Correlated-Q Learning

----- Soccer Game Duplication

Manqing Mao
{GTID: mmao33}

1. Introduction

Recently, multiagent learning algorithm that learns equilibrium policies in general-sum Markov games has been developed, like Q-learning converges to optimal policies in Markov decision processes. Friend-or-foe-Q algorithm [1] always converges, but it only learns equilibrium policies in restricted classes of games: e.g., two players, constant-sum Markov games, which exhibit minimax equilibria (foe-Q); e.g., coordination games with uniquely-valued equilibria (friend-Q). Correlated-Q (CE-Q) learning, a multiagent Q-learning algorithm, is based on the correlated equilibrium solution concept [2].

In this project, I first introduced four algorithms, namely, Q learning, friend-Q, foe-Q and correlated-Q (CE-Q) in Section 2. Next, I simulated four algorithms and showed implementation details with corresponding pitfalls in Section 3. I compare our results of Q-value difference with results from Greenwald & Hall's paper [2] and also discussed their optimal policies.

2. Background

2.1 Algorithms Introduction

Markov Games, Q Learning: Stochastic games are a generalization of Markov Decision Processes (MDPs) and repeated games. A stochastic game is a tuple $<I, S, (A_i(s))_{s\in S,\ 1\le i\le n}, P, (R_i)_{1\le i\le n}>$, 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 in state s, P is a probability transition function that describes state transitions, conditioned on past states and joint actions. R_i is the i^{th} player's reward for state $s \in S$ for joint actions. We can then define a state-action value function for each player as:

$$Q^*(s,a) = (1-\gamma)R(s,a) + \gamma \sum_{s'} P[s'|s,a]V^*(s')$$

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

Friend-Q: Littman [2] proposes a similar value function which will cause Q-learning to converge to a coordination equilibrium. The following equation is just ordinary Q-learning in the combined action space of the two players.

$$\mathsf{Nash}_1(s,Q_1,Q_2) = \max_{a_1\in A_1, a_2\in A_2} Q_1[s,a_1,a_2]$$

Foe-Q: The foe value function for two player zero-sum Markov games which incorporates von Neumann's minimax function in place of the usual max operation as per. The following equation is minimax-Q and can be implemented via a straightforward linear program.

$$\mathsf{Nash}_1(s,Q_1,Q_2) = \max_{\pi\in\Pi(A_1)} \min_{a_2\in A_2} \sum_{a_1\in A_1} \pi(a_1)Q_1[s,a_1,a_2]$$

Foe-Q learns a Q function whose corresponding policy will achieve at least the learned values,

regardless of the opponent's selected policy. This is a straightforward consequence of the use of minimax in the update rule of Foe-Q.

CE-Q: Correlated-Q allows us to select different equilibrium from 4 sub-variants. In this project, we only consider the utilitarian (uCE-Q). A probability distribution over the joint action space of both agents in which all agents optimize with respect to all other agents' probabilities conditioned on their own.

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

Note that the above equilibrium can be computed via linear programming by incorporating the objective function of choice into the linear programming formulation (i.e., the probability and rationality constraints). Note also, the implementation of CE-Q necessitates the sharing of Q-tables among agents.

2.2 Soccer Game Environment

Generalized Description: The soccer field is a grid. The circle represents the ball. There are two players, whose possible actions are N, S, E, W, and stick. The players' actions are executed in random order. If this sequence of actions causes the players to collide, then only the first moves. But if the player with the ball moves second, then the ball changes possession. 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. 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.

Zero-Sum Game: there do not exist pure stationary equilibrium policies, since at certain states there do not exist pure strategy equilibria.

Pitfall & Challenge: There would be additional code to implement the ownership of the ball for players. (If the player without the ball moves into the player with the ball, attempting to steal the ball, he cannot. But if the player with the ball moves into the player without the ball, the former loses the ball to the latter.)

