Recitation 10 (Midterm Review) April 3

1 First Order Logic

1. Vocab check: are you familiar with the following terms?

   (a) Objects

   Things that can be described with relations, and can be returned from functions

   (b) Relations

   Sentences that describe relationships between objects, e.g. Fatherof(a, b). Returns a boolean.

   (c) Functions

   Describe an object using another object, e.g. Fatherof(a). Returns an object.

   (d) Constants

   Specific objects in the world

   (e) Variables

   Lower case letters that can take the value of different objects

   (f) Interpretation

   A mapping of variables to objects

   (g) Connectives

   Describe relationships between sentences (same as operators in propositional logic: $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$)

   (h) Equality

   Given an interpretation, two terms are equal if they are mapped to the same object

   (i) Quantifiers

   $\exists, \forall$

   (j) Atomic Sentence

   Sentences consisting of only one relation (no connectives)

   (k) Unification

   Binding of variables with known objects with some substitution θ (similar to interpretation)

2. Which of the following FOL sentences correctly expresses its corresponding English sentence?

   (a) There was a student at CMU who never did 281 homework but passed the class.
   $\exists x, \text{IsStudent}(x, \text{CMU}) \wedge \neg\text{DoesHW}(x, 281) \implies \text{Pass}(x, 281)$

   False. This sentence is true as long as there exists an x who is not a student at CMU or does 281 homework. This should instead be $\exists x, \text{IsStudent}(x, \text{CMU}) \wedge \neg\text{DoesHW}(x, 281) \wedge \text{Pass}(x, 281)$.

   (b) If a student likes Pat, they'll pass the class.
   $\forall x, \text{Student}(x) \wedge \text{Likes}(x, \text{Pat}) \implies \text{Pass}(x, 281)$

   True.

   (c) All students at CMU who never did 281 homework passed the class.
   $\forall x, \text{IsStudent}(x, \text{CMU}) \wedge \neg\text{DoesHW}(x, 281) \wedge \text{Pass}(x, 281)$

   False - this should instead be $\forall x, \text{IsStudent}(x, \text{CMU}) \wedge \neg\text{DoesHW}(x, 281) \implies \text{Pass}(x, 281)$

3. Which of the following is true with respect to the English sentences (a) and (c) above?

   (a) $\models$ (c) (c) $\models$ (a) Both Neither

   Using the correct formulations of (a) and (c), where
   (a): $\exists x, \text{IsStudent}(x, \text{CMU}) \wedge \neg\text{DoesHW}(x, 281) \wedge \text{Pass}(x, 281)$
   (c): $\forall x, \text{IsStudent}(x, \text{CMU}) \wedge \neg\text{DoesHW}(x, 281) \implies \text{Pass}(x, 281)$
   neither (a) $\models$ (c) nor (c) $\models$ (a) are true.

4. Forward Chaining

   (a) What are the requirements for Knowledge Base in Forward Chaining?

   KB can only contain definite clauses (i.e., clauses with exactly one positive literal).

   (b) What does FOL Forward Chaining return?

   Given a knowledge base KB and sentence α, FOL-FC returns a unification θ for α, or false if no unification is possible.

5. DPLL: determine whether the following are satisfiable or unsatisfiable using DPLL.

   (a) $(P \vee Q \vee \neg R) \wedge (P \vee \neg Q) \wedge \neg P \wedge R \wedge U$

   Assign $\top$ to U (Pure Symbol): $(P \vee Q \vee \neg R) \wedge (P \vee \neg Q) \wedge \neg P \wedge R$
   Assign $\bot$ to P (Unit Clause): $(Q \vee \neg R) \wedge \neg Q \wedge R$
   Assign $\bot$ to Q (Unit clause): $\neg R \wedge R$
   This sentence is unsatisfiable!

   (b) $(P \vee Q) \wedge \neg Q \wedge (\neg P \vee Q \vee \neg R)$

   Assign $\bot$ to R (Pure Symbol): $(P \vee Q) \wedge \neg Q$
   Assign $\top$ to P (Pure Symbol): $\neg Q$
   Assign $\bot$ to Q (Unit clause): T
   This sentence is satisfiable! The model found is $P : \top, Q : \bot, R : \bot$.

