Isak Diaz

# Reinforcement Learning Project 3

Multi-agent Q-learning algorithms pose an interesting problem for reinforcement learning because convergence criteria rely on equilibrium being found within the payoff matrices of multiple agents. These matrices contain numerical values that represent the motivations of different actors. In Greenwald's 2003 paper he discusses Correlated Equilbrium Q-learning which is a is a generalization of multi-agent Q-learning algorithms such as Nash-Q, Foe-Q and Friend-Q. In this paper we try to replicate Greenwald's graphs in relation to 4 algorithms implemented in a Soccer game. His results are in figure 1.

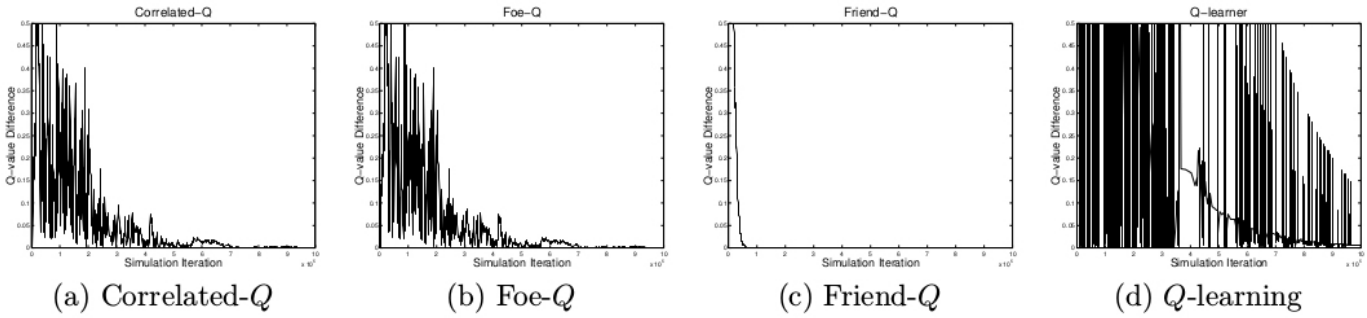


**Figure 3. Greenwald's Q State Value Updates**

First, the soccer game environment was created for a 2 player zero-sum Soccer game. Then we implemented reinforcement learning techniques by starting with a Q-learner. Each player had an independent Q table that was updated using Bellman's equation. The best action for each player was implemented using an epsilon-greedy approach with epsilon starting at 0.9 and linearly reducing to 0.01 by the 1 millionth iteration. Q-learning is not guaranteed to converge and it didn't in this example. The reduction in Q-value difference comes mostly from the reduction in alpha of our algorithm. Alpha started at 1 and eventually decayed to 0.001 linearly. However, a linear reduction in alpha did not provide similar results to Greenwald's paper. Exponentially reducing alpha proved to offer more fidelity to the original experiment but it was still unable to match the curve of the experiment. The large spikes in Q-value change were absent from our implementation of the Q-learner, however the overall learning curve was the same. This is very specific to alpha which Greenwald's paper is careful to point out, however the paper is silent on the exact method of decaying alpha down to the 0.001 value.

The graphs below for the learners represent the Q value update difference of the state s in figure 4 from the perspective of Player A.

The friend Q learner will quickly converge since the opponent is actively helping Player A reach a maximum score. It does this by passing it the ball and letting it score. Very little learning takes place after the few iterations it takes to converge. While the Q -learner performs on-policy learning and only randomizes it's actions according to the epsilon parameter; Friend-Q, Foe-Q and Correlated Equilibrium-Q all perform off-policy learning. The algorithms take random actions and update their Q tables

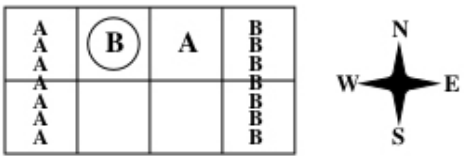


*Figure 4.* Soccer Game. State $s$.

according to the present and discounted future rewards of those actions. The gamma the paper used to discount the future was 0.9 and that was consistent across all of our experiments.

