CS 7624 Project 3 – Multiagent Q-Learning

Arjun Chintapalli[1]
1 Department of Computer Science, Georgia Institute of Technology, arjun.ch@gatech.edu
Github Repo: https://github.gatech.edu/achintapalli3/Project_3

ABSTRACT

This paper documents the results of implementing multi-agent Q-learning algorithms including Correlated-Q (CE-Q), Friend-Q, Foe-Q, and a conventional Q-learner to solve the Grid Soccer Game as described in Greenwald and Hall's paper "Correlated-Q Learning" (2003)[1]. The resultant policies these algorithms respectively take to solve this simple zero-sum stochastic game can be used to understand the convergence properties of the different algorithms.
Key Words: CS 7642, Project 3, Correlated-Q, Foe-Q, Friend-Q

1. INTRODUCTION TO SOCCER GAME

Soccer Game is a simple stochastic game involving two players on 4 x 2 grid that can be used to evaluate the policies that different learners take as well as their convergence. The simple setup of this game is setup as below, where:

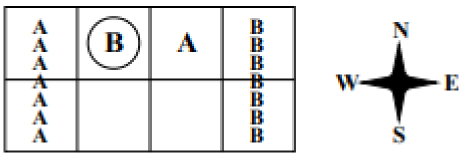

Figure 1: Soccer Game Setup

Player B starts with the Ball and aims to take the ball to the goal on the right side to get a +100 reward and give Player A -100 reward, but if he scores an own goal by heading West he receives -100 and gives Player A +100. Player A's intentions are similarly the opposite, aiming to take the ball and head West. The two player's actions are executed in random order. If both the players head to the same spot and the player with the ball moves second, then the ball changes ownership.

2. INTRODUCTION TO MULTIAGENT Q-LEARNING

Multiagent Q-Learning problems present a unique challenge that can be solved by various algorithms such as Nash-Q, CE-Q, Foe-Q, Friend-Q or a basic Q-Learner. Each of these algorithms seek to learn stationary policies for stochastic games, each with their own convergence criteria and stability conditions. Thus, the optimal algorithm to use for a game should be based on the game conditions as well as the algorithm requirements.

The Soccer Game is a stochastic-zero sum game without deterministic optimal policies. Thus, Q-Learning should fail to find an equilibria policy. However, for this game there do exist Nash equilibria where all Nash equilibria, where no player has an incentive to change his own strategy, are adversarial equilibria, meaning that in addition to the Nash criteria no player is hurt by a change in the strategy of another player. With this game criteria, only Nash-Q, CE-Q and Foe-Q should be capable of finding equilibria policies that include adversarial equilibria policies. Although Friend-Q can converge to a cooperative equilibria policy, this policy would be nonsensical for this game. This hypothesis is further validated in each of the experiments.

Greenwald and Hall describe how their different implemented learners took different policies for the same starting position. Q-Learning was found to not converge; meaning that Q-values for state action tuples kept changing with training and did not stabilize to a constant value. The Friend-Q algorithm was found to converge to a deterministic policy where each player assumes the other would score a goal for them. CE-Q and Foe-Q Learning algorithms converge to non-deterministic polices of randomly either sticking or moving South.

3. EXPERIMENT 1: Q-LEARNING

An on-policy epsilon-greedy Q-Learning algorithm was implemented to play both the players. Each of the players had their own Q table representing the *state x action* space, where there are 112 possible states ([Player A | B has ball] * [Player A possible state] * [Player B possible state] or 2*8*7 = 112 possible states) and 5 actions (North, South, East, West, Stick). Thus, the players took random actions if a randomly generated number was below epsilon, otherwise players took the action that maximized the Q value at that state.

Following each action the Q matrix was updated as follows:

$$Q^{new}(s_t, a_t) = (1-\alpha) \cdot Q(s_t, a_t) + \alpha \cdot \left((1-\gamma) \cdot r_t + \gamma \cdot \max_{a} Q(s_{t+1}, a)\right) \qquad (1)$$

This implemented Q-learner was then tested at the starting state to determine what policy this learner will decide on by plotting the difference in Q values [calculated as $|Q^{t+1}(s, a) - Q^{t}(s, a)|$] at the starting state for Player A taking action South.

