Artificial Intelligence

Spring 2018: Practice Final

Instructions: This exam is governed by the **Emory Honor Code**. This exam is closed book and closed notes. However, you may use 2 sheets of notes. You may also use an ancient calculator (not an app but the actual device). You will have 90 minutes for this exam. The point values are indicated beside each problem.

Name (print): ____________________________________

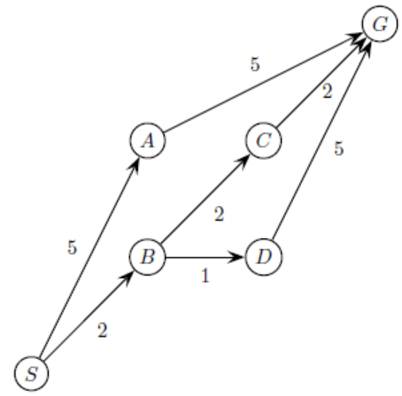
This table is for grading, please leave it blank.

Problem	***Points***	***Score***
1: Search	10	
2: Games	10	
3: Inference and HMMs	10	
4: MDP	10	
5: RL (classical)	10	
6: Neural Networks & Deep RL	20	
Total	80	

This page is intentionally left blank (use it as you like)

Problem 1 (10 pts): Search

For questions a-c, consider the search problem on the right:



(a) (2pts): Which of the heuristics shown are admissible?

(b) (2pts): Give the path A* search will return using heuristic h_2?

Node	h_0	h_1	h_2
S	0	5	6
A	0	3	5
B	0	4	2
C	0	2	5
D	0	5	3
G	0	0	0

(c) (2pts): Which path will greedy best-first search return using h_1?

For questions **d** and **e**, consider the following variations of the **fringe** in the **A* tree search algorithm**. In all cases, *g* is the cumulative path cost of a node *n*, *h* is a lower bound on the shortest path to a goal state (e.g., Manhattan distance). Assume all costs are positive, and all heuristics *h* are admissible.

(i) Standard A*

(ii) A*, but we apply the goal test <u>before</u> enqueuing nodes, rather than after dequeuing

(iii) A*, but prioritize fringe by g(n) only (ignoring h(n))

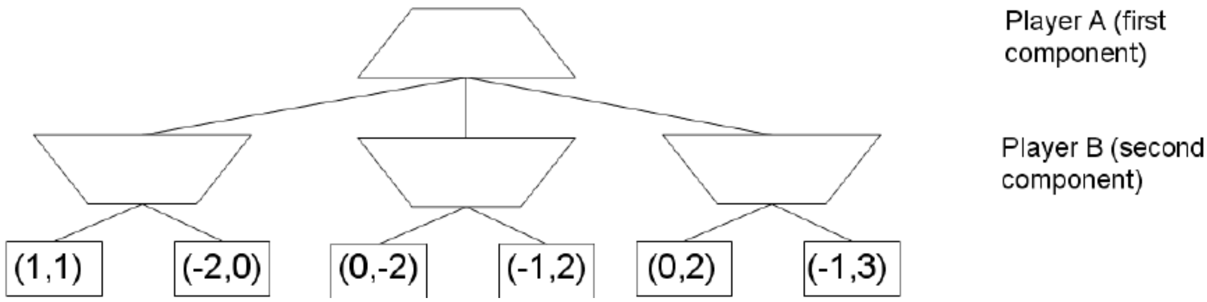
(iv) A*, but prioritize fringe by h(n) only (ignoring g(n))

(d) (2pts): Which of the above are *<u>complete</u>* (always finds goal if exists)?

(e) (2pts): Which of the above are *<u>optimal</u>* (always finds cheapest path to goal)?

