CMPSCI 687 Homework 2

Due October 8, 2018, 11:55pm Eastern Time

Instructions: This homework assignment consists of a written portion and a programming portion. Collaboration is not allowed on any part of this assignment. Submissions must be typed (hand written and scanned submissions will not be accepted). You must use LaTeX. The assignment should be submitted on Moodle as a .zip (.gz, .tar.gz, etc.) file containing your answers in a .pdf file and a folder with your source code. Include with your source code instructions for how to run your code. You may not use any reinforcement learning or machine learning specific libraries in your code (you may use libraries like C++ Eigen and numpy though). If you are unsure whether you can use a library, ask on Piazza. If you submit by October 14, you will not lose any credit. The automated system will not accept assignments after 11:55pm on October 14.

Part One: Written (25 Points Total)

1. (5 Points) Prove that the following two definitions of the state-value function are equivalent:

$$v^\pi(s) \coloneqq \mathbf{E}\left[G_t | S_t = s, \pi\right] \tag{1}$$
$$v^\pi(s) \coloneqq \mathbf{E}\left[G | S_0 = s, \pi\right]. \tag{2}$$

   Ans:
   $v^\pi(s) \coloneqq \mathbf{E}\left[G_t | S_t = s, \pi\right] = \mathbf{E}[\sum_{k=0}^{\infty} \gamma^k R_{t+k} | S_t = s, \pi]$
   $v^\pi(s) \coloneqq \mathbf{E}\left[G_t | S_t = s, \pi\right] = \mathbf{E}[\sum_{t=0}^{\infty} \gamma^t R_t | S_0 = s, \pi]$ Check the sum of R in two equations, we find that the first terms in two summations has the same expected value because reward is stationary. Also the dynamic is stationary, which means the transition function is the same for same state and action. Therefore, we have the same transition function, the same policy, and the same reward, which makes the expected value of the two second term also the same, and so on the third, the forth....thus the two value functions are the same.

2. (5 Points) Prove that the following two definitions of the action-value function are equivalent:

$$q^\pi(s, a) \coloneqq \mathbf{E}\left[G_t | S_t = s, A_t = a, \pi\right] \tag{3}$$
$$q^\pi(s, a) \coloneqq \mathbf{E}\left[G | S_0 = s, A_0 = a, \pi\right]. \tag{4}$$

   Ans:

$$q^\pi(s,a) \coloneqq \mathbf{E}\left[G_t | S_t = s, A_t = a, \pi\right] \tag{5}$$

$$= \mathbf{E}\left[\sum_{s'} P(s,a,s')v^\pi(s') | S_t = s, A_t = a, \pi\right] \tag{6}$$

$$q^\pi(s,a) \coloneqq \mathbf{E}\left[G | S_0 = s, A_0 = a, \pi\right] \tag{7}$$

$$= \mathbf{E}\left[\sum_{s'} P(s,a,s')v^\pi(s') | S_0 = s, A_0 = a, \pi\right]. \tag{8}$$

again, the dynamic is stationary, so the transition function is the same, we proved that the value function is the same for different time steps, therefore the two q definition is also the same.

3. (5 Points) Let M and M' be two MDPs that are identical except for their initial state distributions. Prove that v^π is the same for both MDPs. You may write v^π_M and $v^\pi_{M'}$ to denote the state-value functions for π on M and M' respectively. Ans: even though the initial state distribution is the same, if we check the second definition in problem one, it assumes you start from the state s already, so different initial state distribution does not make a difference here. Ans since all the other dynamics are the same for the two MDP, the value functions are also the same.

4. (2 Points) Prove that multiplying all rewards (of a finite MDP with bounded rewards and $\gamma < 1$) by a positive scalar does not change which policies are optimal.
Ans: From the definition we know that for any state s, if we multiply all reward by k, $v^\pi_{new}(s) = \mathbf{E}[\sum_t k\gamma^t R_t|\pi] = k\mathbf{E}[\sum_t \gamma^t R_t|\pi] = kv^\pi_{old}(s)$.
Multiplying the constant will not change the order of values of the state, so the optimal policy will not change.

5. (2 Points) Prove that adding a positive constant to all rewards (of a finite MDP with bounded rewards and $\gamma < 1$) can change which policies are optimal.
Ans: Check the following MDP with two states s, s_∞ and $\gamma = 0.5$. we always start from s and have two actions to choose:
1.a_1 takes you to s_∞ with reward 3, so for the policy that always choose a_1, the value of s is 3;
2.a_2 takes you to s_∞ with reward 0, or takes you back to s with reward 1, both with 50% chance. The policy that aloways choose a_2 has the vaule of s 2;
clearly the first policy is optimal here. now if we add 31 to all the rewards, the fist policy will have value of s being, while the second policy has value

of s being 64. This changes the optimal policy.

6. (1 Point) Your boss asked you to estimate the state-value function associated with a known policy, π, for a specific MDP. You misheard and instead estimated the action-value function. This estimation was very expensive, and so you do not want to do it again. Explain how you could easily retrieve the value of any state given what you have already computed.
   Ans: Just follow this equation : $v^\pi(s) = \sum_a \pi(s,a) q^\pi(s,a)$.