2 Planning

1. Vocab check: are you familiar with the following terms?

   (a) Predicates

   Conditions that the current state of the world must satisfy to perform an operation.

   (b) Operators

   Perform actions that change the state of the world

   (c) Linear Planning

   Solve one goal at a time.

   (d) Non-linear Planning

   Interleave goals to achieve plans

   (e) Inconsistency

   The effects of two actions negate each other

   (f) Interference

   One action deletes/negates a precondition of the other

   (g) Competing Needs

   The preconditions of two actions negate each other

   (h) Complete

   Can always find a solution whenever one exists

   (i) Sound

   All solutions found are legal.

   (j) Optimal

   Shortest path to goal

2. What are in the knowledge base of logic agents and classical planning problem, respectively?

   Logical agents: Symbols and Implications
   Planning: Predicates and operators

3. What are the 3 components when defining an operator?

   Preconditions, add list, delete list (you could technically define in two: Preconditions, add list)

4. Is linear planning complete? Optimal? What about non-linear planning?

   Linear Planning: Not complete, one goal can immediately undo the other, and the plan stucks there. Not optimal: The plan naively attempts in order where there can be shortcuts among some goals. Non-linear Planning: Complete and optimal. It performs search and would always reach the depth of the solution whenever there exists one. It returns the solution as soon as one is found so there can't be any shorter path (if all interleavings are searched).

5. The monkey-and-bananas problem is faced by a monkey in a laboratory with some bananas hanging out of reach from the ceiling. A box is available that will enable the monkey to reach the bananas if he climbs on it. Initially, the monkey is at A, the bananas at B, and the box at C. The monkey and box have height *Low*, but if the monkey climbs onto the box he will have height *High*, the same as the bananas. The actions available to the monkey include *Go* from one place to another, *Push* an object from one place to another, *ClimbUp* onto or *ClimbDown* from an object, and *Grasp* or *Ungrasp* an object. The result of a *Grasp* is that the monkey holds the object if the monkey and object are in the same place at the same height. We want to formulate this problem using GraphPlan.

   (a) Define the initial and goal states.

Initial state:
At(Monkey,A), At(Bananas,B), At(Box,C), Height(Monkey,Low), Height(Box,Low), Height(Bananas,High), Pushable(Box), Climbable(Box)

Goal state:
Have(Monkey, Bananas)

(b) Write the six action schemas, i.e., define the preconditions and effects for each possible action the monkey can take.

i. *Action*(Go(x,y))

A. **Preconditions:** At(Monkey,x)

B. **Add List:** At(Monkey,y)

C. **Delete List:** At(Monkey,x)

ii. *Action*(Push(b,x,y))

A. **Preconditions:** At(Monkey,x), At(b,x), Pushable(b)

B. **Add List:** At(b,y), At(Monkey,y)

C. **Delete List:** At(b,x), At(Monkey,x)

iii. *Action*(ClimbUp(b))

A. **Preconditions:** At(Monkey,x), At(b,x), Climbable(b)

B. **Add List:** On(Monkey,b), Height(Monkey,High)

C. **Delete List:** Height(Monkey,Low)

iv. *Action*(Grasp(b))

A. **Preconditions:** Height(Monkey,h), Height(b,h), At(Monkey,x), At(b,x)

B. **Add List:** Have(Monkey,b))

v. *Action*(ClimbDown(b))

A. **Preconditions:** On(Monkey,b), Height(Monkey,High)

B. **Add List:** Height(Monkey,Low)

C. **Delete List:** On(Monkey,b), Height(Monkey,High)

vi. *Action*(UnGrasp(b))

