Correlated Q-Learning – Soccer Game

Brad Smith

I. Introduction

Recent modern challenges in reinforcement learning have centered around developing effective solutions to problems with multiple agents affecting an environment. The reason for this is rooted in the difficulties of solving general sum games of arbitrary players for optimal policies. Practical solutions for multiple Markov Decision Process (MDP) agents require expansion of Q-value function updates to solve for joint policies resulting from equilibria. One such method is called Correlated Equilibrium (CE) Q-learning, which is a solution method for general-sum Markov games that can be solved for in polynomial time using linear programming. Amy Greenwald developed a set of four specific correlated equilibrium mechanisms. This report validated Greenwald's results for comparing utilitarian CE-Q learning to Friend-Q, Foe-Q, and standard Q-learning in the context of a simple soccer game with two opposing agents [1].

II. Methods

The soccer game environment was used for analysis of various multi-agent solution techniques initially introduced by Greenwald in 2003. This specific environment was chosen since Greenwald had generated results for the same problem, thus verification of result could be reasoned, ambiguities cleared up, and reasons for differences explored. Additional support for implementation was derived from Greenwald's 2007 revisit of the soccer problem [2].

A. Soccer Game Environment

The soccer game is a grid world environment represented in Figure 1. The field is a 2x4 grid within which both players, denoted by A and B, are free to move according to the cardinal directions: North, South, East, and West. The agents can also choose to stick (not move) [1]. The ball is constrained to be collocated with one of the two players at the start of the game and at no point during the game can the players occupy the same coordinate. Initial ball possession was randomized. To restrict initial states resulting in a goal, both players were forced to start at a random location in the center two columns. This could be overridden to test hardcoded initial positions.

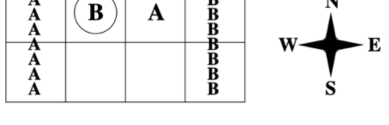


Figure 1: The Soccer Game Grid World [1].

At each turn, there is a random 50% draw to determine which player gets to move first. A consistent random seed was used at the beginning of training and incremented each episode. If both players intend to move into the same coordinate, then the priority is given the player with the first move, and the player with the second move must remain in its current location. If one player possesses, the ball continues to move coordinates with that player. In the case that the player with possession moves into the coordinate containing the other player, possession is yielded to the other player. On the converse, the player without the ball cannot bump into the player with possession and obtain the ball. The left and right most columns are the goals for player A and B, respectively. A goal results in a reward of +100 for the scoring and -100 for the player who was scored on, defining this as a *zero-sum game*, meaning the rewards are opposite and equal for the two players [1].

Some additional logic was included in this implementation that wasn't initially explained in the literature. Actions that would take a player out of bounds were restricted so that the player would simply remain in the coordinate it was at prior to attempting to take the invalid move. In the case that a player took an invalid move back into the same grid and had the ball, a collision with the other player was determined to not result in a change of possession. Although these are not explicitly known to be the same constraints Greenwald used, maintaining consistency was believed to make consistent qualitative results obtainable.

B. Correlated Equilibria

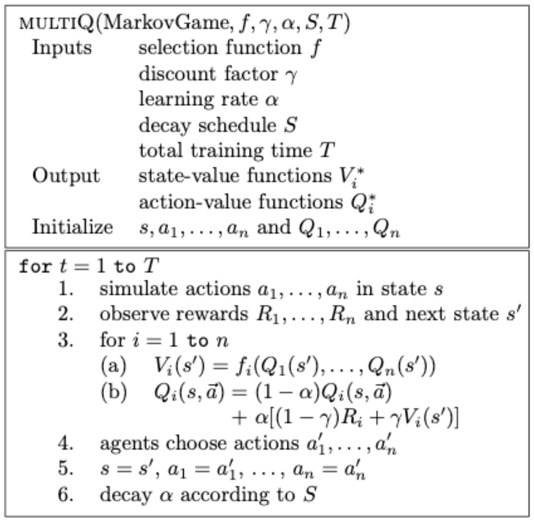
Correlated Q-Learning for multi-agent learning is based on the same fundamental reinforcement learning concepts as exists for single agent MDP problems. An extension is applied to account for the fact that in a multi-agent environment, there are potentially different actions, rewards, and future value estimates that each agent might have regarding the current state of the environment, so computing future expected reward is nontrivial. The state of the environment is consistent and observed by both agents. For the soccer game, the state, s, was an encoding unique to each observed location of the ball, player A, and player B. The Q-learning update for multiple agents is shown in (1).

