Replication of Greenwald and Hall's Correlated-Q Learning Experiment

Beom Jin An

ban37@gatech.edu

Git Hash: 82ad42590e9c1d38868f2aba4f62e52f7494572f

I. Introduction

The paper, "Correlated-Q Learning (2003)" by Amy Greenwald and Keith Hall, introduces a new concept of a multiagent Q-learning algorithm. Correlated-Q learning (CE-Q) is based on the correlated equilibrium concept, which combines both the Nash-Q and Friend-and-Foe-Q concepts.

This project will attempt to replicate Greenwald and Hall's "soccer game" experiment on Correlated-Q learning. The experiment highlights the difference between well-known Q learning, Friend-Q, Foe-Q, and Correlated-Q. The replication requires an in-depth understanding of Correlated-Q learning and the impact of hyperparameters on the result. Also, an efficient method of linear programming must be used to replicate the experiment successfully.

A. Correlated Equilibrium

The Nash equilibrium (NE) is a vector of independent probability distributions over the agents' actions. These probabilities are optimized to the other agent's probabilities. The Correlated equilibrium is similar to the Nash equilibrium, but CE-Q allows dependencies among the agents' probability distributions. The correlated equilibrium allows for dependencies in the agents' randomizations as CE is a probability distribution over the joint space of actions. Also, CE can be computed using linear programming, which makes computation much more manageable. In some cases. CE can achieve higher rewards than NE by avoiding positive probability mass on less desirable outcomes, and CE can also converge after many games [1].

B. Variants of CE-Q

There are four variants of CE-Q that use different objective functions.

These are called utilitarian, egalitarian, republican, and libertarian, respectively. All these equations can be solved using linear programming [2].

1. maximize the *sum* of the players' rewards:

$$\sigma \in \arg\max_{\sigma \in \mathrm{CE}} \sum_{i \in I} \sum_{\vec{a} \in A} \sigma(\vec{a}) Q_i(s, \vec{a}) \tag{9}$$

2. maximize the *minimum* of the players' rewards:

$$\sigma \in \arg\max_{\sigma \in \mathrm{CE}} \min_{i \in I} \sum_{\vec{a} \in A} \sigma(\vec{a}) Q_i(s, \vec{a}) \tag{10}$$

3. maximize the *maximum* of the players' rewards:

$$\sigma \in \arg\max_{\sigma \in \mathrm{CE}} \max_{i \in I} \sum_{\vec{a} \in A} \sigma(\vec{a}) Q_i(s, \vec{a}) \tag{11}$$

4. maximize the *maximum* of each individual player i's rewards: let $\sigma = \prod_i \sigma^i$, where

$$\sigma^i \in \arg\max_{\sigma \in \mathrm{CE}} \sum_{\vec{a} \in A} \sigma(\vec{a}) Q_i(s, \vec{a}) \tag{12}$$

Thus, $\mathrm{CE}_i(\vec{Q}(s)) = \left\{ \sum_{\vec{a} \in A} \sigma(\vec{a}) Q_i(s, \vec{a}) \right\}$, where σ satisfies either Eq. 9, 10, 11, or 12.

For this experiment, we will be using the utilitarian CE-Q.

C. Equilibrium Policies

In general, all the learning algorithms should converge. In an environment like a grid game, on-policy Q-learning consistently converges to equilibrium policies where the two players perfectly coordinate their behaviors [3].

D. CE-Q Learning

In grid games, the CE-Q learning variants all match precisely with those of Friend-Q. A centralized version of CE-Q even allows for consistent coordination, and it does not allow collision between the players. However, the libertarian variant of CE-Q does not necessarily coordinate the two players. The grid game allows for some coordination to achieve the best result, and that is why the CE-Q produced a result identical to what Friend-Q would. However, in a game where the coordination between two players does not provide benefits, Foe-Q learning might take precedent. In the following grid game experiment called "Soccer game," Greenwald and Hall show that CE-Q learning can also generalize Foe-Q learning [4].

II. Experiment

A. Soccer Game

The soccer game is a zero-sum grid game where deterministic equilibrium does not exist. There are two players whose possible actions are N, S, E, W, and stay. If the two players collide, the player who moved first takes the spot. However, if the player with the ball moves second and collides, the ball changes possession to the other player. If the player with the ball moves into a goal state, that player receives a 100, and the other player gets a -100.

Since there are no deterministic policies, the example in Figure 4 shows that the possible deterministic policy would indefinitely block each other from moving. However, if player B act non-deterministically, it can move past player A [5].

