6.s058/16.420
Representation, Inference and Reasoning in AI

Final Exam

Practice Question Solutions

December, 2021

Answer the questions in the spaces provided on the question sheets. Please keep the exam packet together (don't un-staple).

You are permitted to use two sheets of paper with notes on (both sides), and a calculator and a timer. If you use your phone for the calculator and timer, please restrict yourself to these functions.

Name: _____

Kerberos: _____

Question	Points	Score
1	6	
2	12	
3	8	
4	7	
5	13	
6	10	
7	8	
8	8	
9	20	
10	4	
11	8	
12	8	
13	16	
14	16	
Total:	144	

1 Logic

1. (6 points) Which of the following are true statements about constraint satisfaction problems (CSPs)?

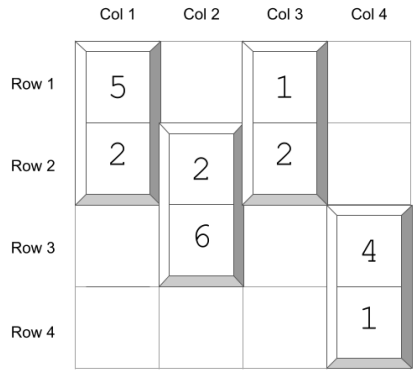
   1. CSP solutions are unique.
   2. Backtracking search is guaranteed to find a CSP solution if one exists.
   3. A binary CSP is one where all variables have two elements in their domain.
   4. There exists an algorithm that is strictly better than the stupidest possible algorithm in terms of time complexity in all CSPs.
   5. If constraint propagation (AC-3) terminates and all variables have one remaining possible value, those values comprise a solution.
   6. It is possible to convert a CSP into a probabilistic graphical model, but it is not necessarily possible to convert a probabilistic graphical model into a CSP.

   Mark all that apply. ◯ 1 √ **2** ◯ 3 ◯ 4 √ **5** √ **6**

2. (12 points) In this problem, we will consider a game of placing M *dominoes* onto an $N \times N$ grid.

   - A domino has a *top* and a *bottom*. Each is annotated with a number between 1 and 6 (inclusive).
   - Each domino can be placed anywhere on the grid so long as it does not overlap with other dominoes.
   - A domino can only be placed in one vertical orientation, with the top on top.
   - If two dominoes touch in the grid, the number on the halves that touch must be equal.

   Given a grid and a set of dominoes, the objective is to find a placement of all dominoes onto the grid that satisfy the criteria above. For example:



   We will approach this game with propositional logic. Consider the following symbols:

   - For each domino $d = 1, 2, \ldots M$, each row $r = 1, 2, \ldots, N$ and each column $c = 1, 2, \ldots, N$ let `domino-d-r-c` be a symbol representing the placement of the *top* of that domino at that cell.
   - For each domino $d = 1, 2, \ldots M$, and each number $i = 1, 2, \ldots, 6$ let `top-d-i` and `bot-d-i` represent that the top and bottom respectively of domino d are annotated with the number i.

   In the questions that follow, you may find the following notation useful:

   - For a set of propositional symbols $S = \{\texttt{p}, \texttt{q}, \texttt{r}\}$, $(\bigvee_{x \in S} x)$ is the same as $(\texttt{p} \vee \texttt{r} \vee \texttt{q})$.
   - Similarly, $(\bigwedge_{x \in S} x)$ is the same as $(\texttt{p} \wedge \texttt{r} \wedge \texttt{q})$.

   (a) Using the symbols above, write a propositional logic sentence that captures the following meaning: "If the top of domino $d = 1$ is placed at row $r = 4$ and column $c = 3$, then the top of domino $d = 2$ cannot be placed at $r = 4$, $c = 3$."

Solution: `domino-1-4-3` $\Rightarrow \neg$ `domino-2-4-3`

(b) Write another for: "Domino $d = 1$ must be placed *somewhere* on the $N \times N$ grid".

Solution: $\bigvee_{r=1,\ldots,N-1} \bigvee_{c=1,\ldots,N}$ `domino-1-r-c` (informal notation also accepted)

(c) Write another for: "If the top of domino $d = 1$ is placed at row $r = 4$ and column $c = 3$, and the top of $d = 2$ is placed to the right at $r = 4$, $c = 4$, then the top numbers must match".

Solution: `domino-1-4-3` $\wedge$ `domino-2-4-4` $\Rightarrow \bigvee_{i=1,2,\ldots,6}$ (`top-1-i` $\wedge$ `top-2-i`)

(d) Write another for: "If the top of domino $d = 1$ is placed at row $r = 4$ and column $c = 3$, and the **bottom** of $d = 2$ is placed at $r = 4$, $c = 4$, then the respective numbers must match".

Solution: `domino-1-4-3` $\wedge$ `domino-2-3-4` $\Rightarrow \bigvee_{i=1,2,\ldots,6}$ (`top-1-i` $\wedge$ `bot-2-i`)

Suppose now we are performing inference with DPLL. Let's say that we have a partial assignment {`domino-2-4-3` $\mapsto$ True} and we are looking at just the following CNF sentence in isolation:

$$(\neg\texttt{domino-1-4-3} \vee \neg\texttt{domino-2-4-3}) \wedge (\neg\texttt{domino-1-4-3} \vee \neg\texttt{domino-1-1-1})$$

(g) Are there any pure symbols? If so, which one(s)?

Solution: Yes, all symbols are pure in this example.

(h) Are there any unit clauses? If so, which one(s)?

Solution: Yes, the first clause is unit because it effectively contains a single literal since `domino-2-4-3` is already assigned.