3. Experimental Duplicates

3.1 Experimental Pitfall & Challenges

There are 3 main challenges to implement four algorithms, and some of them would affect final convergence results. **(1) ε Greedy Policy NOT required:** I choose to random pick action for each player instead of using ε greedy policy in Q learning. This is because it doesn't really matter what's the actions they took as long as there is chance to hit every state since this Q is off policy learning. Same as in the Friend-Q implementation. **(2) Hyper-parameter Tuning:** Considering ε greedy policy not used in our implementation, 5 hyper-parameters in total are used. Since these 5 parameters are the same for 4 algorithms so that we are going to tune parameters to match 4 figures together. **(3) Fine Tuning Required but Limited Time Budget:** Each time computation for foe-Q or CE-Q takes more than 1.5 hours so it is not easy to do fine tuning before the deadline. Since Greenwald indicates that $\alpha \to 0.001$ in the experiment with the number of time steps of 10^6. The discount factor γ is also not designated in the paper but is informed in Greenwald's another

paper [3], γ = 0.9. I tried my best to fit data with Fig. 3 from the paper [2] and listed the 5 parameters below.

Hyper-parameters	Symbol	Value	Ref
Learning Rate	α	0.2	[5]
Learning Rate Decay	α_{DECAY}	0.9999954	[5]
Learning Rate Minimum	α_{MIN}	0.01	[1]
Discount Factor	γ	0.9	[3]
Number of Time Step	*num_steps*	10^6	[1, 3, 5]

3.2 Experiments Implementation Description

Q Learning: I initialized Q table for player A. For each time step, each player randomly selects his/her action. This is because it doesn't really matter what's the actions they took as long as there is chance to hit every state since this Q is off policy learning, as mentioned in Section 3.1. These actions are composed into a joint actions pair and get current rewards R_i and s' are observed according to T and R. Corresponding Q table is updated by using the following Eqn, where i =1 means of Player A.

$$Q_i(s, a_i) \leftarrow Q_i(s, a_i) + \alpha[r_i + \gamma \max_{a'} Q_i(s', a') - Q_i(s, a_i)]$$

At the end of each time step, by decaying α, we disallow large changes in the agents' Q-values, which makes changes in their policies less and less frequent.

Friend-Q: I implemented friend-Q similarly with Q learning but with one big difference. Note that unlike Q-learning, the action set is now defined as the joint over both players. Thus, for each time step, unlike Q learning, (which calculate its max Q by single action), friend-Q updates its maxQ by searching the maximum values from the action pair.

Foe-Q: The foe-Q algorithm implementation mixed strategies by sampling from a probability distribution over actions conditioned on the state. I compute expected values in terms of the maximal expected reward and minimized over the possible action selections of the opponent. Next, I must solve the linear program which imposes the probability constraints. To solve a linear programming problem, I used cvxopt.modeling [4]. By adding constraints that the probabilities of each action larger than 0 with the sum of probabilities equals to 1, and Q value of the current player is going to be minimum.

Correlated-Q: In zero-sum games, minimax strategies will coincide with the adversarial Nash equilibria of the game [1, 3]. The Nash-Q definition of the value function discussed in [1] generalizes this notion. Correlated-Q learning, thus, converges to the same solution as foe-Q learning —the Q-values learned by the two algorithms are identical. Thus, CE-Q learns minimax equilibrium policies in this two-player, zero-sum game. CE-Q generalizes Nash-Q in general-sum games, since the set of correlated equilibria contains the set of Nash equilibria; CE-Q also generalizes minimax-Q in zero-sum games, where the set of Nash and minimax equilibria coincide.