$$Q_i(s,\vec{a}) = (1-\alpha)Q_i(s,\vec{a}) + \alpha[(1-\gamma)(R_i + \gamma V_i(s')] \quad (1)$$

Here, the subscript i denotes existence for each of the N agents in a Markov game [1]. The action space, $\vec{a}$, is a vector of each agent's action when the current state is s. R_i generalizes that the reward signal for each agent may be unique. The element that complicates the multi-agent learning process is $V_i(s')$, the estimated value of the next state. The reason for this is that it is known for single agent MDPs, there exists an optimal policy where all Q-values are maximized (Q*), but this need not be the case to satisfy the system of equations from (1). For general sum games of more than one player, the Nash Equilibrium represents the same characteristic, conceptually, for an arbitrary number of players in a game. The Nash equilibrium is the result in a general sum game where all agents are optimized with respect to the other, and no player has a better action to take; it can be thought of as the solution to the n-player minimax problem. The issue with Nash equilibria is that there is no known efficient solution method. Attention is turned toward Correlated Equilibria (CE), where agents can be conditioned to the probabilities of the action probability distributions of other agents and optimize accordingly. Unlike with Nash, CE can be solved for in polynomial time using linear programming [1]. The correlated equilibrium is computed by a central observer who has access to both player's Q-value functions and uses them to solve for a correlated equilibrium. The correlated policy provides the two players a recommended action according to a joint probability distribution across the players' actions: the resulting policy is more beneficial to both agents than independently generated policies. Comparisons were made between solutions for CE and other multi-agent learning techniques to draw distinctions in the benefits of the CE solution approach for the soccer game problem.

C. Generalized Multi-Agent Q Learning

Greenwald introduced a generalized form of the multi-agent Q-learning algorithm, which is shown in Figure 2. This algorithm can be applied to each of the algorithms of interest (CE Q-learning, Foe-Q, Friend-Q and Q-learning) by changing the definition of the function f_i. The value function is shown being provided a vector of each agent's Q-table, although not required for every learning algorithm.



```
MULTIQ(MarkovGame, $f, \gamma, \alpha, S, T$)
  Inputs      selection function $f$
              discount factor $\gamma$
              learning rate $\alpha$
              decay schedule $S$
              total training time $T$
  Output      state-value functions $V_i^*$
              action-value functions $Q_i^*$
  Initialize  $s, a_1, \ldots, a_n$ and $Q_1, \ldots, Q_n$

for $t = 1$ to $T$
    1. simulate actions $a_1, \ldots, a_n$ in state $s$
    2. observe rewards $R_1, \ldots, R_n$ and next state $s'$
    3. for $i = 1$ to $n$
        (a) $V_i(s') = f_i(Q_1(s'), \ldots, Q_n(s'))$
        (b) $Q_i(s, \vec{a}) = (1 - \alpha)Q_i(s, \vec{a})$
                $+ \alpha[(1 - \gamma)R_i + \gamma V_i(s')]$
    4. agents choose actions $a_1', \ldots, a_n'$
    5. $s = s'$, $a_1 = a_1'$, $\ldots$, $a_n = a_n'$
    6. decay $\alpha$ according to $S$
```

Figure 2: Multiagent Q-Learning defined by Greenwald [1].