We have a SAT solver at our disposal. The solver takes in a propositional logic sentence and returns whether the sentence is satisfiable, and if so, a model. Suppose we have now generated all the propositional sentences needed to represent the game rules and the number labels for a given set of dominoes and grid size. Call the conjunction of these sentences `background-sentence`.

(i) Describe how we can use the SAT solver to figure out how to place the dominoes on the grid, if such a placement is possible.

Solution: Query the SAT solver with `background-sentence` $\wedge$ ($\bigwedge_d \bigvee_{r,c}$ `domino-d-r-c`).

3. (8 points) Consider the following axioms:

   1. If you take 6.s058, you will love logic.
   2. If you take 16.420, you will love logic.
   3. All students take either 6.s058 or 16.420, but not both.
   4. Spongebob is a student.

(a) Formulate the axioms above in first-order logic using the predicates `Takes6s058`, `Takes16420`, `Student`, and `LovesLogic`.

Solution:

- $\forall x.\texttt{Takes6s058}(x) \Rightarrow \texttt{LovesLogic}(x)$.
- $\forall x.\texttt{Takes16420}(x) \Rightarrow \texttt{LovesLogic}(x)$.
- $\forall x.\texttt{Student}(x) \Rightarrow ((\texttt{Takes6s058}(x) \vee \texttt{Takes16420}(x)) \wedge \neg(\texttt{Takes6s058}(x) \wedge \texttt{Takes16420}(x))$.
- $\texttt{Student}(\text{spongebob})$.

(b) Convert your axioms into CNF form.

Solution: Clauses:

1. $(\neg\texttt{Takes6s058}(x) \vee \texttt{LovesLogic}(x))$
2. $(\neg\texttt{Takes16420}(y) \vee \texttt{LovesLogic}(y))$
3. $(\neg\texttt{Student}(z) \vee \texttt{Takes6s058}(z) \vee \texttt{Takes16420}(z))$
4. $(\neg\texttt{Student}(z) \vee \neg\texttt{Takes6s058}(z) \vee \neg\texttt{Takes16420}(z))$
5. $\texttt{Student}(\text{spongebob})$

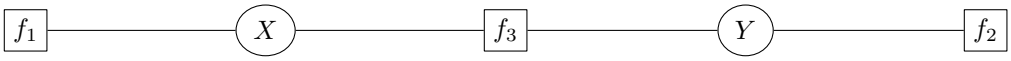
(c) Prove that Spongebob loves logic using resolution.

Solution:

- Suppose (6) $\neg\texttt{LovesLogic}(\text{spongebob})$.
- From (5) and (3) we conclude (7) $\texttt{Takes6s058}(\text{spongebob}) \vee \texttt{Takes16420}(\text{spongebob})$.
- From (7) and (1) we conclude (8) $\texttt{Takes16420}(\text{spongebob}) \vee \texttt{LovesLogic}(\text{spongebob})$.
- From (8) and (2) we conclude (9) $\texttt{LovesLogic}(\text{spongebob})$.
- From (6) and (9) we conclude False. (Okay to omit this step.) □

2 Factor Graph

4. (7 points) Consider two random variables X and Y, both defined over the same domain $\mathcal{S}$. Their joint distribution can be represented as the following factor graph.



In this figure, rectangular nodes are factors; circular nodes are random variables.

(a) Write a formula for the joint distribution of $p(X = x, Y = y)$, where $x \in \mathcal{S}$, $y \in \mathcal{S}$, represented as a function of the factor potentials f_1, f_2, and f_3.

Solution:

$$p(X = x, Y = y) = \frac{f_1(x) f_2(y) f_3(x, y)}{\sum_{x' \in X} \sum_{y' \in Y} [f_1(x') f_2(y') f_3(x', y')]}$$

(b) Write a formula for the conditional distribution of $p(X = x|Y = y)$, represented as a function of the factor potentials f_1, f_2, and f_3.

Solution:

$$p(X = x|Y = y) = \frac{f_1(x)f_2(y)f_3(x, y)}{\sum_{x' \in X} [f_1(x')f_2(y)f_3(x', y)]}$$

(c) Consider the following claim. For any $x \in S$:

Claim 1.

$$\sum_y p(X = x|Y = y) = p(X = x).$$

Prove the claim above or provide a counter-example.

Solution: The claim is false.

Consider the following joint distribution, defined over $x, y \in S = \{0, 1\}$.

$$p(X = 0, Y = 0) = 0.5, p(X = 1, Y = 0) = 0, p(X = 0, Y = 1) = 0.5, p(X = 1, Y = 1) = 0.$$

Thus,

$$p(X = 0|Y = 0) = 1, p(X = 0|Y = 1) = 1.$$

It's not even a distribution.

3 Inverted Hidden Markov Model

5. (13 points) In this section, we will apply a variant of Hidden Markov Models to a sequence labeling task. Specifically, we are interested in inferring the Part-Of-Speech (POS) labels for an input sentence. Consider the vocabulary containing three words: {the, cat, sat}, and four possible POS taggings {⟨S⟩, ARTICLE, NOUN, VERB} (⟨S⟩ is a special tag representing the begining of a sentence). For example, the groundtruth POS tags for the sentence *the cat sat* should be ARTICLE-NOUN-VERB.

An "Inverted Hidden Markov Model" (IHMM) can be defined using the following directed graphical model.

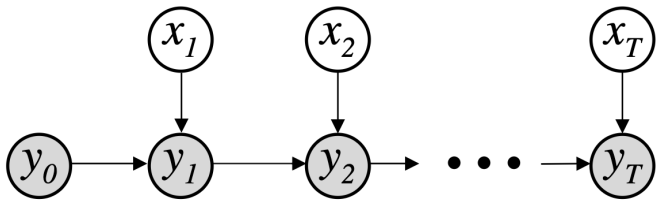


Figure 1: Inverted Hidden Markov Model (IHMM).

