Correlated Q-Learning

Joseph Su

Georgia Institute of Technology

January 6, 2019

Abstract

In this paper we describe and replicate the work in [1]. Specifically we apply several variants of Q-learning techniques to validate convergence to equilibria on a two-player, zero-sum soccer game, a Markov decision process. We note the result differences, and explore and substantiate the reasons behind such deviation and how we arrive at the same empirical conclusion on several others. We make use of related literature in [3] and [4] as reference to commence our work.

I. Introduction

In many multi-agent problems the agents must take random actions to explore their environment before knowing whether there are rewards, making these problems a Markov decision problem. Q-learning, a specific instance of learning techniques in reinforcement learning, is useful to solving these problems. Q-learning assumes that the agent inhabits in a Markov process to stochastically optimize its future optimal rewards by taking actions in certain states. There are mixed policies hence equilibria in these problems. Some equilibria may be preferred by some players while other equilibria shunned upon by others. This dynamic is akin to the level of selfishness versus generosity in players - our learning goal is to ensure all players converge to the same strategy leading to a successful gameplay - hence the goal of Q-learning, and that of identifying optimal strategies to necessitate coordination in a multi-agent game.

II. A Markov Game: Soccer

The Markov game soccer described in [1] is a two-player, zero-sum game that does not reach any deterministic equilibrium. In this game we have our state space, S, action space, A, and reward space, R. The collision and ball possession transfer dynamics are described in [1]. In the same literature, Fig. 3a-d reflect Player A's means of error distribution with A taking action S and Player B sticking initially.

III. Q-Learning

Q-learning is a model-free technique in reinforcement learning that focuses on finding optimal policy in any MDP. [1] explored ϵ-greedy variant of Q-learning leading to an on-policy exploitation of optimal policy. Of the many illustrated Q-learning variants in [1], our paper will focus on Friend-or-Foe-Q (FF-Q), Correlated-Equilibrium-Q (CE-Q), and the general Q-learning by applying them to the aforementioned stochastic soccer game.

i. Assumptions

We begin our work making assumptions on the learning rate, α, and exploration rate, ϵ, both of which were mentioned in [1] but without sufficient details. Firstly, learning rate α determines how fast or slow an agent moves towards the future optimal rewards, which affect the rate of learning. ϵ controls how fast or slow we converge to a solution. There are various means to anneal learning rate to decay it exponentially [2]. Good intuition suggests either a step, exponential, or $1/T$ decay would be suitable to mimic the decay patterns in [1]. Through trial and errors, the closest decay function was found to be $\alpha = \frac{\alpha_0}{1+kT}$ where α_0 and k are hyperparameters and T is the iteration. The same decay pattern is used for ϵ as well. Lastly we use $\gamma = 0.9$ as Littman did in [3] to compute future rewards.

IV. Methodology

The soccer game is stochastic consisting of a tuple $(I, S, (A_i(s))_{s\in S, 1\leq i\leq n}, P, (R_i)_{1\leq i\leq n})$ [1]. I is a set of n players, S is a set of states, $A_i(s)$ is the ith player's action sets at s, P is our state transition function based on past states and joint actions, and R_i is our reward for the ith player at $s \in S$. Each player can be considered an MDP. In a multi-agent Markov game we have

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

where the joint actions as defined in [1] as $\vec{a} \in A(s) = A_1(s) \times \ldots \times A_n(s)$.

The state space is computed combinatorially via $|S| = |I| \times n \times (n-1)$ where I is the number of agents hence of ball possessions, and n is the grid size. For a two-player zero-sum soccer game, we have $|S| = |I| \times P_2^n = 2 \times P_2^8 = 112$. The action set A is similarly computed as $|I| \times \sum_{a\in A} a$, where $A = \{N, W, E, S, sticking\}$. This evaluates to $|A| = 25$.

i. Game Dynamics

We define our goal states, G, where red indicates goal for the first player, and blue for the second player, as seen in Fig. 1 $G_{first} = \{0, 4\}$ and $G_{second} = \{3, 7\}$. Using the same

