Project 3

Nicolas Cofre
nicolas.cofre@gatech.edu

Git hash of last commit
2ac6b1ead28c52fd8ec67a911eb2d65be
be0e00b

***Abstract*—This project consists in the replication of the results for the soccer grid game by Greenwald et al (2003) in their Correlated Q-learning paper, hereafter "the paper". Their correlated Q-learning is compared with Friend and Foe Q learning by Littman and ordinary Q-learning.**

I. The Game and the Environment

The game consists of a grid of 2x4, with the leftmost column representing the goal for player A and the rightmost one the goal for player B. If a player scores in the corresponding goal it gets a reward of 100 and the opponent -100, if it scores an own goal it would get -100 as reward and the opponent 100.

I have used the structure of Gym to create the environment and defined the states in the grid with the following ids:

0	1	2	3
4	5	6	7

Figure 1: The grid IDs.

Then, the goal for player A (player 0 in the code) is defined by states 0 and 4 and the goal for player B (player 1 in the code) by the states 3 and 7. The state is compose by the position ID for each player and the ID of the player with the ball, so for instance the initial position in the paper is (2,1,1) using my convention:

0	B	A	3
4	5	6	7

Figure 2: The initial state.

I have defined the action number and the movements as follows:

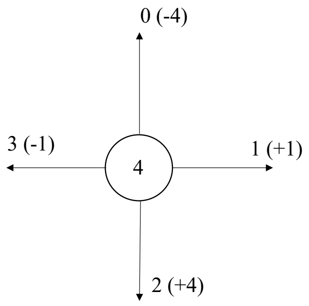
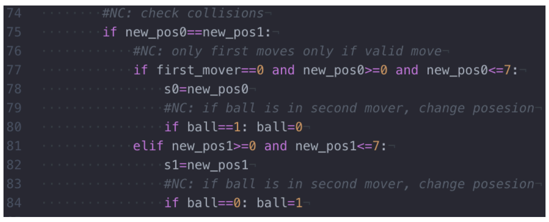


Figure 3: The action space convention.

This figure explains the effect on the coded environment and the convention used for the actions. For instance, the action N is coded as the action ID 0 and the effect on the environment is to decrease the position by 4, so if a player is in 4 and moves north, the new position would be 0. Note that a valid position and therefore movement is only one that move to a position ID in the grid between 0 and 7. If the movement is invalid, the player does not move.

The initial position of the players is as shown in the paper example, . The goals are invalid initial states, thus at initialization only the four squares in the midfield are possible in order to get rid of trivial initial states in which the game starts with a goal or own goal.

Following the collission rule in the paper, if the actions make the players collide, the ball changes possession if the player with the ball moves second. The order of the actions is random, with a uniform distribution among the two players. I have explicitely coded this with no further assumptions about the collisions, as it can be seen in the environment code below:



```
        #NC: check collisions
        if new_pos0==new_pos1:
            #NC: only first moves only if valid move
            if first_mover==0 and new_pos0>=0 and new_pos0<=7:
                s0=new_pos0
                #NC: if ball is in second mover, change posesion
                if ball==1: ball=0
            elif new_pos1>=0 and new_pos1<=7:
                s1=new_pos1
                #NC: if ball is in second mover, change posesion
                if ball==0: ball=1
```

Figure 4: Collisions case.

II. The Experiment

For each of the following methods, the experiment consists in training using the given algorithm and tracking the updates in the state (2,1,1) or state s, illustrated in my figure 2 and action 2,4 which corresponds to player A going S and player B not moving. After each update we store the difference in the value as

$$Error_t \coloneqq |Q_{t-1}(s,a) - Q_t(s,a)|$$

Note that in the case of q learning, a corresponds only to 2, the action of player A, without taking into consideration the action of player B.

III. Q Learning

The paper states that the intuition about the lack of convergence in the q-learner is clear. Continuing with the soccer examples, my intuition about the lack of convergence is explained with a new example below.

Let´s consider a penalty kick game, if you think like a q learner, shoot right and score. Then you might shoot right next time. If your opponent q learner goalkeeper sees he can save diving right, then you would see that your reward from shooting right is not as good as before, and that shooting left is better, as your opponent as learned that it should dive right. So now, you are shooting left, then your opponent q learner would learn that it should dive left, and then we repeat this forever. Thus, the value for a given action would never converge in this case, shooting right was good till the moment the goalkeeper q learner knew it should dive right, then the good action was left.