In the IHMM model, the y nodes correspond to the output label sequence, and the x nodes correspond to the input sentence.

(a) Write down the probability distribution for the following conditional distribution, based on the directed graphical model defined above.

$$p(\mathcal{Y}|\mathcal{X}) = p(Y_0, Y_1, Y_2, \cdots, Y_T|X_1, X_2, \cdots, X_T).$$

Solution:

$$p(Y_0, Y_1, Y_2, \cdots, Y_T | X_1, X_2, \cdots, X_T) = p(Y_0) \prod_{t=1}^{T} p(Y_t | X_t, Y_{t-1}).$$

For the rest of this section, we will assume $p(Y_0 = \langle \text{S} \rangle) = 1$ and focus on inferring $Y_1, Y_2, \cdots, Y_T$. We will develop a compact graphical representation to describe the conditional probability distributions $p(Y_t | X_t, Y_{t-1})$, showing as the weights on the directed edges. For example, for any t, $p(Y_t = \text{NOUN} | X_t = \text{cat}, Y_{t-1} = \text{ARTICLE}) = 0.9$.

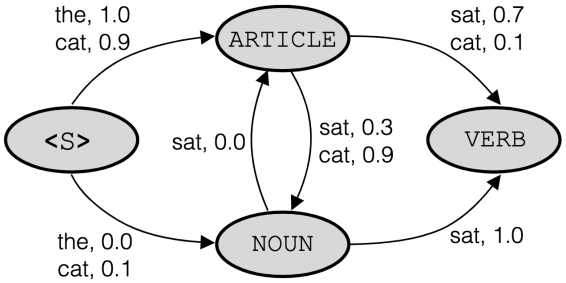


Figure 2: The normalized "conditional probability" function $p(Y_t | X_t, Y_{t-1})$.

(b) What's the most likely part-of-speech (POS) sequence $\mathcal{Y} = (Y_1, Y_2, Y_3)$ for the input sentence $\mathcal{X} = (X_1, X_2, X_3) =$ *the cat sat*? What's the corresponding conditional probability $p(Y_0, Y_1, Y_2, \cdots, Y_T | X_1, X_2, \cdots, X_T)$?

Solution: Most likely label: ARTICLE NOUN VERB.
Conditional probability: $1.0 \times 0.9 \times 1.0 = 0.9$.

(c) What's the most likely part-of-speech (POS) sequence $\mathcal{Y} = (Y_1, Y_2)$ for the input sentence $\mathcal{X} = (X_1, X_2) =$ *cat sat*? What's the corresponding conditional probability $p(Y_0, Y_1, Y_2, \cdots, Y_T | X_1, X_2, \cdots, X_T)$?

Solution: Most likely label: ARTICLE NOUN.
Conditional probability: $0.9 \times 0.7 = 0.63$.

Now, let's consider an "unnormalized" transition model.

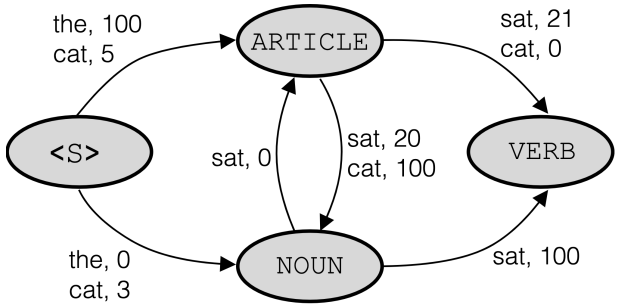


Figure 3: The unnormalized "score" function $s(Y_t, X_t, Y_{t-1})$.

Here, the edge weights will define a score function for transitions $s(Y_t, X_t, Y_{t-1})$. Note: they are not probabilities!

Now let's consider a "highest-score" label sequence. We define the score of an output sequence $y_1, y_2, \cdots, y_T$ as:

$$score(\mathcal{Y}|\mathcal{X}) = score(Y_0, Y_1, Y_2 \cdots, Y_T|X_1, \cdots, X_T) = s(Y_0) + \sum_{t=1}^{T} s(Y_t, X_t, Y_{t-1}),$$

where $s(Y_0 = \langle \mathrm{S} \rangle) = 0$, $s(Y_0 = x) = -\infty$ for any $x \neq \langle \mathrm{S} \rangle$. We will use s to represent the score for each individual transition $s(Y_t, X_t, Y_{t-1})$, and *score* to represent the score for the entire sentence $score(\mathcal{Y}|\mathcal{X})$

(d) What's the highest-score output sequence for the input sentence *the cat sat*? What's the corresponding score?

Solution: Highest-score label: ARTICLE NOUN VERB.
Score: $100 + 100 + 100 = 300$.

(e) What's the highest-score output sequence for the input sentence *cat sat*? What's the corresponding score?

Solution: Highest-score label: NOUN VERB.
Score: $3 + 100 = 103$.

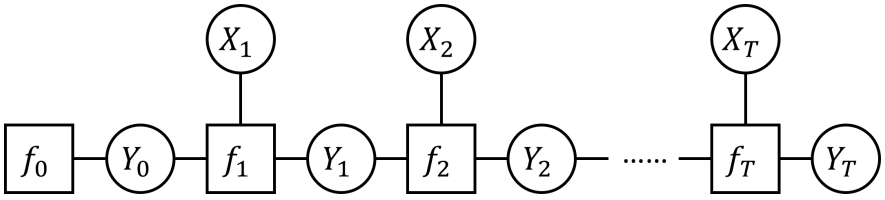
(f) We now define the probability of an output sequence as:

$$p'(\mathcal{Y}|\mathcal{X}) = p'(Y_0, Y_1, Y_2 \cdots, Y_T|X_1, \cdots, X_T) = \frac{\exp\left(score(\mathcal{Y}|\mathcal{X})\right)}{\sum_{\mathcal{Y}'} \exp\left(score(\mathcal{Y}'|\mathcal{X})\right)},$$

where the $\sum_{\mathcal{Y}'}$ in the denominator sums over all possible output sequence $\mathcal{Y}'$ of length T (or $T+1$ if you consider y_0).