The Q-value update is identical to that as originally introduced in (1). In the algorithm, the variable $\vec{a}$ is the action space, $\vec{a} = (a_{-i}, a_i)$, the vector of actions taken by both agents in each state, from player *i*'s perspective. This is illustrated in line 3b, where after agent *i* take its action, it observes the actions taken by the other agents, in the case of the soccer game the other player. This succinct definition was designed to be expandable to multiple different types of multi-agent learning techniques, so some of the implementation specific details of each algorithm's design required various methods for executing each step in Figure 2. The specifics for each algorithm for multi-agent learning are expanded in the following two sections.

D. Utilitarian Correlated Q-Learning

The primary focus of Greenwald's work was to introduce the modifications to Figure 2 to solve for correlated equilibriums. This solution required implementation of a linear program that defined the constraints and the objective function of the system of inequalities of interest. The linear program represents the role of the observer, who has access to each player's Q-table and can compute a joint probability distribution, π_s, where the objective function is defined depending on the type of equilibrium desired.

To solve for the equilibrium at each time step, constraints and variables were established. Greenwald formulated a generalized approach to multi-agent constraints in 2007 with a complete derivation [2]. The derivation is deferred to the literature, but the key result is that for each player *i*, for each action that player *i* can take (a_i), and each alternative action that player *i could* have taken, the following inequality must be true:

$$\begin{aligned} &\sum_{a_{-i} \in A_{-i}(s)} \pi_s\,(a_{-i}, a_i) Q_i\big(s, (a_{-i}, a_i)\big) \\ &\geq \sum_{a_{-i} \in A_{-i}(s)} \pi_s\,(a_{-i}, a_i) Q_i\big(s, (a_{-i}, a'_i)\big) \end{aligned} \tag{2}$$

The variable π_s is the joint probability distribution of the opponent's action, a_{-i} and agent *i*'s action, a_i. This is joint probability distribution of a mixed strategy for both players are solved for an equilibrium. The equation states that when summing over all the possible moves by the opponent, conditioned on a_i, there is no incentive to select a different action, represented by a'_i. Another way to frame this is that that expected value of unilaterally deviating away to any of the other action options, a'_i, the expected value should be *at least as good*. This is the core goal of an equilibrium, that there is no better option. For the soccer game, there are 2 players, each with 5 actions and 4 alternative actions to consider, leading to a total of 2x5x4 = 40 unique versions of (2) to solve as a linear system of inequalities. There are 25 unknowns for each $\pi_s(a_{-i}, a_i) = \pi_s(\vec{a})$, since there are 5 actions that each agent could take. By moving everything to the left side, the form of (3) is generated, as it is a format that can be programmed into a linear program solver.

$$0 \geq \sum_{a_{-i} \in A_{-i}(s)} \pi_s\,(\vec{a})[Q_i\big(s, (a_{-i}, a'_i)\big) - Q_i\big(s, (\vec{a})\big)] \tag{3}$$

As mentioned before, determining which equilibria to solve for is the driving factor behind the linear program's objective function. The constraints could have multiple solutions, so an additional level of detail is needed to retrieve an equilibrium with known behavior types. The constraints define the relationships that must hold for the solution, but a method is necessary for driving the constraints to a unique solution. In Greenwald's work, four different methods were introduced as equilibria selection mechanisms:

1. Maximize the *sum* of players' rewards: <u>utilitarian</u>
2. Maximize the *minimum* of the players' rewards: <u>egalitarian</u>
3. Maximize the *maximum* of the player's rewards: <u>republican</u>

4. Maximize the *maximum* of each player *i*'s rewards libertarian

The first selection mechanism, utilitarian, was used for the soccer game, with the linear program objective function defined according to (4) [2].

$$max_{\pi_s \in \Delta(A(s))} \sum_{j \in N} \sum_{a \in A(s)} \pi_s(a)\, Q_i(s,a) \qquad (4)$$

Here, $A(s)$ is the powerset of actions at the given state, and a represents a possible action profile from that powerset. For example, the following substitution into (2) and (3) is valid representation: $a = (a_{-i}, a_i)$ [2]. Equation (4) finds the joint probability of actions which maximizes the summation over all action pairs for all Q-tables, seeking out the probability distribution that, if actions are jointly sampled from, will benefit both agents the most. Once a joint probability distribution is solved for, the suggested action for both players is sampled for, and taken with probability of $1 - \epsilon$ to allow for random action selection as well.