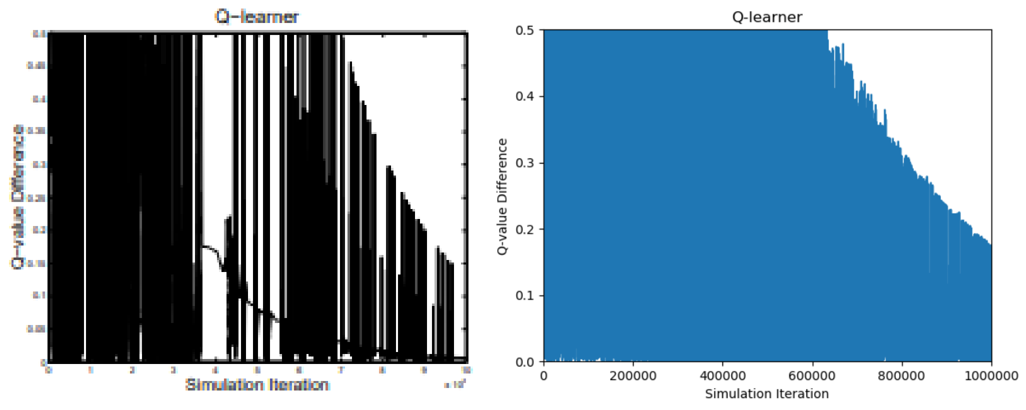

Figure 2: Q-Learning Results (Q-Error vs Time) [Greenwald on Left and Experimental on Right]

The above results indicate that the implemented Q-learner had the same non-convergent behavior as was described in [1]. The reason for this behavior is that this Q-learner fails to account for the other agent's actions\learning and so at each step each Q-learner keeps learning the changes in the other learner's policies as opposed to settling on a stationary policy. If the two players were able to coordinate their behavior in the Soccer Game, then the Q-learner would have been able to equilibrium policies. The only reason that the Q-differences decrease with time is that alpha is decayed at each time step. Thus, the magnitude of the Q-update decreases with each iteration. Although this learner is not optimal for this game due to its divergent behavior, it had by far the fastest run time, taking only ~30 seconds to run.

4. EXPERIMENT 2: FRIEND-Q

An off-policy Friend-Q algorithm was implemented such that both the players respectively assumed that the other player would help him maximize rewards. Thus, both the players took random actions and only a single Q matrix was maintained. This Q matrix was a representation of the 112 possible states, Player A's possible actions and Player B's possible actions, essentially a *112 x 5 x 5* matrix.

The Q table update had the same equation as for Q learner [Eq. 1] but the max Q was taken to be the maximum Q value of the joint action space of Player A and B at the new state. Thus, Player A assumes that Player B will take the action to maximize Player A's reward and takes the corresponding action.

$$Q^{new}(s_t, a_t^1, a_t^2) = (1-\alpha) \cdot Q(s_t, a_t^1, a_t^2) + \alpha \cdot \left((1-\gamma) \cdot r_t + \gamma \cdot \max_{a_1 \in A_1, a_2 \in A_2} Q(s_{t+1}, a_{t+1}^1, a_{t+1}^2)\right) \quad (2)$$

After implementing this algorithm, difference between Q values for the initial state and Player A heading South and Player B sticking were plotted similarly to [1].

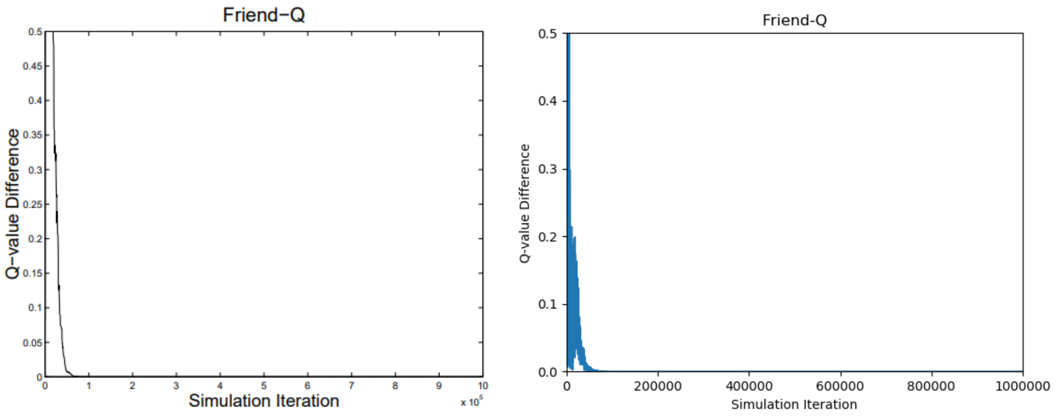

Figure 3: Friend-Q Results (Q-Error vs Time) [Greenwald on Left and Experimental on Right]

The Friend-Q algorithm converges very quickly to a deterministic policy as described in the paper. An inspection of the resulting Q matrix reveals that Player A learns to assume that Player B will move West and score an own-goal giving Player A +100 reward, and thus Q values for Player A's action are the same assuming Player B heads West. Clearly, Friend-Q is not well optimized for adversarial zero-sum games, since both the players are competing against each other. Even in the case of cooperative games, Greenwald [1] describes that Friend-Q performs poorly because the learners can't coordinate actions and run into each other negating rewards in the long run.

