Correlated Q Learning:
On Multiagent Reinforcement learning in a Zero Sum Markov Game

Manikanta Reddy Dornala[1]
CS 7642 - Reinforcement Learning, Georgia Institute of Technology

Multi-Agent games are particularly interesting as the best way to play them is guided by an equilibrium that heavily depends on the payoff each agent receives. The celebrated one of such equilibria is the Nash Equilibrium. Here we study a generalization of multi-agent Q-Learning methods, Correlated Equilibrium which encompasses Nash-Q and Friend-Foe-Q. We apply the algorithms to a simulated game of Soccer, a Zero Sum Game for which no deterministic equilibrium policies exist.

I. INTRODUCTION

Reinforcement learning enables control agents to learn different strategies to interact with the environment through a sequence of observation, decisions, and reflections. The goal of any such agent is to observe the current state of the environment and take an action that maximizes the expected cumulative future reward. The mathematical representation of this statement is given by the Bellman equation for Q-value as

$$\begin{aligned} V^*(s) &= \max_{a' \in A(s)} Q^*(s', a') \\ Q^*(s,a) &= \sum_{s'} T(s,a,s')(R(s,a) + \gamma \max_{a'} Q^*(s', a')) \end{aligned} \tag{1}$$

The optimal policy or the optimal decision function can then be obtained by maximizing the Q-value function.

$$\pi^*(s) = \operatorname*{argmax}_a Q^*(s,a) \tag{2}$$

The Q-value can be approximated by the Q-Learning[?] algorithm that iteratively updates Q-values in a Q-table which converge to true optimal values over time.

Algorithm 1: Off Policy Q Learning (ϵ Greedy)

Initialize: Q(s,a), $\pi(s)$ and an initial state s
while *policy not converged* **do**
  $a = \begin{cases} random(\mathcal{A}ction) & \text{with probability } \epsilon \\ \max_{a'} Q(s', a') & \text{otherwise} \end{cases}$
  Take action a and then observe new state s'
  $y = R(s,a) + \gamma \max_{a'} Q(s', a')$
  $Q(s,a) = Q(s,a) + \alpha(y - Q(s,a))$
  $\pi(s) = \operatorname{argmax}_a Q(s,a)$, s = s'
  decay α, ϵ
end

Note that update for a time(t) depends on the observation from time(t+1) ($\max_{a'} Q(s', a')$). Δ at any time(t) doesn't depend on the the action to be taken at time($t+1$), which will be dependent on the policy(π) at $t+1$. Hence the term **Off** Policy. In an **On Policy** Q Learning the $\max_{a'} Q(s', a')$ is replaced by simply $Q(s', a)$.

The above algorithm is suited to Single-agent problems. In order to apply the Q-Learning algorithm can be modified in multiple ways, mostly changing the Value function.

II. MULTI AGENT Q LEARNING AND GAMES

A multi-agent problem can be modeled as a game of rewards. These games force the agents to evolve a strategy to decide their policy. The strategies themselves are based off on Equilibrium conditions set forth by their payoffs. This can be done by simply modifying the Value function for each agent in the algorithm.

$$V_i(s) = f(Q_1, Q_2, ..., Q_3) \tag{3}$$

For a simple two agent zero sum game, the Friend Q^1 algorithm is given by

$$\begin{aligned} V_1(s) &= max_a Q_1(s,a) \\ V_2(s) &= max_a Q_2(s,a) \end{aligned} \tag{4}$$

This is suited to coordination games with unique equilibrium as each agent tends to expect the other agent will perform an action that'll be beneficial to him.

On the other hand is the Foe Q tries to maximize the minimum of an agent's expected reward. This makes him aggressive towards the other player.

$$\begin{aligned} V_1(s) &= max_{\sigma_1 \in \Sigma_1(s)} min_{a_2 \in A_2(s)} Q_1(s, \sigma_1, a_2) = -V_2(s) \\ Q(s, \sigma_1, a_2) &= \sum_{a_1 \in A_1(s)} \sigma_1(a_1) Q(s, a_1, a_2) \end{aligned} \tag{5}$$

For a n-player, general sum game. we redefine the Value function as follows.

$$V_i(s) \in NASH(Q_1(s), Q_2(s), ..., Q_n(s)) \tag{6}$$

Notice the $\in$ operator, as there could be multiple equilibrium positions, we have to choose one. In Zero Sum Games, the Nash Equilibrium and Minimax strategy coincide. But the above replacements are case specific. In order to generalize Greenwald proposed an alternative definition in Markov games.[2]

$$V_i(s) \in CE_i(Q_1(s), ..., Q_2(s)) \tag{7}$$

CE_i denotes the i^{th} player's reward according to ***some*** *Correlated Equilibrium* in the general sum game determined by the rewards. This definition encapsulates all the above replacements as well. Since a Nash Equilibrium is also a Correlated Equilibrium.