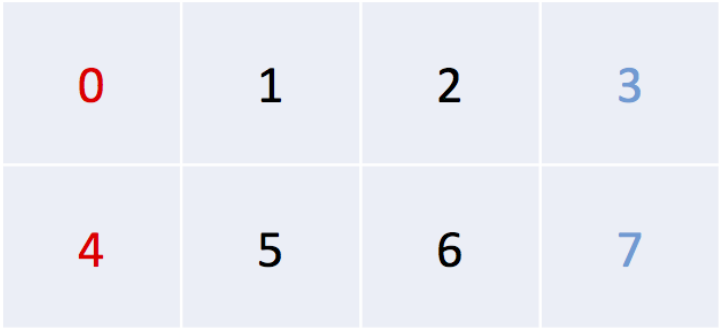

Figure 1: *Positional indices for Player A and Player B in the soccer grid*

indexing system, we assign indices to our actions: sticking means inactive which means 0 movement, $E = 1$, $W = -1$, $N = -4$, and $S = 4$. For example, a player at position 2 moving to S receives a +4 to the new position at $2 + 4 = 6$. Additionally we impose boundary constraint allowing only moves within the grid; no off-grid move is allowed. Collision is detected when both players move into the same position on grid. The transition probability distribution accounts for 50% of either player taking the first action to collide with another. According to [1] when collision occurs only the first player moves, assuming his next move is valid; this implies the second player sticks on collision. However, when the first player collides with the second player who is sticking, both players stick after the collision. Lastly during collision, the ball changes possession only when the second player has the ball. Given the above prescribed dynamics, the transition probability distribution $P[s'|s, \vec{a}]$ can be computed over the entire $S \times A$ space.

ii. Reward Space

We extract reward tuple (r_A, r_B) over the entire S based on the scoring system in [1]. For example, when Player A is in possession of the ball landing on either 0 or 4, a score of 100 is awarded to Player A, and -100 to Player B. Thus the entire $R_i(s, \vec{a})$ for each ith player's reward for state $s \in S$ is defined. As stated $|R| = |S|$. For example, at s_0 we have $(0, 0)$.

iii. Q-Value iterations

Recall $Q : S \times A \rightarrow \mathbb{R}$ where $s \in S$ and $a \in A$. We choose s_0 based on our defined grid system. In Fig. 2, Player B is at index 1 and Player A at index 2. According to [1], a_0 for Player A receives a 4 (going S) and a_0 for B receives a 0 (sticking).

Q-learning commences after the state initializations. During each learning episode, the agent first chooses an action on ϵ-policy, and observes the new reward and state to arrive at its learned value. The core of Q-learning algorithm is the value iteration to update Q continuously. Learning ensues until either player scores the goal - at which point the episode is completed and the next is to begin. This process iterates until convergence or when the total training time T is over.

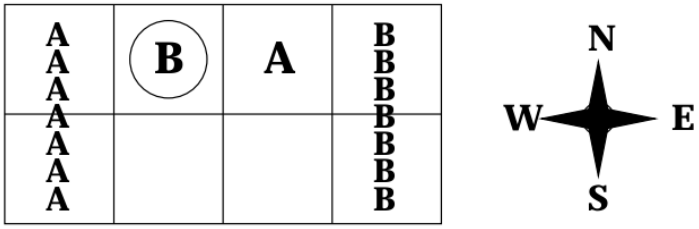

Figure 2: *Soccer game initialization state in [1]*

iv. ERR Measurement

The moving means of distribution of errors (ERR) are logged at the end of each Q-learning iteration. That is, ERR = $|Q_i^t(s, \vec{a}) - Q_i^{t-1}(s, \vec{a})|$, over the span of 1MM iterations for most of our replication efforts. In [1], ERR is also termed Q-Value Difference in charts.

V. Results

Fig. 3 depicts the outcome of Q-learning after 1MM iterations. In comparing with [1], our result is closely on par however both the amplitude and latitude are slightly off. This could be due to how our chosen learning rate was annealed and the ϵ-policy used. Overall Fig. 3 shows a tighter mesh than [1], insinuating more perturbation hence divergence than the published result.

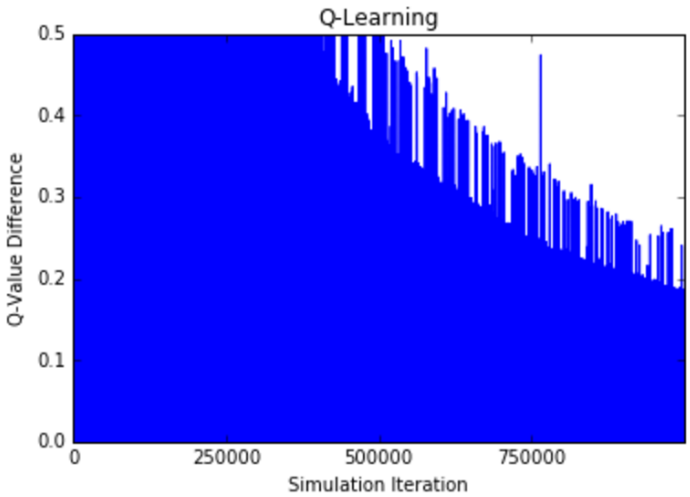

Figure 3: *Q-Learning outcomes after 1,000,000 iterations.*

VI. Friend-or-Foe-Q

According to [4], for a two-player soccer game the Q update is performed with

$$Q_i(s, a_1, \ldots, a_n) := (1 - \alpha_t) Q_i[(s, a_1, \ldots, a_n)] + \alpha_t (r_i + \gamma Nash_i(s, Q_1, \ldots, Q_n)) \quad (2)$$

