**Correlated-Q learning in Non-Deterministic Zero-Sum Games**

Adluri Surya Teja

Hash Code - 1e22fd6d89b757cfa388029b1916173468331dd2

**1 Introduction**

The purpose of this report is to replicate and analyze the experiments conducted in the paper "Correlated - Q Learning" (Greenwald, Hall, and Serrano 2003) [1]. The paper introduces an algorithm known as Correlated - Q (CE-Q) Learning for general-sum and constant-sum multi-agent scenarios. This algorithm extends the state of the art of Reinforcement Learning (RL) in multi-agent environments, where RL agents should adapt to the environment and also adapt to other agents' actions, be it coordination or opposition. Thus, CE-Q or any other Multi-agent RL algorithm extends to the single-agent RL algorithms like Q-Learning. Like the CE-Q algorithm, Littman's [2] friend-or-foe-Q (FFQ) algorithm was also developed for multi-agent scenarios. In the original paper [1], the authors attempt to demonstrate the performance of CE-Q learning by experimenting with the four algorithms – CE-Q, Foe-Q, Friend-Q, and Q-learning. The experiments are conducted on the testbed of a non-deterministic zero-sum game called soccer. The following few sections in this report detail the environment and explain each of the algorithms mentioned above, and later sections deal with the experiments and analysis of the results. The following section describes a few terminologies used in this paper.

* 1. **Definitions**

**Markov Game** framework formalizes multi-agent interaction, extending on the Markov decision process (MDP) idea for a single-agent framework. A Markov game consists of finite state space, and each state follows the Markov property. N agents take N actions at each state, and then the game transitions into the next state based on every agent's actions. Each state also associates rewards to each agent; when those rewards add up to zero or a constant value, it is called a **zero-sum game** or a **constant-sum game**, or when it adds up to different values based on different scenarios in the game, it is called a **general-sum game**. Each agent has a Policy that maps states to probability distributions over its actions. The ***value function*** is the same as in an MDP, a discounted sum of total rewards for each agent.

In single-agent games, an agent's optimal policy can be achieved by maximizing its value function. But, in multi-agent scenarios, maximizing the agent's value function may not lead to the optimal behavior of the agent. Hence, the agent must consider the actions of the opponent while figuring out his optimal policy. As a result, finding the optimal policy for each agent in a Markov game is not ideal. Hence, a concept known as **equilibrium** is designed to represent the optimal policy in the Markov game. Hu and Wellman [3] introduced a concept knows as **Nash equilibrium (NE),** a vector of independent probability distributions over action space concerning one another's probabilities. For example, in a classic rock-paper-scissors game, Nash equilibrium would be choosing 1/3rd times each action; this policy results in an optimal value for all the agents. A game can have more than one Nash equilibrium, and each equilibrium may produce different rewards for the players. Hu and Wellman proposed **Nash-Q learning** in the paper [3] to find the Nash equilibrium, but Nash-Q only converges in restrictive conditions. Also, a game can have a **pure or mixed strategy**. Pure NE has a deterministic policy, and mixed NE has a probabilistic policy like the example above.

**2 Algorithms**

As mentioned above, in this paper, four algorithms are implemented in a zero-sum game environment discussed in the next section. The algorithms use the Q-learning update rule to learn the optimal action values iteratively, as shown in equation (1). is a bootstrapped value that estimates the following state: this value function computation is different for four different algorithms, which is discussed below.

(1)

**2.1 Q – Learning**

Q learning is a core RL algorithm that is designed for single-agent reinforcement learning. In a multi-game setting, Q-Learning can be applied by not considering the opposite player's actions while computing the best policy of a player. In this project, Q learning is applied to the soccer game where each player uses Q learning to learn the optimal policy and takes actions epsilon-greedily from the learned policy. Q-learning update takes place according to equation (1), replacing bootstrapped with equation (2).

