CCIS 4100: Midterm

3/3/2017

This exam comprises 4 multi-part questions. You have the entire class period (~1 hour 30 minutes and some change) to complete it; it must be turned in by 5:05pm. Hopefully you won't need the whole period! But in any case, **budget your time accordingly; answer questions concisely.** You may use a calculator, but you shouldn't need one.

Questions 3 and 4 use a shared game/setup, so do invest the time to read carefully about the game introduced in Question 3.

All answers need to be written on this exam. You may use the blank pages interspersed.

Good luck, the robots are rooting for you!

Your name: ______________________________________

1 Definitions and Concepts (*20 points*)

Answer the following short-answer questions **concisely**, as in 2 sentences per question *at most.*

a. What is the basic aim of reinforcement learning? (*2 points*)
Answer: To learn an optimal policy from observed (state, action, reward) triplets.

b. Define *arc-consistency.* (*2 points*)
Answer: X_i is arc-consistent w.r.t. X_j iff for every value v_i that it may still be assigned, there exists at least one value v_j that X_j might be assigned that is consistent with v_i (does not violate binary constraints on X_i, X_j).

c. What makes a heuristic *admissible* for A* search? (*2 points*)
Answer: It needs to be optimistic, i.e., never over-estimate the true distance to the goal.

d. When is a search algorithm *complete*? Is Depth First Search (DFS) complete? (*2 points*)
Answer: A search algorithm is complete if it is guaranteed to find a solution if one exists. Yes.

e. Describe a scenario (i.e., a search state problem) in which iterative deepening search performs much worse than DFS. (*2 points*)
Answer: One such scenario is when the solution is to be found at the bottom left of the tree (assume DFS branches left first). In this case, DFS will go straight to the solution, while iterative deepening will explore large swathes of the search tree.

f. Explain the relationship between Uniform Cost Search (UCS) and Breadth First Search (BFS). (*2 points*)
Answer: UCS reduces to BFS if edge costs are uniform.

g. In Constraint Satisfaction Problems (CSPs), what is the heuristic often used to select variables? Once selected, what is the heuristic then often used to select a corresponding value? (*2 points*)

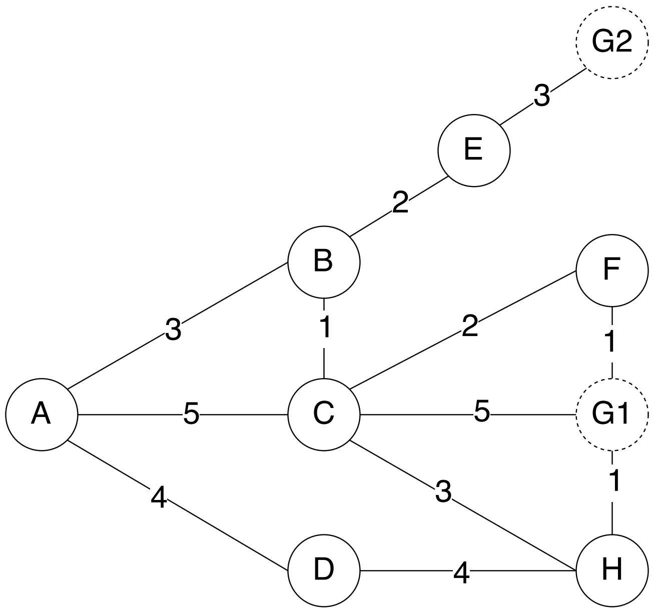
Answer: To select a variable, choose the most constrained variable; to assign it a value, choose the least constraining option from its domain.

h. In the context of Markov Decision Processes (MDPs), what is the effect of the *discounting factor* γ, why did we introduce this? (*2 points*)
Answer: This makes the agent prefer collecting rewards sooner rather than later. Recall that this prevented the 'thrashing' behavior we saw in PacMan when he didn't care when he would collect the pellet (only that he eventually would).

i. In the context of adversarial search, what are evaluation functions and why/when are they used? (*2 points*)
Answer: These are functions that attempt to provide an estimate regarding the min or max value of a given node, usually using a linear function of 'features' that describe the corresponding state. These can be used to avoid having to explore impractically large search trees.

j. Why is temporal difference (TD) learning of Q-values (Q-learning) usually preferable to TD-learning of values? (*2 points*)
Answer: Q-values allow us to decide how to act; the state values themselves do not tell us this.