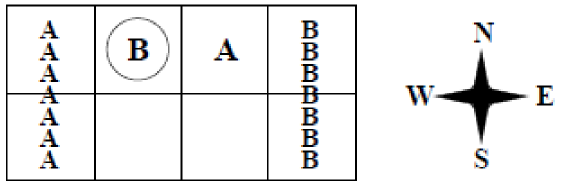


Figure 4. Soccer Game. State s.

B. Methods

We are trying to see the difference between Q-learning, Foe-Q, Friend-Q, and CE-Q for this experiment. Unlike the grid game, the soccer game is a battle between two players where coordination does not help the players. Whereas the Battle of the Sexes game in the first grid game provided a result identical to Friend-Q, the soccer game is expected to provide a result identical to Foe-Q. To conduct this experiment, I had to replicate Greenwald and Hall's methods. This required me to select hyperparameters that would reflect the paper's results. Also, I had to read the analysis and the methods of the original experiment carefully.

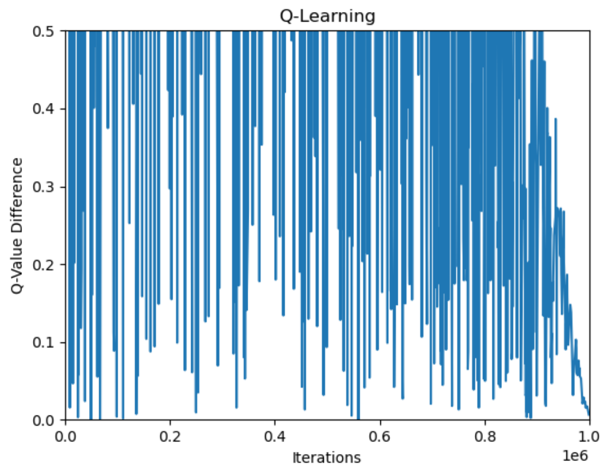
Using the non-deterministic policy in Figure 4, we can find the Q-value difference between that policy and the calculated Q-values would converge over time. The policy at this state is for player A to take the S action and player B to stay. I will using this function to calculate the error between the Q-values of the known state and the learned states.

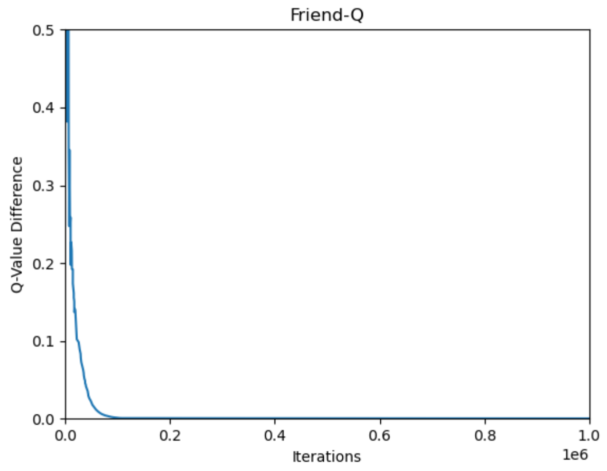
$$\mathrm{ERR}_i^t = \left|Q_i^t(s,\vec{a}) - Q_i^{t-1}(s,\vec{a})\right|$$

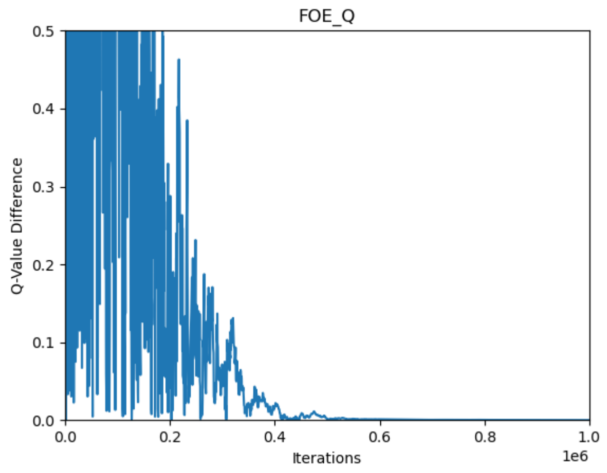
For the Q-learning, I will be using the Q-value update function from paper.

