CS 7642 - Reinforcement Learning
Correlated Q Learning

Prasanna Venkatesan Srinivasan
MS in Computer Science at Georgia Tech
psrinivasan48@gatech.edu
Git Hash: fcaa57110790e8b9217c3a473b1e1044d3c129ec

1 Introduction

Q-Learning is used to derive an optimal policy for an agent acting on an MDP. For Markov Games, which involve multiple players at a time, the payoffs of the adversaries of an agent will have to be considered for proposing a policy that aims to achieve a state of equilibrium. Variation of Q-Learning algorithms like Friend-Q, Foe-Q and CE-Q have been proposed to formulate an equilibrium policy for an agent. The purpose of this report is to analyze performance of these algorithms on a simple, zero-sum markov game of perfect information.

2 Background

2.1 Q-Learning

Q function (Q(s,a)) computes the long term expected reward for an agent being in a particular state and executing a particular action. The action that maximizes the Q value has to be selected by the agent at every time-step. The main advantage of Q learning is that it enables itself to be applied to problems that are dynamic, i.e., it doesn't require the agent to be fully aware of the environment. It lets an agent explore and update its belief about the environment from time to time. This make it suitable for Markov games where the opponent's behaviour might not be predefined.
Q-Learning algorithm is an off-policy Temporal Difference Control algorithm which is defined by the following equation:

$$Q(S_t, A_t) \leftarrow (1-\alpha)Q(S_t, A_t) \\ \alpha((1-\gamma)R_{t+1}+\gamma max_a Q(S_{t+1}, a)-Q_t(S_t, A_t)) \tag{1}$$

Where Q(s, a) is the $quality$ of executing an action a from state s by the agent. As the agent takes the long term expected reward into account, there is a discount factor γ which can be fine tuned according to the expected behavior of the agent. Smaller values of γ lead to short sighted policies where the agent only worries about the expected rewards in the near future, as against larger values that lead to the agent being too proactive. Sutton[1] demonstrates that the Q function updated according to (1) ultimately converges to Q_*, the optimal action-value function.
The main disadvantage of the application of multi-agent Q-Learning on a Markov Game setting is that, the algorithm does not take into account the actions and the rewards obtained by the adversary for computing the Q Table. It only aims to find a deterministic optimal strategy which may not be available on a markov game setting.

2.2 Nash Equilibrium and FFQ

Strategies in which no player has the incentive to deviate out of it, are said to be in Nash equilibrium. n players with strategies $S_1, S_2, ..., S_n$ and $S_i^* \epsilon S_i$ are in Nash equilibrium if and only if the following condition is satisfied.

$$\forall_i S_i^* = argmax_{Si} utility_i(S_1^*, ..., S_i, ...S_n*)$$

From the above equation, it can be inferred that no agent has the affinity towards any of other possible strategies if it follows a strategy in Nash equilibrium. A Nash equilibrium will survive elimination of strictly dominated strategies i.e., In an n-player pure strategy game, if the elimination of strictly dominated strategies get rid of all but one combination of strategies, the resulting combination is in Nash equilibrium. For any finite Markov game, there is atleast one Nash equilibrium. FriendQ and FoeQ strategies are used to devise combination of strategies that would preserve Nash equilibrium. An agent using

FriendQ Learning assumes that the other agent acts with a sense of altruism., i.e., the assumption is that other agent will be co-operative. It adds another dimension to the Q-Table it maintains. While traditional Q-Learning maintains its entries against (s, a) pairs, FriendQ maintains a table with entries assigned against the tuple (s, a_1, a_2) where a_1 represents the agent's action and a_2 represents the opponent's action.

$$Q(S_t, A_{t1}, A_{t2}) \leftarrow (1-\alpha)Q(S_t, A_{t1}, A_{t2}) \\ \alpha((1-\gamma)R_{t+1} + \gamma max_{a1,a2}Q(S_{t+1}, a_1, a_2) - \\ Q_t(S_t, A_{t1}, A_{t2})) \quad (2)$$

FoeQ assumes an adversarial opponent. It produces a minimax strategy - i.e., the agent is supposed to choose an action that maximizes its expected long term discounted rewards, assuming that the opponent will truly try to reduce its value.