In this algorithm we try to find the optimum policy and values for each state using the following rule

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(R + \gamma \max_{a'} Q(s',a'))$$

In all the algorithms used here, we have scaled the reward R by (1- γ) as done in the paper. The paper states on policy Q learning and therefore I have actually used SARSA:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(R + \gamma\, \mathrm{Q}(s',\mathrm{a}'))$$

Where a' is drawn from the same epsilon-greedy policy used for the behavior. The initial value for the tables is not specified and I have used 0, as for the decay rate I have used exp(ln(0.001)/1e6) in order to reduce it to 0.001 after 1 million iterations as stated in the paper. The initial alpha is not specified and the same for the exploration rate, I have assumed 1 for both. $\gamma = 0.9$ as stated in the paper for the grid games. The results for this experiment are the following,

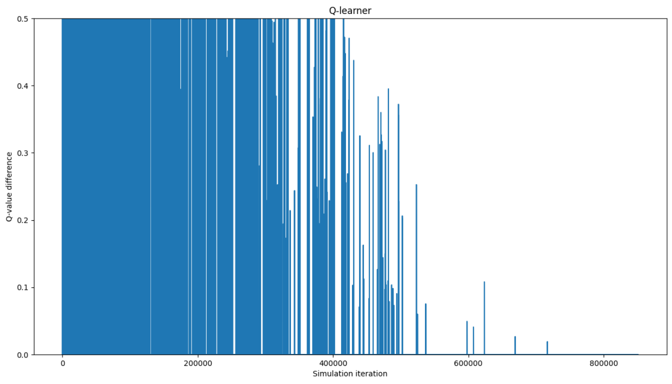


Figure 5: Q learning result.

The results seem similar to what is in the paper, only decreasing due to the decrease of α in contrast to the fast decrease in the case of Friend Q learning shown later. The difference can be due to a higher decrease rate of α in my case, which I make 0.001 after 1 million iteration.

IV. Friend Q Learning

In this case, the algorithm is basically the same as in Q learning, but the maximization is done over the joint space of actions, this means that the players have a common maximization objective and try to coordinate in the best action for this common objective. This is clearly not suited for the soccer game where a good outcome for one player is a bad outcome for the other. Friend Q learning is useful for a cooperative game, which is a game with a common goal. This is an adversarial game in which one player wins and the other loses and thus each player is not trying to maximize the utility of the opponent. If player A considers the opponent a friend, then the maximization that occurs is the following:

$$\max_{a_A \in A_A, a_B \in A_B} Q_A(s, a_A, a_B)$$

Where A_A and A_B are the action space for player A and B, respectively. This code is basically the same as the one from the previous section but optimizing over 25 actions (5x5) instead of only 5, this is optimizing over the joint set of actions. I have used the same parameters as in Q learning. The results of the experiment for the method is the following,

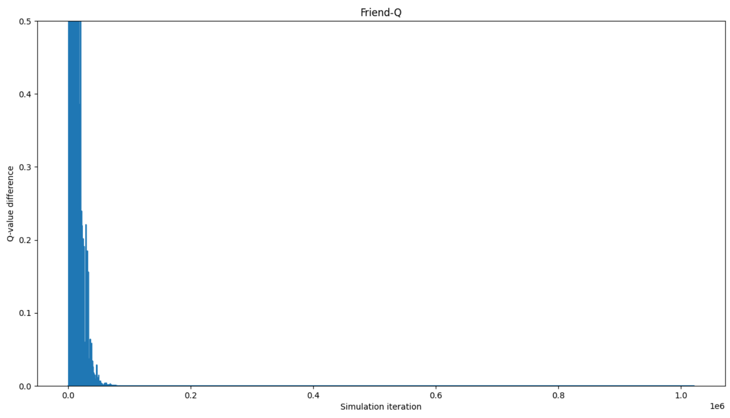


Figure 6: Friend Q learning result.

This looks very similar to what the paper shows, the difference seems to be that I have many more iterations without any update, and then the error is just 0 in many points. I have the intuition that the paper is plotting conditioning on the state s being updated, if I had more time, I would explore that option. But on the other hand, such a modification of the data does not seem subtle enough not to be stated explicitly in the paper.

V. Foe Equilibrium

In this equilibrium, a player tries to maximize his utility assuming the opponent will try to minimize it. Thus, player A is trying to solve the following problem:

$$\max_{p} \min_{a_B \in A_B} \sum_{a_A \in A_a} p(a_A) Q_A(s, a_A, a_B)$$

s.t.

$$\sum_{a_A \in A_a} p(a_A) = 1$$

$$p(a_A) \geq 0$$

Player A tries to find a probability distribution over his actions in order to maximize his utility assuming the worse response in pure strategies of his opponent. This can be solved using linear programming as in the homework 6 for the Nash equilibrium of the two player, symmetric, zero-sum game.