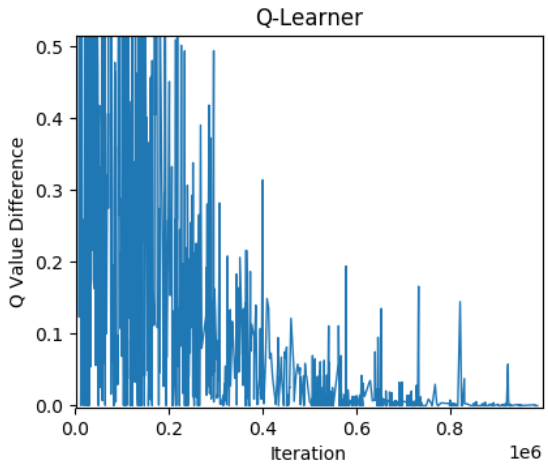


**Figure 1. Q Learner**

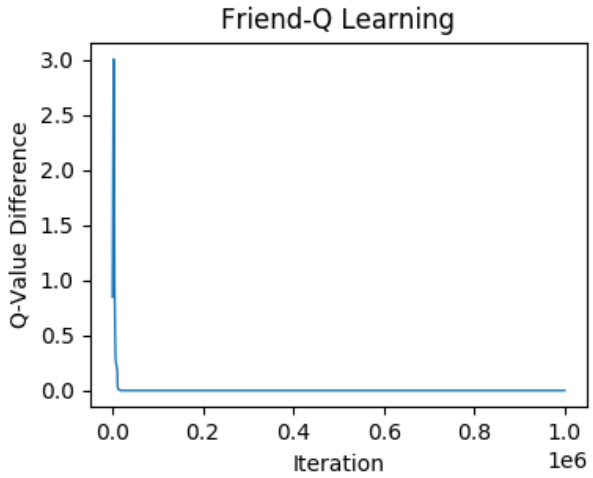


**Figure 2. Friend-Q Learner**

The third algorithm was Foe-Q which was modeled with a single Q table. The adversary was assumed to be acting solely to reduce Player A's returns. This implies that Foe's rewards are the negation of Player A's returns which accurately reflects the zero-sum nature of the game. This is crucial because Nash equilibrium and minimax strategies coincide in zero sum games which is why our Foe-Q algorithm has no issues converging. Running the minimax algorithm allowed us to model an adversarial player. A series of linear equations was constructed and linear programming was used to solve the probabilities for each action to take in S prime. These probabilities were multiplied by the Q values of those future actions. Then the current state's Q value was updated using discounting according to gamma and the learning rate.

The minimax solution for a zero sum game proves it's versatility in that the specific probabilities for each action are calculated in a way that forces a rational opponent's actions to be predicted probabilistically. In a pure strategy minimax zero sum game, the result is the same even if the opponent can predict what the player will do. For this implementation of Foe-Q, the strategy was mixed and the players both chose actions without the other knowing. However, the mixed strategy that the player chooses has a corresponding best strategy for the opponent. It's like a pure strategy game in that there is a unique solution that provides the best possible payoff.
To solve a correlated equilibrium, a set of linear equations were created that represented the constraints of each player. Neither player would be compelled to choose an alternate action given a signal by a third party. Their rewards would be lower if they simply chose to ignore the advice given. Creating a set of linear equations for the probabilities of each possible action pair allows us to solve the optimal distribution of actions. The equations chosen must simply weigh the benefits of choosing another row according to the probabilities associated with the other player also choosing another column. Both players will find themselves in an equilibrium since following correlated strategy pairs will always be in their best interest.

In this experiment we chose V which is the expected value of the next game state to be the future Q values as the set of attainable payoffs under the correlated strategy. The aggregate of linear equations forms a convex hull which we can easily maximize to propose a set of actions that would produce a higher reward for each player than acting independently. This expected reward is what we update the Q equation with for all future values of the state action pairs.