2 Search (*25 points*)



h1	h(A)	h(B)	h(C)	h(D)	h(E)	h(F)	h(H)
Value	6	3	2	1	1	1	1

Figure 1: A search graph with multiple goal states and heuristic values under heuristic function $h1$.

Consider the search graph above where node A is the start state and there are two goal states, G1 and G2. Values on edges denote path costs. The table provides the values of a heuristic function, which is a proxy or estimate of the distance to a goal state.

a. Using the A* search algorithm, which goal state is reached first? Show all calculations performed by A*. (*15 points*)

Answer: A* uses the function $f(n) = g(n)+h(n)$ to evaluate which node should be expanded next.
Expanding different paths
A : $f(A) = g(A) + h(A) = 0 + 6 = 6$

Moving ahead with the only option expanding node A.
A - B : $f(B) = g(B) + h(B) = 3 + 3 = 6$
A - C : $f(C) = g(C) + h(C) = 5 + 2 = 7$
A - D : $f(D) = g(D) + h(D) = 4 + 1 = 5$

Selecting the path with the minimum $f(n)$ value, A - D.
A - D - H: $f(H) = g(H) + h(H) = 8 + 1 = 9$

Next minimum $f(n)$ value is 6 for path A - B.
A - B - E : $f(E) = g(E) + h(E) = 5 + 1 = 6$
A - B - C : $f(D) = g(C) + h(C) = 4 + 2 = 6$

As both the paths have same path breaking the tie randomly and selecting path A - B - E
A - B - E - G2 : $f(G2) = g(G2) + h(G2) = 8 + 0 = 8$

Although we have reached a goal state, the search does not end until the goal state is chosen for expansion.
A - B - C - F : $f(F) = g(F) + h(F) = 6 + 1 = 7$
A - B - C - G1 : $f(G1) = g(G1) + h(G1) = 9 + 0 = 9$
A - B - C - H : $f(H) = g(H) + h(H) = 7 + 1 = 8$

Next, we pick the path with minimum $f(n)$ value.
A - B - C - F - G1 : $f(G1) = g(G1) + h(G1) = 7 + 0 = 7$

We have reached both the goal states now, we pick the path with the minimum value. Hence, goal state G1 is chosen for expansion and this is when the search ends. Finally, we conclude that the search reaches goal state G1 first. The path obtained to reach the goal state would be A - B - C - F - G1.

b. Is the h1 heuristic admissible? Justify your answer. (*5 points*)

Answer: Yes, h1 heuristic is admissible because the heuristic function values at each node is less than equal to the actual cost of reaching the goal state from that node. It does not overestimate the distance to the goal state at any node.
c. Is h1 heuristic consistent? If not, provide an example of where the heuristic

is inconsistent. (*5 points*)

Answer: No, h1 heuristic is inconsistent because the heuristic function value at node D doesn't satisfy the consistency criteria $h(D)-h(H) <= ActualCost(D-H)$.

