Correlated Q

Zehan Li[1, 2]
[1]*OMSA, Georgia Institute of Technology, North Avenue, Atlanta, GA 30332, USA*
[2]*Department of Physics and Astronomy, University of Pittsburgh, Pittsburgh, PA 15260, USA*
(Dated: April 13, 2020)

I. INTRODUCTION

The Nash equilibrium is to derive a set of strategy for the players to reach the equilibrium state by using the linear programming. We obtain the equilibrium conditions by solving objective functions with a set of LP constraints. Here, we find the equilibrium of the environment of the soccer game by using four different alogirthms: Correlated-Q, Foe-Q, Friend-Q and Q-learning to reproduce the Figure 3 of Greenwald's paper.

II. GAME THEORIES

A. Correlated equilibrium

Nash equilibrium is a fundamental concept in the theory of games and the most widely used method of predicting the outcome of a strategic interaction in the social sci- ences. A game (in strategic or normal form) consists of the following three elements: a set of players, a set of actions (or pure-strategies) available to each player, and a payoff (or utility) function for each player. The payoff functions represent each player's preferences over action profiles, where an action profile is simply a list of actions, one for each player. A pure-strategy Nash equilibrium is an action profile with the property that no single player can obtain a higher payoff by deviating unilaterally from this profile.

From the paper, the key feature of Nash equilibrium is a vector of independent probability distributions over actions, in which all agents optimize with respect to one another's probabilities. Different from that, a correlated equilibrium (CE) is more general than a NE, since it permits dependencies among the agents' probability distributions, while maintaining the property that agents are optimizing. An everyday example of a correlated equilibrium is a traffic signal. For two agents that meet at an intersection, the traffic signal translates into the joint probability distribution (stop,go) with probability 0.5 and (go,stop) with probability 0.5. No probability mass is assigned to (go,go) or (stop,stop). Besides, another property that makes correlated equilibrium stand out is that Linear Programming (LP) is efficient in solving for the correlated equilibrium. Unlike Nash equilibrium where no efficient method of computation is known, correlated equilibrium can be formulated as a set of constraints and an objective function for an LP solver to solve.

B. Markov games

The Greenwald's paper states that stochastic games generalize repeated games and Markov decision processes (MDPs). A stochastic game is a tuple $< I, S, (A_i(s))_{s\in S, 1\leq i\leq n}, P, (R_i)_{1\leq i\leq n} >$, where I is a set of n players, S is a set of states, $A_i(s)$ is the i-th player's set of actions at state s, P is a probability transition function that describes state transitions, conditioned on past states and joint actions, and $R_i(s, \vec{a})$ is the ith player's reward for state $s \in S$ and joint actions $\vec{a} \in A(s) = A_1(s) \times \ldots \times A_n(s)$. Stochastic games for which the probability transitions satisfy the Markov property are called Markov games: i.e., for $\vec{a}_t = (a_1, \ldots, a_n)_t, P[s_{t+1}|s_t, \vec{a}_t, \ldots, s_0, \vec{a}_0] = P[s_{t+1}|s_t, \vec{a}_t]$. The Q-value of the Markov games is defined as

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

In that paper, Greenwald proposed an alternative definition of the value function in Markov games:

$$V_i(s) \in CE_i(Q_1(s), \ldots, Q_n(s)), \quad (2)$$

where $CE_i(X_1, \ldots, X_n)$ denotes the i-th player's reward according to some correlated equilibrium in the general-sum game determined by the rewards $X_1, \ldots, X_n$.

III. MULTIAGENT Q-LEARNING

In MDPs, the special case of Markov games with only a single agent, the corresponding local update procedure, known as value iteration, is well understood: Given Q-values at time t for all $s \in S$ and for all $a \in A(s)$, namely $Q_t(s, a)$, at time $t + 1$,

$$V^{t+1}(s) := max_{a\in A(s)} \pi_s^t(a) Q^t(s, a), \quad (3)$$

$$Q^{t+1}(s, a) := (1-\gamma)R(s, a) + \gamma \sum_{s'\in S} P[s'|s, a] V^{t+1}(s'). \quad (4)$$

