Recreation of ‘Correlated-Q Learning’

James Corwell

Git Submit ID:
df2b2ca58779d11
660e7e232e597c
b7cbd429ee0

***Abstract*—This paper serves to demonstrate the reproducibility of the results found in “Correlated Q-Learning” by A. Greenwald and K. Hall.**

I. Introduction

The major purpose of this paper is to compare several different modes of Correlated Q-Learning on a 2 player zero-sum Markov game. We rely on the two-player discrete soccer game as defined in ‘Correlated Q-Learning’ (1). CE-Q is proposed as a generalization of Nash-Q and FF-Q.

Correlated Q-Learning algorithm computes the ‘correlated equilibria’, which represents the probability distributions over 2-player’s joint-action states. For this paper, utilitarian CE-Q will be implemented. Under the utilitarian CE-Q algorithm, the two players are working to achieve a common goal. This paper relies on the goal being to maximize the sum of the rewards between both players according to the

We can compare four variants of multi-agent Q-Learning. Friend Q-Learning believes that the opponent will try to maximize the other player’s reward. Foe-Q learning believes that the opponent will minimize the other player’s reward. Q Learning does operate in any positive or negative faith towards the other player.

II. Methods and Implementation

Greenwald and Hall describe many variants of grid games in the original paper. Our task is to utilize the gridded markov ‘Soccer Game’.

The Soccer Game

The soccer game is a type of grid game. Grid games are a type of general-sum game in which deterministic equilibria exist. The soccer game is zero-sum game without deterministic policies.

We can picture the gridded soccer game as follows, where the A and B denotes the player, the ends of the grid are the goals, and the circle represents who has control of the ball currently.

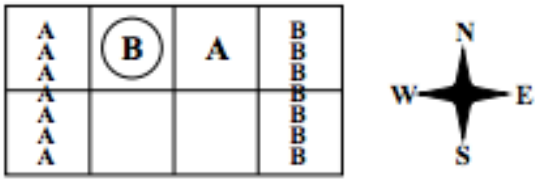


Fig 1 depicting the soccer world from ‘Correlated Q-Learning’ (1)

Players can move in any direction or stay in place. The players receive reward for holding the ball while moving into the goal. If the player with the ball moves into the player without the ball, the ball changes possession.

The game is terminated when an agent receives some form of reward. Reward for an agent can be obtained by moving the ball into the other player’s goal, or moving the ball into your own goal. An open source implementation of the Soccer Environment was utilized and adapted for my Agents (2).

The Learners

Greenwald and Hall provide us with a template for using Q-Learning in a Markov game:

MULTIQ(MarkovGame, f, γ, α, S, T)

Inputs	selection function f
	discount factor γ
	learning rate α
	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 α according to S

Table 1. Basic outline for multi-agent Q-learning to be adapted for each agent. [1]

The Value functions for each of the agents are defined as follows.

Foe-Q Learning

$$V_1(s) = \max_{\pi \in \Pi(A_1)} \left(\min_{a_2 \in A2} \sum_{a_1 \in A_1} \pi(a_1) Q_1(s, < a_1, a_2 >) \right)$$

([2], Section 4.3.3)

Friend-Q Learning

$$V_i(s) = \max_{a_1 \in A_1, a_2 \in A_2} \left(Q(s, < a_1, a_2 >) \right)$$

([2], Section 4.3.3)

Utilitarian CE-Q Learning

$$\sigma \in \arg\max_{\sigma \in \mathrm{CE}} \sum_{i \in I} \sum_{\vec{a} \in A} \sigma(\vec{a}) Q_i(s, \vec{a})$$

where a is <a1, a2> from previous eqs.
([2], Section 4.3.4)

Q – Learning

$$V^*(s) = \max_{a \in A(s)} Q^*(s, a)$$

([1] Naïve Value selection)

Design considerations

In selecting the policies for our agents, Greenwald et al. state;

"experiments reveal that off-policy correlated-Q, foe-Q, friend-Q ($\alpha \to 0.001$ and $\gamma = 0.9$.) and on-policy Q-learning (i.e., ε-greedy, with $\varepsilon \to 0.001$, $\alpha \to 0.001$, and $\gamma = 0.9$) all converge empirically"

This is in reference to experiments with other Markov games. The experiments here use the aforementioned values as suggestions.

