Project 2: CS8803 - O03 Reinforcement Learning

Saad Khan (skhan315@gatech.edu)

July 24, 2016

1 Introduction

The purpose of this project report is to experimentally replicate Multi-agent Correlated Q-Learning put forward by Amy Greenwald and Keith Hall in their 'Correlated Q-Learning' paper published in 2003. The focus of this report will be to apply Correlated Q, Friend & Foe Q and classic Q-Learning to the "Soccer Game" example discussed by them in the second half of the paper. This piece of research attempts to learn equilibrium policies for Markov games as does classical Q-learning for regular MDPs. Following parts of this section cover important concepts pertaining to the report before we delve into the experiments.

1.1 Markov Games & Correlated Equilibrium

Markov or stochastic games can be visualized as multi-player MDPs and can be represented in form of tuple $<I,S,(A_i(s)),P,R_i>$, with I as number of players, S as states, $A_i(s)$ as joint action vector for the players, P the probabilistic transition function and $R_i(s, \vec{a})$ as the i^{th} player's reward. Instead of individual player's action, here the actions are considered to be a joint vector for all the players.

The paper presented by Greenwald and Hall extends the idea of a multi-agent learning technique to what they call as Correlated Q-learning. This method generalizes both Nash-Q and Friend & Foe-Q methods presented by Hu and Wellman in 1998 and Littman in 2001 respectively. Expanding on Nash-Q algorithm, which is based on Nash equilibrium (NE), learns via a vector of *independent* probability distributions (PD) over actions where all the agents optimize based on each others probabilities. These independent PD vectors made things complicated as no efficient method for computation for Nash equilibria was known at the time, this is where Correlated equilibrium (CE) came for the rescue, representing PD over *joint* space of actions, which can easily be computed using linear programming.

1.2 Multi-agent Q-Learning

Single agent Q-learning in case of MDPs can be generalized for multiple agents using Q updates for a particular agent i based on equation 1 as shown in Figure 1.

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

$$V_1(s) = \max_{\sigma_1 \in \Sigma_1(s)} \min_{a_2 \in A_2(s)} Q_1(s,\sigma_1,a_2) = -V_2(s) \quad (2)$$

$$V_i(s) = \max_{\vec{a} \in A(s)} Q_i(s,\vec{a}) \quad (3)$$

$$\sigma \in \arg\max_{\sigma \in CE} \sum_{i \in I} \sum_{\vec{a} \in A} \sigma(\vec{a}) Q_i(s,\vec{a}) \quad (4)$$

$$Q(s,a) \leftarrow Q(s,a) + \alpha \underbrace{(R' + \gamma \max_{a} Q(s',a)}_{\text{learned value}} - \underbrace{Q(s,a)}_{\text{old value}} \quad (5)$$

$$\text{ERR}_i^t = \left| Q_i^t(s,\vec{a}) - Q_i^{t-1}(s,\vec{a}) \right| \quad (6)$$

Figure 1: [Equations] Correlated Q, Friend & Foe Q and classic Q-Learning

The most critical part of the update, which governs the type of equilibrium being applied is based on the selection of equations 2, 3 or 4 implementing Foe-Q, Friend-Q and CE-Q learning algorithms respectively. Greenwald and Hall utilized the above mentioned equations to convey the results of multi-agent Q-learning alongwith regular Q-learning also shown in Figure 1 as **equation 5**. The intention is to apply these equations on to a special case of Markov Games: *Two-Player, Zero-Sum* in form of grid soccer and compare how these converge based the error metric shown above in equation 6.

1.3 Grid Soccer

Description : As shown in Figure 2 [LEFT], 4 X 2 grid soccer, is a simple approach to show the power of multi-agent Q-learning through simple gameplay. The game always initiates in the state shown above with ball ownership with player B.