More generally, in Markov games, given Q-values at time t for all $i \in N$, for all $s \in S$, and for all $a \in A(s)$, namely $Q_i^t(s, a)$; given a policy π^t; and given a selection mechanism f, that is, a mapping from one-shot games into (sets of) joint distributions; at time $t+1$,

$$V_i^{t+1}(s) := max_{a \in A(s)} \pi_s^t(a) Q_i^t(s,a), \tag{5}$$

$$Q_i^{t+1}(s,a) := (1-\gamma) R_i(s,a) + \gamma \sum_{s' \in S} P[s'|s,a] V_i^{t+1}(s'), \tag{6}$$

$$\pi_s^{t+1} \in f(Q^{t+1}(s)). \tag{7}$$

The algorithm of the multiagent Q-Learning is listed below,

Algorithm 1 Multiagent Q-Learning

```
1: Input the selection function $f$, discount factor $\gamma$, learning rate $\alpha$, decay schedule $S$, and total training time $T$.
2: Initialize $i = 0$, $Q, V, \pi$ and error list.
3: while $i < T$ do
4:     while True do
5:         Generate new actions for two players using $\epsilon$-greedy
6:         $i++$
7:         Perform the new actions and get the new states, rewards, and done
8:         if done then
9:             Calculate $|Q_i^t(s,a) - Q_i^{t-1}(s,a)|$ and error list. Break and start a new game.
10:        else
11:            Calculate new $Q$ and solve for $V$ using function $f$ and linear programming
12:            Decay $\alpha$
```

IV. SOCCER GAMES ENVIRONMENT

The soccer field is a grid. There are two players, whose possible actions are N, S, E, W, and stick. Players choose their actions simultaneously. Actions are executed in random order. If the sequence of actions causes the players to collide, then only the first player moves, and only if the cell into which he is moving is unoccupied. If the player with the ball attempts to move into the player without the ball, then the ball changes possession; however, the player without the ball cannot steal the ball by attempting to move into the player with the ball. Finally, if the player with the ball moves into a goal, then he scores +100 if it is in fact his own goal and the other player scores -100, or he scores -100 if it is the other player's goal and the other player scores +100. In either case, the game ends.

There are no explicit stochastic state transitions in this game's specification. However, there are "implicit" stochastic state transitions, resulting from the fact that the players actions are executed in random order. From each state, there are transitions to (at most) two subsequent states, each with probability 1/2. These subsequent states are: the state that arises when player A (B) moves first and player B (A) moves second.

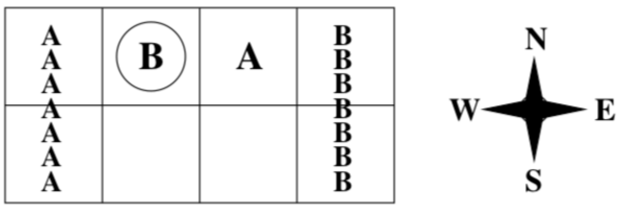

Figure 1: Soccer Game. The circle represents the ball. If player A moves W, he loses the ball to player B; but if player B moves E, attempting to steal the ball, he cannot.

Unlike in the grid games, in this simple soccer game, there do not exist pure stationary equilibrium policies, since at certain states there do not exist pure strategy equilibria. For example, at the state depicted in Fig. 1, any pure policy for player A is subject to indefinite blocking by player B; but if player A employs a mixed policy, then player A can hope to pass player B on his next move. We list the algorithm for the soccer environment below.

Algorithm 2 Soccer Game

```
1: System Initialization with current coordinates of both players, ball possession, new action selections from player A and B and scores
2: Randomly decide which player moves first, and start moving the first player
3: if The first moving player is overlapping with the rest one after moving then
4:     Change the ball possession if the ball belonged to the first player before the collision
5: else if The first player is not on the edge then
6:     Move the first player to the new coordinate
7:     if First player scores then
8:         Update the scores and done
9: Start moving the second player
10: if The second moving player is overlapping with the rest one after moving then
11:     Change the ball possession if the ball belonged to the second player before the collision
12: else if The second player is not on the edge then
13:     Move the second player to the new coordinate
14:     if Second player scores then
15:         Update the scores and done
16: Return the states, scores and done
```

V. EXPERIMENTS

A. Q-Learning

The Q-learning follows the standard structure of the off-policy Q-table update. We construct Q_A and Q_B for the two players respectively, where the dimensions are (8, 8, 2, 5). The first two dimensions represent the 8 possible positions for player A and B. The third dimension is the indicator of ball possession, and the last dimension is the

