Assignment 1

Reinforcement Learning

CISC 453/474, Fall, 2019

Due Tuesday, October 22, 2019 before Midnight

Two-Player Matrix Games

In class we briefly discussed the "*Prisoners' Dilemma*", a game involving two criminals that need to decide whether they will cooperate with the police and defect on the accomplice or whether they will cooperate with the accomplice and lie to the police. The rewards for this game can be represented in matrices: R_1 for player 1, and R_2 for player 2. One example of reward matrix for this game is

$$R_1 = \begin{bmatrix} 5 & 0 \\ 10 & 1 \end{bmatrix}$$

Where the first row represents cooperation with the accomplice and lying to the police, whereas the second row represents defection and confession to the police. The rewards are from player 1's perspective. Therefore, if both players cooperate (element $r_{1,1}$), both go to jail for a short period and have a large reward. If player 1 cooperates and player 2 does not, player 1 goes to jail for several years and has a very small reward (element $r_{1,2}$). If player 1 confesses and player 2 cooperates (element $r_{1,2}$), player 1 has a very large reward, i.e., it does not go to jail at all. Finally, if player 1 and 2 confess (element $r_{2,2}$), both of them have a small reward, i.e., they go to jail for a couple of years. It should be clear that the reward matrix for player 2 is

$$R_2 = R_1^T = \begin{bmatrix} 5 & 10 \\ 0 & 1 \end{bmatrix}$$

It turns out that this same approach can be used for several other games. One example is the matching pennies game. In this game the two players hold a penny each. They independently show one side of the penny (actions are "*head*" or "*tail*"). If both pennies show the same figure (two *heads* or two *tails*), player 1 wins the penny and has a reward of 1. If the pennies show different figures, player 2 wins the penny and player 1 has a reward of -1. The reward matrix for player 1 is

$$R_1 = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

Then, player 2's reward matrix is $R_2 = -R_1$.

Finally, another game is *Rock-Paper-Scissors*. The idea is to display your hand as either a rock, scissors, or paper. Then, paper *covers* rock, rock *breaks* scissors, and scissors *cuts* paper. If both players display the same thing, it is a tie. The reward matrix for player 1 is:

$$R_1 = \begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{bmatrix}$$

Again, player 2's reward matrix is $R_2 = -R_1$.

You will implement an algorithm to iteratively calculate the optimal policies and the value of the game for each player in all three games.

Algorithm

Programming Language Requirement: Python 3 or MATLAB

(For all simulations, the initial policies are:

- Prisoner's Dilemma: $p^1(0) = p^2(0) = [0.5 \quad 0.5]^T$
- Matching Pennies: $p^1(0) = p^2(0) = [0.2 \quad 0.8]^T$
- Rock-Paper-Scissors: $p^1(0) = p^2(0) = [0.6 \quad 0.2 \quad 0.2]^T$

Also, $\alpha = 0.001$. Run the simulations for at least 50,000 steps.)

The algorithm is a variation of the ones we saw in class. It is an every-visit update algorithm that does policy iteration. Observe that this is not an associative problem, therefore, states do not need to be represented. The update rules for each player j are:

$$p_c^j(k+1) = p_c^j(k) + \alpha r^j(k)\left[1 - p_c^j(k)\right], \qquad if\ action\ c\ is\ taken\ at\ time\ t$$

$$p_o^j(k+1) = p_o^j(k) - \alpha r^j(k) p_o^j(k), \qquad for\ all\ other\ actions\ o \neq c$$

This algorithm works directly on the policies. Implement the algorithm and for each one of the three games described above, calculate:

a) The policies for each player. Show graphs in your report. (10 pts)
b) Are the policies calculated optimal? Demonstrate your answer. (10 pts)

Modify the algorithm so another term is added:

$$p_c^j(k+1) = p_c^j(k) + \alpha r^j(k)\left[1 - p_c^j(k)\right] + \alpha\left\{\mathbb{E}\left[p_c^j(k)\right] - p_c^j(k)\right\}, \qquad if\ action\ c\ is\ taken\ at\ time\ t$$

$$p_o^j(k+1) = p_o^j(k) - \alpha r^j(k) p_o^j(k) + \alpha\left\{\mathbb{E}\left[p_o^j(k)\right] - p_o^j(k)\right\}, \qquad for\ all\ other\ actions\ o \neq c$$

Make sure of calculating $\bar{p}_i^j = \mathbb{E}\left[p_i^j(k)\right]$ iteratively (**hint**: you can use the same learning rate α to calculate the mean). After you implement the algorithm, calculate:

c) The policies for each player. Show graphs in your report. (10 pts)
d) Are the policies calculated optimal? Demonstrate your answer. (10 pts)
e) What is the value of the game? (5 pts)
f) Why do you think the results are different? (5 pts)

Grading

The assignment is worth 100 points divided as follows:

1. The answers above are worth 50 points. Observe that the answers need to be supported by your code. If the code is not provided, or if it is incorrect, you will not receive marks for answering the questions.
2. The quality of your code is worth 20 points. Even if your code is correct, you may not receive full marks if the quality of the code is low.
3. The remaining 30 points are for your report (explanation and quality of answers) and your participation in your group. If you do not participate in the solution of the assignment, you will not receive these points.