Draw a factor graph $\mathcal{G}$ and define the factor potentials based on function s so that this factor graph $\mathcal{G}$ represents the probability distribution $p'(\mathcal{Y}|\mathcal{X})$. (The number of variables in each factor should be strictly smaller than 4.)

Solution:



We will define factor f_0 for Y_0, and $f_0(Y_0 = \langle \mathrm{S} \rangle) = 1$; $f_0(Y_0 = x) = 0, \forall x \neq \langle \mathrm{S} \rangle$.
We will define factor f_t for each tuple (Y_t, X_t, Y_{t-1}).

$$f_t(Y_t, X_t, Y_{t-1}) = \exp\left(s(Y_t, X_t, Y_{t-1})\right).$$

(g) Let's assume the conditional probability $p(Y_t|X_t, Y_{t-1})$ and the score function $s(Y_t, X_t, Y_{t-1})$ are related in the following way:

$$p(Y_t|X_t, Y_{t-1}) = \frac{\exp\left(s(Y_t, X_t, Y_{t-1})\right)}{\sum_{Y_t'} \exp\left(s(Y_t', X_t, Y_{t-1})\right)}.$$

Under this assumption, is the "score-based" distribution: $p'(\mathcal{Y}|\mathcal{X})$ identical to the original IHMM distribution you wrote down in (a): $p(\mathcal{Y}|\mathcal{X})$? That is, for any $s(Y_t, X_t, Y_{t-1})$ and $p(Y_t|X_t, Y_{t-1})$ satisfying the condition, $p(\mathcal{Y}|\mathcal{X}) = p'(\mathcal{Y}|\mathcal{X})$.

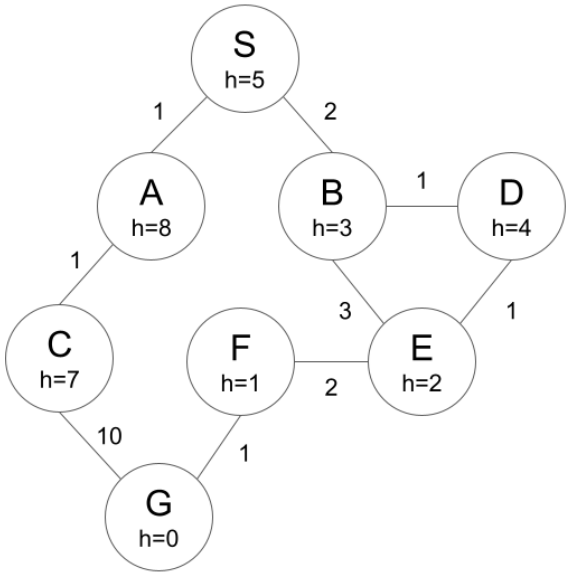
Solution: False. You can see this simply by comparing (c) and (e): the values assigned to edges roughly satisfy this condition. For example:

- for the edge(⟨S⟩, cat): $\frac{\exp(5)}{\exp(5)+\exp(3)} \approx 0.9$.
- for the edge(ARTICLE, sat): $\frac{\exp(21)}{\exp(21)+\exp(20)} \approx 0.7$.

The key difference between $p(\mathcal{Y}|\mathcal{X})$ and $p'(\mathcal{Y}|\mathcal{X})$ is that p' defers the normalization at the global level and p does normalization at each step t.

4 Search

6. (10 points) Consider the search graph shown below. S is the start state and G is the (only) goal state. All edges are bidirectional. Heuristic values are written in the circles.



Make the following assumptions:

- Ties are broken alphabetically in search. For example, a partial plan S→X→A would be expanded before S→X→B.
- When a node is expanded (popped from the frontier/fringe) in graph search, if a node with this state has been previously expanded, then the node is pruned (discarded). Note that this is not the same as never visiting (adding to the frontier/fringe) a state twice.
- All algorithms terminate when a node corresponding to a goal state is expanded, not when the node is added to the frontier/fringe.

(a) For **depth-first search**, report all the states that are expanded in the order that they are expanded.

Solution: S, A, C, G

(b) For **depth-first search**, report the path found.

Solution: S, A, C, G

(c) For **breadth-first search**, report all the states that are expanded in the order that they are expanded.

Solution: S, A, B, C, D, E, G

(d) For **breadth-first search**, report the path found.

Solution: S, A, C, G

(e) For **uniform-cost search**, report all the states that are expanded in the order that they are expanded.

Solution: S, A, B, C, D, E, F, G

(f) For **uniform-cost search**, report the path found.

Solution: S, B, D, E, F, G

(g) For **greedy best-first search**, report all the states that are expanded in the order that they are expanded.

Solution: S, B, E, F, G

(h) For **greedy best-first search**, report the path found.

Solution: S, B, E, F, G

(i) For **A* search**, report all the states that are expanded in the order that they are expanded.

Solution: S, B, D, E, F, G

(j) For **A* search**, report the path found.

Solution: S, B, D, E, F, G

7. (8 points) Let's bake some cakes with PDDL! Here's what we know:

- We have boxes of cake mix. One box makes one cake.
- To make a cake, we need to put the cake mix in a pan and put the pan in the oven.
- Each pan can hold one cake mix. Once the cake is baked, we'll leave it in the pan.
- An oven can hold infinitely many pans, and baking happens instantaneously.

