STUDENT NAME: ______________________________

STUDENT ID: ______________________________

MIDTERM EXAMINATION

Reinforcement Learning - Winter 2019

March 14, 2019

You are allowed one double-sided "cheat sheet". No laptops, calculators or cell phones are allowed.

Read all the questions before you start working. Please write your answer on the provided exam (you can use both sides of each sheet). If you have a question, raise your hand. Partial credit will be given for incomplete or partially correct answers. Please be sure to define any new notation you introduce.

There are 4 questions with multiple parts, totalling 100 points.

Good Luck!

1. [30 points] **Problem formulation**

   Consider a sandwich shop in a small town with a fixed population of N people. Customers arrive at times governed by an unknown probability distribution. Each customer can order a sandwich with a certain type of bread (chosen from 5 types) and filling (chosen from 4 types). Customers pay a given price for each sandwich. If a customer cannot get the desired sandwich, he or she will never come back to the store again. Ingredients need to be discarded 3 days after purchase. The store owner wants to figure out a policy for buying ingredients in such a way as to maximize long-term profit, and hopes to use reinforcement learning for this task.

   (a) [12 points] What is the state space and action space for this problem? What is the reward function?

   (b) [3 points] Would you use a discounted, undiscounted or average-return criterion? Justify your answer.

   (c) [5 points] Would you use dynamic programming or reinforcement learning for this problem? Justify your answer

   (d) [5 points] Between Monte Carlo and temporal-difference learning, which method would you prefer? Justify your answer.

   (e) [5 points] Is function approximation required to solve this problem? Justify your answer

2. [30 points] **Bellman equations and dynamic programming**

   Suppose we are in an MDP and the reward function has the structure $r(s, a) = r_1(s, a) + r_2(s, a)$. In this question, we investigate whether we could solve the problems defined by reward functions r_1 and r_2 independently and then somehow combine them in order to solve the problem defined by r. We are using the discounted setting

   (a) [10 points] Suppose you are given the action-value functions q_1^π and q_2^π corresponding to the action-value function of an arbitrary, fixed policy π under the two reward functions. Using the Bellman equation, explain if it is possible or not to combine these value functions in a simple manner to obtain q^{pi} corresponding to reward function r.

   (b) [10 points] Suppose you are given the optimal action-value functions q_1^* and q_2^*. Using the Bellman equation, explain if it is possible or not to combine these value functions in a simple manner to obtain q^* which optimizes reward function r

   (c) [10 points] Suppose you are given the optimal policies π_1^* and π_2^*. Explain if it is possible or not to combine these policies in a simple manner to obtain policy π^* which optimizes reward function r.

3. [30 points] **An alternative learning algorithm**

   In this question, we will consider a learning algorithm which attempts to learn a Q-function, but instead of using the usual Q-learning target, it uses as target a mixture of $(1-\epsilon)$ times the maximum Q-value, plus ϵ times the average action value at the next state.

   (a) [5 points] Draw the one-step backup diagram of the algorithm and write out its update rule

   (b) [5 points] Is this algorithm on-policy or off-policy? Justify your answer.

   (c) [5 points] Write the two-step version of the algorithm.

   (d) [5 points] Write the single-step function approximation version of the algorithm

   (e) [10 points] Do you expect this algorithm to be more or less stable than $Q - learning$ when used with function approximation? Would you expect it to be convergent with function approximation? Explain your answer.

4. [10 points] **Function approximation**

   Consider two function approximators, one which discretizes the state space into k bins, and one which takes each of those bins and splits it in half (hence producing a feature vector of size $2k$). Suppose we want to use the approximators to estimate the value function of a fixed policy, using on-policy data. Suppose you have a small number of samples n. Explain the impact that you expect to see on the two algorithm by using $TD(\lambda)$ with $\lambda > 0$ compared to doing $TD(0)$.