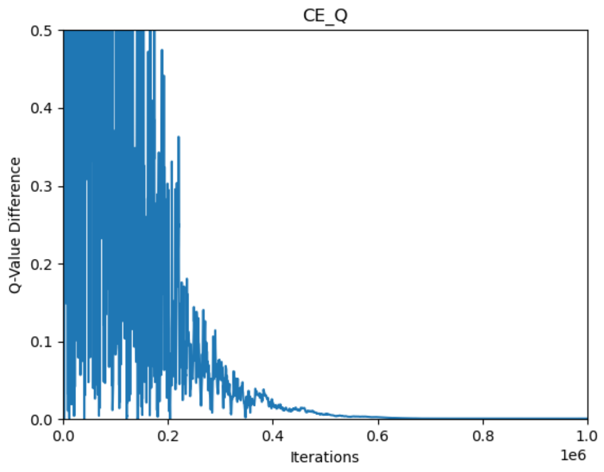
$$V_j(s') = \textstyle\sum_{a\in A_{s'}} \pi^*_{s'}(a) Q_j(s',a)$$
$$Q_j(s,a) = (1-\alpha)Q_j(s,a) + \alpha[(1-\gamma)R_j(s,a) + \gamma V_j(s')]$$

C. Results









D. Analysis

All variants of the CE-Q learning should converge at any state in this grid game since all states have equivalent values. Thus, the CE-Q objective functions will all produce identical outcomes. In addition, CE-Q learns Q-values that converge exactly identical to what Foe-Q would converge to.

Q-learners ignores the opponent's actions, which means the learner never converges because it does not know that the opponent's actions matter significantly. The Q-learning graph only seems to converge because the learning rate decreases, but it never stabilizes to a solid value. Since there is no known deterministic policy in this game, the values will never converge.

The CE-Q and Foe-Q graphs were supposed to imitate each other since they both converge to a non-deterministic policy. The agents are capable of making moves based on the probability distribution of the actions, thus moving in different directions to avoid indefinite blocking. Therefore, both algorithms learn to get the minimax equilibrium policies for this two-player zero-sum game.

The Friend-Q graph converges a lot quicker than the others. This algorithm allows only one player to maximize the rewards while the other player helps that one player. In this case, there is a deterministic policy where it helps one player to achieve the biggest reward. The "friendly" player will make moves that will assist the other player by moving out of its way or giving the ball to the other player. In some sense, the "friendly" player can step into their own goal to score for the other player.

III. Replication and Pitfalls

The hardest part of the replication was to build a sufficient algorithm to reproduce the CE-Q. The Q-learning and Friend-Q algorithms were simpler to implement. They are only required to focus on one agent. The Q-learning calculates for both agents, but only one agent's Q-values are used for the final error calculation.

A. Replication

The replicated Q-learning graph matches the original result closely but not precisely. The original Q-learning graph seems to decrease its values a lot smoother whereas the replicated graph seems messy and has a sudden drop at the end. I believe that this is the result of the hyperparameters not matching. There must be a disparity in the alpha decay or even the epsilon decay. However, the replicated graph does show that the Q-values did not converge, which was what the experiment intended.

The replicated Friend-Q graph matches the original graph very closely. The Q-value difference converges very quickly and in a similar manner. The paper explains that this happens because the two players are working towards the same goal, so the extensive learning is not required.

The replicated Foe-Q graph matches the experiment's result. The drop around 200,000 iteration is similar, and the graph also converges to 0.

The replicated CE-Q graph also matches the experiment's result. To make the Foe-Q and CE-Q identical, I had to set a seed for randomization, which meant I need to set a seed for the soccer game environment and the randomized actions for the players. Even with the seed, I was not able to create identical graphs of Foe-Q and CE-Q. The reason behind this is the randomization that selects which player will move first. I found that if I set a seed for the function that randomly chooses the first player, I get Q-values that are all zeros. Thus, I had to remove the seed for this randomization which must have resulted in the two graphs looking differently.

B. Hyperparameters

The first hyperparameter is the alpha, the learning rate. The paper highlights that the alpha should stop decaying at 0.001, but it does not specify how. In another paper by Greenwald and Hall in 2005, it explains that the $\alpha = n(s, a)$, where $n(s, a)$ is the number of visits to state-action pair [6]. However, I found that this method converges too fast. Since it converged too fast, the resulting graph showed a horizontal line of 0's. For the CE-Q, Foe-Q, and Friend-Q, I tried using a geometric decay of 0.9999, which proved to provide a similar result as the original. For Friend-Q, to make the graph converge fast like how the original experiment did, I had to change the initial alpha value to 0.2. I came across this initial alpha value by testing out which one would give me the closest shape to the experiment. I believe that Friend-Q did not learn that much since both players are working towards the same goal. It is obvious to both players what to do immediately. For the Q-learning, the geometric decay was too fast, so I had to use a linear decay to produce a similar graph. When the alpha converged too quickly, the results seemed to converge to 0.

