Replicating Multi-Agent Soccer Game Experiments

1. Introduction

In "Correlated-Q learning" by Greenwald [1], performance of Correlated-Q (CE-Q) is compared against Foe-Q, Friend-Q, and Q-learning (QL) in a multi-agent zero-sum soccer game. This paper verifies Greenwald's findings via implementation to better understand these algorithms.

2. Environment

Soccer game environment is a 2x4 grid with 2 players, whose actions are N, S, E, W, and stick:

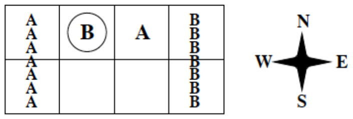


Figure 1. Soccer game environment [1]. This is the state "s" experiments reference.

Players' actions are executed in random order with state updated after each player action. Player does not move if attempts move to occupied space (e.g. steal). If player with ball moves to opponent's space, player does not move and ball's possession changes. Player scores if any player delivers ball to his goal (e.g. player B to goal B), in which he receives +100 while other player -100, and the game ends.

2.1 Implementation

To implement, state is encoded as 3-digit value (0-177) with each digit representing ball possession, and two players' spaces, sequentially. State is updated after each player action to simplify simultaneous moves. Game is restarted randomly with players starting in rational opposing states. Actions are taken deterministically, but order is stochastic. Player sticks if attempts to move out of grid. Following these rules, environment's step function returns <r, s', done> with player actions as input.

3. Algorithm

Markov games are multi-agent stochastic games in MDP-like environments. As such, they can be solved using value iteration with generalized Bellman equation as follows:

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

Figure 2. Generalized Bellman equation for value iteration in Markov game [1].

The four algorithms (CE-Q, Friend-Q, Foe-Q & QL) differ in their selections of value function ($V_i(s')$) or ways of maximizing reward, which are formulated as follows:

$$V_i(s) = \max_{\vec{a}\in A(s)} Q_i(s,\vec{a}) \qquad (6) \qquad\qquad V_1(s) = \max_{\sigma_1\in\Sigma_1(s)} \min_{a_2\in A_2(s)} Q_1(s,\sigma_1,a_2) = -V_2(s) \qquad (5)$$

$$\sigma \in \arg\max_{\sigma\in \mathrm{CE}} \sum_{i\in I} \sum_{\vec{a}\in A} \sigma(\vec{a}) Q_i(s,\vec{a}) \qquad (9) \qquad\qquad V^*(s) = \max_{a\in A(s)} Q^*(s,a) \qquad (2)$$

Figure 3. Value function selection for Friend-Q (6), Foe-Q (5), CE-Q (9) and QL (2) [1].

In Friend-Q (6), player optimistically assumes opponent acts to help maximize reward. In opposite extreme, Foe-Q (5) computes reward pessimistically, in which player maximizes his reward assuming opponent sabotages it (e.g. Minimax). CE-Q (9) generalizes NE by allowing dependency between players' actions. Thus, knowing opponent's possible strategies, player chooses optimal set that neither player deviates (NE). QL (2) chooses value greedily.

4. Experiment & Algorithms Implementation

All algorithms are implemented using Greenwald's procedure for multiagent Q-Learning as follows:

```
MULTIQ(MarkovGame, $f, \gamma, \alpha, S, T$)
Inputs      selection function $f$
            discount factor $\gamma$
            learning rate $\alpha$
            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 $\alpha$ according to $S$
```

Figure 4. General procedure for solving markov games [1].

First, parameters are initialized as follows: γ=0.9, α=1.0, Q_i :=0, $\vec{a}$~[0,1]. For each step, players take actions in implemented environment, which generates players' respective rewards (R_n) and next state (s'). If states are non-terminal, players' Q-tables are updated for experience <s, $\vec{a}$, R_n, s'> using Bellman equation ("Figure 2") and desired value selection ("Figure 3"); otherwise, Q-tables are updated with V_i(s') = 0 and environment is restarted. New actions are selected with α decay. This repeats for 10^6 steps.

4.1 Specifications

All methods besides QL are off-policy, which means players' actions are randomly selected. Off-policy and zero-sum game property (V_1 = -V_2) allows those algorithms to be implemented using only player A's Q-table of size=(177, 5, 5). In contrast, QL's action is selected using ε−greedy (ε=0.01) so both players' Q-tables are used with sizes=(177,5). Decay of α is computed as 1 / (1 + t/N)) such that α → ~0.001 at 10^6 t step. As Foe-Q and CE-Q require mix strategies, their value functions are solved using LP (e.g. CVXOPT) by following constraints:

Foe-Q Constraints:

$$v - \sum_{i=1}^{m} a_{ij} x_i \leq 0 \qquad \text{for all } j = 1, \ldots, n$$
$$\sum_{i=1}^{m} x_i = 1$$
$$x_1, \ldots, x_m \geq 0 \quad \text{and} \quad v \in \mathbb{R}.$$

CE-Q Constraints:

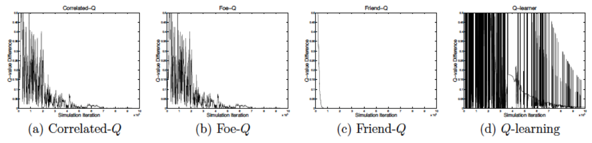
$$\Sigma R_{i,j} * \pi_{i,j} \geq \Sigma R_{i\prime,j} * \pi_{i\prime,j}$$
$$\Sigma \pi_{i,j} = 1$$
$$\pi_{i,j} \geq 0$$

Figure 5. LP constraints for Foe-Q and CE-Q [1].