E. Alternative Multi Agent Learning Strategies

Correlated equilibria learning using uCE Q-learning was compared to 3 other alternative methods: Q-learning, Friend-Q, and Foe-Q. Q-learning was design according to standard implementation for off-policy tabular Q learning [3]. Friend-Q is a naïve agent implementation which assumes the opponent is trying to assist in achieving the goal. Friend-Q was implemented by selecting actions based off the highest overall value in each player's Q-table for a given state. Here, the agent is optimistic that the opposition will select actions which optimize its own future expected reward. The implementation method for this was to select the action with the maximum reward over all possible agent actions, using the same Q-table definition for uCE Q-learning.

Foe-Q learning is a little more complex and requires the use of linear programming to determine the value of player A according to (5). The constraint here is to maximize the minimum value of the Q-table value. This equation was used to derive constraints and objective functions for the Foe-Q learning algorithm. Foe-Q, as it intuitively sounds, solves for the values assuming that the opposing agent is trying to maximize damage to the current agent. A linear program was written to solve this for each Foe-Q player. The behavior of Foe-Q is to assume that your opponent is trying to hurt you, so you plan according to that worst possible outcome.

$$V_A = max_{\sigma_A \in \Sigma_A(s)} min_{a_B \in A_B(s)} Q_A(s, \sigma_A, a_B) = -V_B \qquad (5)$$

F. Implementation – Python

The soccer game environment (*SoccerGameEnv*) was developed in python and was defined similarly to an OpenAI Gym API, with the step command taking both player A and player B actions and returning both players' reward signals. This common environment was used for training in all four learning agent scenarios. The python library CVXPY was used to solve the linear programs for the correlated equilibria problem while the python library CVXOPT was used for foe-Q, since the *x* vector in linear programming for the Foe-Q problem cannot constrain the variable *V* from (5) to be greater than or equal to 0, which made using CVXPY more complicated.

III. Experiments

The learning capabilities of four different learning forms were compared for the soccer game environment: Q-Learning, Utilitarian Q-Learning, Foe-Q Learning and Friend-Q Learning. In all cases, an agent of the same type is trained when playing against another agent of the same type. Some comparisons to uCE-Q vs a random agent was drawn to validate the learning process. The metric of performance was the Q-Value Difference, which was recorded according to relative error of Q values in consecutive simulation iterations according to (5). The difference was always computed from player A's perspective for player being sticking (STK) and player A taking the south action (S). The exception to (5) is Q-learning where each agent's Q table does not include the actions of the other player; for this case, the error was computed using the south action at each state. All plots were created with consistent x and y ranges to stick consistently with how Greenwald presented the results in 2003 [1].

$$\Delta QValue_A^t = \left| Q_A^t(\hat{s}, (STK, S)) - Q_A^{t-1}(\hat{s}, (STK, S)) \right| \qquad (5)$$

Since replication of Greenwald's 2003 results were desired, the same hyperparameters were used. The initial learning rate was $\alpha = 0.01$, decayed by a factor of 0.9999 to a constant value of $\alpha = 0.001$. The discount factor was set to $\gamma = 0.9$. Action selection was taken using ϵ-greedy, setting the probability of taking a random action to $\epsilon = 0.99$ and decaying it by 0.9999 to a constant value of $\epsilon = 0.001$. All algorithms trained for 1,000,000 iterations: the number of total steps taken by the agents. Greenwald only defined constant hyperparameters (like discount factor being 0.9) and the limit of learning rate and ϵ-greedy parameters, so the scheduling of starting points and decay rates were adjusted as needed to extend training over the number of steps Greenwald used [1].