If the opponent is a friend we have, in the combined action space of two players:

$$Nash_1(s, Q_1, Q_2) = \max_{a_1 \in A_1, a_2 \in A_2} Q_i[s, a_1, a_2] \quad (3)$$

otherwise if the opponent is a foe we have a minimax-Q:

$$Nash_1(s, Q_1, Q_2) = \max_{\pi \in \prod(A_1)} \min_{a_2 \in A_2} \sum_{a_1 \in A_1} \pi(a_1) Q_i[s, a_1, a_2] \quad (4)$$

Eq. 4 can be solved with a linear program. In FF-Q the agent knows whether his opponent is a "friend" or "foe". In the former case we look for a coordination equilibrium, and in the latter, an adversarial equilibrium. For this experiment we only need to maintain a joint Q table. In the "friend" case both players work together to maximize one player's value.

Lastly, FF-Q has several advantages over Nash in that FF-Q takes place independently of other agents' actions; FF-Q does not require the learning of Q functions for other players, which can be intensive computationally, and the outcome can be computed agilely by solving a system of linear equations.

i. Friend-Q

The Friend-Q algorithm is similar to that of Q-learning's, as discussed in [4] with the difference being Eq. 3, which converts the Q algorithm into the following learning path:

$$Q_i^{t+1}(s, a_1, \ldots, a_n) \leftarrow (1 - \alpha_t) Q_i[(s, a_1, \ldots, a_n)] + \alpha_t (r_i + \gamma Nash_i(Q_t(s'))) \quad (5)$$

Fig. 4 shows Friend-Q convergence in under 100,000 iterations. The protracted view of the same Friend-Q for 1MM iterations looks similar to the published result [1]. Note Friend-Q provides no guarantees to learned values; this is evident in the early iterations where ERR » 0.5. However after sufficient iterations Friend-Q converges, a sign of coordination taking over the significance of non-compatibility [4].

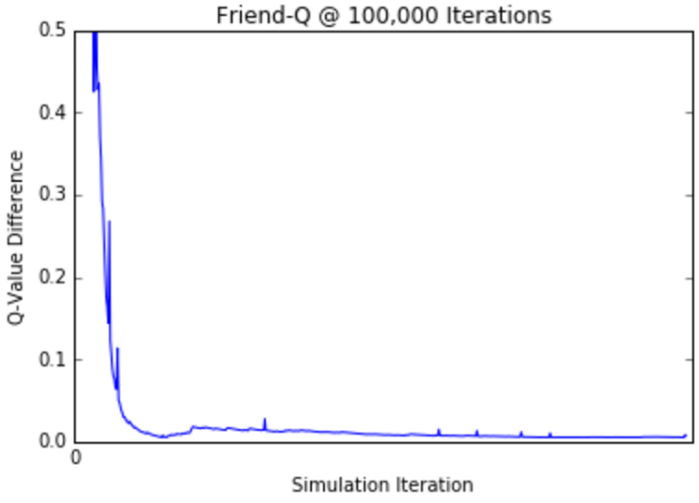

Figure 4: *Friend-Q @ 100,000 Iterations*

ii. Foe-Q

Fig. 5 depicts the result of Foe-Q after 1MM iterations, which show convergence. This graph shows some deviations from the published result [1]. In short Foe-Q uses a mixed strategy involving minimax-Q over the players' action space, solvable by linear programming. The most visible difference is the absence of higher-elevation fluctuations in early iterations. This connotes to two things: (1). ERR ≈ 0 in early iterations. This means either coordination or misfires from the foe whose action inadvertently helps his opponent in maximizing his expected reward. (2). The π space for both players are the same and non-zero as many spots in early iterations. This leads to uniform Vs hence ERR ≈ 0. It is assumed that our minmax-Q setup, rather than our LP solver, is the culprit for skewing the trend and the two aforementioned issues, as our result for μCE-Q in Fig. 6 shows no problem. This problem may be resolved with a few tweaks in the followup work. Overall, Foe-Q converges as expected at iterations bypassing 500,000, at an annealing rate of $\alpha = \frac{\alpha_0}{1+kt}$, where $k = 0.01$ and $\alpha_0 = 1$.

