CS 7642 Reinforcement Learning and Decision Making (Project 3): Comparison of Q-learning, FF-Q, and Correlated Q in Zero-Sum Multi-Agent Stochastic Game

Qisen Cheng

Computer Science Department, Georgia Institute of Technology,
Atlanta, Georgia, USA 30318, qcheng35@gatech.edu

Abstract. This work, as part of the course requirement of CS7642, compares different learning algorithms for zero-sum multi-agent stochastic game, by replicating the comparison results in [1]. In particular, ordinary Q-learning method is used as a baseline method to show the advantages of using Friend-Foe-Q (FF-Q) or Correlated-Q in such learning tasks. The performance of these methods are compared in terms of convergence, by playing the soccer game in [1]. The soccer game is a zero-sum two-player stochastic game, which serves as a good testbed for the comparison.

Keywords: CS7642, reinforcement learning, zero-sum two-player stochastic game, Q-learning, FF-Q, Correlated Q, convergence

1 Introduction

The four learning methods involved in this report are 1) ordinary Q-learning, 2) Friend-Q learning, 3) Foe-Q learning, and 4) correlated Q learning. These methods mainly differ in their ways to view and treat the opponent in the two-player game setting [1]. In the ordinary Q-learning, the opponent's action is ignored during learning, which means the two players try to learn the optimal Q values without consideration of each other. In the Friend-Q learning (Equation 1) [2], each player tends to be optimistic, which means each player considers the opponent will help him even with sacrifice in their own purpose. In the contrast, the Foe-Q learning (Equation 2) [2] method leads to extremely pessimistic view of the opponent – each player will try to minimize the gain for the other. Correlated-Q is a more comprehensive frame of algorithms involving four variants, which have different emphasis regarding the relation between two players, respectively. The four variants (Equation 3) are referred to as 1) utilitarian, 2) egalitarian, 3) republican, and 4) libertarian [1].

$$\mathsf{Nash}_1(s, Q_1, Q_2) = \max_{a_1 \in A_1, a_2 \in A_2} Q_1[s, a_1, a_2] \tag{1}$$

$$\mathsf{Nash}_1(s, Q_1, Q_2) = \max_{\pi \in \Pi(A_1)} \min_{a_2 \in A_2} \sum_{a_1 \in A_1} \pi(a_1) Q_1[s, a_1, a_2] \tag{2}$$

1. maximize the *sum* of the players' rewards:

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

2. maximize the *minimum* of the players' rewards:

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

3. maximize the *maximum* of the players' rewards: (3)

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

4. maximize the *maximum* of each individual player i's rewards: let $\sigma = \prod_i \sigma^i$, where

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

1.1 Two-player Soccer Game

Two-player soccer game (TSG) serves as a simple but informative setup for exploration of different learning algorithms. TSG can be depicted as following (Fig. 1): a finite number of states are determined by the ball possession (between two players), and the grid location (2 rows x 4 columns) of each player. Due to each grid can only be occupied by one player, there are (8x7x2 =112) different states in total. Each player can choose from 5 different actions (namely go North, South, West, East, or Stay) at each step. The game is a zero-sum game since the rewarding rules make the sum of the payoffs between the players to be zero, regardless of which player scores. Another important feature of the game is that the action of each player is taken by random order, though they are considered to act simultaneously. This randomness of action order and rewarding rules leads to no deterministic equilibrium policy of each player. Instead, the players can take mixed policy (which can be represented as a probability distribution over all available actions) in order to achieve better results. By considering the fact that the two of the four learning algorithms (ordinary Q-learning and friend-Q) only give deterministic policy, it can be expected that the performance of these two learning algorithm should be less optimal.

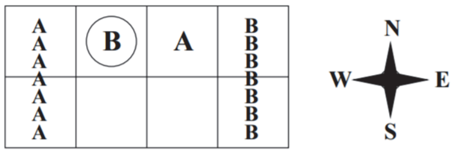

Fig. 1: Illustration of two-player soccer game in [1].

In this report, the background is introduced in section 1. The implementation details are described in section 2. Section 3 shows the experimental results and discussions.

