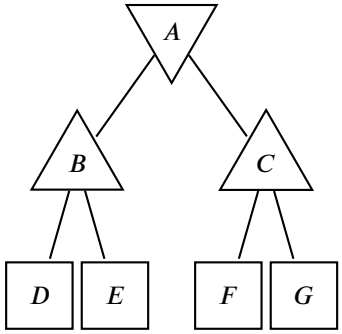
(j) [2 pts] Recall that in Monte Carlo tree search, we allocate more rollouts to game states that are more promising (i.e. better utility for Pacman).

Inspired by this idea, Pacman considers running gradient descent for a longer time when using the neural network to evaluate game states that are more promising. Would this idea work?

- ◯ Yes, because this causes the neural network to focus its training on promising game states.
- ◯ Yes, because training for a longer time leads to better evaluations.
- ◯ No, because running gradient descent for too long always leads to overfitting.
- ◯ No, because gradient descent runs during training, not evaluation.

Q8. [13 pts] Potpourri

(a) [4 pts] Select all true statements about the game tree shown.



Suppose that the values in the terminal nodes have the property that $D < E < F < G$, and the agents do not know this.

Assume that alpha-beta pruning visits nodes from left to right.

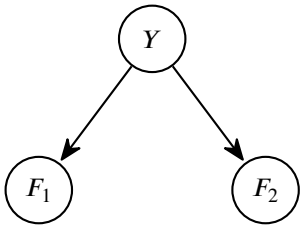
- ☐ Two actions are needed to transition from state A to state E.
- ☐ The value at the root node will always be E.
- ☐ When running alpha-beta pruning, node G can always be pruned.
- ☐ When running alpha-beta pruning, node F can sometimes be pruned.
- ◯ None of the above

(b) [4 pts] Consider a standard HMM (from lecture) with states $X_1, \ldots, X_N$ and evidence $E_1, \ldots, E_N$.

Suppose we're running particle filtering with a large number of particles. At time step t, we have a particle at state $X_t = x_t$, with weight 0.7. Select all true statements about this particle.

- ☐ We've just finished a time elapse update, and have not performed the observation update for time t yet.
- ☐ 0.7 is the probability of the particle visiting all the states it's visited so far: $P(x_1, \ldots, x_t)$.
- ☐ If this particle is drawn during resampling, the resulting new particle will still be at state x_t.
- ☐ 0.3 is the probability that this particle disappears during resampling.
- ◯ None of the above

Consider the following Naive Bayes' model and training data. Y, F_1, and F_2 are binary random variables.



F_1	F_2	Y
$+f_1$	$+f_2$	$+y$
$+f_1$	$-f_2$	$+y$
$-f_1$	$+f_2$	$-y$

(c) [2 pts] What is $P(+y \mid +f_1, +f_2)$?

- ◯ 0
- ◯ 1/3
- ◯ 2/3
- ◯ 1/2
- ◯ 1
- ◯ Not enough information

(d) [3 pts] Select all equations that are true about the model, regardless of the training data used.

- ☐ $P(F_1) = P(F_1|F_2)$
- ☐ $P(Y) = P(Y|F_1)$
- ☐ $P(F_1|Y) = P(F_1|Y, F_2)$
- ◯ None of the above