IV. Results

The results from running the four experiments were used to validate the conclusions that Greenwald originally made about correlated equilibria solutions to Markov games and how they compare to non-correlated methods. A discussion of the pitfalls and challenges of replicating Greenwald's results are first detailed.

A. Linear Programming Implementation

The process for implementing the linear program for correlated equilibrium was one of the greatest challenges for the multi-agent learning problem. Specifically, interpreting constraints correctly from (3), correctly maximizing (4), and the development using CVXPY and CVXOPT proved difficult. There were multiple areas to go down incorrect paths. One key detail that helped alleviate issues with obtaining results was the realization of the action profile being relative to each player. The implemented Q tables from the decentralized learning algorithm were implemented as a matrix with the columns being the "self" action and the rows being the opponent's action. This helped with implementation since the learner could be an object with two instances, one for player A and one for player B. The key mix-up initially, was that when the joint probabilities are computed, an assumption must be made about which player is the row player and which player was the column player, otherwise constraint values are multiplied by the symmetric counterpart of the joint probability (i.e., being multiplied by the (3,2) instead of the

(2,3) action pair). No assumption can be made that these should be equal since the Q values of the two players are not equivalent. Fixing some of the programmatic insertions of Q-values for the column player (chosen to be A) and the row player (chosen to be B) established consistency with which to compute the joint probability. This ultimately benefited implementation of other agents once reasoned through.

B. *Correlated Action Selection*

Another pitfall of the algorithm development process understanding how to generate the action for the next simulation cycle (line 5 in the decentralized q learning algorithm in Figure 2). This was left vague in the paper as Greenwald intended for this algorithm to be generalized to the various methods for learning (Q-learning, Foe-Q, Friend-Q, etc.). In the line of thinking about how actions are selected in Q-learning, there was initial confusion on whether the individual players' Q-tables would be needed to generate the next action. An initial mistake was made here for utilitarian CE-learning, where different permutations of the maximum values of the Q-tables for the two agents were used along with ϵ-greedy to select actions. This was erroneous because the key detail of correlated equilibria solutions is that the mixed strategy is determined by an observer entity. This entity is like a referee, whom has access to both Q-tables belonging to player A and player B and generates a recommended move from a mixed strategy joint probability distribution. This recommendation is based off constraints derived from the expected values of the action-value functions (Q-tables), so the application of these baked into the linear program solution. This was realized, and the algorithm was correct to sample from the joint probability distribution derived from the linear program and provided to respective players as a recommended action to take. In addition, since unexplored states were always initialized with Q-values of 0.0, random exploration was still deemed necessary, so the recommended action was selected according to the ϵ-greedy schedule.

C. *Q-Learning*

Results from Q-Learning for the soccer game are shown in Figure 3. There is clear indication that no learning occurred due to the divergent nature of the graph. The decreasing magnitude of the oscillations is attributed to decaying learning rate values with consecutive episode and not agent learning. Although not shown on the graph, the Q-Value differences converged from approximately 1.0 to a value of about 10% of the starting value (0.1) before the agent begins acting on-policy. This roughly aligns with the learning rate decay limit (0.001) as a percentage of the original value (0.01). This check is useful for determining if expected values are decreasing because of learning or diverging with a decreasing size in the maximum attainable value due to the decreasing value of the learning rate. The proposed reason for poor agent learning in this instance is because stochasticity of the agent turn-taking process makes it impossible for each agent to predict fully how the state of the environment will change when determining what move to make.

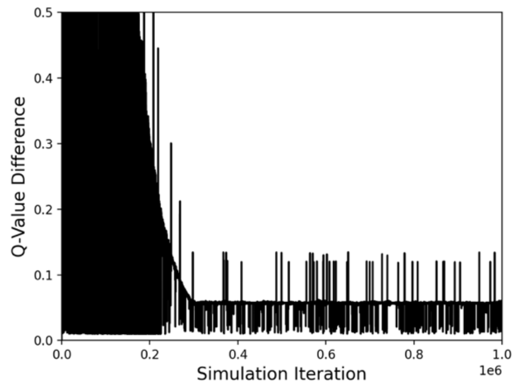


Figure 3: Q-Learning (filtered out delta = 0).