A. **Preconditions:** Have(Monkey,b)

B. **Delete List:** Have(Monkey,b))

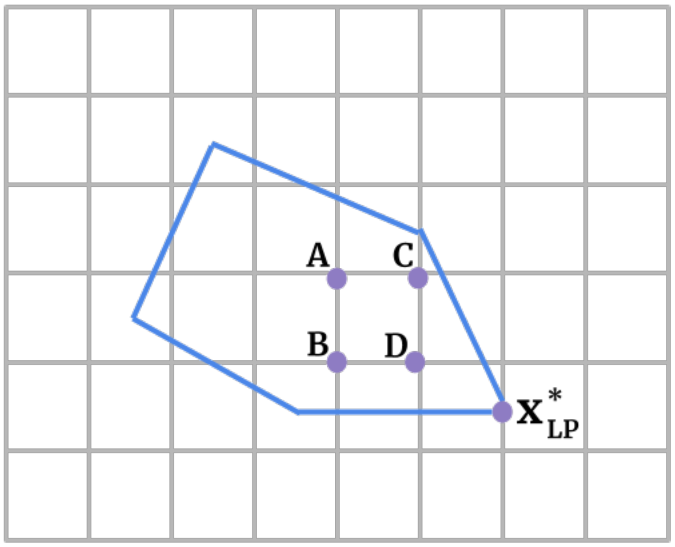
3 IP

1. What is the relationship between branch and bound and pruning?

   In branch and bound, we stop recurring down a branch when the LP returns a worse objective than the best feasible IP objective seen so far. This is similar to pruning in Minimax because we know we can do better by taking another branch, so we prune.

2. Consider the integer programming problem illustrated by the figure below.

   - The blue lines represent the boundaries of the feasible region and the gray vertical and horizontal lines represent the integer values for each axis.
   - Let x^*_{LP} be the unique point that minimizes the relaxed linear program. It happens to lie on a vertical gray line and a horizontal blue line.
   - Let x^*_{IP} (unlabeled) be the unique point that minimizes the integer program.
   - When running branch and bound, we will explore the x_i less than constraint subtree before the x_i greater than constraint subtree ("less" being to the left and down).



   (a) Which of the points (A, B, C, or D) can possibly be x^*_{IP}?

   C or D
   Since x^*_{LP} is the unique point that minimizes the related linear program and not the vertex directly left of it, we know that the horizontal component of the cost vector will point left. Thus, C will be of lower cost than A, and D will be of lower cost than B. This means that A or B can never be the optimal integer solution.

   (b) Which of the points from your answer to part (a) needs to be x^*_{IP} in order for branch and bound to find the IP solution at the minimum possible depth?

   D

   If we know that D is the unique point that minimizes the integer program, we know that the vertical component of the cost vector has to point up. If we run branch and bound knowing that, we find the optimal IP solution at depth 2.

However, if we know that C is the unique point that minimizes the integer program, we know that the vertical component of the cost vector has to point down. If we run branch and bound, we find the optimal IP solution at a depth greater than 2.

4 MDPs

1. Concepts:

   (a) What does the Markov Property state?

   Markov Property states that action outcomes depend on the current state only. It states that action outcomes do not depend on the past.

   (b) What are the Bellman Equations, and when are they used?

   The Bellman Equations give a definition of "optimal utility" via expectimax recurrence. They give a simple one-step lookahead relationship amongst optimal utility values.

   (c) What is a policy? What is an optimal policy?

   A policy is a function that maps states to actions. $\pi(s)$ gives an action for state s. An optimal policy is a policy that maximizes the expected utility if an agent follows it.

   (d) How does the discount factor γ affect the agent's policy search? Why is it important?

   γ determines how much the value of a state should take into account future states. The higher the discount factor, the more one state would value distant states. Having $0 \le \gamma < 1$ also helps our algorithms converge.

   (e) What are the two steps to Policy Iteration?

   Policy evaluation and policy improvement.

   (f) What is the relationship between $V^*(s)$ and $Q^*(s,a)$?