The problem can be stated as a linear programming problem as it follows:

$$\min_{x} -c^T x$$

s.t.

$$Ax \leq b$$

Where similarly as in the HW6 we have (2 more actions, that is the difference),

$c^T = [1,0,0,0,0,0]$

A=

1	One shoot game matrix with opposite sign Q(s,:,:)*(-1)				
1					
1					
1					
1					
0	1	1	1	1	1
0	-1	-1	-1	-1	-1
0	-1	0	0	0	0
0	0	-1	0	0	0
0	0	0	-1	0	0
0	0	0	0	-1	0
0	0	0	0	0	-1

$$b^T = [0,0,0,0,0,1,-1,0,0,0,0,0]$$

Q(s,:,:) corresponds to the matrix of the one shoot game, q values in the state s which is a 5x5 matrix with the actions for player A in the rows and player B in columns. We use this one shoot game to compute the values for the state, which are used for the update of the Q value itself. Note that V in this case corresponds to the first value of our vector x and that the value function has 2 action indices.

In summary the update rule is

$$Q(s,a,o) \leftarrow (1-\alpha)Q(s,a,o) + \alpha(R + \gamma V(s'))$$

With V the optimum value from the LP stated above. I have used a totally exploratory behavior policy i.e. taking random actions, and the initial value for the V table is 0. The rest of the parameters are the same as in the previous cases. The result for this experiment is the following,

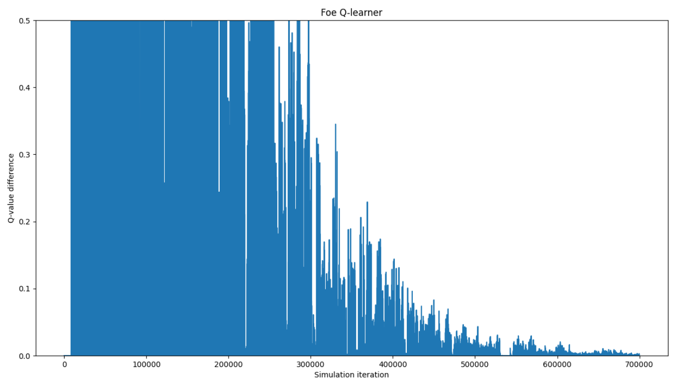


Figure 7: Foe Q learning result.

VI. Correlated Equilibrium

The correlated equilibrium takes into account the joint distribution over actions. I have computed the correlated equilibrium according to the following linear problem:

$$\max_{p} \sum_{a\in A} p(a) \sum_{i\in N} u_i(a)$$

Subject to:

$$\sum_{a\in A} p(a) = 1$$

$$p(a_A) \geq 0$$

$$\sum_{a\in A} p(a)u_i(a) \geq \sum_{a\in A} p(a)u_i(a'_i, a_{-i})$$

The last inequality is called rationality constraint and means that following the correlated signal is better than not following it. In a symmetric game like this one, it only suffices to solve the problem for one agent, i.e. satisfying the rationality constraints for one of the players.

$\sum_{i\in N} u_i(a) = 0$ in this zero-sum game. Therefore, any probability distribution satisfying the constraints is a correlated equilibrium. Note that this means that we are maximizing the total expected utility of the players as a whole. Note also that the rationality constraint is exactly the minimax problem as the game is symmetric,

$$\sum_{a\in A} p(a)u_i(a) \geq \sum_{a\in A} p(a)u_i(a'_i, a_{-i})$$

This can be seen as a minimax from the perspective of the opponent, as the game is symmetric using the fact that $u_{-i} = -u_i$.

$$\sum_{a\in A} p(a)u_i(a) \leq \sum_{a\in A} p(a)u_i(a_i, a'_{-i})$$

Thus, we end up with the same problem.

VII. Conclusion

The experiments seems to be fairly replicated in behaviour, although figures are not exactly as in the paper the conclusions are the same. Q-learning not converging except by the forced decrease in learning by the decreasing α. Friend Q learning converging very fast after around 100000 iterations. The minimax case also converging as in the paper. This results are important as they show the lack of convergence of ordinary q learning methods for multiagent non-cooperative or adversarial games.