2 Implementation

This section describes the implementation details of four different learning methods, which also includes the discussion on challenges in and difference between each method.

1. **Soccer game environment.**

Implementation. The implemented environment involves two classes, which capsulate the functions of the game and the agent (player), respectively. In the game class, it mainly determines the transitions between states based on the game rules, which includes the determination of action order and ball possession between players and the movement of each player in the grid. In each of the two instances based on agent class, the attributes (i.e. locations, Q value table, etc.) and functions (i.e. Q-table updating, action selection, etc.) of each player (A and B) are abstracted.

Challenges. The tricky parts in the implementation of soccer game environment stem from some "vague" definitions of rules for edge cases in the game. For example, it is not crystal clear when certain player with the ball steps into the goal area but blocked by the opponent – does he goal first before losing ball? For this particular case, the implementation enforces that the possession of ball is considered first – so that he could not score but bounced back and lose the ball. In addition, it is also important to consider that the switch of ball can lead to immediate scoring of the new holder of the ball – namely "own assistance" rather than "own goal".

2. States, actions and Q-table.

Implementation. As discussed in the introduction (section 1.1), there are totally (8x7x2 = 112) different states for each of the four learning algorithms. In addition, each player has 5 options of actions at each step. The keys to the entries of Q-table should be the combination of the states and actions. Based on the treatment of opponent, the Q-table of each algorithm differs with each other. In all the four implementations, each player keeps a Q-table for his own reference. Due to the ignorance of opponent's action in the ordinary Q-learning, the entries in the Q table of this algorithm are differentiated by the pair of different state and the action of only the player himself. Thus it is only a "two dimensional" Q-table. All the Q-tables of Friend-Q, Foe-Q and CEQ involve the action of opponent as part of the key – the tables can be considered as "three dimensional". Each entry has key formed by a three-element tuple, (state, action of myself, action of opponent), in these tables.

Challenges. The most challenging implementation of Q-table is the one used in CEQ. The algorithm requires each player knows the exact rewards and Q value of the opponent, which generally needs to have two Q tables for each player – one for own Q-table, and another as a copy of opponent's. However, this implementation can be simplified by considering the zero-sum property of the game. This property leads to corresponding entries to be just opposing in sign but same in absolute value. Thus, each player can still hold only one Q-table and learn from his own Q-table together with direct inference of his opponent's Q-table.

3. Learning algorithms and parameters.

Implementations for ordinary Q and Friend Q. The ordinary Q-learning algorithm is simple due to the fact that each player is basically updating his own Q-table with no interference. The Friend-Q is similar to the ordinary Q-learning to some extent – each player is considering his opponent would help him, so that only the global maximum Q value learned at each state would be the choice for updating the V of each state. This algorithm actually does not need to "know" action of the opponent, it just uses the most optimistic estimation on the opponent in every step – which means the opponent is somehow "fixed" in the situation of being always helpful. Thus the ordinary Q and Friend-Q learning methods could use just discrete table search to update the Q table, which will result in deterministic policy if the learning is converged.

Implementation for Foe-Q. The Foe-Q tries to find the optimal mixed policy – which is essentially optimal distribution ($\vec{\pi}_{a|s}$) over all the available actions. Because of the infinite possibilities of distribution, simple table search is not working for Foe-Q. Instead, we can introduce a variable (V) for the expectation of value for each player (*V(s)*) at certain state, and regard the expectation of value as dot product of the marginal probabilities of each action taken by the player himself and the Q value of corresponding entries in the Q table. The expectation of V is conditioned on the action of the opponent – in terms of optimization, this means the maximization of V is constricted by minimization from the mixed action of the player's opponent. Further, the difficult max-min problem can be simplified by transforming the minimization as linear constrictions for maximization, which mathematically means the assumed expectation of V (variable U) should only less than or just equal to the conditional expectation of value for each possible action. Thus, this gives 5 constrictions ($\mathrm{U} \leq \sum \vec{\pi}_{a|s,o} \cdot \overrightarrow{\mathrm{Q}}_{a|s,o}$) for each player. Other constrictions come from the non-negative nature and summation of $\vec{\pi}_{a|s}$ being 1. Since the objective function and all the constrictions are linear, the optimization problem is a linear programming problem. The actual action taken by each player at each step is sampled based on the calculated marginal distributions.