(2)

**2.2 Friend-or-Foe-Q (FF-Q) Learning**

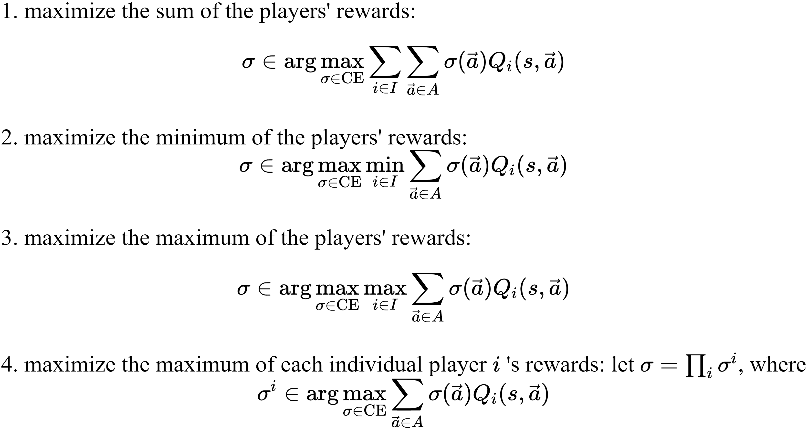
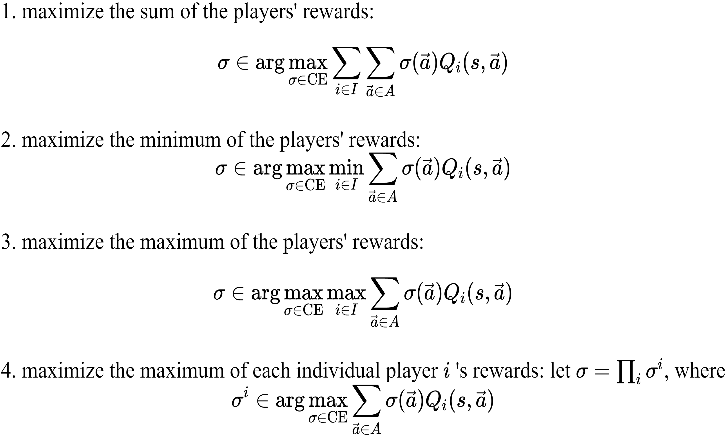
FF-Q learning is proposed by Michael L. Littman in his paper Friend-or-Foe Q-Learning in General-Sum-Games [2]. In the paper, Littman presents two algorithms: Friend-Q and Foe-Q, which work well when the other agents can be treated as friends or foes. These algorithms require that the agent be told whether he is facing a "friend' or a "foe." However, these algorithms perform well in general-sum conditions and offer more significant convergence guarantees than Nash-Q [3] algorithm. As mentioned in section 1.1, the Nash-Q algorithm only converges in restrictive assumptions because multiple equilibria exist in a game. For example, a game may have **coordination equilibrium** and **adversarial equilibrium**. When a general equilibrium learning algorithm like Nash-Q is used in these scenarios, the algorithm will learn different equilibriums at different iterations, resulting in non-convergence. If an algorithm is told what type of agent to expect, then the algorithm attempts to learn only that particular equilibrium and could be able to converge to optimal values. Using this idea, Littman proposed Friend-Q and Foe-Q as shown in equations (3) and (4), which are later experimented in a zero-sum soccer environment to analyze the result of CE-Q learning.

(3)

(4)

**2.3 CE-Q Learning**

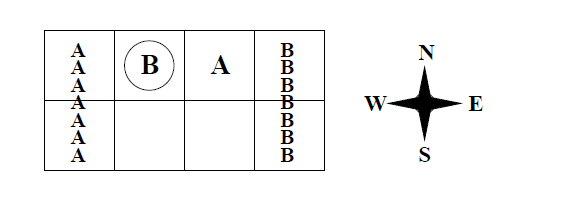
A **correlated equilibrium** (CE) is another type of equilibrium proposed in the original paper (Greenwald, Hall, and Serrano 2003) [1]. It is similar to NE, but it also allows dependencies among the agent's probability distributions while ensuring that the agents optimize their actions. This means each agent chooses its action according to the observation of the value of the same public signal, which accounts for dependency among the agent's actions. CE can be computed easily with linear programming since it is a convex optimization problem with rationality and probability constraints over the reward matrices. To find the correlated equilibrium, the authors introduced a new learning algorithm named CE-Q learning, which attempts to solve the problem of difficulty in learning equilibrium policies in Markov Games. As discussed in the previous section, challenges in learning occur mainly due to the presence of multiple types of equilibrium in a game. To effectively address the issue, authors [1] introduced four variants of CE-Q, each variant is based on the different equilibrium selection functions, and the variants are called utilitarian, egalitarian, republican, and libertarian CE-Q learning. These four variants ensure that the learning target is a unique equilibrium throughout the learning process. In this project, the first variant of CE-Q is applied in a soccer game environment, and the results are analyzed in the later sections. The below equations represent the four variants of CE-Q learning. All the equilibrium can be computed using linear programming by choosing one of the below objective functions and converting them into probability and rationality constraints. Equation 5.1 represents uCE-Q which is implemented in this project.