Here is a draft PDDL domain:

```
(:predicates
    (isPan ?pan) (isOven ?oven) (inBox ?mix ?box) (inPan ?mix ?pan) (isBaked ?mix)
)

(:action pourInPan :parameters (?mix ?box ?pan)
 :precondition (and (inBox ?mix ?box) (isPan ?pan))
 :effects (and (not (inBox ?mix ?box)
               (inPan ?mix ?pan)))

(:action putInOven :parameters (?mix ?pan ?oven)
 :precondition (and (inPan ?mix ?pan) (isOven ?oven))
 :effects (and (isBaked ?mix)))
```

(a) This draft is flawed. Write a PDDL problem where no valid plan should exist, but planning with the domain above would result in a plan found. To receive full credit, include all parts of the PDDL problem specification with proper syntax.

Solution: Any problem where the number of desired cakes exceeds the number of pans. E.g.:

```
(:objects mix1 mix2 box1 box2 pan1 oven1)

(:init
    (inBox mix1 box1)
    (inBox mix2 box2)
    (isPan pan1)
    (isOven oven1)
)

(:goal (and (isBaked mix1) (isBaked mix2)))
```

(b) Describe how to remedy the original PDDL domain.

Solution: Add a predicate like `(panEmpty ?pan)`. Make `(panEmpty ?pan)` a precondition of `pourInPan` and a delete effect too, and add it to the initial state as appropriate.

8. (8 points) Which of the following are true statements?

1. If a PDDL problem has no solution, constructing the RPG may never terminate.
2. If a PDDL problem has only one goal literal, then $h_{max} = h^*$.
3. If a PDDL problem has only one goal literal, then h_{add} is admissible.
4. Greedy best-first search with an admissible heuristic is guaranteed to find an optimal-cost plan.
5. Uniform-cost search is guaranteed to find an optimal-cost plan.
6. If a heuristic is consistent, then it is admissible.
7. Consider the following heuristic: given a PDDL domain, create a relaxed domain where all operator preconditions are removed; plan with these operators and report the cost of the plan found. This heuristic is admissible.
8. The heuristic in the previous question will equal the number of goal literals not yet achieved at any state.

Mark all that apply. ◯ 1 ◯ 2 √ **3** ◯ 4 √ **5** √ **6** √ **7** ◯ 8

5 MDPs

9. (20 points) After your finals, you are relaxing on the couch watching TV. There is a remote control that you can reach while sitting on the couch to turn on the TV. Unfortunately, both the TV and the remote are buggy: the TV turns off randomly, and the clicking the remote often has no effect. You can also get up off the couch and turn on the TV manually.

   Let's formulate this scenario as an infinite-horizon MDP. To start:

   - $\mathcal{S} = \{\texttt{tvOn}, \texttt{tvOff}\}$
   - $\mathcal{A} = \{\texttt{clickRemote}, \texttt{pressTV}, \texttt{doNothing}\}$
   - $\gamma = 0.9$

   The transition distribution $P(s' \mid s, a)$ is specified as follows:

   - If $a = \texttt{clickRemote}$, the TV switches between on and off with 0.4 probability, and stays the same with 0.6 probability.
   - If $a = \texttt{pressTV}$, the TV switches between on and off with 1.0 probability.
   - If $a = \texttt{doNothing}$ and the TV is on, it switches off with 0.05 probability.
   - If $a = \texttt{doNothing}$ and the TV is off, it stays off with 1.0 probability.

   The reward function $R(s, a, s')$ is specified as follows, starting at 0:

   - If $s' = \texttt{tvOn}$, $+3$ is added to the reward.
   - If $a = \texttt{clickRemote}$, -1 is added to the reward.
   - If $a = \texttt{pressTV}$, -2 is added to the reward.

   You are considering the following policy:

   - $\pi(\texttt{tvOn}) = \texttt{doNothing}$
   - $\pi(\texttt{tvOff}) = \texttt{clickRemote}$

   (a) Write a system of linear equations to evaluate (compute the value function for) π. You do not need to solve the system.

   Solution: $V(\texttt{tvOn}) = (0.05 * 0.9)V(\texttt{tvOff}) + 0.95(3 + 0.9V(\texttt{tvOn}))$
   $V(\texttt{tvOff}) = 0.6(-1 + 0.9V(\texttt{tvOff})) + 0.4(2 + 0.9V(\texttt{tvOn}))$

   This policy seems reasonable, but having just aced an exam that covered MDPs, you are determined to compute a good policy in a principled way.

   In the course of performing value iteration, you have the following optimal value estimates:

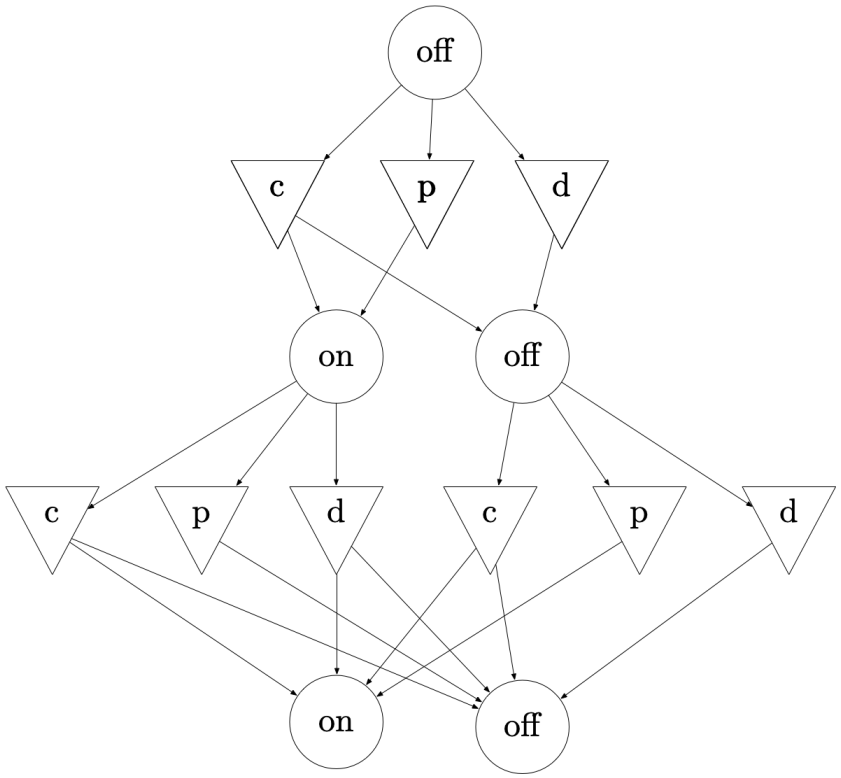
   - $V^*(\texttt{tvOn}) = 2$
   - $V^*(\texttt{tvOff}) = 1$

   (b) What would the new optimal value estimate for $V^*(\texttt{tvOn})$ be after one more iteration of value iteration?

   Solution:
   $$V^*(\texttt{tvOn}) \leftarrow \max \begin{cases} 0.4(-1 + 0.9V^*(\texttt{tvOff})) + 0.6(2 + 0.9V^*(\texttt{tvOn})) & [\texttt{clickRemote}] \\ -2 + 0.9V^*(\texttt{tvOff}) & [\texttt{pressTV}] \\ 0.05(0.9V^*(\texttt{tvOff})) + 0.95(3 + 0.9V^*(\texttt{tvOn})) & [\texttt{doNothing}] \end{cases}$$
   $$V^*(\texttt{tvOn}) \leftarrow \max \begin{cases} 0.4(-1 + 0.9) + 0.6(2 + 1.8) & [\texttt{clickRemote}] \\ -2 + 0.9 & [\texttt{pressTV}] \\ 0.05(0.9) + 0.95(3 + 1.8) & [\texttt{doNothing}] \end{cases}$$
   $$V^*(\texttt{tvOn}) \leftarrow 4.605$$

(c) Unfortunately, all good TV watching sessions must come to an end. Let us now consider a finite-horizon version of the above MDP. Everything remains the same, except now $H = 2$ (so we will take two actions) and $\gamma = 1$. We also know that the TV is off in the initial state. We will want to perform expectimax search to compute an optimal partial policy. Show the full AODAG *without any value annotations*. Do not do any calculations here!

Solution:



10. (4 points) The version of value iteration that we studied in class maintains a value function estimate V^*. One drawback of this approach is that it requires a separate final step to convert V^* into Q^*, and then finally into π^*. In this question, we will consider a different version of value iteration that maintains Q^* instead, and does not ever explicitly represent V^*. Below is almost complete pseudocode:

```
input: S, A, P(s' | s, a), R(s, a, s'), γ
initialize Q*(s, a) ← 0 for all s ∈ S, a ∈ A
while not converged do
  for s ∈ S do
    for a ∈ A do
      Q*(s, a) ← [TODO]
    end for
  end for
end while
return Q*
```

Fill in the missing TODO to complete the algorithm.

Solution: $\sum_{s' \in \mathcal{S}} P(s' \mid s, a)(R(s, a, s') + \gamma \max_{a' \in \mathcal{A}} Q^*(s', a'))$

11. (8 points) Which of the following are true statements about MDP heuristics?

1. MDP heuristics are approximate value functions.
2. An admissible MDP heuristic is less than or equal to 0 at all states.
3. In real-time dynamic programming (RTDP), the heuristic is irrelevant after the first iteration (one trajectory collection) because it is only used to initialize the value estimate.
4. For any MDP, there exists a heuristic s.t. RTDP converges to an optimal policy after one iteration.
5. Stochastic rollouts can be used to obtain a heuristic in any MDP.
6. When performing expectimax search in an **infinite**-horizon MDP with heuristic leaf evaluations, the first action selected will not change if we add a constant to all heuristic outputs.
7. When performing expectimax search in an **indefinite**-horizon MDP with heuristic leaf evaluations, the first action selected will not change if we add a constant to all heuristic outputs.
8. When performing expectimax search in **any** MDP with heuristic leaf evaluations, the first action selected will not change if we multiply a positive constant to all heuristic outputs.

Mark all that apply. √ **1** ◯ 2 ◯ 3 √ **4** √ **5** √ **6** ◯ 7 ◯ 8

12. (8 points) Each of the following questions have a sequence of statements about Monte Carlo techniques for MDP planning. Your job is to identify the *first* statement that is incorrect in each group, if any.

(a)
1. We motivated multi-armed bandits (MAB) as a special case of sparse sampling, which is a planning method for MDPs.
2. The policies that we studied for MDP planning were all *Markov*: each policy depends only on the current state, not the history of states, actions, and rewards.
3. The MAB strategies that we studied (namely ϵ-greedy and UCB) were also *Markov*: the action selected at each step depends only on the current state, not the history.
4. There is no non-Markov MAB strategy that is strictly better than UCB.

Mark **one**: ◯ 1 ◯ 2 √ **3** ◯ 4 ◯ All correct

(b)
1. Sparse sampling builds a AODAG/tree whose size is exponential in the horizon.
2. The memory complexity of sparse sampling is therefore also exponential in the horizon.
3. In finite-horizon MDPs, MCTS also has memory complexity that is exponential in the horizon.
4. In infinite-horizon MDPs, MCTS has unbounded memory complexity.