5. EXPERIMENT 3: FOE-Q

An off-policy Foe-Q was implemented such that both the players respectively assumed that the other player was a foe. Like Friend-Q each player maintains a *112 x 5 x 5* matrix Q matrix representing the possible states, their actions and their opponent's actions. In contrast to Friend-Q, Foe-Q players seek to maximize the Q values amongst their own actions and minimize the Q-values of the opposing the player. Thus, the Q update is as follows:

$$Q_1^{new}(s_t, a_t^1, a_t^2) = (1-\alpha) \cdot Q_1(s_t, a_t^1, a_t^2) + \alpha \cdot \left((1-\gamma) \cdot r_t + \gamma \cdot \max_{\pi \in \Pi(\mathrm{A}_1)} \min_{\mathrm{a}_2 \in \mathrm{A}_2} \sum_{\mathrm{a}_1 \in \mathrm{A}_1} \pi(\mathrm{a}_1) Q_1(s_{t+1}, a_{t+1}^1, a_{t+1}^2) \right) \quad (3)$$

This minmax problem then can be derived as linear program for this Soccer Game world that involves 11 constraints. After implementing this algorithm, difference between Q values for the initial state and Player A heading South and Player B sticking were plotted similarly to [1].

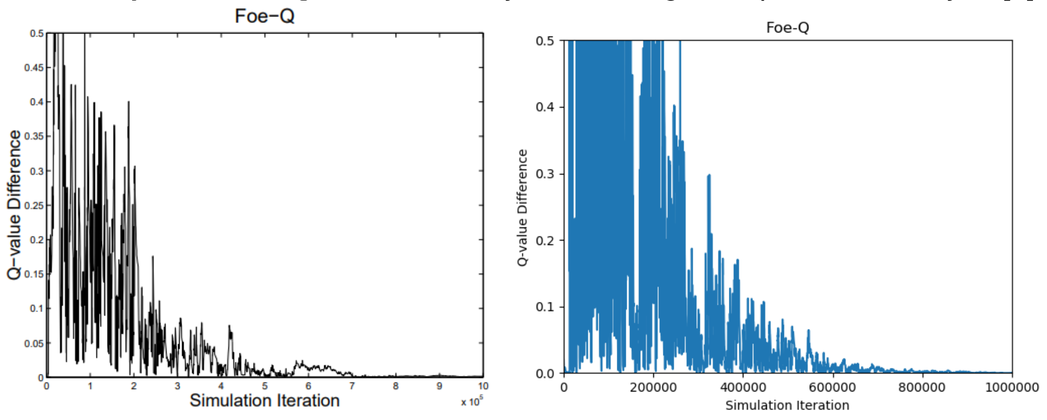

Figure 4: Foe-Q Results (Q-Error vs Time) [Greenwald on Left and Experimental on Right]

The Foe-Q results closely match the published results in [1] and could better match them if the hyperparameters such as α and the decay schedule were further tuned. Foe-Q converges to a stationary policy and inspecting the Q matrix reveals that for this state Player A has high Q-values for both sticking and heading South, like the published results. These results indicate that the implemented algorithm has found the adversarial equilibria for this game. These results though can’t be generalized to other games as Foe-Q will fail to converge to an equilibria policy if the game is not a two-player constant sum game, as then algorithms such as CE-Q would be optimal.

6. EXPERIMENT 4: UTILITARIAN CORRELATED-Q

A utilitarian Correlated-Q Learner was implemented such that the learner learns a strategy that maximizes the rewards of all players. Similar to Foe-Q and Friend-Q, a *112 x 5 x 5* matrix Q matrix is maintained, but for each Q update each player needs the other players Q values in order to calculate the action that maximizes both players rewards. Thus, the Q-update changes to the form:

$$Q_1^{new}(s_t, a_t^1, a_t^2) = (1-\alpha)\cdot Q_1(s_t, a_t^1, a_t^2) + \alpha\cdot((1-\gamma)\cdot r_t + \gamma\cdot V_i(s_{t+1})) \qquad (4)$$

Where $V_1(s_{t+1})$ is the summation of Q values over the joint action space:

$$V_1(s_{t+1}) = \sum_{a_1 \in A_1, a_2 \in A_2} \pi(a_1, a_2) Q_1(s_{t+1}, a_1, a_2) + \pi(a_1, a_2) Q_2(s_{t+1}, a_1, a_2) \qquad (5)$$

Here the probabilities $\pi(a_1)$ must satisfy the conditions of being at least zero, summing to one and rationality constraints for each player (Player 1 in this case):

$$\sum_{a_1 \in A} \sum_{a_2 \in A} \pi(a_1, a_2) Q_1(s_{t+1}, a_1, a_2) \geq \sum_{a_1 \in A} \sum_{a_2 \in A} \sum_{a_3 \in A \text{ if } a_3 \neq a_2} \pi(a_1, a_3) Q_1(s_{t+1}, a_2, a_3) \qquad (6)$$

Utilitarian CE-Q seeks to maximize the sum of the rewards of all players considering all actions, which can be represented as follows:

$$\max_{\pi_s \in \Delta(A_s)} \sum_{j \in Players} \sum_{a_1 \in A_1, a_2 \in A_2} \pi_s(a) Q_j(s_{t+1}, a_1, a_2) \qquad (7)$$

In total for this Soccer Game world, I derived 67 constraints. After implementing this algorithm, difference between Q values for the initial state and Player A heading South and Player B sticking were plotted similarly to [1].