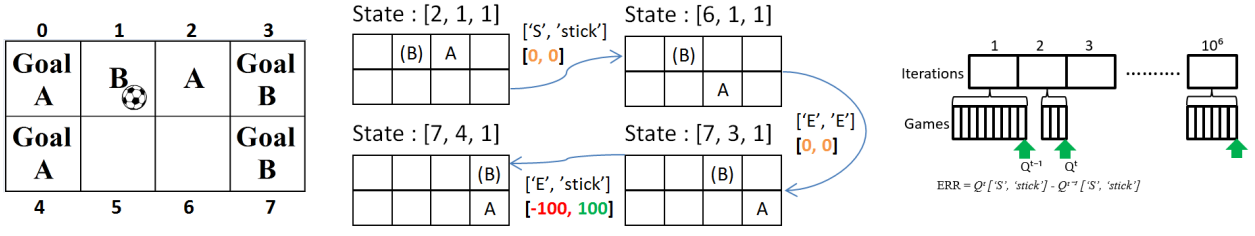

Figure 2: [Grid Soccer] LEFT: game in start state, CENTER: example walk through, RIGHT: error calculation

Second state of the game is where player B stays in cell 1 and player A moves to cell 6 via joint action ['S', 'stick']. Following this, two players choose actions simultaneously, executing them in random order to follow moves that lead to either player winning the

game by achieving goals (opposing or own goal). This can be realized as a golden goal situation in two scenarios: *1.* the scoring player receives a reward of 100 for scoring in the opposing goal while the other (loser) gets -100, *2.* the scorer (loser) gets -100 for scoring an own goal while the other player (winner in this case) receives 100, hence, zero-sum game.

To illustrate the understanding, a simple example walk through of a game is shown in Figure 2 [CENTER]. The sequence of moves, leads to player B winning the game and achieving 100 points at the end with player A receiving -100. The state representation is a tuple which shows grid positions for the players followed by the indication of ball ownership. The joint actions show the movement of the players and joint reward array shows the reward allocation in the final state with no rewards for all the intermediate states.

2 Experiments

2.1 Implementation

Based on the paper, four different experiments were to be designed to prove that Friend-Q, Foe-Q, CE-Q algorithms converge while classic Q-learning fails to converge in the multi-agent scenario even after 1 million iterations of learning. The implementation of the soccer game, the multi-agent learning algorithm and error calculation was done using code written in python. States and action pairs were generated based on piazza post **392**, proposed by Marlon Thomas; 2 x 8 x 7 = 112 states and 5 x 5 pair of actions. All implementations for the learning algorithms were done following Table1(Greenwald and Hall 2003) and its variant for updates as shown in Figure 1 above. Error calculations were computed using Q values for current state ['S', 'stick'] and comparing it to Q values of previous state ['S', 'stick'] iterations as shown by Figure 2[RIGHT. Finally, the results of the experiments were exported to MS Excel where the plots were generated.

Assumptions : *1.* Certain assumptions were made regarding player movement in corner/edges as nothing in particular was mentioned about this by the authors. *2.* From algorithm aspect, an assumption about the learning rate/decay schedule was also made as authors do not explicitly provide decaying schedule. *3.* For Q-learning, no specific ϵ-greedy conditions were set, so an assumption for this had to be made also.

Pitfalls and Resolutions : *1.* The authors did not clearly mention handling the situation when player without the ball is in the corner/edge and tries to move outside the grid. I thought of either making the player stick or randomly choose to move inwards (I chose the later), i.e. if player is in cell 0 and random action is west (which it cannot take), it would then choose randomly between three options, either to move east, south or stick. Similarly, if a player is in cell 6 and random action is south, it would then choose randomly between four options, either to move east, north, west or stick; this helped with more interactions between the players and ultimately quicker finish to the game. *2.* There was real struggle to choose the best decay schedule for the learning rate, tried out different options, from exponential, to linear and finally decided on the geometric scheme proposed in section 6.2, Littman 1994. *3* Deciding on greedy selection criteria also consumed a lot of time, finally, ϵ=0.01 was chosen based on Greenwald, Hall and Zinkevich **2007 paper** section 7.1 and the selection scheme was based on this **link**.