Greenwald and Hall also suggest that in experiments with other grid games, that they received empirical convergence using off-policy for CEQ, friend-Q, foe-Q and on-policy for Q-Learning. [ref. 1 sec 4]. As such, agents were designed such that Q-Learning utilized on-policy (epsilon-greedy) action selection, while the rest of the agents were off-policy.

Measurements were taken only when a particular state action pair was taken. It is largely irrelevant which S-A is required for measurement; any should suffice. The purpose of recording only at certain S-A pairs is to show that the difference in Q-values eventually stop changing for a specific S-A-A pair. The original paper stated the S-A-A pair for measurement to be S-south-stick, but the experiments here make measurements on S-south-south.

In designing Correlated Q Learning Agent, after discussion in the class forum I made the decision that the first player's Value matrix is equal to the negation of the second Player's Value matrix [4][1].

$$V_1(s) = \max_{\sigma_1 \in \Sigma_1(s)} \min_{a_2 \in A_2(s)} Q_1(s, \sigma_1, a_2) = -V_2(s) \quad (5)$$

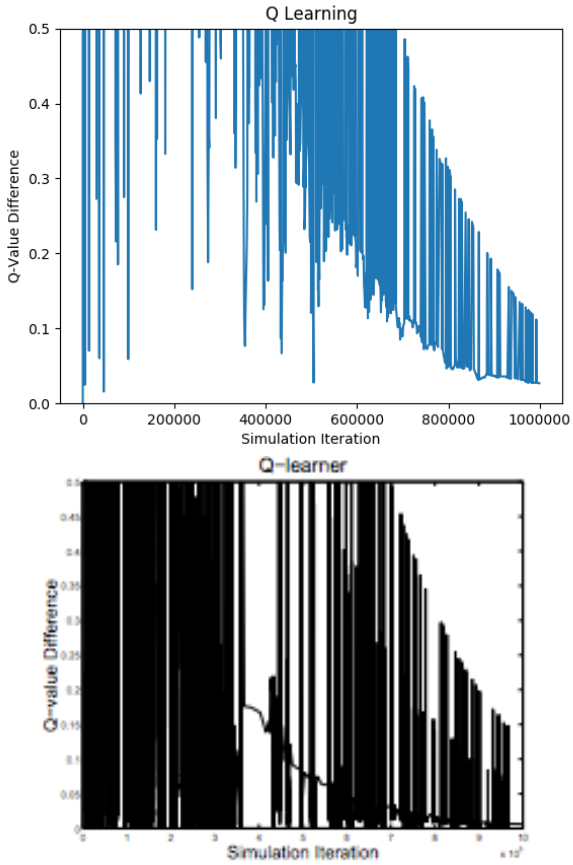
Greenald states in a Markov game the Value at a given state is opposite for each opponent, we utilize this in our model for uCEQ. [1, equation (5)]

III. RESULTS

In this section, the generated results will be shown in comparison with the published results of "Correlate Q-Learning".