(5)

**3. Experiment**

**3.1 Zero-Sum Game – Soccer environment**

****In this project, a soccer environment is created as a testbed to test the four algorithms mentioned above – CEQ, Foe-Q, Friend-Q, and Q-learning. The soccer game is a zero-sum grid game, which consists of two players A and B, and the starting state of the game is initialized, as shown in figure 1.

A’s Goal (0,1) (0,2) B's Goal

Figure 1 Soccer Game with initial state and actions

The game starts with players B and A at positions (0,1) and (0,2). The circle around player B represents the possession of the ball, and the game starts with player B in control of the ball. Each player has five different actions to choose from, and a player can choose to go in one of the directions of N, S, E, W, or he can choose to stick in the current position. The first column represents A's goal, and the fourth column represents B's goal, and which player gets to move first is chosen randomly. For example, sometimes player B will move first, and other times player A will move first, and depending on which player moves second, the possession of the ball can be interchanged. For example, if player B has the ball and moves second and proceeds to A player's grid, A will get possession of the ball. However, the player who makes a move to steal the ball fails to do so. When a player with the ball moves to the goal, he scores a goal and gets the reward of +100, and his opponent gets the reward of -100. If any player scores own goal, then he gets -100, and the opponent gets +100.

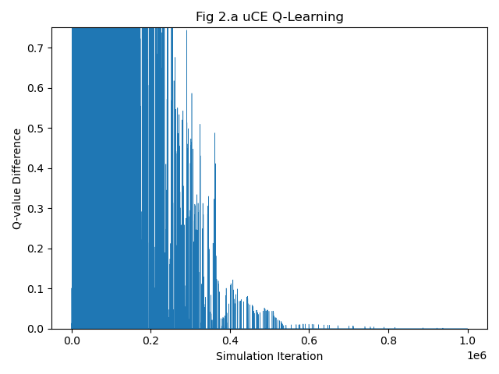
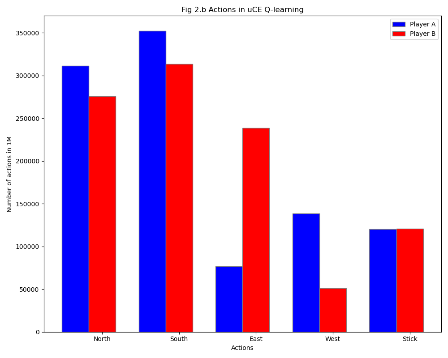
Furthermore, only non-deterministic equilibrium policies are optimal in this game because there is no deterministic action that enables a player to score. For example, A can block B continuously, and B cannot score, so B should find a non-deterministic policy like the rock paper scissor example mentioned in the introduction. An assumption about players leaving the grid area is made as no instruction is detailed in the paper [1]. Whenever a player chooses the action that leaves the grid, the player stays in the same cell, equivalent to the stick action. When the action values are tied for any state, players will choose the first action, which is going north; this detail is also assumed as it was not mentioned in the original paper [1].