2.2 Experiment 1 & 2 - Correlated-Q and Foe-Q

Description : Correlated-Q (CE-Q) learning algorithm, as the name suggests is based on the correlation equilibrium solution concept. In case of the soccer game experiment run by the authors, it converges to non deterministic polices for both players, given the first joint action: player A moves south and player B sticks. CE-Q was run using calculations based equations 1 and 4 provided in Figure 1. The primary idea was to maximize the sum of rewards of both the agents. For Foe-Q, based on Minimax-Q algorithm presented by Littman 1994, the goal was to maximize the worst case (minimum) over all that the opponent can play. This algorithm was implemented using the information provided in Littman 1994, Figure 1.Like CE-Q, Foe-Q also converges to non deterministic policies for the players involved.

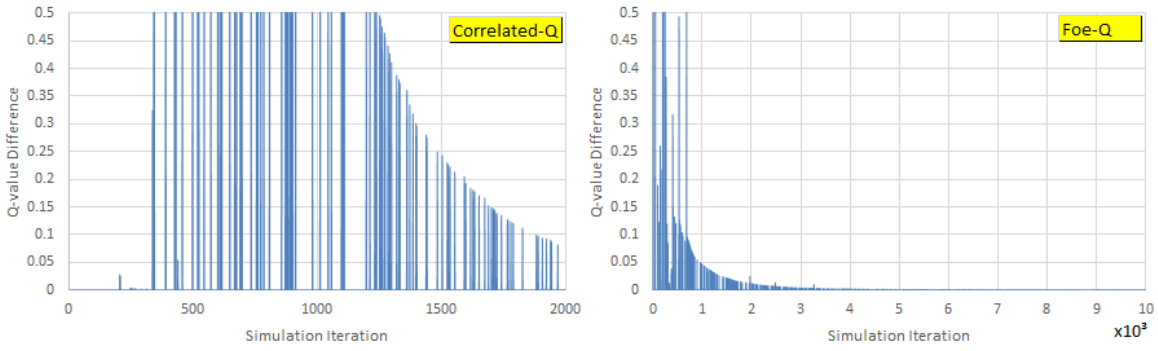

Figure 3: LEFT: Correlated-Q, RIGHT: Foe-Q

Pitfalls : There was a real struggle to understand even the basic notion of what was going on with CE-Q and Foe-Q algorithms. In addition to this, linear programming aspect of this experiment was another mountain to climb. In order to take a head start on this project, BURLAP/Java was considered as the first choice due to earlier suggestions in Piazza as it had LP already implemented. With an extremely steep learning curve, it was dropped and python (the go to language) was chosen, only to be completely bemused by linear programming packages. As suggested in piazza post **418** extensive time was spent to get head around PuLP in order to complete the linear programming part with minimal success.

Results and Observations : The plot generated for Correlated-Q learning shows a convergence pattern, even with the 2000 iterations and probably would have converged if there was enough time to run more iterations. On the other hand, the 10,000 iterations ran for Foe-Q do show convergence.

Significant Differences/Comparison : If looked closely, both plots for CE-Q and Foe-Q, are very similar for the first 2000 iterations, confirming what the authors say about CE-Q learns policies that coincide exactly with Foe-Q. Although, plot for CE-Q has not converged, it still has a decaying effect. Complete 1 million iterations were not performed for both these experiments, so it is hard to compare these plots to the originals.

2.3 Experiment 3 & 4 - Friend-Q and Q-Learning

Description : Friend-Q was presented by Littman in his **2001 paper** and this learning technique works on the principle of coordination equilibrium between the agents in play. As actions are in form of joint pair (players take them simultaneously to move to different state), the objective was to simply maximize Q values (as per equation 3 in Figure 1) in a given state based on the actions taken by both the players, hence, no linear programming was required.

Q-Learning applied to the soccer game is based on the Q-update equation 5 shown in Figure 1. The Q update in this case has two parts as it is based on difference of previous state's Q value (old) and current state's (learned) value which is discounted by a discount factor of 0.9, relying heavily on future reward. TQ error updates after each game ends were taken as shown in Figure 2 [LEFT].