Mark **one**: ◯ 1 √ **2** ◯ 3 ◯ 4 ◯ All correct

(c)
1. MCTS is a special case of upper-confidence trees (UCT).
2. The aspect of MCTS distinguishing it from other UCTs is the Explore function.
3. The Explore function in MCTS uses ideas from UCB.
4. MCTS with UCB is guaranteed to optimize cumulative regret.

Mark **one**: √ **1** ◯ 2 ◯ 3 ◯ 4 ◯ All correct

(d)
1. Sparse sampling and MCTS avoid exhaustive Bellman backups.
2. Sparse sampling and MCTS require only simulator access to the MDP.
3. It is possible to obtain performance guarantees on sparse sampling that are independent of the number of states in the MDP.
4. MCTS is influenced by the reward function when building out the tree/AODAG, but sparse sampling is not.

Mark **one**: ◯ 1 ◯ 2 ◯ 3 ◯ 4 √ **All correct**

6 Prospecting

13. (16 points) **Disclaimer***: this example is completely made-up and probably ridiculous from the petroleum engineering standpoint.*

Your job is to automate the decision-making process for oil exploration and drilling. In a particular site, your goal is to strike pumpable oil, if possible.

- At that site, there might be shallow oil, deep oil, or no oil.
- You can: drill a test well and take a sample (but it won't work to pump oil from a test well), drill a shallow well, drill a deep well, or give up on this site.
- If you drill a test well, you get one of two observations: "oil" or "no oil". The probability of "no oil" when there is no oil is 1. The probability of "no oil" when there is deep oil is 0.3. The probability of "no oil" when there is shallow oil is 0.1.
- If you drill a test well, and there is shallow oil, then there is a 0.2 chance that the floor of the reservoir will rupture and the oil will become deep, otherwise it will stay shallow. Drilling a test well has no other possible effects.
- It costs -10 to drill a test well, -50 to drill a shallow well, and -200 to drill a deep well. A shallow well is guaranteed to hit oil if there is shallow oil. A deep well is guaranteed to hit oil if there is shallow or deep oil. If you hit oil with a shallow or a deep well, it is worth +1000. Giving up is worth 0. There is no discounting.

(a) Write down the state space, action space, and the reward function for this problem.

Hint: there are three states, four actions. You can draw a 3 by 4 table to represent the reward function.

Solution: State space: {no, shallow, deep}. Action space: {giveup, test, shallow, deep}. Reward function:

	giveup	test	shallow	deep
no	0	-10	-50	-200
shallow	0	-10	950	800
deep	0	-10	-50	800

(b) If your initial belief $b = (1/3, 1/3, 1/3)$, what is the belief state resulting from drilling a test well and observing "oil"? Assume that the observation depends on the starting state (before the test well has its potential effect on the state).

Solution: $b' \propto (1/3 \cdot 0, 1/3 \cdot 0.9, 1/3 \cdot 0.7)$. Thus, $b' = (0, 0.5625, 0.4375)$.

(c) If your initial belief $b = (1/3, 1/3, 1/3)$, what is the belief state resulting from drilling a test well and observing "no oil"? Again, assume that the observation depends on the starting state (before the test well has its potential effect on the state).

Solution: $b' \propto (1/3 \cdot 1, 1/3 \cdot 0.1, 1/3 \cdot 0.3)$. Thus, $b' = (5/7, 1/14, 3/14)$.

(d) What is the value of the following policy tree at each of the three possible world states? Drill a test well; if the observation is "oil" then drill a deep well, otherwise give up. Show your math work.

Solution:

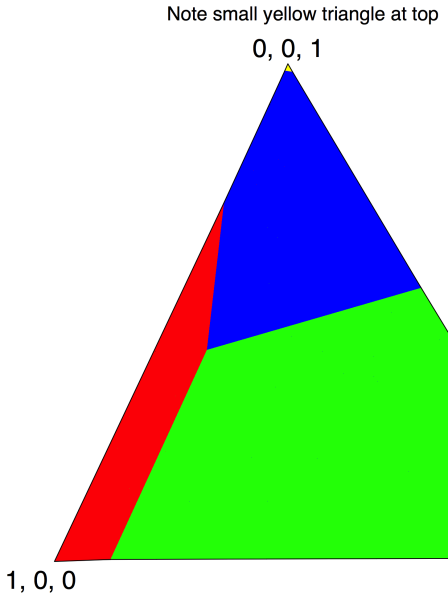
- State = no: -10.

- State = shallow: $-10 + 0.9 \cdot 800 = 710$.
- State = deep: $-10 + 0.7 \cdot 800 = 550$.

(e) The following is a picture of the belief space, partitioned into regions in which different policy trees dominate. The vertices of the belief simplex are labeled with coordinates: the first coordinate is *shallow oil*, the second is *deep oil*, and the third is *no oil*.

For each of the following policies, indicate which color region it corresponds to. Note that there is a tiny yellow triangle at the top.

Policy A : Always drill shallow.
Policy B : Always drill deep.
Policy C : Always give up.
Policy D : Follow the policy in part c.



Yellow:	◯ Policy A	◯ Policy B	√ **Policy C**	◯ Policy D
Blue:	◯ Policy A	◯ Policy B	◯ Policy C	√ **Policy D**
Red:	√ **Policy A**	◯ Policy B	◯ Policy C	◯ Policy D
Green:	◯ Policy A	√ **Policy B**	◯ Policy C	◯ Policy D

7 Tiger in the Room

14. (16 points) Consider the Tiger and Pot-of-Gold POMDP. A robot is faced with the choice of opening two doors (left and right). Behind one door sits a tiger, behind the other door is a pot of gold, and the robot is uncertain which door is which. The robot can open the left door, open the right door, or listen to try to determine which door the tiger is behind.