**3.2 Implementation**

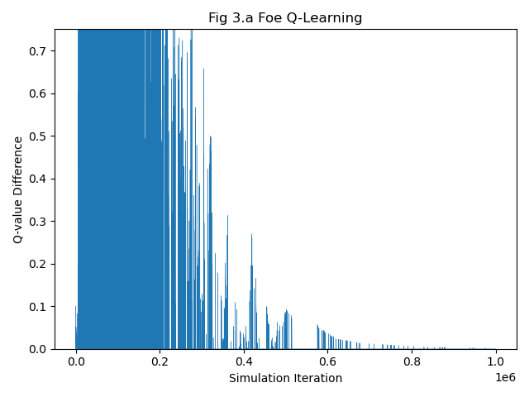
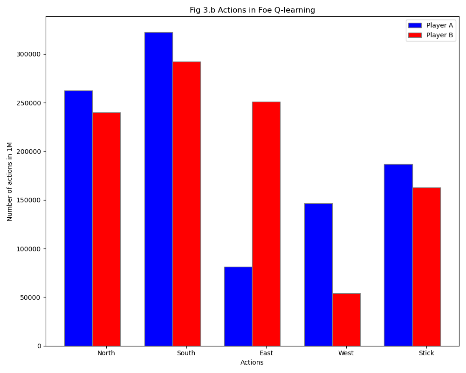
To analyze the performance of the CEQ-learning algorithm, four algorithms Q-Learning (2), Friend -Q (3), Foe-Q (4), and CEQ (5), are implemented to play the zero-sum soccer game. Among CEQ variants, uCEQ from equation 5.1 is chosen to participate in this experiment according to the original paper [1]. To show the convergence properties of the algorithms, a single Q value among all the (state, action) pairs is chosen, and the error is calculated using . The algorithms are then made to run for 1-million-time-steps; each time step constitutes an action taken by both the players. Error is then plotted against each algorithm's time steps, which helps visualize whether the Q-values for each algorithm converge. Following procedure is used for designing the experiments, first, a soccer environment is created as shown in figure 1, already prebuilt ML soccer library [4] is used for this purpose. Then to implement the four algorithms, the same procedure is followed as shown in equation (1), with equations (2), (3), (4), (5.1) are used to calculate the respective V functions.

In Q-learning, a player does not require to maintain the opposite player's actions or Q table as the player attempts to learn the optimal policy using its own (state, action) values. In contrast, player learning through friend-Q needs to consider opposing players' actions in his Q table, as the friend-Q chooses the action that maximizes the combined action space. Similarly, Foe-Q needs to maintain the opposite player's actions and Q-tables to perform minimax operation using linear programming, which is also the case in uCE-Q; the only change is that linear programming is performed such that the agent chooses to maximize the sum of the player's rewards.

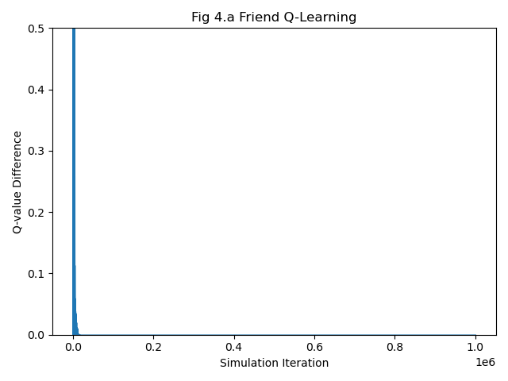
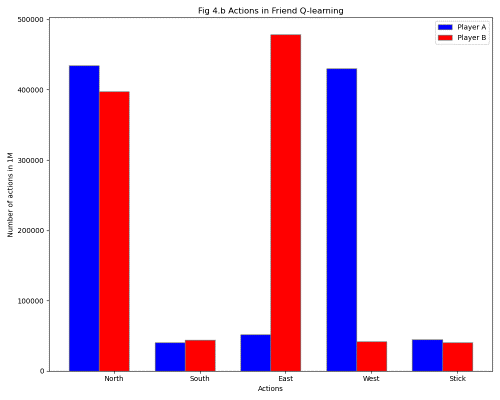
**4. Results and Analysis**

