
REPLICATING GREENWALD AND HALL’S CORRELATED-Q LEARNING IN SOCCER

Carlos Souza*

Georgia Institute of Technology
São Paulo, SP, Brazil
souza@gatech.edu

July 21st, 2019

ABSTRACT

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Keywords Reinforcement learning · Correlated-Q learning · Markov games · Nash equilibrium · Correlated equilibrium

1 Introduction

In 2003, Greenwald and Hall introduced Correlated-Q (CE-Q) learning (Greenwald and Hall 2003), a multiagent Q-learning algorithm based on the correlated equilibrium solution concept designed to learn equilibrium policies in general-sum Markov games. This algorithm generalized both Hu and Wellman’s Nash-Q algorithm (Hu and Wellman 1998), which converges to Nash equilibrium policies under restrictive conditions, and Littman’s friend-or-foe-Q (FF-Q) algorithm (Littman 2001), which always converges but only works in restricted classes of games (e.g. two-player, constant-sum Markov games, which exhibit minimax equilibria). The purpose of this paper is to discuss the results obtained by replicating four algorithms implemented and tested in soccer environment as described by Greenwald and Hall, providing a complete description of the experiments, implementation details and outcomes.

1.1 Background

A Markov Decision Process (MDP) is a tuple $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$, where \mathcal{S} is a finite set of states, \mathcal{A} is a finite set of actions, P is a state transition probability matrix, R is a reward function and γ is the discount factor.

In a MDP, the agent’s objective is to find a policy π that maximizes the expected sum of discounted rewards:

$$v(s, \pi) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}(r_t \mid s_0 = s) \quad (1)$$

where s_0 is the initial state and r_t is the reward at time t .

*Latest Git hash: INCLUDE LATEST GITHASH

It has been proved that there is an optimal policy π^* such that, for any $s \in \mathcal{S}$, Bellman equation holds:

$$v(s, \pi^*) = \max_a \left(r(s, a) + \gamma \sum_{s'} p(s' | s, a) v(s', \pi^*) \right) \quad (2)$$

Finding the optimal policy π^* can be done using iterative searching methods when reward and state transition functions are known. When these functions are not known, the agent can either learn them by interacting with the environment and use the equations above to solve for π^* (model-based reinforcement learning), or directly learn the optimal policy (model-free reinforcement learning). One of the most popular model-free reinforcement learning methods is *Q-learning*. Key equations of this method are:

$$Q^*(s, a) = r(s, a) + \gamma \sum_{s'} p(s' | s, a) v(s', \pi^*) \quad (3)$$

$$v(s, \pi^*) = \max_a Q^*(s, a) \quad (4)$$

$$\pi^*(s) \in \arg \max_a Q^*(s, a) \quad (5)$$

Given $Q^*(s, a)$, the optimal policy π^* can be found by always taking the action $a \in \mathcal{A}$ that maximizes $Q^*(s, a)$ for all $s \in \mathcal{S}$. In Q-learning, at each time step t the agent interacts with the environment and updates its estimate of Q based on the following equation:

$$Q_{t+1}(s, a) = (1 - \alpha_t)Q_t + \alpha_t \left(r_t + \gamma \max_{a'} Q_t(s', a') \right) \quad (6)$$

It has been proved that the above equation converges to the optimal $Q^*(s, a)$ when learning rate α_t decays over time.

1.2 Markov Games

MDPs are single-agent decision problems. Markov Games are essentially n-agent Markov Decision Processes. They generalize MDPs and repeated games. Mathematically they are defined by the tuple $\langle I, \mathcal{S}, (\mathcal{A}_i)_{1 \leq i \leq n}, P, (R_i)_{1 \leq i \leq n}, \gamma \rangle$, where I is a set of n players, \mathcal{S} is the state-space, \mathcal{A}_i is the i th player's set of actions, P is the probability transition function that describes state transitions, conditioned on past state and joint actions, and $R_i(s, \vec{a})$ is the i th player's reward for state $s \in \mathcal{S}$ and joint actions $\vec{a} \in \mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \dots \times \mathcal{A}_n$.

In Markov Games, player i 's Q-values are defined over states and action-vectors $\vec{a} = (a_1, a_2, \dots, a_n)$:

$$Q^*(s, \vec{a}) = r(s, \vec{a}) + \gamma \sum_{s'} p(s' | s, \vec{a}) v(s', \pi^*) \quad (7)$$

Although we can carry over the notion of state-value function from MDPs to Markov Games, we cannot hold Eq.5 true: deterministic actions that maximizes all players' rewards with respect to one another's actions may not exist. Several alternative definitions for the value function have been proposed. Here are some of them, studied in this paper.