- The robot gets +10 reward for opening pot-of-gold door. -100 for opening tiger door. -1 for listening.
- Opening a door resets the tiger to a 50-50 chance of being behind either door.
- When listening, the robot will hear the tiger behind the correct door with a 75% chance. It will hear the incorrect door with a 25% chance.

(a) Write down the state space S, the action space A, the observation space O, the transition function T, the reward function r, and the observation function e (ignore actions that do not yield observations) for this problem.

Hint: there are two states, three actions.

Solution: State space: {tiger-left, tiger-right}. Action space: {open-left, open-right, listen}. Observation space: {heard-left, heard-right}.

Transition function:

	open-left	open-right	listen
tiger-left	Pr[tiger-left] = 0.5	Pr[tiger-left] = 0.5	Pr[tiger-left] = 1
tiger-right	Pr[tiger-left] = 0.5	Pr[tiger-left] = 0.5	Pr[tiger-left] = 0

Reward function:

	open-left	open-right	listen
tiger-left	-100	10	-1
tiger-right	10	-100	-1

Observation function:

	open-left	open-right	listen
tiger-left	N/A	N/A	Pr[heard-left] = 0.75
tiger-right	N/A	N/A	Pr[heard-left] = 0.25

(b) Consider the horizon 1 policy (take one action and then terminate). For the underlying MDP (S, A, T, r), compute the value V and the best action π^* for each state.

Solution:

$$V(\text{tiger-left}) = V(\text{tiger-right}) = 10.$$

$$\pi^*(\text{tiger-left}) = \text{open-right}, \pi^*(\text{tiger-right}) = \text{open-left}.$$

(c) Consider the infinite horizon policy with discount factor 0.9. For the underlying MDP (S, A, T, r), compute the value V and the best action π^* for each state.

Solution:

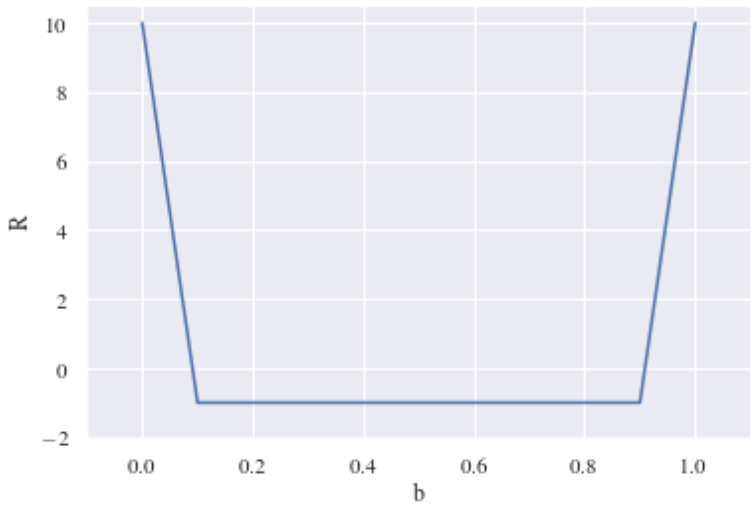
$$V(\text{tiger-left}) = V(\text{tiger-right}) = 100.$$

$$\pi^*(\text{tiger-left}) = \text{open-right}, \pi^*(\text{tiger-right}) = \text{open-left}.$$

(d) Consider the horizon 1 policy (take one action and then terminate). Since we have only two states, we can use a single scalar b to parameterize our belief. Draw a 2D plot for the expected value (under optimal action): use x-axis to denote the belief scalar b, and y-axis to denote the expected value. Mark the optimal action to take for each segments in your drawing. Also mark both ends for each segments in your drawing.

Solution: The curve contains three parts. Let's use b to denote the belief that tiger is in the left door.

$$R(b) = \max\{10b - 100(1-b), -1, 10(1-b) - 100b\}.$$



The left-most region corresponds to "open-left"; the middle region corresponds to "listen"; the right region corresponds to "open-right".

(e) Consider using QMDP to solve this horizon-1 POMDP problem. Specify the policy as a function of b.

Solution: When $b \leq 0.1$, $a =$ open-left. When $b \geq 0.9$, $a =$ open-right. Otherwise, $a =$ listen

(f) Consider using QMDP to solve this horizon-1 POMDP problem. Specify the condition of b such that QMDP yields the optimal action.

Solution: For all $b \in [0, 1]$, QMDP yields the optimal action, by comparing (d) and (e).

(g) Now, use Expectimax over the belief-space MDP to solve this problem. Draw the optimal horizon-2 policy trees for $b = 0.05, 0.2, 0.5$ (draw three trees), and compute the expected return for each tree (at the root node).

Solution: When $b = 0.05$, at root, we should do listen. If we observe right, the next action should be open-left. If we observe left, the next action should be listen. The expected return is $0.95 \cdot 0.75 \cdot 9 - 0.05 \cdot 0.25 \cdot 101 - 2 \cdot (0.05 \cdot 0.75 \cdot 0.95 * 0.25) = 4.6$.

When $b = 0.2$, at root, we should do listen. If we observe right, the updated belief will be $<$ 0.1 (around 0.07). In this case, the next action should be open-left. If we observe left, the next action should be listen. Expected return is $0.8 \cdot 0.75 \cdot 9 - 0.2 \cdot 0.25 \cdot 101 - 2 \cdot (0.8 \cdot 0.25 + 0.2 \cdot 0.75) = -0.35$.

When $b = 0.5$, we should do two listen actions. Expected return is -2.