Last name: Huang First name: Yongjia SID#: 1461069

Collaborators:

CMPUT 366/609 Assignment 3: Dynamic Programming and Monte Carlo methods

Due: Tuesday Oct 10 by gradescope

Policy: Can be solved in groups (acknowledge collaborators) but must be written up individually

There are a total of 100 points available on this assignment, and 10 bonus points.

Question 1. Exercises from SB textbook. **[18 points total]**. This question has two parts corresponding to two exercises from the book.

Exercise 4.3 [9 points, 3 for each equation] (*policy evaluation equations for action-value functions*)

Exercise 4.5 [9 points] *(policy iteration for action values)*

4.3: 4. $q_\pi(s,a) = \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a]$ where $v_\pi(s') = \pi(s',a')\, q_\pi(s',a')$

$$= \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_\pi(s')\right]$$

$$q_{k+1}(s,a) = \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_k(s')\right].$$

4.5: Algorithm for computing q_π

Initialization

$\pi(s), P(s,a)$ arbitrarily for all $s \in S$. $a \in S$.

Repeat forever

choose $S_0 \in S$ and $A_0 \in A(S_0)$ such that pairs probability >0 (Exploring Starts)

for each pair: $q = q(s,a)$

$q(s,a) \leftarrow \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_\pi(s')\right]$

Append result to the list Result[]

for each s:

$\pi(s) \leftarrow \operatorname{argmax}_a q(s,a)$

Question 2. Programing exercise. There are two parts worth **82 points** total.

Part 1. Programming exercise. **[32 points]. Exercise 4.8** from Sutton and Barto second Ed. You are not are not expected to use the RL-Glue interface for this exercise. **But you must use python2!**

Please submit all your code and your plot of the value estimates, and the plot of the final policy.

Part 2. Programming exercise. **[50 points]**.

Implement On-policy Monte Carlo Control with Exploring Starts for action values (described in Section 5.3) on the Gambler's problem described in Chapter 4 (Example 4.3).

We will make two changes from the problem specification in the book. First we will set pr(heads) = 0.55. In addition we will **not allow** the **zero** bet action; the agent must always make a bet of one or greater.

You will use RL-Glue. We will provide you an environment program that implements the gambler's problem in the **assignment directory of the course folder**. Please take a look at the environment code, to better understand what is going on. We will also provide you with the experiment program to run the experiment. You can either use the plotting script provided or use the data generated by gambler_exp.py and plot it some other way. Look at gambler_exp.py to see how the data is stored. All you have to do is modify mc_agent.py correctly, to implement the Monte Carlo control method and then run the experiment and generate the graphs you need. The mc_agent.py is not complete—that's your job—it currently will not run.

Please use the code we have provided. **Do not modify** rl_glue.py, gambler_env.py. You will only need to modify mc_agent.py and possibly gambler_exp.py if you want to change the number of runs or how the data is stored.

We have also made small modifications to rl_glue.py to make it easier to use and understand. The code will be distributed via the course folder. **Don't use the rl_glue code from A1!**

You will plot the Monte Carlo agent's estimate of $v_*(s)$ during training. To do this: plot on the y-axis V(s) for each state on the x-axis. The Monte Carlo Exploring Starts algorithm iteratively updates an action-value function Q(s,a). You will compute V(s) using the following relation: $V(s) = \max_a Q(s, a)$ for all $s \in S$. You will plot V(s) after 100 episodes, 1000 episodes, and 8000 episodes. The plotted V should be produced by averaging over 10 independent runs. That is, your plot should contain 3 lines: V after 100 episodes averaged over 10 runs, V after 1000 episodes averaged over 10 runs, and V after 8000 episodes averaged over 10 runs. The experiment program you have been given does the averaging for you. You just have to make sure agent_message in mc_agent.py does the correct thing. You may experiment with more than 8000 episodes if you like. Our implementation takes 3 to 5 minutes to run 8000 episodes and 10 runs. Please perform at least 10 runs, but feel free to test more runs (typically leading to more clear results). Number of runs and averaging is done in the experiment program which is provided for you.

Hint: the initial policy used for Monte Carlo is important for getting good performance in this problem. Therefore we want to initialize the policy in a smart way. One example: initially pi(s) = min(s, 100-s) (i.e. bet all the needed capital to win). Other initializations might work better. Feel free to explore. Note: this is the initial policy. The agent's goal is to learn and change the policy over time.

Please submit your plot and **all** your code.

Bonus Question. [5 points]. Discuss and compare how the plot produced by Monte Carlo Control with Exploring Starts is similar and different from the plot produced by value iteration (in Question 2, part 1). Discuss why Dynamic Programming is suitable for the Gambler's problem, and why the Monte Carlo method with exploring starts might be less suitable for tasks like the Gambler's problem. Experiment with pr(heads) different than 0.55, to see what happens and report those results and submit those plots.

Bonus Question. Exercise 4.7 of SB textbook. **[5 points]** *(gambler's problem plots)*