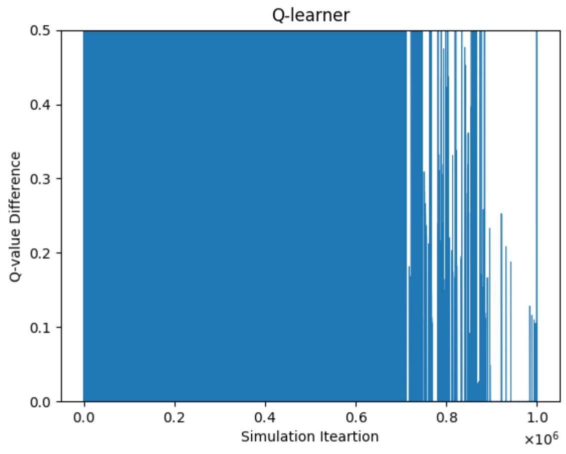

Figure 2: Convergence in the soccer game using Q-learning algorithm.

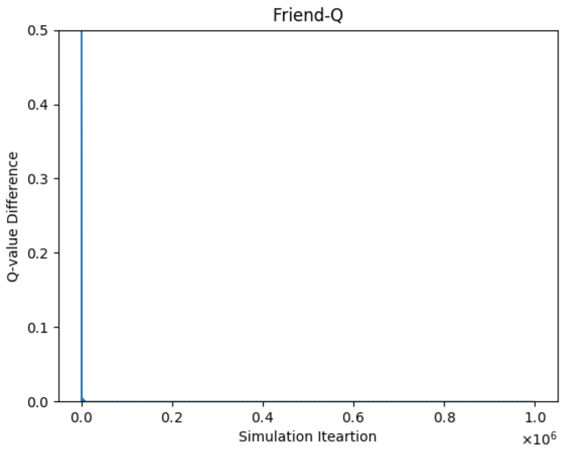

Figure 3: Convergence in the soccer game using friend-Q algorithm.

5 legal actions for the player. The hyper parameters are set up below,

learning rate $\alpha = 1.0$;
$\epsilon = 1.0$;
decay factor $\epsilon_{dec} = 0.999995$;
discount factor $\gamma = 0.9$;
alpha decay $\alpha_{dec} = 0.999995$;
max training episodes number $= 10^6$

The experiment is shown in Fig. 2. The Q-value difference is decreasing as more iterations are implemented. However, it does not imply the convergence to 0. Since the learning rate α is decreasing to 0.001, it is very likely that the decrement of Q-value difference is due to the decreasing of α. As the paper points out, Q-learners compute Q-values for each of their own possible actions, ignoring their opponents' actions. At all times, the amplitude of the oscillations in error values is as great as the envelope of the learning rate. In the game theory, if the agent does not learn from the opponent, it learns from a noisy dynamic environment hence convergence cannot be guaranteed. Intuitively, the rationale for this outcome is clear: Q-learning seeks deterministic optimal policies, but in this game no such policies exist.

The graph can capture the main feature of the figure in Greenwald's paper, although there does not show an "envelope-like" decrement. Since Greenwald did not provide the hyper parameters he used and the seeds of the random settings, we can adjust our decay factors to tune the shape of the plot and make it more similar to the original graph.

B. Friend-Q

The implementation of friend-Q algorithm is that each agent looks for the pair of actions (one for each player) which has the highest estimated value. Effectively, it is assuming that the other agent will act altruistically towards it. In this implementation, the Q-table has dimension of (8,8,2,5,5) where the last two dimensions represent the action space of the opponent and the player. The hyper parameters are set up below,

learning rate $\alpha = 1.0$;
$\epsilon = 1.0$;
decay factor $\epsilon_{dec} = 0.999995$;
discount factor $\gamma = 0.9$;
max training episodes number $= 10^6$

The experiment is shown in Fig. 3. The Q-value difference is decreasing very rapidly compared with the Q-learning algorithm, which shows a fast convergence. As stated in the paper, learning according to friend-Q, player B (fallaciously) anticipates the following sequence of events: player A sticks at state s, and then player A takes action E. Thus, by taking action E, player B passes the ball to player A, with the intent that player A score for him. Player A is in-different among her actions, since she assumes player B plans to score a goal for her immediately.

The graph is very close to the shape in Greenwald's paper. However, different settings of the decay factor of the parameters could result in different convergence speed of the Q-value difference. Here we set the decay on the learning rate as $\alpha \propto 1/i$, where i is the iteration step.