$$Q(S_t, A_{t1}, A_{t2}) \leftarrow (1-\alpha)Q(S_t, A_{t1}, A_{t2}) \\ \alpha((1-\gamma)R_{t+1} + \gamma max_{a1} min_{a2}Q(S_{t+1}, a_1, a_2) - \\ Q_t(S_t, A_{t1}, A_{t2})) \quad (3)$$

2.3 Correlated Equilibrium and CE-Q

Correlated Equilibrium is a more general form of equilibrium in which the probability distributions of joint action spaces are taking into account. This can be assumed as the markov game being deployed with a mediation party that chooses from a uniform (or other probabilistic) distribution of joint action spaces. Greenwald[2] specifies a traffic example where a system of traffic signals at a junction of two roads act as a form of the mediating party. The action combinations that the traffic lights impose are (STOP, GO) and (GO, STOP). This keeps (STOP, STOP) and (GO, GO) out of the equation as they are less probabilistic. By getting rid of less probabilistic action combinations, the expected rewards for agents are known to increase in certain circumstances, as described in the lectures. Correlated equilibrium, being a convex polytope can be effectively solved by linear programming.
The strategy profile S for players p=1,2,3,...,n is,

$$S = \prod_{p=1}^{n} S_p$$

where S_p represents the strategy of player p. Let $x_{i,\bar{s}}$ denote the probability with which player p takes strategy i and others follow strategy $\bar{s}\epsilon S_{-p}$ where S_{-p} denotes all strategies except for player p. Let $u^p_{i,\bar{s}}$ be the expected payoff for the player p in this setup.
Correlated equilibrium imposes the following constraints on the linear program.

$$\sum_{\bar{s}\epsilon S_{-p}} (u^p_{i,\bar{s}} - u^p_{j,\bar{s}})x_{i,\bar{s}} \geq 0, \forall p \text{ and } \forall i, j\epsilon S_p \quad (4)$$

The probability values must be greater than or equal to 0 and must add up to 1.

$$x_s \geq 0, \forall s\epsilon S \quad (5)$$

$$\sum_{s\epsilon S} x_s = 1 \quad (6)$$

The CE-Q Learning is composed of a set of 4 algorithms, than can enable a system of agents to maintain correlated equilibrium.

S.NO	Type	Maximizes
1	utilitarian	sum of rewards for all agents
2	egalitarian	Min of all agents' rewards
3	plutocratic	Maximum of all agents' rewards
4	dictatorial	Maximum of any agent's reward

3 The Problem

3.1 Soccer Game

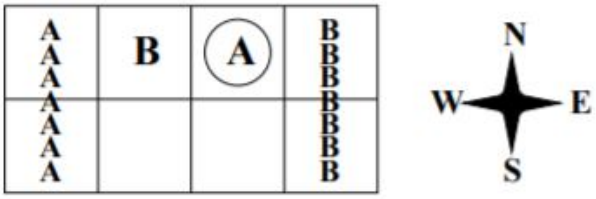

Figure 1: The soccer game

A pictorial representation of the soccer game we are assuming is given in Figure 1. Description of soccer game is given below:

- **States** One of the 8 states in (2 rows and 4 columns) figure 1.
- **Reward** The goal post of B is in column 1 and that of A is in Column 4. Ball reaching any state in column 1 incurs a reward of -100 and any state in Column 4 incurs a reward of

+100 for player A. The game is zero-sum and hence, the reward received is the negative of the reward received by A at any time step.

- **Actions:** The players can execute the following actions:

Action	Displacement
North	[0, -1]
South	[0, 1]
West	[-1, 0]
East	[1, 0]
Stick	[0, 0]

- The players choose their actions simultaneously, but execute them sequentially. Either of Player A or Player B can go first at every time step. The probability of any player executing their action first is 0.5.

- If the players happen to collide by performing a pair of actions, then only the first player is allowed to move. If the player with the ball moves second and then happen to collide, then the ball changes possession.

4 Experiments and Results

4.1 Experiment

For the soccer game discussed in section 3, all the Q-Learning approaches discussed previously were applied. The successive differences in state(s)-action value for the combination (DOWN, STICK) for player1 and player2 was kept track of, and its convergence was monitored over a million iterations. ie.,

$$ERR_i^t = [Q_i^t(s, \vec{a}) - Q_i^{t-1}(S, \vec{a})]$$

Every time the ball hits a goal state, the game was reset to the initial state mentioned in Figure 1.
The main problem with the original article by Greenwald is that the hyper-parameter values and approach to training (handling exploration, exploitation dilemma) have not been mentioned clearly for all 4 algorithms. A single run on all 4 approaches using the same hyperparameter values produced results that were not consistent with that of the original paper. Therefore, every algorithm has been run on a different set of parameters as described in the subsequent sections.