Implementation for CEQ. The CEQ implementation is very similar to that for Foe-Q with differences in shared Q tables between two players and the optimization goal. As stated in descriptions for Q-table above, the CEQ could just use one table for each player considering the zero-sum nature of the game. Compared with Foe-Q, CEQ (utilitarian) tries to optimize the joint expectation of the sum of value from both players at certain state – this leads to correlated equilibrium between the two players. This is also a linear programming problem to solve for the joint distribution ($\vec{\pi}_{a,o|s}$) at each state. This optimization problem has some constrictions in terms of rationality of taking certain action – the assumed expectation of value at each state with certain action taken by one player should be always more or equal to the conditional expectation of value based on other actions (conditioned on actual action). This reflects why the player should take the actual action. After solving the joint distribution, each player can figure out his own marginal distribution over actions by taking the summation over the actions of his opponent. So that each player can update his own Q table based on action sampled from the marginal distributions.

Challenges. The most challenging part is the linear solver. In the code, a python package "cvxopt" with "cvx_glpk" solver is used to solve the linear programming problems at each step for both players. However, the outputs of the solver sometimes involve negative probabilities, which results in the summation of all marginal probabilities not equal 1. These negative probabilities are always of extremely low absolute value (close to 0), which implies that the issue is due to the limited precision and binary representation in numerical calculations, rather than the setup of the solver. To solve this problem, a "clipping" trick is used in the codes – the calculated array of probabilities is renormalized in a range of 0 – 1. This will correct the small negative probability to be zero, whereas keep the value of "correct" probabilities.

3 Experimental results

Experiments and results. The reproduced results of soccer game in [1] are shown in Fig. 2(a-d). It can be seen that the trends of all the four figures are identical to the results in [1]. The Friend-Q, Foe-Q and CEQ (utilitarian) all converge after certain steps, while the ordinary Q-learning does not converge. In particular, the Friend-Q converges relatively faster than the Foe-Q and CEQ (utilitarian). This reflects the fact that Friend-Q actually converge to a non-optimal deterministic policy instead of a mixed policy – action of B converges to "go to E" at the start of the game, in hope of own goal from A.

Parameters. The parameters for all the four algorithms are identical. The reward discount Gemma is 0.9, which is directly from Littman's paper [2], which shows a similar soccer game. This encourages the players to score as early as possible. The bound of learning rate ($\alpha_{\max} = 1$, $\alpha_{\min} = 0.01$) and decay rate (=0.999995) are found in a series of ad-hoc experiments. The decay rate is determined partially with consideration to keep the learning rate large enough until enough updating and relatively small when approaching the optimal solutions. The upper bound of greedy-level ($\varepsilon_{\max} = 1$) and its decay rate (=0.9993) are tuned by running several experiments. The decay rate ensures enough exploration for each player in at least first thousands of steps.

Difference, challenges and analysis. Although the trend of all the figures are similar to that of the results in [1], there are still 3 major differences. First, the absolute Q-difference is different in magnitudes compared to the figures in [1]. This is mainly due to unclear definition and misunderstanding of the error used in [1]. In this work, the absolute Q-difference is the sum of difference over all the states given action of player A ("go to S"), and action of player B ("stay"). It is suspicious that the error in the paper only counts the Q-difference of the starting state rather than over all the certain states, while leads to the difference. Another difference is that the converging sequences (or transient patterns) between the Foe-Q and CEQ are different in this work, which is actually similar to each other in [1]. This might be because of the different setting of parameters. For example, it is just given the trend and minimum value of learning rate ($\alpha \rightarrow 0.001$) in the paper [1], but it does not specify how the learning rate is decreased. In the work, the reduction is assumed to performed by continuously multiplying a decay factor to the learning rate.