There is probability of 0.5 that player A will move first, and the rules of the game because of moving into the other player a drastically different depending on the action order sequence (difference in losing the ball or not, a huge driver in success in the game). These results make it clear that independent learning is not sufficient to have an agent be a successful learner in the soccer Markov game. This result illustrates the shortcomings of single Markov models when acting in an environment with other agents that can also affect the environment, expanding the type of problem into a Markov game. Differences with Greenwald's results are attributed to that the method for decaying learning rate and ϵ (and initial values).

D. *Correlated-Q Learning (Utilitarian)*

The uCE-learning results are in Figure 4. Here, in contrast with Q-learning, the Q-values begin to converge as the delta approaches 0 in the limit. This is consistent with Greenwald's findings and illustrates the value of analyzing the equilibria. Since the environment is stochastic, there is no deterministic policy to learn. It is more effective to the learning process for both agents to keep track of the expected values of the other player's Q-table and take the recommended action from the observer than decide without accounting for the decision-making process. Although not quite matching with Greenwald's results, additional testing revealed artifacts of learning by the agents.

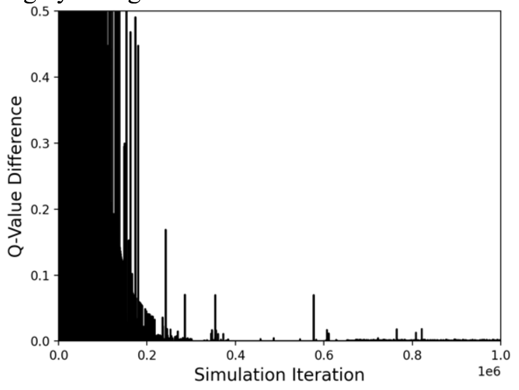


Figure 4: Utilitarian Correlated Q-learning (converges)

When playing A as uCE and B as a random action selector, over 1000 episodes player A won over 75% of the time. When roles were switched, player A won less than 25% of the time. This illustrates that player A using information about player B's state-action values benefits the player. The win percentage was not expected to be perfect, since a random player is still reasonable to win occasionally due to the small number of possible states (player A can't learn to navigate far out of the way of player B to avoid risk in colliding and losing possession). At first, it seemed that the agents should not benefit from the other player not cooperating (even if random), but, since it is a zero-sum game, there is no effective cooperation scenario, by logic, since there is always just one winner and loser. This effect might be different if there was a third goal that had a score of +125 for both players.

E. *Foe-Q Learning*

Foe-Q behaved similarly to CE-Q, showing that ultimately the former converges to a minimax solution, as the players learn how to maximize their worse possible situation with respect to the other player, since there is no proper cooperation policy. Results did not quite line up with Greenwald, so additional testing was performed. First, Q-table values and policies for various "near" terminal states were examined after training. Both players were put in obvious goal scoring opportunities and obvious own goal potential regions. In obvious goal scoring cases, both players' nondeterministic polices converged to a 100% on the move that would score them a goal.

In obvious "own goal avoidance" cases, both players would either take a 50% change of moving down or 50% change of staying put and faking the opponent out. One interesting scenario examined was when the players are arranged as in Figure 1, the player with the ball selected to move south or into the opponent at a 50% chance (taking a chance that the other player will move south), while the player without the ball selected to either stay put or move south at a 50% chance (guessing how to best block the other player's attack). Also, to compare the success of the agents when playing Foe-Q, tests were run to see what would happen if the ball were always to be given to one player at the beginning of each episode: this resulted in about a 70.0-win percentage by the possessing player, showing the advantage of starting off with the ball and continually learning how to better improve. This process of checking out specific scenarios increased confidence level in proper agent behavior in the limit.