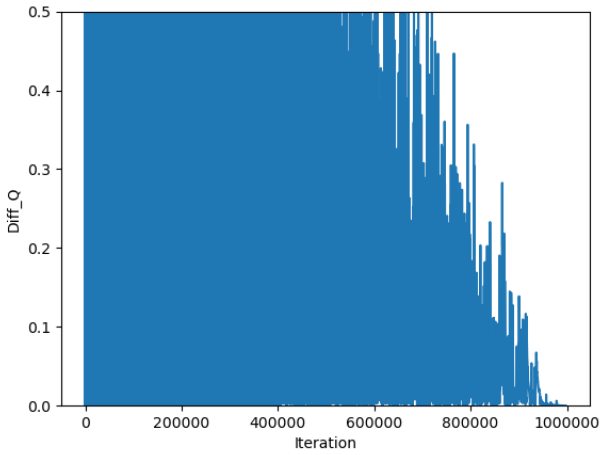

Figure 2: Q-Learning ERR_t^i graph

4.2 Q-Learning

Multi-agent Q-Learning approach applied to the game produced the ERR_t^i graph in Figure 2. The implementation assumed the following set of parameters:

- Learning rate - $\alpha_{start} = 1$, $\alpha_{min} = 0.001$, Linear decay of α, with $\delta\alpha = (\alpha_{start} - \alpha_{min})/10^6$

- Exploration-exploitation probability - $\epsilon_{start} = 1$, $\epsilon_{min} = 0.001$, Linear decay of ϵ, with $\delta\epsilon = (\epsilon start - \epsilon min)/10^6$

- Discount factor - $\gamma = 0.9$

Multi-agent Q Learning was used to obtain a deterministic optimal strategy for player A. From Figure 2, it can be seen that the algorithm does not work well. ERR_t^i decreases with increase in the number of iterations, but does not converge. The non-convergence does not work well because of the fact that the algorithm does nothing to learn about the player's adversary. It aims to identify a deterministic optimal policy which does not seem to exist for the given game.

4.3 FriendQ

The ERR_t^i plot for FriendQ is shown in Figure 3. The following parameter values have been used for implementation:

- Learning rate - $\alpha_{start} = 1$, $\alpha_{min} = 0.001$, $\alpha_{decay} = 0.995$

- Discount factor - $\gamma = 0.9$

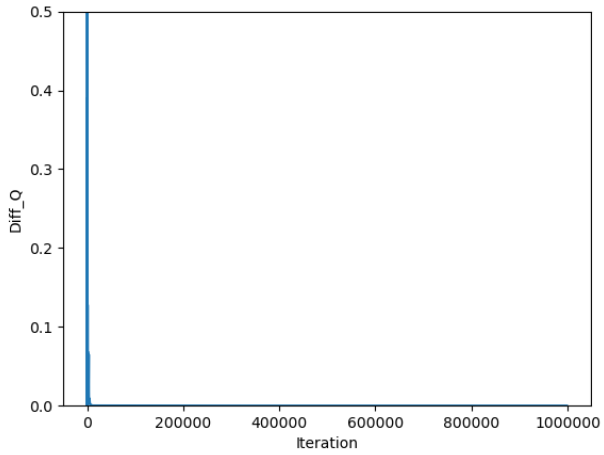

Figure 3: FriendQ ERR_t^i graph

The implementation assumed a constant ϵ value of 1 during all iterations, which means that the agent always explored during the learning process of one million iterations. It can be seen that the algorithm converged to a deterministic strategy quite quickly. Both of the agents are cooperative and worked towards increasing the expected reward for player A. The ERR_t^i was logged everytime, the pair of actions turned out to be (STICK, EAST). The plot closely resembles the one in Greenwald's paper, although the convergence seems to have happened faster. The more similar plot could have been obtained by adjusting the values of α, α_{decay}.