The alpha decay for Foe-Q and CE-Q had to be adjusted better to replicate the graphs. The original graphs had the Q-value difference start converging around 200,000~300,000 iterations. In order to do that in my algorithm, I had to adjust the alpha decay to 0.9999. This alpha decay gave me a similar convergence pattern as the original experiment.

For the discount rate, gamma, I used 0.9 for all algorithm as stated in the paper. For the epsilon, I only needed it for the Q-learning. I used a geometric decay of 0.9999 for the epsilon that starts at 0.9.

C. Linear programming

I used the cvxopt module to compute the linear program. I first made an algorithm that takes in one matrix 5x5 of the two players' action values for the Foe-Q, which is an adversarial equilibrium. The one player is trying to minimize the other player's reward while the other player is trying to maximize that minimization. I also needed to put some constraints. One constraint is to ensure that each action's probabilities are greater than 0, and the other constraint is to make sure that the sum of the probabilities equals 1.

For the CE-Q linear programming, it uses the utilitarian CE-Q, where it maximizes the sum of all agents' rewards. The algorithm for computing a CE in polynomial time depends on the dual has a polynomial number of variables. You have to solve with the ellipsoid, which polynomially many steps should determine that the program is infeasible. There is a solution of CE that is a convex combination of polynomially many product distributions. Furthermore, an oracle for computing $U^T X$ yields a polynomial-time algorithm for computing the correlated equilibria. The steps for CE-Q is to run k step of the ellipsoid method producing k candidate points y_i, compute distribution $x_1, x_2, ...x_k$ such that $x_i^T U^T y_i = 0$, and let X be a matrix of rows x_i and compute $\alpha \geq 0$ such that $(UX^T)\alpha \geq 0$ [7].

However, it wasn't easy to implement linear programming into the code. I had to play around with the shape of the matrices using linear algebra. Using sufficient resources online, I was able to follow a guide on how to add constraints and multiply the two matrices. Surprisingly, the CE-Q linear programming calculation using cvxopt was fast enough that I was able to run 1,000,000 iterations in under 20 minutes.

D. Algorithm

The Q-learning algorithms were much more straightforward than making the linear programming algorithm. I've used the pseudocode provided in another paper to replicate it. But I realized that I did not necessarily need the value, function, V, to be able to update the Q-values. I used the expected values calculated using linear programming and directly input those values into the Q-value update function. I did not see the necessity to store these values in another array.

For the error calculation, I've used the state depicted in Figure 4 of the paper. I used the error calculation formula to find the Q value difference over time.

E. Pitfalls

The hardest part of the replication was finding the right hyperparameters and the right algorithms. I realized that any deviation from the original settings of the experiment would mean a completely different result. I played around with the alpha decay mostly to replicate the graphs' behavior. I still yet understand exactly how the decay was performed in the original paper, but I was able to achieve something very similar. My graph failed to replicate the first 200,000 iterations. The original graph had the Q-value difference under 0.5 in the first couple iterations; however, my graph seems to go over that 0.5 and out of the graph's boundary for the first couple iterations. This probably means that alpha decay methods are different.The original paper must have had the alpha values to decay in a manner not merely linear or geometric. Some other paper mentions that the decay should happen depending on the number of state-action pair the players visited. I've tried my best to replicate that, but the result was worse. Thus, I stucked to a geometric decay that allowed me to control exactly where the graph starts to converge suddenly, around iteration 200,000 ~ 300,000.

IV. Conclusion

I replicated the graphs very closely, and I learned a lot about CE-Q, Friend-or-Foe-Q learning, and linear programming. This project's success relied on my understanding of how these learning algorithms react to certain hyperparameters and iterations and implement linear programming. I was able to use the previous project's knowledge to truly implement how the hyperparameters impact the shape of the results. Overall, this project gave me insight into how reinforcement learning can be quite complicated, even with the simplest problem. However, I know that reinforcement learning can achieve things that we normally thought were impossible by a computer.

References

[1] A. Greenwald, K. Hall, "Correlated-Q Learning", pp. 1, 2003.
[2] A. Greenwald, K. Hall, "Correlated-Q Learning", pp. 3, 2003.
[3] A. Greenwald, K. Hall, "Correlated-Q Learning", pp. 3, 2003.
[4] A. Greenwald, K. Hall, "Correlated-Q Learning", pp. 3, 2003.
[5] A. Greenwald, K. Hall, "Correlated-Q Learning", pp. 6, 2003.
[6] A. Greenwald, K. Hall, M. Zinkevich, "Correlated-Q Learning", pp. 21, 2005.
[7] A. Shkolnik, A. Sabreri, "Correlated Equilibria", pp. 3, 2009.