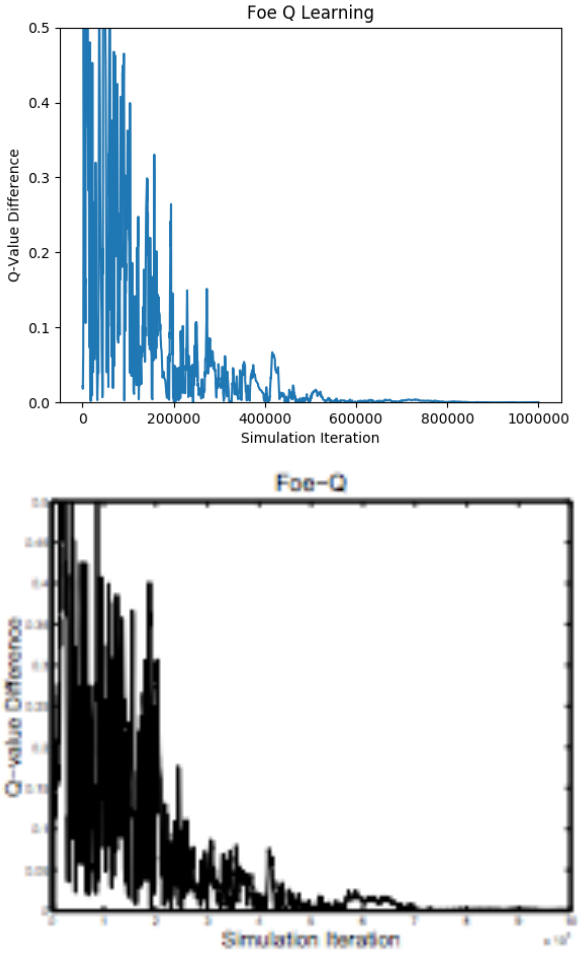
The y-axis is the temporal difference of Q-values between iterations represented by the following relationship:

$$\mathrm{ERR}_i^t = \left|Q_i^t(s,\vec{a}) - Q_i^{t-1}(s,\vec{a})\right|$$



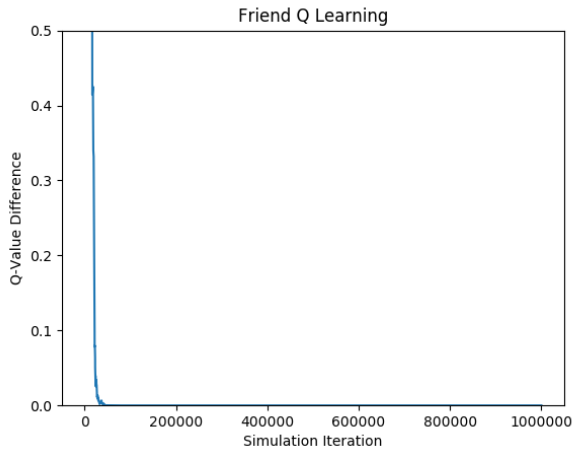
(d) Q-learning

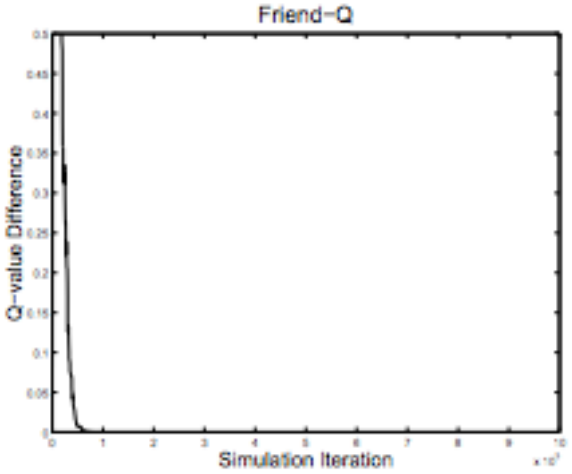
The results shown here show that Q learning does not converge to equilibrium for the selected state action pair, just as suggested in 'Correlated-Q Learning.' This serves as the base case in which other algorithms can be measured. Both the reference graph and the experimental graph are tried at 10^6 iterations, and the experimental graph is pruned to 1/5 of the original data for clarity.