4.4 FoeQ

The ERR_t^i plot is displayed in Figure 4. The following parameters were assumed for the purpose of implementation:

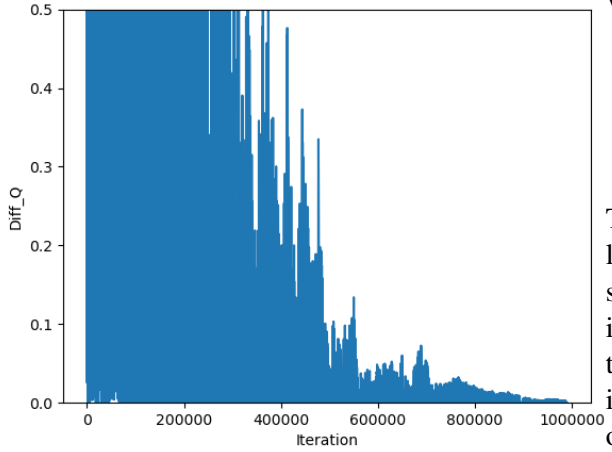

Figure 4: FoeQ ERR_t^i graph

- Learning rate - $\alpha_{start} = 1$, $\alpha_{min} = 0.001$, $\alpha_{decay} = 0.999995$
- Discount factor - $\gamma = 0.9$

The agent is assumed to only explore during the learning process ($\epsilon = 1$). FoeQ is a minimax strategy learning algorithm. A system of linear equations was constructed using the Q-Table for a given pair of states (one for each player) and were solved to obtain the state-values to be used in equation (3). It can be seen that the plot closely resembles to that of the Greenwald's paper.

4.5 uCEQ

The ERR_t^i plot from the implementation of CEQ is shown in figure 5. The following parameters were assumed for the purpose of implementation:

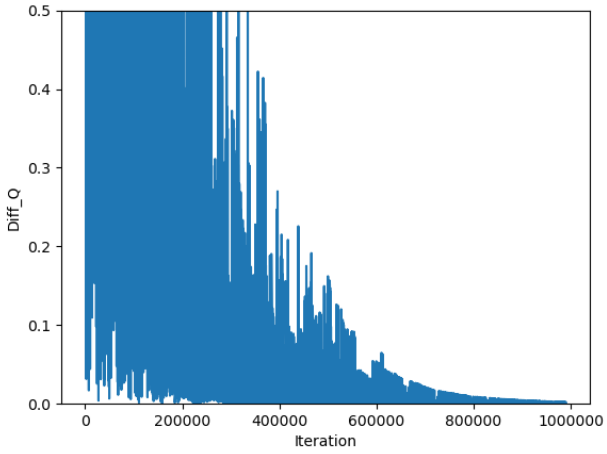

Figure 5: CEQ ERR_t^i graph

- Learning rate - $\alpha_{start} = 1$, $\alpha_{min} = 0.001$, $\alpha_{decay} = 0.999993$
- Discount factor - $\gamma = 0.9$

The agent is assumed to only explore during the learning process ($\epsilon = 1$). CEQ assumes that the set of actions are signalled to the players by using a third party that probabilistically chooses actions from the joint action spaces, which leads to increase in the expected reward obtained by the set of players in equilibrium. A linear program was constructed using a system of equations similar to that FoeQ, but with additional constraints encoded

by the equations 4, 5, 6. It can be seen that the plot looks similar to that of the Greenwald's paper. It can be seen that, for the given hyperparameters that are similar, both FoeQ and CEQ begin to converge after almost the same number of runs.

5 Conclusion

This report explored the concept on Q-Learning on a simple zero-sum game. It analyzed the pitfalls of generalized multi-agent-Q, demonstrated FriendQ and FoeQ methodologies of attaining Nash equilibrium. It finally introduced the concept of correlated equilibrium and CEQ-learning which has been shown to produce almost the same rate of convergence as the FoeQ minimax algorithm. The graphs are quite close to the ones in Greenwald's paper, with minor deviations, possibly due to selection of hyparameters.

6 References

1. Richard S. Sutton and Andrew G. Barto. Introduction to Reinforcement Learning (2st. ed.). MIT Press, Cambridge, MA, USA.

2. Greenwald, Amy, Keith Hall, and Roberto Serrano. "Correlated Q-learning." ICML. Vol. 20. No. 1. 2003.

3. Littman, Michael L. "Friend-or-foe Q-learning in general-sum games." ICML. Vol. 1. 2001.