Friend-Q. Littman's friend-Q value function is suited to coordination games, for which all the players' reward functions are equivalent, with uniquely-valued equilibria:

$$v_i(s) = \max_{\vec{a} \in \mathcal{A}} Q_i(s, \vec{a}) \quad (8)$$

Foe-Q. At the opposite, Littman studied two-player, zero-sum Markov Games and von Neumann's minimax value function:

$$v_1(s) = \max_{\sigma_1 \in \Sigma_1(s)} \min_{a_2 \in \mathcal{A}_2(s)} Q_1(s, \sigma_1, a_2) = -v_2(s) \quad (9)$$

where $\Sigma_i(s)$ is the probabilistic action space of player i at state s and $Q(s, \sigma_1, a_2) = \sum_{a_1 \in \mathcal{A}_1} \sigma_1(a_1) Q(s, a_1, a_2)$.

Nash-Q. Hu and Wellman proposed the following definition of value function for the general case of n-player, general-sum games:

$$v_i(s) \in \text{NASH}_i(Q_1(s), Q_2(s), \dots, Q_n(s)) \quad (10)$$

where $\text{NASH}_i(X_1, X_2, \dots, X_n)$ represents the i th player's reward according to some Nash equilibrium in the general-sum game determined by the reward matrices X_1, X_2, \dots, X_n .

Correlated-Q. Greenwald and Hall proposed a generalization of Hu and Wellman’s *Nash-Q* definition:

$$v_i(s) \in \text{CE}_i(Q_1(s), Q_2(s), \dots, Q_n(s)) \quad (11)$$

where $\text{CE}_i(X_1, X_2, \dots, X_n)$ represents the i th player’s reward according to some *correlated equilibrium* in the general-sum game determined by the reward matrices X_1, X_2, \dots, X_n .

A Nash equilibrium (NE) is a vector of *independent* probability distributions over actions, in which all agents optimize with respect to one another’s probabilities. A correlated equilibrium (CE), in contrast, allows for the possibility of dependencies in the agents’ randomizations: a CE is a probability distribution over the *joint* space of actions, in which all agents optimize with respect to one another’s probabilities, conditioned on their own.

While there is no efficient method of computing Nash equilibria, correlated equilibria can be easily computed using linear programming.

1.3 Multiagent Q-Learning

A multiagent Q-learning algorithm generalization template from MDPs to Markov Games is presented below:

Algorithm 1 Multiagent Q-Learning

Require: selection function f , discount factor γ , learning rate α , total training time T

```

Initialize  $s$ 
Initialize  $Q_1, Q_2, \dots, Q_n$ 
Initialize  $V_1, V_2, \dots, V_n$ 
for  $t \leftarrow 1, T$  do
  Agents choose actions  $a_1, a_2, \dots, a_n$  following  $\epsilon$ -greedy policy
  Simulate actions in state  $s$ 
  Observe rewards  $r_1, r_2, \dots, r_n$  and next state  $s'$ 
  for  $i \leftarrow 1, n$  do
     $v_i(s') = f_i(Q_1(s'), Q_2(s'), \dots, Q_n(s'))$ 
     $Q_i(s, \vec{a}) = (1 - \alpha)Q_i(s, \vec{a}) + \alpha[(1 - \gamma)R_i + \gamma v_i(s')]$ 
  end for
   $s = s'$ 
  Decay  $\alpha$  and  $\epsilon$ 
end for

```

This algorithm takes as input an equilibrium selection function f , as shown previously from Eq.8 to Eq.11. Specifically for correlated equilibrium, Greenwald and Hall presented four different correlated-Q learning variants, based on four different selection mechanisms. In the experiment section, we simulated the *utilitarian* (*uCE-Q*) variant, which aims to maximize the *sum* of the players’ rewards:

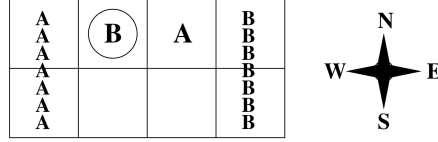
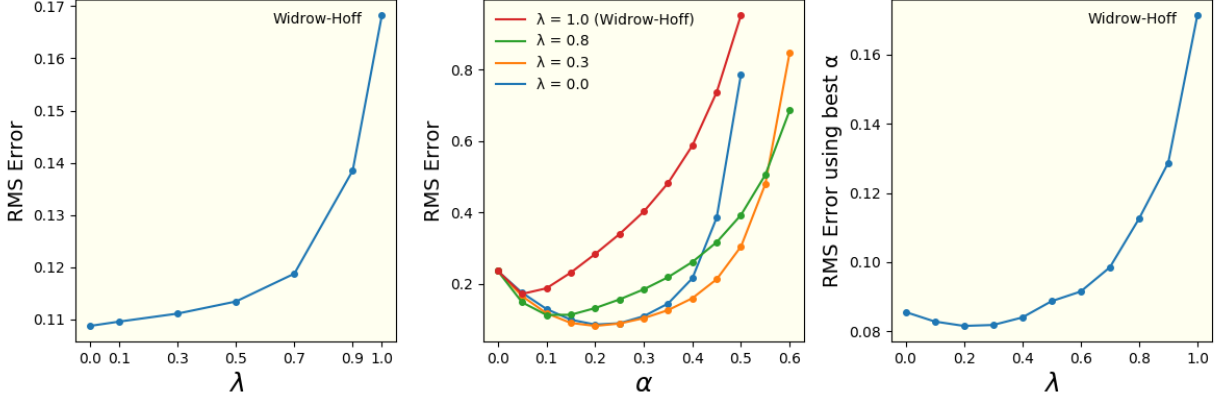
$$\sigma \in \max_{\sigma \in \text{CE}} \sum_{i \in I} \sum_{\vec{a} \in \mathcal{A}} \sigma(\vec{a}) Q_i(s, \vec{a}) \quad (12)$$

1.4 Soccer Environment

Greenwald and Hall simulated different multiagent-Q learning algorithms in a soccer grid game, a zero-sum game for which deterministic equilibrium policies do not exist. The soccer field is a 2-rows by 4-columns grid, as shown in Figure1. There are two opposite players, who can choose five different actions: move North, East, South, West or stick. The ball, represented by the circle, is always with one of the two players. At each time step, both players act in random order. The players cannot occupy the same grid space at the same time: if the sequence of actions cause them to collide, only the first moves. But if the player with the ball moves *second*, trying to occupy a space which is already occupied, the ball changes possession. If a player moves with the ball into his goal, he scores +100 and his opponent -100; if he moves with the ball into his opponent’s goal, he scores -100, and his opponent +100.

2 Methods

To replicate Greenwald and Hall’s results, we followed the exact protocols described in the article. First, we implemented the four different algorithms shown in original article’s Figure 3: Q-learning, Friend-Q, Foe-Q and Correlated-Q (utilitarian). We simulated 10^6 time steps for each algorithm, using discount rate $\gamma = 0.9$.

Figure 1: Soccer Game. State control s .Figure 2: Replication of Figures 3, 4 and 5 from Sutton’s original paper. In the left, average error on random walk problem under repeated presentations. In the center, average error on random walk problem after 10 episodes for different values of α and γ . In the right, average error at best α on random walk problem for different values of γ .

In each simulation, we decayed the learning rate α from 0.1 to 0.001, multiplying it by 0.999995 at each time step. At each time step, every agent chose his action following a ϵ -greedy policy: we started the simulations with $\epsilon = 1$ and decayed it until 0.001, by also multiplying it by 0.999995.

To solve the linear programming set of inequalities, required in Foe-Q and uCE-Q algorithms, we used Python’s PuLP optimization package. Despite its user-friendliness and fast learning curve, this library proven to be very slow in simulating the algorithms. For future work that requires linear programming package and a massing number of calculations, a different solution must be used, such as Python’s CVXOPT.

3 Results

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum. Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris. Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

4 Conclusion

References

- [1] Junling Hu and Michael P. Wellman. “Multiagent Reinforcement Learning: Theoretical Framework and an Algorithm”. In: *Proceedings of the Fifteenth International Conference on Machine Learning*. ICML '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 242–250. ISBN: 1-55860-556-8. URL: <http://dl.acm.org/citation.cfm?id=645527.657296>.
- [2] Michael L. Littman. “Friend-or-Foe Q-learning in General-Sum Games”. In: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 322–328. ISBN: 1-55860-778-1. URL: <http://dl.acm.org/citation.cfm?id=645530.655661>.
- [3] Amy Greenwald and Keith Hall. “Correlated-Q Learning”. In: *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*. ICML'03. Washington, DC, USA: AAAI Press, 2003, pp. 242–249. ISBN: 1-57735-189-4. URL: <http://dl.acm.org/citation.cfm?id=3041838.3041869>.