III. CORRELATED EQUILIBRIUM

A Nash equilibrium simply put is a probability distribution over actions generated such that the agents optimize with respect to one others probabilities. Othe other hand, Correlated Equilibrium is a more general equilibrium which permits dependencies among multiple agents probabilities, all while individual agents optimize. One might ask why choose CE over Nash. It's not because it is more general but because CE is easily computable for multi-agent problems via Linear Programming as opposed to Nash.

A famous illustration of CE is given by the traffic lights example. consider two cars that are at a junction on opposite ends. They can both now take actions $(GO, STOP)$ If both take GO they crash and receive a -10 each. If one takes a GO and the other a STOP, the one that takes a GO receives a 5 and the other a 1. If both wait by taking a STOP, both receive a -1.

Since a Nash Equilibrium is one where no agent has any incentive to change its strategy. This game has two Nash Equilibrium, A1 take GO and A2 takes STOP or A1 takes STOP and A2 takes GO, but both prefer a GO. Hence will crash or will wait indefinitely,

A CE is attained when a player takes an action (A) receives a reward that is at least that of taking any other action (not A) assuming the player takes the first action (A).

A CE would give a joint probability distribution over these two essentially forcing the game into one of its Nash Equilibrium. Much like a fair traffic light, that sometimes STOPs the first agent and sometimes the second agent.

	2. GO	2. STOP
1. GO	-10, -10	3, 1
1. STOP	1, 3	-1, -1

The Correlated equilibrium is given by solving the following equations

$$\begin{aligned}
&\pi_{GG} \geq 0, \pi_{GS} \geq 0, \pi_{SG} \geq 0, \pi_{SS} \geq 0 \\
&\pi_{GG} + \pi_{GS} + \pi_{SG} + \pi_{SS} = 1 \\
&R^1_{GG} * \pi_{GG} + R^1_{GS} * \pi_{GS} \geq R^1_{SG} * \pi_{GG} + R^1_{SS} * \pi_{GS} \\
&R^1_{SG} * \pi_{SG} + R^1_{SS} * \pi_{SS} \geq R^1_{GG} * \pi_{SG} + R^1_{GS} * \pi_{SS} \\
&R^2_{GG} * \pi_{GG} + R^2_{SG} * \pi_{SG} \geq R^2_{GS} * \pi_{GG} + R^2_{SS} * \pi_{SG} \\
&R^2_{SS} * \pi_{SS} + R^2_{GS} * \pi_{GS} \geq R^2_{SG} * \pi_{SS} + R^2_{GG} * \pi_{GS}
\end{aligned} \tag{8}$$

solving the above set, without any other constraints gives

$$\begin{aligned}
\pi_{GG} &= \pi_{SS} = 0 \\
\pi_{SG} &= \pi_{GS} => 0.5
\end{aligned} \tag{9}$$

Well its a fair traffic light! Half of the time it allows Player 1 to GO and the other to STOP and rest of the time the other way. But it never allows the players to take the same action at the same time which is worse for both.

IV. CE Q LEARNING

There are 4 variants of CE Q Learning where further conditions ensure the equilibrium is unique.

1. Maximize the sum of Players rewards (Utilitarian $uCE - Q$)

$$\sigma \in \underset{\sigma \in CE}{\operatorname{argmax}} \sum_{i \in I} \sum_{a \in A} \sigma(a) Q_i(s, a)$$

2. Maximize the minimum of player's rewards (Egalitarian $eCE - Q$)

$$\sigma \in \underset{\sigma \in CE}{\operatorname{argmax}}\, min_{i \in I} \sum_{a \in A} \sigma(a) Q_i(s, a)$$

3. Maximize the maximum of player's rewards (Republican $rCE - Q$)

$$\sigma \in \underset{\sigma \in CE}{\operatorname{argmax}}\, max_{i \in I} \sum_{a \in A} \sigma(a) Q_i(s, a)$$

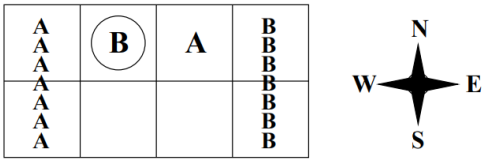
4. Maximize the maximum of *individual* player's rewards (Libertarian $lCE - Q$)

$$\begin{aligned}
\sigma &= \prod_i \sigma^i \\
\sigma^i &\in \underset{\sigma \in CE}{\operatorname{argmax}} \sum_{a \in A} \sigma(a) Q_i(s, a)
\end{aligned} \tag{10}$$

Now $CE_i(Q(s)) = \sum_{a \in A} \sigma(a) Q_i(s, a)$ where σ is obtained by either of the four conditions.

V. SOCCER GAME

To study the behavior of the Correlated Q Agents we implement a simple Two Player Soccer Game. The game is a simple $2x4$ grid with the state determined by the possession of the ball by the player. There are five actions (N, E, W, S, Stay) available for each player. A reward of 100 is awarded when a player either scores or the other player self-goals. In either case, the opponent receives a -100 and then the game ends. There are more rules regarding stealing the ball, collision, and others.



There exists a special state exists as depicted in the image where an action South is taken by A and player B sticks. We base our experiments on the Q value of this particular state action pair.