C. Foe-Q

Foe-Q is a minimax strategy; an agent is trying to maximize its reward given the assumption that other agents are working to minimize it. The structure of minimax (maxmin) and can be solved through linear programming (LP). The experiment is shown in Fig. 4. The Q-value difference is decreasing as more iterations are

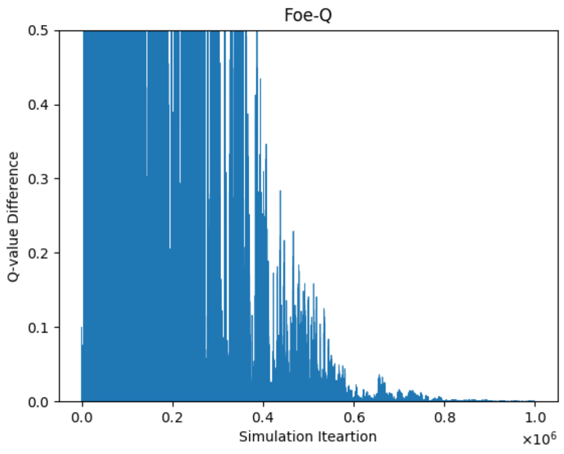

Figure 4: Convergence in the soccer game using foe-Q algorithm.

implemented, and it is convergent to 0, where the convergence property of minimax-Q and is guaranteed to converge. The hyper parameters are set up below,

learning rate $\alpha = 1.0$;
decay factor $\epsilon_{dec} = 0.999993$;
discount factor $\gamma = 0.9$;
alpha decay $\alpha_{dec} = 0.999993$;
max training episodes number $= 10^6$

The graph is consistent with the figure 3(b) in Greenwald's paper, and converges at about 6×10^5 steps. The slight difference between the two plots could be due to the initialization of the parameters. As the maxmin (Foe-Q) algorithm requires an LP solver to calculate value function and probability distribution over actions, different solver used might contribute to such difference.

D. Correlated-Q

As stated in the paper, the difficulty in learning equilibria in Markov games stems from the equilibrium selection problem: how can multiple agents select among multiple equilibria? Four variants of correlated-Q learning are introduced: utilitarian (uCE-Q), egalitarian(eCE-Q), republican(rCE-Q) and libertarian (lCE-Q), based on four correlated equilibrium selection mechanisms. All these equilibria can be computed via linear programming by incorporating the objective function of choice into the linear programming formulation. The experiment is shown in Fig. 5. The Q-value difference is decreasing as more iterations are implemented, and it is convergent to 0. CE-Q learns the same set of Q-values as Foe-Q, and since Foe-Q is the minimax equilibrium (maxmin), Greewald also draws the conclusion that CE-Q learns minimax equilibrium policies in the two-player, zero-sum game. The hyper parameters are set up below,

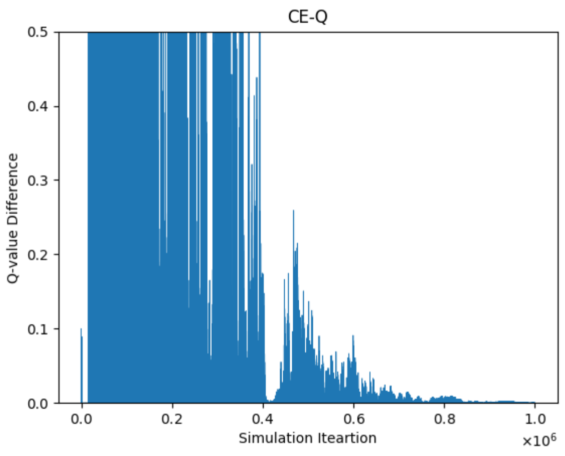

Figure 5: Convergence in the soccer game using foe-Q algorithm.

learning rate $\alpha = 1.0$;
decay factor $\epsilon_{dec} = 0.999993$;
discount factor $\gamma = 0.9$;
alpha decay $\alpha_{dec} = 0.999993$;
max training episodes number $= 10^6$

The graph is consistent with the figure 3(a) in Greenwald's paper, and converges at about 8×10^5 steps. Similar to the case in foe-Q algorithm, the slight difference between the two plots could be due to the initialization of the parameters and different solver used.

VI. CONCLUSION

In this work, we discuss the game theories, correlated equilibrium and Markov games. As an illustration, we build the soccer environment, which is a zero-sum game for which there do not exist deterministic equilibrium policies. We discuss the general framework of multiagent Q-learning and the formulations of four implemented Q-learning algorithms: Q-Learning, Friend-Q, Foe-Q and CE-Q, and reproduce the figures in Greenwald's paper. The detailed analysis is also provided to explain the differences among the figures in this paper and the ones in Greenwald's paper.