**4.1 uCE-Q Learning**

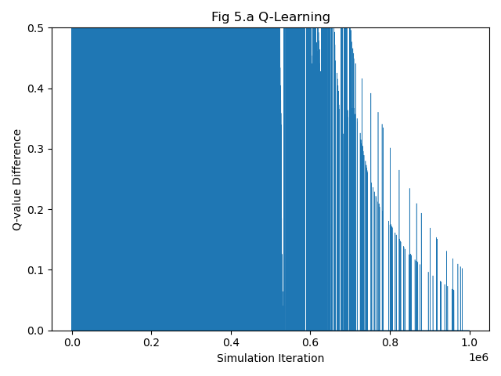
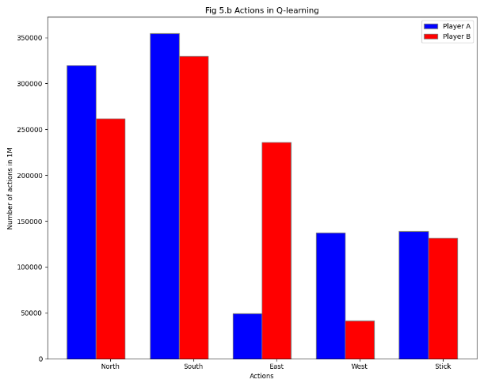
CE Q-learning is implemented using the initial learning and epsilon rates as 1.0 and decaying according to the decay schedule as mentioned in [5], where decay = . Like the original paper [1], the CE-Q algorithm is used to find the correlated equilibrium (CE) in the zero-sum soccer game. In addition to the experiment in the original paper, another extra experiment is conducted to find the policies to which these algorithms will converge. Accordingly, Fig 2.a represents the error values of the Q function for player A taking action south and player B sticking, whereas Fig 2.b represents different actions taken by the uCE-Q agent throughout the course of learning. The Q value function is converged as expected, which is shown in fig 4.a. The convergence of the uCE-Q process indicates that the CE-Q algorithm can successfully learn non-deterministic policies in a multi-agent environment. This is possible for a CE-Q algorithm because it keeps its equilibrium target fixed, while training, as characterized by the four variants in (5), here uCEQ is used, which marked the equilibrium target set according to equation 5.1. This claim can be further justified using figure 2.b, where both the actions of sticking and moving south were chosen by both the agents almost the same number of times. Hence, it is the correlated equilibrium of the soccer game. However, action North is also highly popular among the players that is because players are choosing this action whenever Q values in the table are equal, this is one of the assumptions of selecting an action with similar Q values, and it is just the expected behavior which is mentioned in the soccer environment section.

**4.2 Foe-Q Learning**

Foe-Q Learning algorithm has also experimented on the testbed of soccer environment with the same set of hyperparameters, and the results are almost identical to the CE-Q algorithm. Like the CE-Q algorithm, Foe-Q also converged, as shown in figure 3.a. Foe-Q offers strong convergence guarantees in this soccer setting because of the zero-sum nature of the game and the presence of adversarial equilibria in the game. The Foe-q learning algorithm will learn by treating the opponent as a foe and maximizing the worst-case scenario induced by the opponent. By initially deciding the opponent is an enemy, the algorithm was stuck to only one equilibrium target, which offered the stability to converge. However, exciting findings have been found from figure 3.b, where the policies both players converged to are identical to the policies converged using the uCEQ algorithm as shown in figure 2.b. Similar to figure 2.b in uCEQ, North action is also popular due to the initial states where Q values of all the possible states are not populated enough; by ignoring that, the following most equally chosen actions are moving south and sticking. In the paper, it is also mentioned that both the uCEQ and Foe-Q algorithms will converge to the policy of randomizing between taking action south and action stick, but no empirical evidence is given to prove that point, however, using figures 2.b and 3.b, it can be verified. This result also shows that the uCEQ algorithm can act as a Foe in zero-sum or constant-sum games and finds the adversarial or minimax equilibrium.