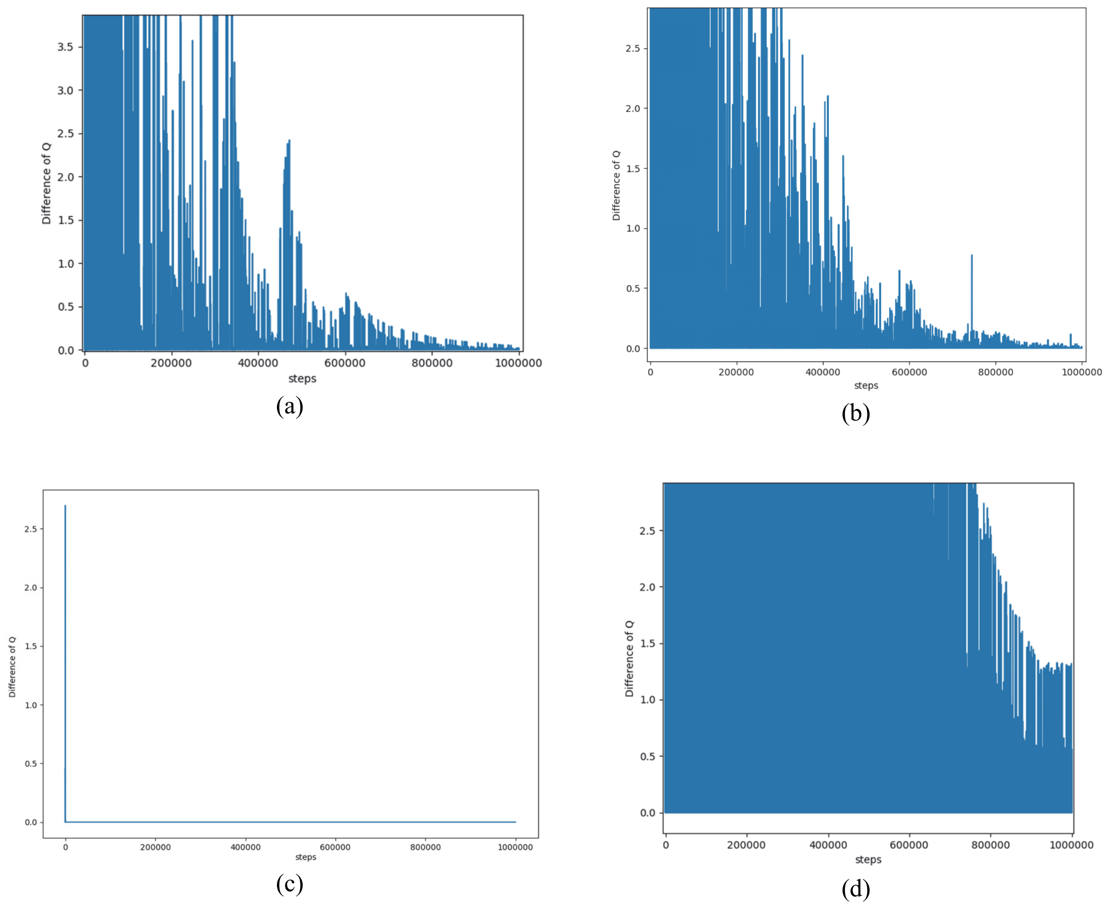

Fig. 2: Results of reproduced results in [1]. (a) Convergence sequence of CEQ. (b) Convergence sequence of Foe-Q. (c) Convergence sequence of Friend-Q. (d) Learning sequence (not converging) of ordinary Q-learning.

References

1. Amy Greenwald and Keith Hall. 2003. Correlated-Q learning. In Proceedings of the Twentieth International Conference on International Conference on Machine Learning (ICML'03), Tom Fawcett and Nina Mishra (Eds.). AAAI Press 242-249.
2. Michael L. Littman. 1994. Markov games as a framework for multi-agent reinforcement learning. In Proceedings of the Eleventh International Conference on International Conference on Machine Learning (ICML'94), William W. Cohen and Haym Hirsh (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 157-163.