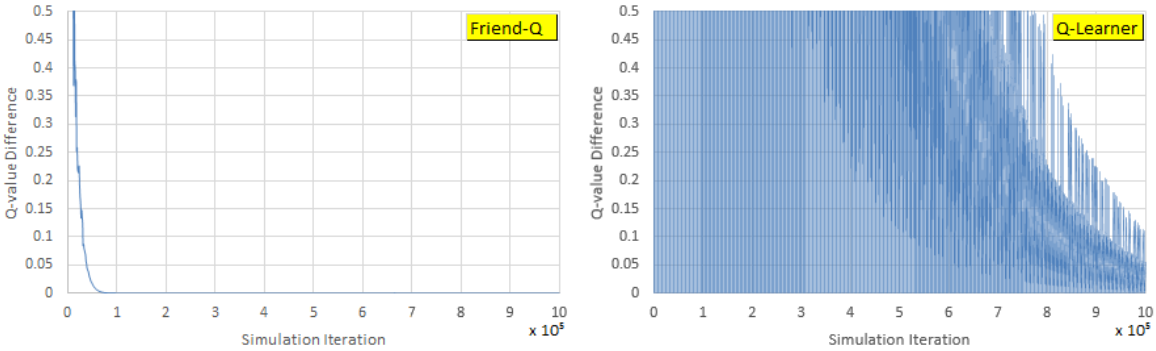

Figure 4: LEFT: Friend-Q, RIGHT: Q-Learner

Results and Observations : Figure 4 [LEFT] shows the results for Friend-Q when run on the soccer game starting at state shown in Figure 2. Algorithm was run for 1 million iterations and it can be seen that Friend-Q converges. The biggest challenge in replicating this graph was to choose the appropriate decay schedule for the learning rate. An attempt was made to try out a simple linear decaying function based on arithmetic progression, however, this did not help in convergence. Finally, as mentioned in the previous section, decay schedule mentioned by Littman in his **1994 paper** was chosen. This decay schedule was tuned by a factor in order for the Friend-Q algorithm to learn in a way to replicate the plot in 2003 paper as closely as possible.

Plot for Q-Learning is shown in Figure 4 [RIGHT] and in contrast to Friend-Q, it can be seen that classic Q-Learning fails to converge. Although, the Q-Learning algorithm results do show a decreasing trend but even after million iterations it does not fully converge in this non-deterministic scenario, highlighting the fact that Q-Learning is always looking for deterministic policies. Decay schedule used in this case was also taken from Littman 1994 and ϵ=0.01 was chosen based on the paper mentioned above in section 2.1.

Significant Differences/Comparison : The generated plot for Friend-Q very closely resembles the plot in the paper. In order for clarity and to observe the decaying trend due to the learning rate, Q-value differences were sampled at intervals of 1,000. It can clearly be seen that the algorithm converges well before 100,000 iterations coinciding with the original plot. The only major difference observed was fluctuations in the difference values in the region where the algorithm was converging while the original plot remained smooth, i.e. less than 50,000 iterations.

For Q-Learning, at first, it was hard to compare the created to the original plot in the paper, but sampling interval of 500 iterations for Q-ERR helped signify the common points between the two plots. It can clearly be seen that generated Q-Learning plot, just like the original plot, is showing internal decaying trends for iterations greater than 300,000. To observe this, I had to thin down the line width in MS Excel. The only major difference observed between the generated and the original plot seems to be the decaying sequence between 350,000 - 450,000 iterations in original plot which is not present in the later, could be due to the randomization with which gameplay was executed.

3 Conclusion

Overall this was a very challenging project, if it had not been for some of the fellow course mates, I would not have made it so far. Piazza and slack chats were a great help. In terms of the challenges faced in the project, linear programming, decay schedule for the learning rate and sampling for the plots took major chunk of the time. However, I learned a lot about markov games in general. There were a lot important details missing from the experiments described in the paper, so overall attempt was to come as close as possible to the published results.