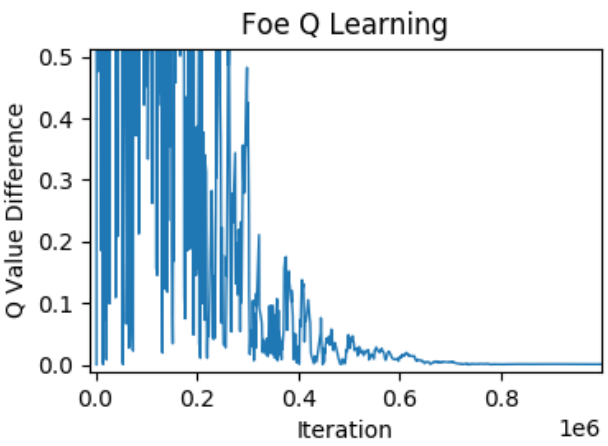


**Figure 5. Foe -Q**

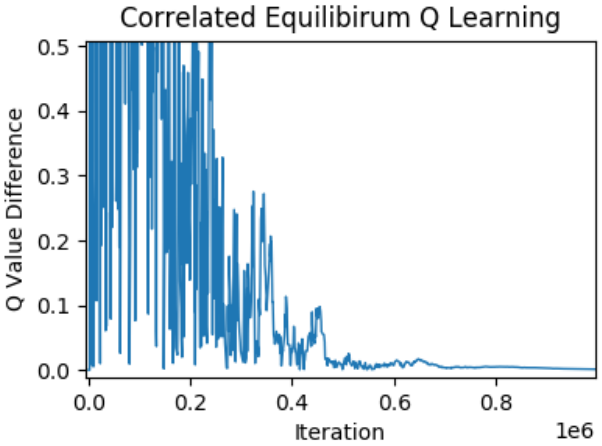


**Figure 6. CE-Q**

In the figures above we see the convergence of Foe-Q near 400k iteration mark while Correlated Q starts to converge near the same mark. The paper states that both Foe-Q and Correlated-Q learning both converge to the same Q values. I found this to be true with my final Q values between the two algorithms only varying on average by 0.2525 percent. Below is a table of the differences of each Q value.

[[ 0.00239241 0.00226303 0.00301427 0.00258774 0.00315265]
[ 0.0019054 -0.00640892 0.00198563 0.00293548 0.00235877]
[ 0.00256838 0.00273445 0.00329514 0.00397931 0.00300159]
[ 0.00310586 0.00275013 0.0023437 0.00341574 0.00346418]
[ 0.00382976 0.00375354 0.00283107 0.00273053 0.00313998]]

**Table 1. Foe-Q and CE-Q Value Differences (Percentages with 1 = 100%)**

Trying to replicate the Foe-Q graph for correlated-Q proved difficult. Looking back on the assignment, I wish I had used a random seed for all the experiments since each run of the experiment produced updates at different iterations due to the random actions. If I had used a random seed then the updates would have happened at the exact time and it would have been interesting to see if the value updates were the same. This would have been interesting since Foe-Q uses a single Q table and does minimax while CE-Q uses two Q tables and solves a much more complicated set of equations regarding each player's motivations. The graphs look very similar but not as similar as the two graphs did in Greenswald's paper.

Overall all the graphs matched closely to Greenwald's paper. The assumptions on how to decay alpha and epsilon changed the graphs dramatically and needed some fine tuning. The biggest problem with recreating the graphs was how to treat the opponent. For Q, Friend-Q, and Foe-Q there was only a single Q table used for the main player. Player B had random actions for regular Q-learning. Friend-Q simply chose the action that would benefit us the most, and Foe-Q chose the action that would be most detrimental to our agent. The choice to use a single Q table was not obvious at first for the original Q learner. Originally, I chose two tables and had both player's do what was in their best interest. This however proved subpar results.