CS 6364 Homework 6

October 26, 2021

Deadline for the first submission: **Nov-08-2021**.
All assignments **MUST** have your name, student ID, course name/number at the beginning of your documents.
Your homework **MUST** be submitted via Blackboard with file format and name convention as follows:

HW#_Name_writeup.**pdf** (for writing part)
HW#_Name_code.**zip** (for coding part)

If you have any questions, please contact me.

For the following questions, if you need a GPU to run, Google provide a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. Here is the link: https://colab.research.google.com/notebooks/welcome.ipynb?hl=enscrollTo=5fCEDCU_qrC0

A Markov decision Process is defined by $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{P}, \gamma)$, where $\mathcal{S}$ denotes the set of possible states, $\mathcal{A}$ denotes the set of possible actions, $\mathcal{R}$ denotes distribution of reward given (state, action) pair, $\mathbb{P}$ denotes transition probability, and γ is a discount factor. In this homework, given a simple 5×5 grid game (see below), the upper left (i.e. state 1-1) and lower right (i.e. state 5-5) corners are terminal states. Therefore, there are $5 \times 5 = 25$ states (i.e. $|S| = 25$) and each is denoted as $s(i, j)$ where i and j represent i-th row and j-th column, respectively. Four possible actions are $\{right, left, up, down\}$. We set a negative reward $r = -5$ for each transition , a discount factor $\gamma = 0.7$, and the probability of an initial state $p(s_0)$ equals $\frac{1}{25}$. Our goal is to reach one of the terminal states (smiling states) in least number of actions. In other words, it aims to find the optimized policy π^* that maximized cumulative discounted reward. The initial Q table can be randomly defined by you.

Q1 [Value iteration algorithm] Using the Bellman's equation, implement value iteration algorithm to iteratively update the Q table until convergence.

Q2 [Deep Q-learning] This question functions the same as question 1. The only difference is to replace with a simple four-layer (i.e. three hidden layers and one output layer) fully-connected deep neural network to approximate the Q table. The three hidden layers contain 32, 8, and 16 neurons respectively and all applied ReLU activation functions. The output function is linear.

Q3 [Deep Q-learning with experience replay, bonus question] This question functions the same as question 2, except that you need to implement the Deep Q-learning with experience replay. Note that if you finish this bonus question, the points that you obtained in this HW will be doubled.

Both two questions are programming assignments. Please use Pytorch for the implementation of the second one. Please submit your code along with initial and finial Q-tables for both questions.