Problem 2 (10 pts): Games

The standard Minimax algorithm calculates worst-case values in a zero-sum two player game, i.e. a game in which for all terminal states s, the utilities for players A (MAX) and B (MIN) obey $U_A(s)+U_B(s) = 0$. In the zero-sum case, we know that $U_A(s) = -U_B(s)$ and so we can think of player B as *minimizing* UA(s). In this problem, you will consider the non zero-sum generalization, in which the sum of the two players' utilities are not necessarily zero. Because player A's utility no longer determines player B's utility exactly, the leaf utilities are written as pairs (U_A, U_B), with the first and second component indicating the utility of that leaf to A and B respectively. In this generalized setting, A seeks to maximize U_A, the first component, and B seeks to **maximize** U_B, the second component (note: there is no "minimizer").



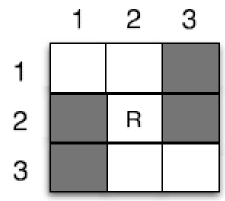
(a) (5pts) Propagate the terminal utility pairs up the tree using the appropriate generalization of the minimax algorithm on this game tree. Fill in the values (as pairs) at each of the internal node. Remember, each player maximizes their own utility only. Hint: just as in minimax, the utility pair for a node is the utility pair of one of its children.

(b) (2pts) Briefly explain why no alpha-beta style pruning is possible in the general non-zero sum case. Hint: think first about the case where $U_A(s) = U_B(s)$ for all nodes.

(c) (3 points) For Minimax, we know that the value v computed at the root (say for player A = MAX) is a worse-case value, in the sense that, if the opponent MIN doesn't act optimally, the actual outcome v_0 for MAX can only be better, never worse, than v. In the general non-zero sum setup, can we say that the value U_A computed at the root for player A is also a worst-case value (as in Minimax), or can A's outcome be even worse than the computed U_A if B plays suboptimally? Provide brief justification with an example to justify your answer.

YES or NO?
Justification:

Problem 3 (10pts): Inference and HMMs



Robot Localization Grid

Suppose you are a robot navigating a maze (see figure 1), where some of the cells are free and some are blocked. At each time step, you are occupying one of the free cells. You are equipped with sensors which give you noisy observations, (w_U, w_D, w_L, w_R) of the four cells adjacent to your current position (*up, down, left,* and *right* respectively). Each w_i is either *free* or *blocked*, and is accurate 80% of the time, independently of the other sensors or your current position. [1]

Imagine that you have experienced a motor malfunction that causes you to randomly move to one of the four adjacent cell with probability $\frac{1}{4}$.[2]

a) Suppose you start in the central cell in figure 1. One time step passes and you are now in a new, possibly different state and your sensors indicate (*free, blocked, blocked, blocked*). Which states have a non-zero probability of being your new position?

b) Give the posterior probability distribution over your new position. You **must** write down the formula before plugging in numbers to get full credit. Hint: don't forget to normalize!

Problem 4 (10 pts): Markov Decision Furby

You are designing an evil Furby doll (shown on right) to be *maximally annoying*. [*]That is, the goal is to maximize the total annoyance reward (*R*), imparted by the toy to the owner's family. The Furby has two states: ON and OFF. If the Furby is destroyed, it stays in OFF state, and there is no future reward. Otherwise, it stays in ON state, since this torture device's battery never runs out. It starts in the ON state.

This Furby has two actions:

***L*: L**oud annoying laugh, with immediate annoyance reward +10.

***Q*:** Quiet annoying laugh, with immediate annoyance reward +4.

The probability of the annoyed family destroying the Furby after action ***L*** is 0.5, and after action ***Q*** is 0.25.

(a) (3 pts). Draw the state transition diagram for MDF, clearly marking the **states**, **q-states**, **transition probabilities**, and the **rewards** associated with each action.

(b) (2pts) Write down the recurrence equation for the expected total reward for the policy **πL**="**always-L**", that is, always do action **L**. Assume discount γ is 1 (no decrease in "reward" over time)

(c) (2pts) Estimate the total rewards for the policy **πL** = "**always-L**". Assume discount γ is 1.

V(πL) =

(d) (3pts) What is the optimal policy for this MDF, and associated optimal value? Assume discount γ is 1. Show sufficient work to justify your answer.