(b) Foe-Q

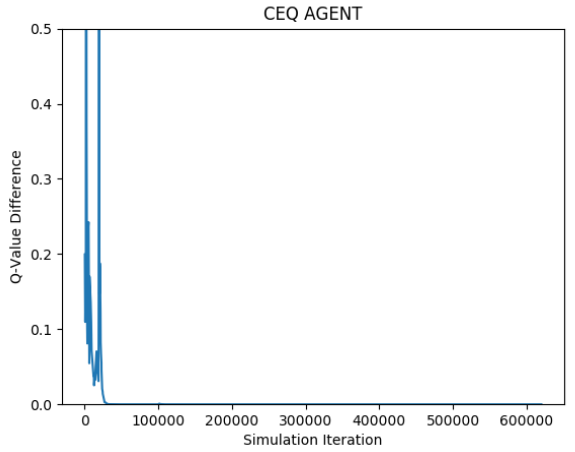
Here we can see similar results between the two Foe-Q learning instances. Both the reference and the generate experiments are run for 10^6 iterations. The results replicate the reference well. Notably my agent has a larger has a smaller 'tail' than the original.

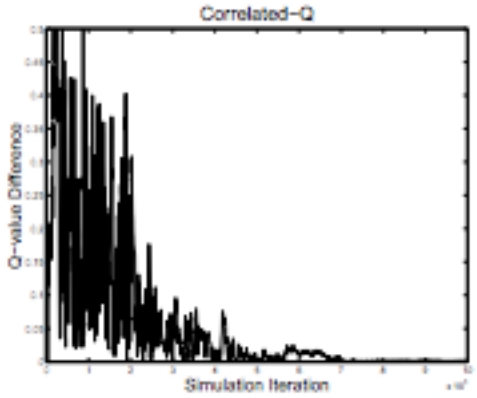




(c) Friend-Q

Friend-Q converges quickly, as shown in the original paper. Here we can see that friend-Q learning converges very quickly, closely mimicking the results from the paper. These graphs look astonishingly similar.





(a) Correlated-Q

Correlated Q-Learning was a failure. Though convergent, the behavior does not match the historic results, nor does it closely mimic the behavior of Foe-Q learning. Note that my game was not run for 10^6 iterations due to several runs of poor performance and eventually running out of time.

IV. Conclusion/Further

This paper serves to show the viability of the research conducted in 'Correlated Q-Learning.' The results are highly indicative that that when controlling for randomized trial possibilities, and hardware limitations that the implementation of the agents and soccer environment are akin to that described in 'Correlated Q-Learning'. Despite the incomplete Correlated-Q agent, we have generally displayed the reproducibility of the original experiments, as well as the *robustness*. Something of note is that despite the differing measurements in S-A pairs from the original paper, the results are not dependent upon measurement state.

Difficulties

To improve the scope of this project, it would be better to run these experiments from a AWS instance, or on a higher faster CPU/GPU locally. The Linear Programming required to implement Foe-Q and CEQ were computationally expensive. The failure with CEQ learning was the complexity of designing the constraints. I need to conduct more research on how to model these constraints with linear programming. The original paper was vague in the implementation description. The next step is

to perhaps study and reverse-engineer the BURLAP implementation [5].

References

[1] A. GREENWALD , K. HALL , 'CORRELATED Q LEARNING', HTTPS://WWW.AAAI.ORG/PAPERS/ICML/2003/ICML03-034.PDF

[2] Lisa Jing Yan and Nick Cercone, "Thoughts on Multiagent Learning: From A Reinforcement Learning Perspective", HTTP://WWW.CS.TORONTO.EDU/~LYAN/CSE-2010-07.PDF

[3] LITTMAN 'FRIEND-OR-FOE Q-LEARNING IN GENERAL-SUM GAMES" ,HTTP://CITESEERX.IST.PSU.EDU/VIEWDOC/DOWNLOAD?DOI=10.1.1.589.8571&REP=REP1&TYPE=PDF

[4] Piazza discussion regarding assigning values in CEQ Learning
https://piazza.com/class/jl0wfj4bb1x4yc?cid=712

[5] BURLAP CEQ LEARNING
http://burlap.cs.brown.edu/