3 Reinforcement learning (*30 points*)

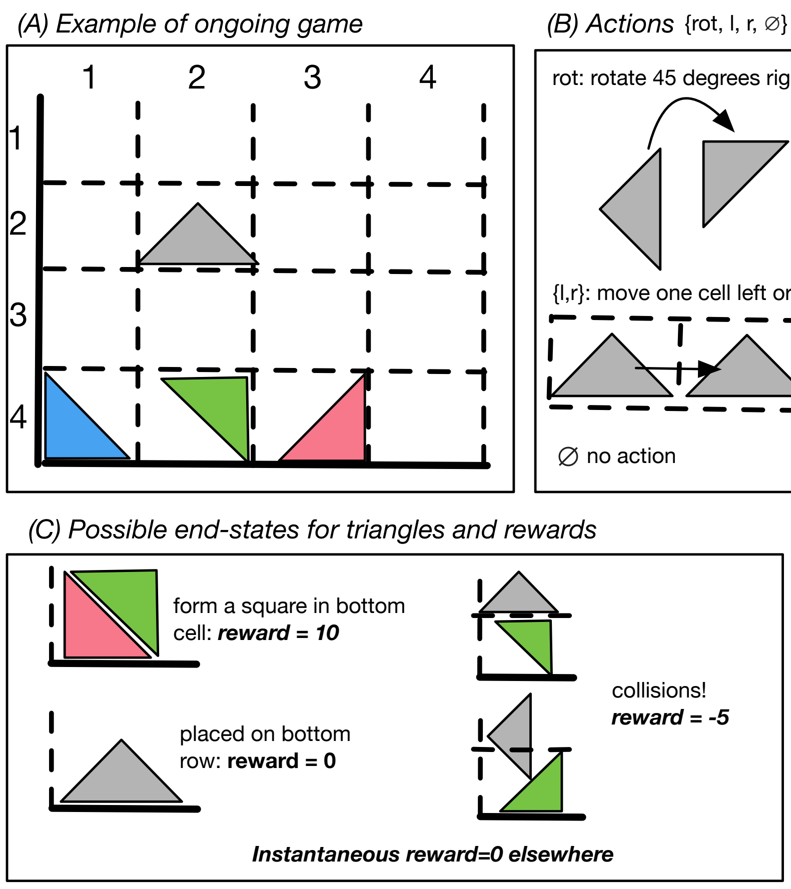

(A) Example of ongoing game
1
2
3
4
1
2
3
4
(B) Actions {rot, l, r, ∅}
rot: rotate 45 degrees right
{l,r}: move one cell left or right
∅ no action
(C) Possible end-states for triangles and rewards
form a square in bottom cell: reward = 10
placed on bottom row: reward = 0
collisions! reward = -5
Instantaneous reward=0 elsewhere

Figure 2: A simple tetris-like game played on a 4x4 grid.

Consider a simple tetris-like game played on a 4x4 grid, as pictured in Figure 2, (A). At the start the grid is clear. One triangle 'falls' from the top at a time; it is placed in a random cell at the top of the grid to begin with. At each time step, the falling triangle will move 1 square down. You get points for placing two triangles in a bottom grid cell such that they form a square, as shown in (C). (Note that the colors are purely aesthetic here.) If you do this, the corresponding cell clears and you get 10 points.

Falling triangles can either: (1) reach the bottom row and form a square, (2) reach the bottom row in an empty cell, or (3) attempt to enter a bottom row cell already occupied in an incompatible way. In this game, (3) happens whenever the falling triangle attempts to enter a bottom row cell already occupied by a triangle that is *not* positioned with one of its legs along the bottom; i.e., only the positions depicted in cells (4,1) and (4,3) in example (A) can result in square formation; (4,2) can not. Even in the former cases, the falling triangle must be positioned correctly to avoid a collision. When a falling triangle 'collides' with one in the bottom row, a reward of -5 is incurred.

We are going to formulate this game as a Reinforcement Learning (RL) task, where the aim is for the agent to learn how to play well. The actions available to the agent are $\{rot, l, r, \emptyset\}$, as shown in the figure. You can assume these are deterministic. Note that you can only rotate the triangle 45 degrees right at a time. Also note that if $\emptyset$ is taken, the triangle will still automatically move 1 square down (it always does, regardless of the action).

a. As defined, one triangle at a time falls and eventually reaches one of 3 'terminal' states, at which point another triangle drops. For simplicity, we will assume that the RL agent treats each new triangle as an independent 'game' to be played. This in contrast to treating all T steps as one longer game, spanning multiple triangle descents (in this case, a reward would only be experienced at the end of play as your final score, which would have spanned multiple triangles). Is this a reasonable simplification? Make a concise, concrete argument to support your position, that is, for or against the strategy of observing rewards every time an individual triangle reaches an end-state. (*5 points*)

Answer:
Arguments in both directions were acceptable. Here is a sample of rationales for each.
This is NOT a reasonable simplification, because:

- Basically, any rationale pointing out that this will make it difficult for the agent to learn how to place the *first* triangle in an empty bottom row.

This is a reasonable simplification, because:

- Rewards are experienced more proximately to Q-states. For instance, if we correctly place the second triangle to form a square with the first, then we will immediately experience this reward. Whereas if T is long, the reward signal for this (successful) action may be quite far in the future!
- When squares are formed they are cleared anyway. Thus this strategy should still allow the agent to learn to place itself at a right-angle when the bottom row is empty and then on top of this when one such triangle has been positioned. The optimal player will just repeat this simple greedy strategy anyway; no longer range patterns need to be learned.
- This approach allows one to learn *while playing* in a given round. Thus we can, e.g., update Q-value estimates during play; this would not be possible if we wait until T steps expire to observe a reward.