In brief, Foe-Q maximizes v given upper bound by opponent's actions and CE-Q chooses strategy of optimal payoff for both players. This results in solving 12 inequalities and 6 variables (v & A's action likelihood), and 68 inequalities and 26 variables (v & all mix strategies) for Foe-Q and CE-Q, respectively.

5. Experiment Results

Experiments computes player A's Q-value difference between each step when A takes action S at state s as shown in "Figure 1" for 10^6 steps. For α decay, N=5000, 5000, 2000, and 100 (or α → 0.005, 0.005, 0.002, & 0.0001, see **Pitfall**) for Foe-Q, CE-Q, QL, and Friend-Q, respectively. Results are as follows:



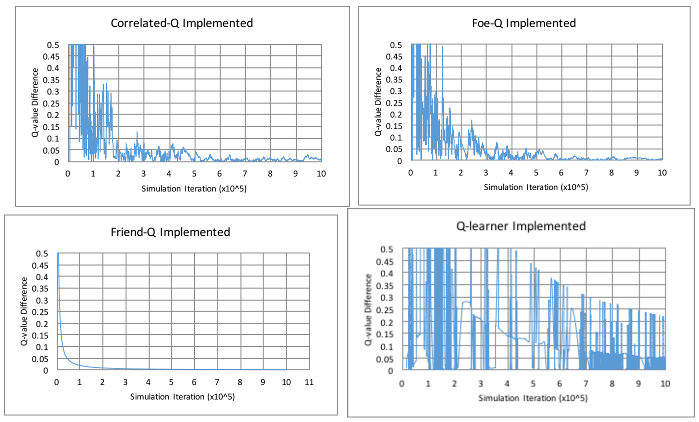


Figure 6. Greenwald's (top) vs. implemented (bottom) results for soccer game.

5.1 Foe-Q & Correlated-Q

As shown, implemented CE-Q and Foe-Q are relatively identical (e.g. small difference due to randomness). This is because in non-zero games, maximizing player's reward also minimizes that of opponent. Thus, CE-Q's joint action space has same probability distribution as that of Foe-Q.
Both implementations agree with Greenwald's with Q-value difference fluctuating significantly between 0 and ~0.35 and gradually converges as alpha decreases. However, they still differ from Greenwald's with slightly less (~0.05) fluctuation. This is likely due to different α decay as reference's decay more sharply.

5.2 Friend-Q

Similar to reference, implemented Friend-Q quickly converges monotonically. This supports Greenwald's finding that players find a deterministic, but irrational policy that assumes opponent help score. However, implementation converges at ~10^5 steps slower than reference. This is due to different form of alpha decay (e.g. linear vs. proportional) as indicated by less sharpness in Q-value curve for reference.

5.3 Q-Learner

Implemented QL agrees with Greenwald's finding in that QL do not converge even as alpha decays due to learners ignoring opponent's actions. Curve of oscillation's mean (e.g. attempt of convergence by lower alpha) and shape in implementation are identical to reference's. However, reference's alpha decay creates a sharper oscillation. Like Friend-Q, this means alpha decays slightly differently in implementation.

6. Pitfall & Assumptions

Many pitfalls in replications arise due to under-specification in "Correlated-Q learning" [1]. For example, α decay function is not explicitly defined (e.g. α → 0.001 without mention of form or steps). Extended paper uses alpha decay of 1/n(s,a), but for different experiments (e.g. magnitude lower iterations), and initial implementation failed replication using this form as alpha decayed too quickly. Thus, this paper assumes α → ~0.001 of linear form is suitable as convergence between algorithms is of interest. To resolve, a range of alpha decays were tested to replicate reference as shown in result.
Reference's use of terminology is also contradictory. For example, unlike reference, QL is conventionally "off-policy" as it as it assumes greedy value selection even if action is not always greedy. This paper assumes definitions used in extended paper [2].
Another pitfall is reference does not specify state constraint for random restart. To resolve, implementation assumes restart states are constrained to soccer game rules (e.g. players are on opposing spaces). This condition affects randomness by exploring more or less possible states. It is also assumed that only Player A's Q-table is needed for all algorithms but QL because they are off-policy.

7. Conclusion

In zero sum Markov games, Foe-Q, CE-Q and Friend-Q converge to policy quickly. Results are identical for Foe-Q and CE-Q as their probability distribution on joint action space is identical due to zero sum. Friend-Q converges quickly to an irrational optimistic policy. In contrast, QL does not converge as learners ignore opponent. Replicating research is challenging as details are often absent. This requires making best assumptions and verifications.

8. Video Presentation

https://youtu.be/3QjvAOmmy9w

9. Reference

[1] A. Greenwald. *Correlated-Q Learning*. Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.
[2] A. Greenwald. *Correlated-Q Learning*. Brown University Technical Report (2005) CS-05-08, 2005.