**4.3 Friend Q-Learning**

Friend Q-learning is performed using the same epsilon rate as 1.0, but the initial learning rate is taken as 0.2 to replicate the figure 3.C (here figure 4.a) from the paper [1], as the higher learning rate may cause the agents to directly learn a policy, reducing the absolute error in one step. Both learning rate and epsilon are decayed according to the same decay schedule mentioned in [5]. Similar to previous algorithms, two experiments were conducted. Fig 4.a represents the error values of player A acting South and player B sticking, whereas Fig 4.b represents different actions taken by the Friend-Q agent through the course of learning. Although Fig 4.a demonstrates the convergence of friend-Q learning, the algorithm will not converge to optimal values and may even converge to worst-case policy, which is what happened in the case of the soccer game. If Fig 4.b is observed, as identical with every algorithm, players took north action often as the argmax function returns 0 in case of ties. But barring that north action, player B always moved east, and player A always moved west; they both are giving each other possession of the ball expecting his friend (opponent in real) will score for him. This process happened endlessly, and the algorithm converged to this worst-case policy at the end. This phenomenon is due to the presence of the adversarial equilibrium in the zero-sum soccer game, whereas friend-Q is trying to find the coordination equilibrium.

**4.4 Q-Learning**

Q-learning is also performed using the same initial learning and epsilon rates as 1.0 and decaying according to the same decay schedule. Q-learning did not converge, which is expected as shown in figure 5.a. Although in the graph Fig 5.a, at the end of the training, it looks like the values are converging, this low error value is due to decreasing learning rate. The q-learning is not expected to converge in the non-deterministic soccer game scenario because q-learning tries to find the deterministic actions for its players to maximize their reward, i.e., to score a goal. But as mentioned in the previous sections, this game requires the players to have a non-deterministic policy as the opposing player can block any deterministic action taken by the player. This scenario is observed in fig 5.b, where players chose the south action mostly as they are trying to move to the south to escape the opposing player. Depending on who gets to move first determines who scores more goals; in this case, player B could have scored more goals as he has moved to East many times compared to Player A moving to West. In any case, this algorithm did not yield an equilibrium policy as the players cannot converge to optimal actions that can be interpreted from fig 5.b. It is also evident that Q-learning cannot converge and could not minimize the error for the player action of moving south from the initial state, as shown in fig 5.a.

**5 Pitfalls and Assumptions**

The major challenge faced while implementing the experiments was the lack of clarity of the decay schedule of alpha and epsilon; thus, identical graphs from the original paper could not be replicated. However, following the decay schedule mentioned by Littman [5] was enough to prove the end goals of the experiment. Also, the graph in the original paper [1] precisely demonstrates the similarities between Foe-Q and uCE-Q. However, lack of clarity on parameters leads to slightly different charts 2.a and 3.a. However, to strongly prove the similarities between them, another experiment was designed (2.b and 3.b) and analyzed in the previous sections. Another assumption made in the experiment is about the discount rate (gamma); in this project discount rate of 0.9 is used throughout, which is chosen by manual exploration. The 0.9 value can also be justified theoretically, as the players need to be maximizing the immediate rewards as each iteration of the game will end in just a few steps. Apart from the above, several other assumptions have to be made about the soccer environment that lacked clarity, detailed in section 3.1.

**6 Conclusion**

The experiments conducted in this project prove the adaptability of CE-Q algorithms in constant-sum games; similarly, CE-Q algorithms also offer convergence guarantees in general-sum games, which is discussed in the original paper [1]. This project proves that in constant-sum games, the set of correlated equilibria is same as the minimax equilibria, which can be observed from the similarities between Foe-Q and uCE-Q algorithms from the section 4.1 and 4.2. This report also proves and analyzes that following a unique equilibrium throughout the learning process will offer stronger convergence guarantees.

**7 References**

[1] Amy Greenwald, Keith Hall, and Roberto Serrano. "Correlated Q-learning." In: ICML. Vol. 20. 1.2003, p. 242.

[2] M. Littman. Friend or foe Q-learning in general-sum Markov games. In Proceedings of Eighteenth International Conference on Machine Learning, pages 322– 328, June 2001.

[3] J. Hu and M. Wellman. Multi-agent reinforcement learning: Theoretical framework and an algorithm. In Proceedings of the Fifteenth International Conference on Machine Learning, pages 242–250, July 1998.

[4] Prebuilt Soccer environment - https://github.com/pdvelez/ml\_soccer.

[5] Markov games as a framework for multi-agent reinforcement learning – Michael L. Littman - 1994.