   $V^*(s) = \max_a Q^*(s,a)$

2. In a certain country there are N cities, all connected by roads in a circular fashion. A wandering poet is travelling around the country and staging shows in its different cities. She can choose to move from a city to each of the neighboring ones or she can stay in her current city i and perform, getting a reward r_i. If she chooses to travel, she will have a success probability of p_i. There is a $1-p_i$ chance she will encounter a dragon along the way, which means she will have to turn back and wait the next day. If she is successful in travelling, she gains a reward of 0 for the day. And if she is unsuccessful at travelling, she can still perform a little bit when she gets back, giving her a reward of $r_i/2$.

   Let $r_i = 1$ and $p_i = 0.5$ for all i and let $\gamma = 0.5$. For $1 \le i \le N$, answer the following questions with concrete numbers:

   i) What is the value $V^{\text{stay}}(i)$ under the policy the wandering poet always chooses to stay?

   $$V^{\text{stay}}(i) = r_i + \gamma V^{\text{stay}}(i) = 1 + 0.5V^{\text{stay}}(i)$$
   $$V^{\text{stay}}(i) = 2$$

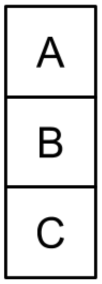
   ii) What is the value $V^{\text{next}}(i)$ under the policy the wandering poet always chooses to go to the next city? You may assume that the values of each state converge to the same value.

   $$V^{\text{next}}(i) = 0.5(\frac{1}{2} + 0.5V^{\text{next}}(i)) + 0.5(0 + 0.5V^{\text{next}}(i+1))$$
   $$V^{\text{next}}(i) = V^{\text{next}}(i+1) = \frac{1}{2}$$

5 Reinforcement Learning: Just Add Water

Rikki the mermaid is trying to learn a general policy for swimming to Mako Island while avoiding Charlotte[1], the inferior and annoying mermaid. Rikki always starts in grid A shown below.

Rikki's possible environment states are of the form (x, y), where $x \in \{A, B, C\}$, representing her grid position, and y is 1 if Charlotte is present in grid C, 0 if she is nowhere on the map. Rikki's possible actions are swim (S) or heat (H). She observes the episodes shown to the right:



s	a	s'	r
(A, 1)	S	(C, 1)	-4
(A, 1)	S	(B, 1)	0
(B, 1)	S	(C, 1)	-4
(A, 0)	S	(C, 0)	2
(A, 0)	S	(B, 0)	0
(B, 0)	S	(C, 0)	2
(A, 1)	S	(B, 1)	0
(B, 1)	H	(B, 0)	0
(B, 0)	S	(C, 0)	2

1. Which of the following are techniques Rikki can use to try finding an optimal policy (without having to use any other process in addition to the technique)?

   - Policy iteration
   - TD learning
   - Estimate T and R, then policy iteration
   - Q-learning
   - Value iteration, then policy extraction
   - Approximate Q-learning

   Estimate T and R, then policy; Q-learning, Approximate Q-learning.
   We can't perform policy iteration or value iteration without access to the transition and reward functions.
   TD-learning would help us evaluate the states, but we wouldn't be able to use it to learn an optimal policy without the transition and reward functions.

2. Let $Q(s, a) = 0$ initially for all Q-states (s, a), and that $(C, 0)$ and $(C, 1)$ are terminal states. Let $\gamma = \alpha = 0.5$. Now calculate $Q(s, a)$ for the entries in the following table after Q-learning.

s	(A, 1)	(A, 0)	(B, 1)	(B, 0)	(A, 1)	(B, 1)
a	S	S	S	S	H	H
$Q(s,a)$						

s	(A, 1)	(A, 0)	(B, 1)	(B, 0)	(A, 1)	(B, 1)
a	S	S	S	S	H	H
$Q(s,a)$	-0.5	0.5	-2	1.5	0	0.25

[1]who sucks

Note that without even being intrinsically valuable (i.e., without resulting in a reward in and of itself), Rikki's agent learns that the heat action is good. As you'd imagine, this is a super useful property when teaching an agent something like walking - the action space is so large, and it's hard to say which actions are "good" beforehand.