b. Define a state-space that will be sufficient to learn to play this game. Assuming a naive encoding, what is the size of this space? (You do not need to evaluate this; writing it as a product is fine.) (*10 points*)

Answer: For this problem, a state needs to encode: (1) the position in the grid of the falling triangle; (2) its orientation (one of 8, since we can move 45 degrees at a time); (3) which of the four bottom row cells are occupied and (4) the orientation of the occupying triangles. So naively, this results in $4 \times 4 = 16$ position states $\times$ 8 orientation $\times\ 2^4 = 16$ bottom row states $\times$ 8 orientation states, or: $16 \times 8 \times 16 \times 8$ unique states (roughly 16,000 states).

c. Temporal Difference Q-Learning (TDL) (*10 points*)

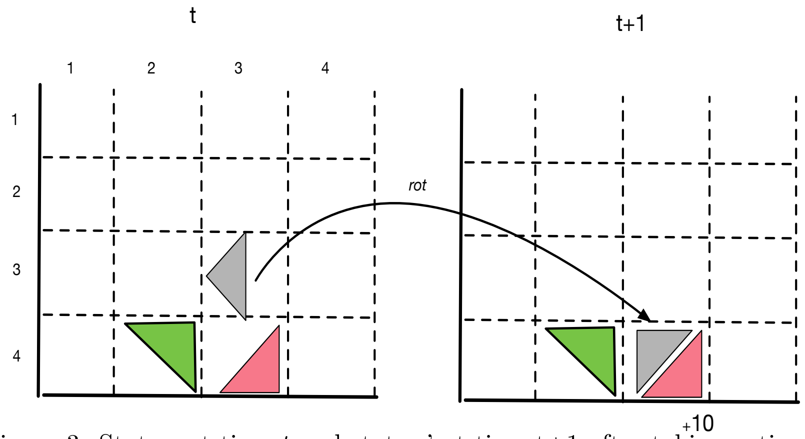


Figure 3: State s at time *t* and state s' at time t+1 after taking action *rot*.

Assume the board state at time *t* is as depicted in Figure 3 (left), so you are at (3,3) and then select the action *rot* and end up in the state depicted in the right of the Figure (time t+1). results in a reward of +10. Suppose we currently have an estimate $\hat{Q}(s, rot) = 5.0$; further suppose our learning rate α is .5. What is the estimate after this event?

Answer: This is just a straight-forward application of sample-based Q-learning. In general: $\hat{Q}(s,a) \leftarrow (1-\alpha) \cdot \hat{Q}(s,a) + \alpha \cdot R(s,a,s') + \gamma \cdot max_a \hat{Q}(s',a')$, but here this reduces to simply $\hat{Q}(s,a) \leftarrow (1-\alpha) \cdot \hat{Q}(s,a) + \alpha \cdot \text{reward} = (.5) \cdot 5 + (.5) \cdot 10 = 7.5$.

d. In light of your answer to (b), what's a potential problem with standard *Q*-learning here? (Consider what happens if we move to an even larger board, say, 16x16.) What could we do, concretely, to address this problem? (*5 points*)

Answer: The state space – and hence Q learning table – is very large, and completely unmanageable if the board size increases further. We would want to use generalized Q-learning, i.e., rather than store a table for state/action pairs, we would represent these via *features*. For example, such features might include indicator features for each of the bottom rows, the distance to the bottom, etc.

4 Game playing (*25 points*)

We will now extend the simplified tetris-like game introduced above to a two-player game; you will now face off against an adversarial agent. The adversary has access to the same 4 moves $\{rot, l, r, \emptyset\}$ that she also plays while the triangle is dropping; so at each time step, you make a move and then the adversary does. The adversary wants to minimize the score.

a. Assume again that the board is in the state depicted in Figure 3 (left hand side; time t). You are going to draw a partial minimax tree from this state. Specifically, this is to include (1) all possible moves you have, and then (2) expand the possible countermoves your adversary has *assuming you have chosen rot*. I am **not** asking you to draw the *entire* two-move minimax tree! (We assume her move follows yours and no further moves are possible once the bottom row is reached). Fill in the terminal nodes in the tree with values. Please order edges in the tree, from left to right, as given, i.e.,: $\{rot, l, r, \emptyset\}$. What is the min value for the adversary given your play of *rot*? (*7 points*)

Answer:

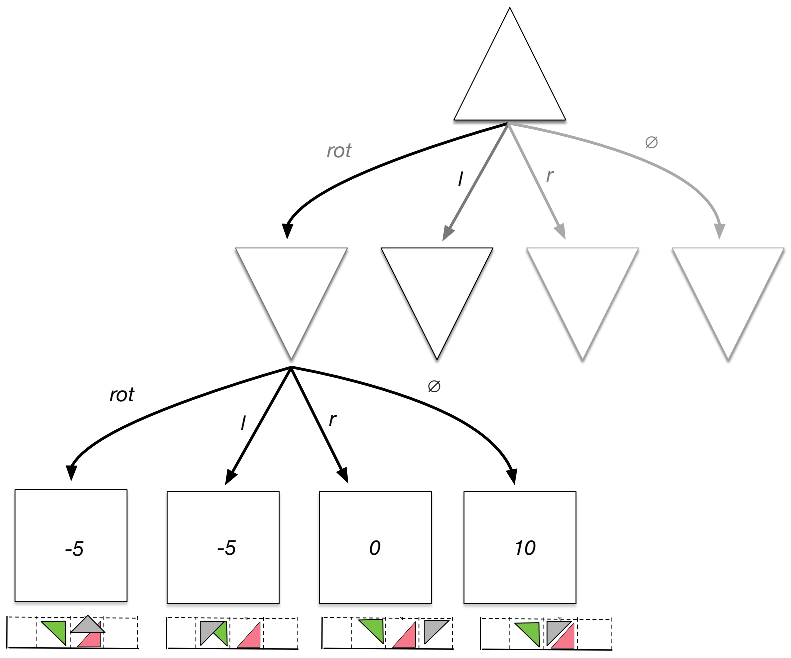


Figure 4: Tetris Game Tree with an adversarial agent

The min value is -5.

b. Now assume you choose to move left (l). Expand min's node from this starting place. Using α-β pruning, and the result from the preceding question (a): can we prune any of the edges here? If so, which? (*8 points*)
Answer:

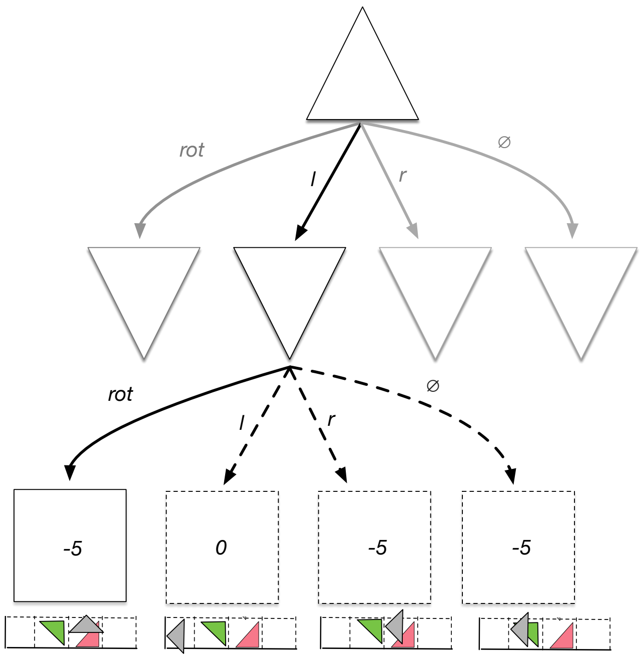


Figure 5: Expanding the min node assuming we've taken action l.

The α value from (a) is -5. We can prune the three rightmost moves (dotted lines and boxes) because min can choose *rot* which gives $-5 \leq \alpha$.

c. According to mini-max, is *rot* preferable to l? Why or why not? (*3 points*)

Answer: Mini-max says they are equivalent (both have a terminal value of -5 under optimal play by our opponent).

d. Now assume that the adversary is playing nondeterministically. Specifically, assume she selects the optimal play with probability 0.80 (4/5) and does nothing (that is, plays $\emptyset$) otherwise. Calculate the expected values of the *rot* and l actions. According to expecti-max, is one of these preferable to the other? (*7 points*)

Answer:
For *rot*: $\frac{4}{5} \cdot -5 + \frac{1}{5} \cdot (10) = -4 + 2 = -2$
For l: $\frac{4}{5} \cdot -5 + \frac{1}{5} \cdot -5 = -5$.
Thus *rot* is preferable under expecti-max.