FYI: Infinite geometric series: $a+ar+ar^2+ar^3+ar^4+ \dots = a/(1-r)$ (converges to this closed form for r<1)

[*] Aww it's so cute, you say? Not so much. See here: https://www.youtube.com/watch?v=B_0FWuJuKpk

Problem 5 (10 pts): Reincarnating Q-Learning Furby 2.0

Same states and actions as in Problem 3 above, but with the following changes:

- Furby 2.0 has been *upgraded* with an annoyance sensor, which measures how annoyed the family is, and returns a reward score ranging from 0 (no reward) to +10 (maximum annoyance)
- Furby 2.0 now has reincarnation capability (if destroyed, the Furby re-orders itself from Amazon, and resets itself to ON state). All the (Q-Values) are saved on AWS cloud storage, so training continues…
- The transition probabilities and action rewards are no longer known.

To maximize the total annoyance, the Furby 2.0 implements the Q-Learning algorithm.

(a) 8pts: Compute the Q-Values (that you drew in 3a), **after each training episode below**, provided as sequence of tuples (State, Action, NextState, Reward). Assume α=0.5, and γ =1. Show work.

Episode 1: (ON, Q, ON, +4), (ON, L, OFF, +10)

Q-Values:

E1 Q(on, Quiet) Q(on, Loud) || show work here on how you got the values on left
0 0 0 ||

Episode 2: (ON, Q, ON, +4), (ON, Q, OFF, +4)

Q-Values:

E1 Q(on, Quiet) Q(on, Loud) || show work here how you got the values on left
? ? ? ||

(b) 2 pts: What **policy** does the Furby learn after the 2 episodes of training above?

Problem 6 (20 pts): Neural Networks and Deep RL

(a) (4 pt) Circle true or false for each statement about a perceptron classifier in general. Assume weight vectors are initialized to 0s.

(i) (true or false) Can produce non-integer-valued weight vectors from integer-valued features

(ii) (true or false) Estimates a probability distribution over the training data

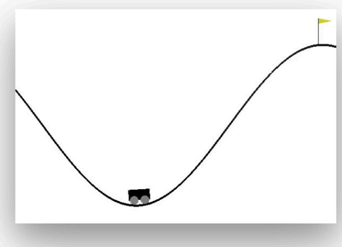
(iii) (true or false) Assumes features are conditionally independent given the class

(iv) (true or false) Perfectly classifies any training set eventually

(b) (6 pts) Construct by hand a simple feed-forward neural network, which computes the XOR function of two inputs *a* and *b*. Make sure to specify what sort of units you are using, and weights that would give the desired behavior. The XOR input/output table is provided -→

a	b	XOR
1	1	0
0	1	1
1	0	1
0	0	0

For the problems below, refer to the "mountain car" control problem.



Aim: learn how to drive the car up a big hill to the goal on right, where the car's engine is weak (requires building up enough speed first by going up the little hill on left, away from the goal. The car starts standing at bottom of hill (0,0)
Environment representation:
Observations: car's (position, velocity) (real numbers/scalars)
Actions (accelerator): a real number between (-1,1)
Rewards: -0.1 time penalty, only 1 reward of +100 iff reaches the goal.
Best score possible: about 90 (at least 100 time steps required to reach goal)

(d) (2 pts): Can exact Q-learning work here in finite time? If yes, describe how. If not, explain why not.

(e) (2 pts): Can approximate Q-learning work here in finite time? If yes, describe how. If not, explain why not.

(f) (4 pts): Draw a **simple** policy neural network that could be used to directly learn an effective policy for this problem. Make sure to specify exactly the input layer to match the observations, and output layer to match the action space.

(g) 2 pts: Let's use a **Policy Gradient** (PG) method to train your network. Write 1-2 lines of pseudocode just for the parameter update step, for you network above. Must specify loss w.r.t. to total episode reward. For simplicity, assume back-propagation is implemented and available.

End of Exam.