3.3 Results Comparison & Optimal Policy Description

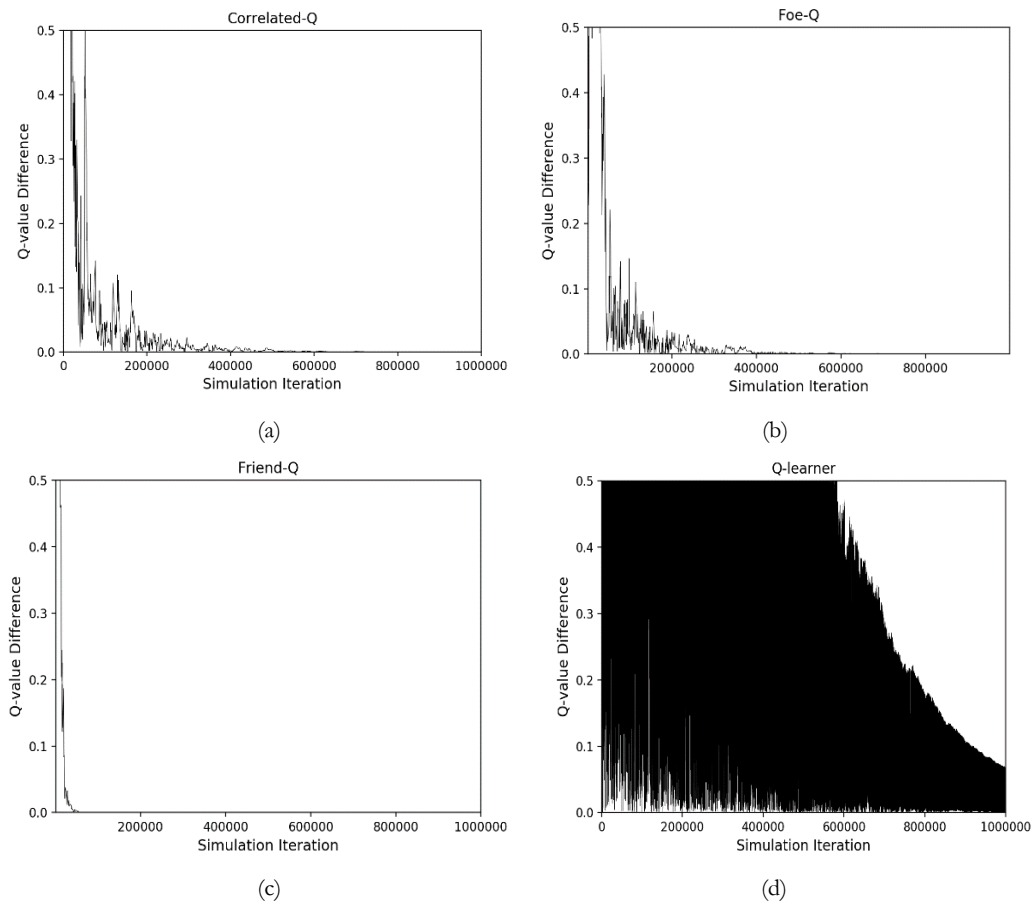

Figure 1. Convergence in the soccer game: (a) uCE-Q; (b) foe-Q; (3) friend-Q; and (4) Q learning.

The Q values difference for uCE-Q, foe-Q, and friend-Q shown in Figs. 1(a), (b), and (c), respectively, reflecting player A's Q-values corresponding to state *s*, with player A taking action *S* and player B sticking. Q-learners compute Q-values for each of their own possible actions, ignoring their opponents' actions. The Q values difference is shown in Fig. 1(d) reflect player A's Q values, corresponding to state *s* and action *S*.

CE-Q & Foe-Q: (1) Result Comparison: From Fig. 1 (a) & (b), we can see that CE-Q and Foe-Q converge for this problem, what's more, they are almost indistinguishable in this game [1, 3]. This is because uCE-Q converge to independent ***minimax*** equilibrium policies at state s, (like what Foe-Q exactly did) although in general, correlated-Q can learn correlated equilibrium policies, even in zero-sum Markov games. However, our CE-Q and Foe-Q converge faster than Fig. 3(a), Fig.3 (b) from [1]. This is maybe due to learning rate α and α corresponding decay rate. **(2) Optimal Policy:** CE-Q and Foe-Q converge to the same mixed policies for both players, with each player randomizing between sticking and heading south.

Friend-Q: (1) Result Comparison: From Fig. 1 (c), friend-Q converges to a pure policy for B but its policies are not rational. It perfectly matches with the Friend-Q from [1]. **(2) Optimal Policy (Score for Me):** Friend-Q converges to a deterministic policy for player B at state *s*, namely action E. 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.

Q Learning: (1) Result Comparison: From Fig. 1 (d), we can see that Q-learning does not

converge for this problem, although the Q-value differences are decreasing. It matches well with the Q learner from [1]. However, our Q learner difference value decreases faster than Fig. 3(d) from [1]. This is maybe due to learning rate α since the amplitude of the oscillations in error values is as great as the envelope of the learning rate [1]. **(2) Optimal Policy:** No optimal policy exists. This is because Q learning does not learn equilibrium policies at zero-sum game.

Reference:

[1] Amy Greenwald and Keith Hall. Correlated-q learning, 2003.

[2] Michael L. Littman. Friend-or-foe q-learning in general-sum games, 2001.

[3] Amy Greenwald, Keith Hall and Martin Zinkevich, 2005.

[4] http://cvxopt.org/userguide/modeling.html

[5] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning, 1994.