3. According to the observed episodes, what is Rikki's optimal policy?

s	(A, 1)	(A, 0)	(B, 1)	(B, 0)
$\pi^*(s)$	H	S	H	S

4. Suppose Rikki uses ϵ-greedy search, and that her environment is deterministic. She starts with $\epsilon = 0$ and doesn't change it. Is she guaranteed to learn the optimal policy? Why or why not?

   Rikki is not guaranteed to learn the optimal policy because she will never explore potentially better actions and states.

5. Now suppose Rikki sets $\epsilon = 1$ and doesn't change it. Is she guaranteed to learn the optimal policy? Why/why not?

   Rikki will learn the optimal policy because she will eventually explore all actions and states. However, she will never act according to the optimal policy because she always moves randomly. This method will accumulate more regret than $\epsilon < 1$.
   This is the idea behind ϵ-greedy search with decaying ϵ, a strategy commonly used in practice: explore more at the beginning to learn, then slowly begin acting according to the learned policy.

6. Rikki's no ordinary girl, so she decides to use approximate Q-learning, with $\alpha = \gamma = 0.5$. Suppose she has three features f_1, f_2, f_3, and corresponding weights w_1, w_2, w_3.
   Her approximate function is

$$Q_w(s,a) = w_1^2 f_1(s,a) + e^{w_2 f_2(s,a)} + w_3^3 f_3(s,a).$$

   Initially, $w_1 = 2, w_2 = 0, w_3 = 3$. Suppose $f_1((A,1),S) = f_2((A,1),S) = f_3((A,1),S) = 1$.

   (a) What is the approximate Q-value for $((A,1),S)$?

   $Q_w((A,1),S) = 2^2(1) + e^{0(1)} + 3^3(1) = 32.$

   (b) What is the value of $((A,1),S)$ according to Rikki's first observation $((A,1),S,(C,1),-4)$? Suppose $max_{a'} Q_w((C,1),a') = 4$.

   Using her first observation, Rikki's approximate Q-value for Q-state $((A,1),S)$ is given by the equation $Q_{sample}(s,a) = R + \gamma(max_{a'} Q_w(s',a'))$.

$$Q_{sample}((A,1),S) = -4 + 0.5(4) = -2.$$

   Note that this represents something like Q_{sample}, i.e. a new evaluation of this Q-state based on the observation (where "sample" refers to $((A,1),S,(C,1),-4)$).

   (c) What are the new weights after the first observation?

First note that

$$\frac{\partial Q_w}{\partial w_1} = 2w_1 f_1(s,a),\ \frac{\partial Q_w}{\partial w_2} = f_2(s,a)e^{w_2 f_2(s,a)},\text{ and } \frac{\partial Q_w}{\partial w_3} = 3w_3^2 f_3(s,a).$$

Thus each of the weights will be updated using the equation

$$w_i = w_i + \alpha(R + \gamma max_{a'} Q_w(s',a') - Q_w(s,a))\frac{\partial Q_w(s,a)}{\partial w_i}.$$

$\alpha(R + \gamma max_{a'} Q_w(s',a') - Q_w(s,a)) = 0.5(-2-32) = -17$ is the same for all 3 weights. (Note that the $Q_w(s,a)$ we use here are the old, Q-values prior to the update in part (b).)

$$w_1 = 2 - 17(2(2)(1)) = -66$$

$$w_2 = 0 - 17(1 \times e^{0(1)}) = -17$$

$$w_3 = 3 - 17(3(3)^2)(1) = -456$$

Intuitively, these are what we'd expect because our approximation function (using the original weights) evaluated the Q-state $((A,1),S)$ highly positively (part (a)), but the actual reward was negative, in turn resulting in a decreased Q-value for that state after performing the update (part (b)). Thus, the weights shift to reflect this "unpleasant surprise", in attempt to evaluate similar states (similar feature values) more negatively in the future.