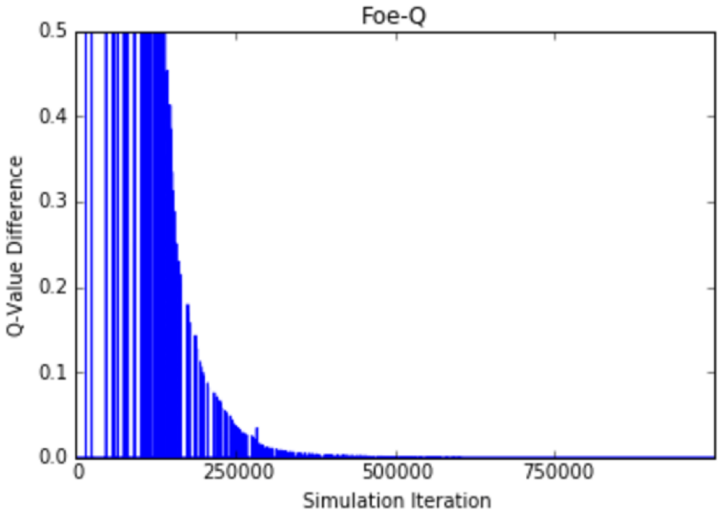

Figure 5: *Foe-Q after 1,000,000 Iterations*

VII. Correlated-Q

We explore the *utilitarian* variant of CE-Q as in [1]. This algorithm is based on the Correlated Equilibrium solution to maximize the player rewards:

$$CE_i(\vec{Q}(s)) = \{ \sum_{\vec{a} \in A} \sigma(\vec{a}) Q_i(s, \vec{a}) \} \quad (6)$$

$$\sigma \in argmax_{\sigma \in CE} \sum_{i \in I} \sum_{\vec{a} \in A} \sigma(\vec{a}) Q_i(s, \vec{a}) \quad (7)$$

Fig. 6 depicts the result solved using Python's PuLP library. We apply μCE-Q with over 1,000,000 iterations. Our result resembles [1] but again differs on amplitude and latitude. This is most likely due to our chosen learning rate and ϵ adoption, which may be quite different from the literature's.

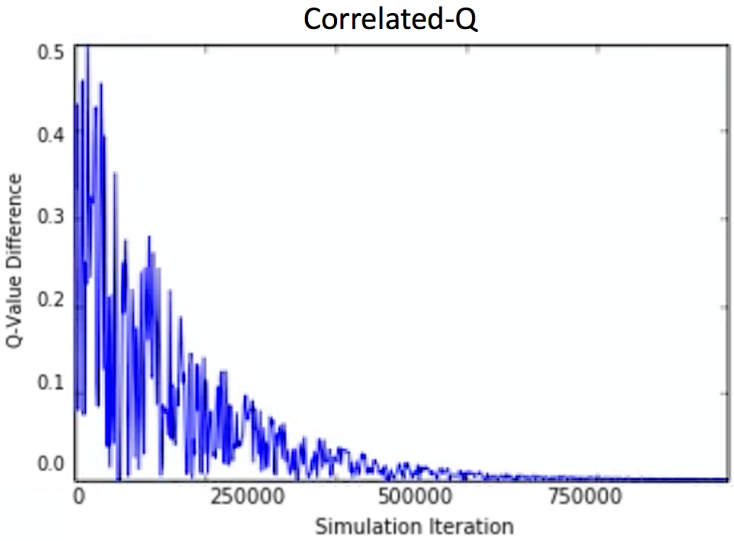

Figure 6: *μ-CE-Q after 1,000,000 Iterations*

VIII. Conclusion

Reproducing literature work requires making assumptions and taking new approaches. One such assumption we made is the decay schedule on our learning rate and ϵ-greedy to make our experiments on-policy. These chosen values greatly impact the rate of convergence and divergence making it difficult to compare our results against [1]. In addition, a multitude of references such as [3] and [4] had to be used to connect the dots; they contain different mathematical formulations making it hard in our replication effort to produce matching results to the tee.

References

[1] Greenwald, A. and Hall, K. (2003). Correlated-Q learning. *AAAI Spring Symposium*, 242–249.

[2] "Convolutional Neural Networks for Visual Recognition" *Annealing the Learning Rate*, retrieved on July 22, 2016 from http://cs231n.github.io/neural-networks-3

[3] Littman, M L. (1994). Markov games as a framework for multi-agent reinforcement learning *Proceedings of the eleventh international conference on machine learning.* Vol. 157, 242–249.

[4] Littman, M L. (2001). Friend-or-Foe Q-learning in General-Sum Games *Proceedings of the Eighteenth International Conference on Machine Learning* 322-328.