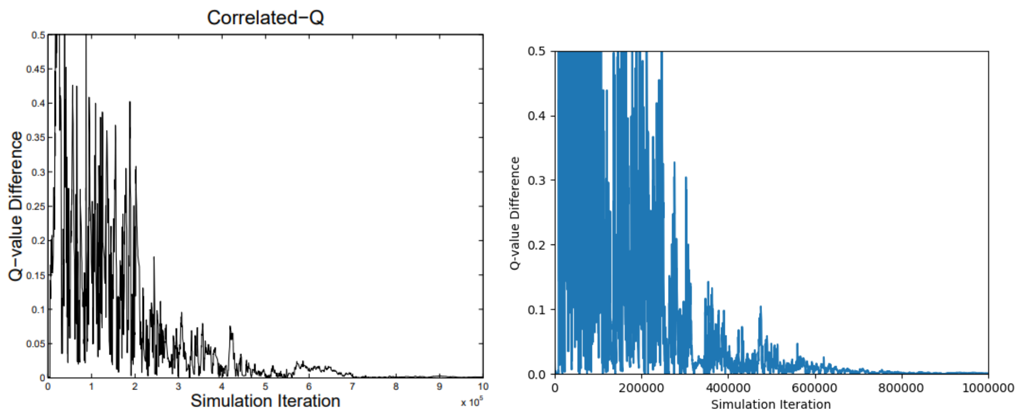

Figure 5: CE-Q Results (Q-Error vs Time) [Greenwald on Left and Experimental on Right]

The interesting part of the CE-Q algorithm is that in the process of maximizing the rewards of both players CE-Q also learns the minimax strategy just like Foe-Q. Additionally, CE-Q can also learn the cooperative strategy in cooperative games as shown by other grid games in [1]. Thus, CE-Q finds correlated equilibria that for the cases of constant sum games such as Soccer Game contain minimax equilibria, and for cases of general sum games CE-Q can also find correlated equilibria that contain Nash and coordination equilibria.

This comes though with much greater computational cost, as this algorithm took much longer to solve than Foe-Q. Additionally, CE-Q requires players to share Q matrices between each other and thus coordinate. Thus, comparing this algorithm with the prior algorithms is unfair because if those were implemented to coordinate as well then, they may have found the equilibrium policies as well. Additionally, I came across iterations where the linear program solver could not solve the setup problem because it was primal infeasible, which is a further limitation to the CE-Q approach in that it can’t solve at all time steps. Although indeed Correlated Equilibria are easier to calculate than Nash Equilibria, adversarial equilibria are still much faster to calculate.

7. CONCLUSION

Despite getting close results, I was not able to exactly replicate Greenwald’s published results due to the lack of details on hyperparameters and exact game mechanics. Although Greenwald states that α was decayed to 0.001, she didn’t give the exact α used or the decay schedule in her experiment. The tuning of these hyperparameters was easy enough in the case of Q-learner because it took less than a minute to run, but the Foe-Q and CE-Q algorithms each took over an hour to run. Greenwald only gives a cursory glance over the game mechanics, and I’m unsure if I am replicating his exact mechanics. For example, I assumed that if the first moving player runs into the other player, the second player can still move to another location. The paper only mentioned that the ball ownership changes if the second moving player has the ball but didn’t mention if the second moving player’s actions are implemented. I also had to assume that the x-axis of the graphs in figure 3 of [1] was the time step at each iteration as opposed to each episode in the training. Furthermore, the error metric used to plot the y-axis of these graphs was also unclear and was just assumed to be the absolute difference.

Although the CE-Q and Foe-Q are published to have given the same exact results with the same Q matrix, the CE-Q took significantly longer to run due to having more constraints to solve each linear program.

REFERENCES

[1] Greenwald, Amy, Keith Hall, and Roberto Serrano. "Correlated Q-learning." *ICML*. Vol. 3. 2003.
[2] Greenwald, Amy, Keith Hall, and Martin Zinkevich. "Correlated Q-learning." *Brown University Technical Report*. 2005.
[3] Littman, Michael L. "Friend-or-foe Q-learning in general-sum games." *ICML*. Vol. 1. 2001.