7. (5) Consider a finite MDP with bounded rewards, where all rewards are negative. That is, $R_t < 0$ always. Let $\gamma = 1$. The MDP is finite horizon, with horizon L, and also has a deterministic transition function and initial state distribution (rewards may be stochastic). Let $H_\infty = (S_0, A_0, R_0, S_1, A_1, R_1, \ldots, S_{L-1}, A_{L-1}, R_{L-1})$ be any history that can be generated by a deterministic policy, π. Prove that the sequence $v^\pi(S_0), v^\pi(S_1), \ldots, v^\pi(S_{L-1})$ is strictly increasing.
   Ans: we can prove this statement by showing that $v^\pi(S_{t+1}) - v^\pi(S_t) > 0$ for t from 0 to $L-2$.
   $v^\pi(S_{t+1}) - v^\pi(S_t) = \mathbf{E}[G_{t+1} - G_t | H_\infty] = \mathbf{E}[-R_{t+1} | H_\infty] > 0$

Part Two: Programming (55 Points Total)

Implement the 687-Gridworld domain described in class and in the class notes, and the cart-pole domain using the (frictionless) dynamics described by **?**, Equations 23 and 24. When implementing Cart-Pole, the state should include the position of the cart, velocity of the cart, angle of the pole, and angular velocity of the pole. You **must** use a forward Euler approximation of the dynamics. If the cart hits the boundary of the track, terminate the episode. **You may not use existing RL code for this problem—you must implement the agent and environment entirely on your own and from scratch.** Four questions ask for a plot. You may report two plots: one for cart-pole and one for 687-Gridworld, where each of these two plots has a curve corresponding to the cross-entropy method and a curve corresponding to first-choice hill-climbing.

Use the following values for the Cart-Pole constants:

- Fail angle = $\pi/2$. (If it exceeds this value or its negative, the episode ends in failure.)
- Motor Force F = 10.0 (force on cart in Newtons).
- Gravitational constant g is 9.8.
- Cart mass = 1.0.
- Pole mass = 0.1.
- Pole half-length $l = 0.5$.

- $\Delta t = 0.02$seconds (time step).
- Max time before end of episode = 20seconds + $10\Delta t$ = 20.2seconds.

Problems:

- (10 Points) Implement the cross-entropy method as described in the class notes and apply it to the 687-Gridworld. Use a tabular softmax policy. Search the space of hyperparameters for hyperparameters that work well. Report how you searched the hyperparameters, what hyperparameters you found worked best, and present a learning curve plot using these hyperparameters, as described in Figure **??**. This plot may be over any number of episodes, but should show convergence to a nearly optimal policy. The plot should average over at least 500 trials and should include standard error (or standard deviation) error bars (say which error bar variant you used).
- (10 Points) Repeat the previous question, but using the cross-entropy method on the cart-pole domain. Notice that the state is not discrete, and so you cannot directly apply a tabular softmax policy. It is up to you to create a representation for the policy for this problem. Report the same quantities, as well as how you parameterized the policy.
- (10 Points) Repeat the previous question, but using first-choice hill-climbing on the 687-Gridworld domain. Report the same quantities.
- (10 Points) Repeat the previous question, but using first-choice hill-climbing (as described earlier in these notes) on the cart-pole domain. Report the same quantities and how the policy was parameterized.
- (5 Points) Reflect on this problem. Was it easier or harder than you expected to get these methods working? In the previous assignment you hypothesized how long it would take an agent to solve the 687-Gridworld problem. Did it take more or fewer episodes than you expected? Why do you think this happened?
  Cartpole Policy: I use a linear equation to represent the policy. Treat state s as a vector, and with W a 4 by 2 matrix and b a length 2 vector, $score = sW + b$. Apply softmax to the score, we get the probability of taking two actions which we can then sample from.
  Hyper parameter search: for both BBO algorithm, I use random search over the space and evaluate the hyper parameter by running several trails and check the performance.
  Cartpole vs grid world: The grid world environment takes more time to evaluate a policy and search hyper parameters. THe first thing is, it has a way bigger parameter space than cartpole, which makes the search harder. Second, even though both policies are stochastic, grid world also has a stochastic transition function, but cartpole is not. Therefore, more run is needed for grid world policy evaluation. More randomness also means

slower execution in Python. I end the grid world episode after around 50 time steps because the discounting will eliminate any change on the return, but it still takes forever for me to run.
CE vs FCHC: CE has way more hyper parameters than FCHC, which makes the searching way harder. It also takes forever to run if you have large N and K at the same time. However, it is important to have large N,K for grid world.
I am using standatd deviation in the plots. I believe it is harder to make the algorithm work. The theory looks good, but it usually does not work well unless you get really good hyper parameters, which is like magic number. the randomness in grid world also hurts the performance, even though the problem seems so easy when you reason on the optiomal policy. BBO algorithm are seemed to be less efficient than I first thought, especially when I use my crappy laptop and got over heat. I underestimate the computation needed.
Grid-CE does converge. Grid-FCHC has really high std that covers the return in the graph, guess I need better hyper parameters. cartpole-FCHC looks good, except the dots for std are connected in a wrong way. I should save the data on disk so that I can correct the plot, since there is bo time for me to re-run the program. I only get good hyper parameter for cartpole-CE. It converges with fewer episodes, but I can only run a few trails before the deadline. I replace the std with green color so it does not hide the return curve. The std for this is pretty high, but the program shows it converges for most of the time.
the hyper parameters and plots are saved in the root directory of the source code.