6 Probability

Recall the example of *marginalization*, which means summing out variables from a joint distribution. Consider three random variables A, B, and C with domains $\{a_1, a_2\}$, $\{b_1\}$, and $\{c_1, c_2, c_3\}$, respectively. Remember that $P(C)$ refers to the table of probabilities of all the elements of the domain C.

1. Express $P(C)$ in terms of the joint distribution $P(A, B, C)$. Your answers should contain summations.

   We can find $P(C)$ by summing out the variables A and B from the joint distribution $P(A, B, C)$. Remember that $P(C)$ is a probability table, so the equations below can take on three values: one for c_1, one for c_2, and one for c_3.

$$P(C) = \sum_{A \in \{a_1, a_2\}} \sum_{B \in \{b_1\}} P(A, B, C)$$

   For convenience, we will abbreviate the above sums as

$$P(C) = \sum_A \sum_B P(A, B, C)$$

2. Express $P(C)$ in terms of $P(B)$, $P(A \mid B)$ and $P(C \mid A, B)$. Reorder any summations in your solution to minimize the number of arithmetic operations.

   By the chain rule, $P(B)P(A \mid B)P(C \mid A, B) = P(A, B, C)$, so we can substitute this into the expression from (1) and get

$$P(C) = \sum_A \sum_B P(B)P(A \mid B)P(C \mid A, B)$$

   We can reorder the sums and factor to get

$$P(C) = \sum_B P(B) \sum_A P(A \mid B)P(C \mid A, B)$$

3. Expand the sums from part (b) to show the two elements of $P(C)$ ($P(c_1)$,$P(c_2)$,and $P(c_3)$) in terms of the individual probabilities (e.g. $P(a_2)$, $P(b_1)$ instead of $P(A)$ or $P(B)$).

   We evaluate these sums by simply plugging in each possible value for a and b and taking the sum. To get the individual element in the table $P(C)$, we plug in the corresponding value (either c_1, c_2, or c_3).

$$\begin{aligned}
P(c_1) &= P(b_1)\left(P(a_1 \mid b_1)P(c_1 \mid a_1, b_1) + P(a_2 \mid b_1)P(c_1 \mid a_2, b_1)\right) \\
P(c_2) &= P(b_1)\left(P(a_1 \mid b_1)P(c_2 \mid a_1, b_1) + P(a_2 \mid b_1)P(c_2 \mid a_2, b_1)\right) \\
P(c_3) &= P(b_1)\left(P(a_1 \mid b_1)P(c_3 \mid a_1, b_1) + P(a_2 \mid b_1)P(c_3 \mid a_2, b_1)\right)
\end{aligned}$$

   Thus we have achieved concrete expressions for $P(c_1)$, $P(c_2)$, and $P(c_3)$ in terms of the given values (the rows in the conditional probability tables).

4. Each of the following terms represents a conditional probability table. What do the entries for each table sum to? (For example, the entries of the table $P(A)$ sum to 1.)

   (a) $P(A \mid B)$

   (b) $P(A \mid C)$

   (c) $P(C \mid a_1, b_1)$

(d) $P(C \mid a_1, B)$

(e) $P(C \mid A, b_1)$

The last section of the probability review sheet may help with these questions!

(a) $P(A \mid B)$ will have two entries, $P(a_1 \mid b_1)$ and $P(a_1 \mid b_1)$. These entries sum to 1.

(b) $P(A \mid C)$ contains entries for $P(A \mid c_1)$, $P(A \mid c_2)$, and $P(A \mid c_3)$. Each of these three "subtables" sums to 1, so the entire table overall sums to 3.

(c) $P(C \mid a_1, b_1)$ contains the entries $P(c_1 \mid a_1, b_1)$, $P(c_2 \mid a_1, b_1)$, and $P(c_3 \mid a_1, b_1)$; this sums to 1.

(d) $P(C \mid a_1, B)$ contains all entries for $P(C \mid a_1, b_1)$, which sum to 1.

(e) $P(C \mid A, b_1)$ contains entries for $P(C \mid a_1, b_1)$ and $P(C \mid a_2, b_1)$; this sums to 2.