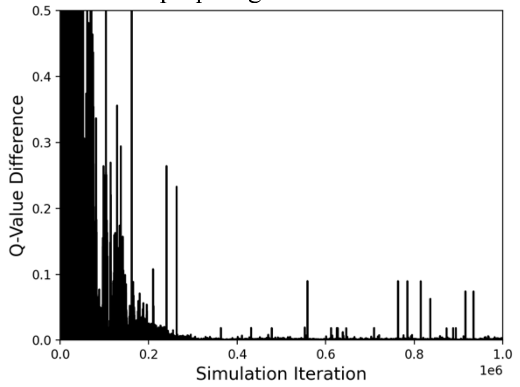


Figure 5: Foe-Q Learning (converges)

The results from Foe-Q show a significant result when seeing the similarities with CE-Q: The correlated problem essentially converged to a Foe-Q type learner because the game is zero-sum: there is no way for the two opponents to mutually benefit one another, so the correlated problem, the unilateral alternatives are those which maximize the minimum possible result with respect to the other player (2). Another key result was that in the limit, Q-table values for both Foe-Q and uCE-Q converged to be the opposite transpose of the other player, which is rightfully validates the expected values for a game with zero-sum (equal and opposite reward structure).

F. *Friend-Q Learning*

Friend-Q learning convergences very quickly, as shown in Figure 6. Convergence is not very recognizable on the scale of the plot shown to illustrate consistency with the other methods. Although not as like Greenwald's 2003 results, this does show the similar behavior traits of Greenwald's 2007 revisit of the soccer game problem [2]. The reason for quick convergence is that the player optimism (selecting the best overall valued action assuming the other agent will select the action to support that) results in moves that turn the ball over quickly, which most of the time will result in an obvious goal scoring opportunity for the benefiting agent to act on. There is no benefit for working with the other player in zero-sum, but in the third goal hypothesized game, there could be a mix.

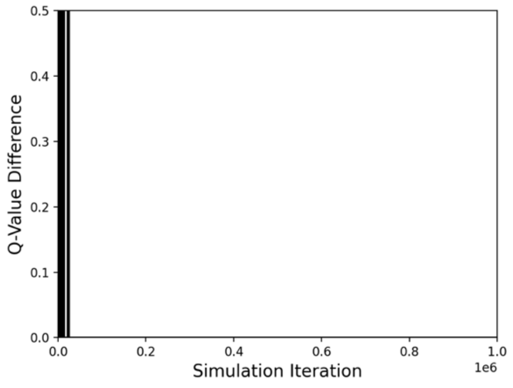


Figure 6: Friend-Q Learning (converges)

V. Conclusion

Although not identical to Greenwald's results, the value of uCE-Q was observed and shown to generalize Foe-Q well. The additional implication is made that possible variants of the soccer game might reflect Friend-Q, implying correlated equilibria can provide more flexible solutions to multi-agent learning problems that include Foe-Q, Friend-Q, and possibly a hybrid mix. The studies into playing the CE-Q agent against an agent taking random actions showed that in the limit the CE-Q agent would win up to 80% of the time, including the earlier episodes in which random exploration was occurring. This verified that effective learning was resulting in the multi-agent learning problem, yielding an earned value for the effort into replication of Greenwald's results and showing the power of correlated solutions for multi-agent learning problems [1].

References

[1] Amy Greenwald, Keith Hall, and Roberto Serrano. "Correlated Q-learning". In: ICML. Vol. 20. 1. 2003, p. 242. https://www.aaai.org/Papers/ICML/2003/ICML03-034.pdf.

[2] Amy Greenwald, Keith Hall, and Martin Zinkevich. "Correlated Q-learning". Journal Of Machine Learning Research. Vol 1. 2007. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.71.9825&rep=rep1&type=pdf

[3] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction.* Second Edition. MITpress. (2020). http://incompleteideas.net/book/the-book-2nd.html.