VI. EXPERIMENTS

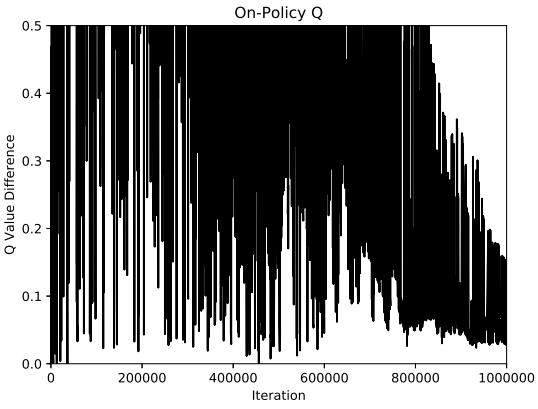
Each agent is simulated for a 10^6 iterations and the error is noted as the absolute difference between the previous Q value of the special state action pair with the current Q value. Player A from here on will be our protagonist.

There was initially a bit of confusion regarding the definition of iteration, whether it was the number of episodes of games simulated or the total number of steps simulated. We initially took the pain of simulating the experiment for a million episodes a lot more than what's necessary and found the graphs to be converging with a tenth of what was expected.

That is when a careful analysis revealed that on an average a game completes in 10 steps and after 10^5 games there isn't anymore learning whatsoever.

A. On Policy ϵ greedy Q Learner

As stated earlier for an On Policy Q Learner the update is given by $R(s,a) + \gamma Q(s',a)$. The agent assumes the problem to be a single agent and updates its own Q-table without worrying about anything else only by observing the outcomes of the actions it takes at different states.

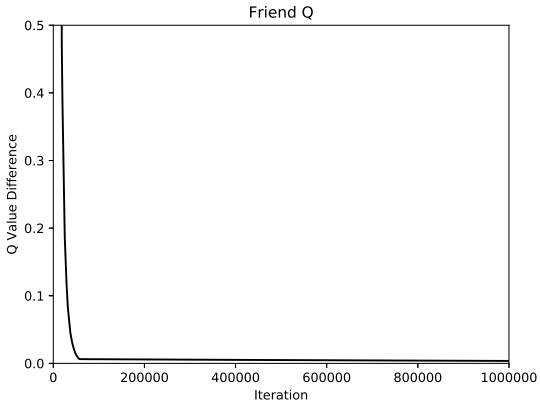


This agent diverges as there is no general deterministic solution to solve this problem. The convergence towards the end is solely due to low α due to decay. The Q Learning agent has a decaying α tending to 0.001 by the end.

The original 2003 paper doesn't mention how the α decays, but the subsequent 2005 paper mentions that α decays as the inverse of number of visits to the state action pairs, but then the Q Learning graph doesn't match with the original (In 2005 the plots are for $2 * 10^5$ iterations only. So we choose to simply decay it exponentially to 0.001 with a decay rate of $0.001^{1/10^6}$ 0.999999. Varying the function of decay rate drastically changes the way the plot behaves, suggesting that the original decay rate is very specific.

B. Friend Q Learner

In the friend Q learning algorithm, we assume that the second player always takes actions that will help our player. This can be implemented by a simple Off Policy Q Learner that takes in the second player's action as input as well. Since we are only trying to find the best Q values both the players always take random actions irrespective of player A's Q table.



As expected Friend-Q quickly converges. This is because of our assumption that player B will score the goal for us, we just have to pass it on to B in our special state. Such Optimism much wow.

Interesting varying the decay rate of alpha, in this case, doesn't offer much. No matter what it is, the algorithm quickly converges within a 10^5 iterations or even faster but no later.

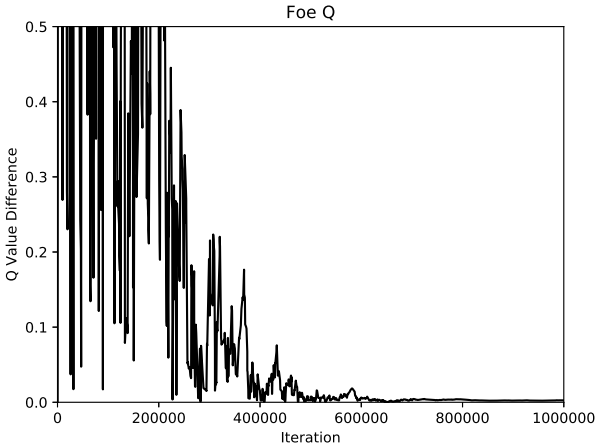
VII. FOE Q

As mentioned earlier Foe Q both players try to minimize the other player's returns. This in-turn implies that each other value functions are simply negations of the other. So we need to only maintain one Q Table for player A and we also know the Q table for B. The minmax objective is obtained by solving a set of constraints via linear programming.

$$\begin{aligned} maximize &: c^T x \\ subject\ to &: Ax \leq b \\ &: x \geq 0 \end{aligned} \tag{11}$$

$$A = \begin{pmatrix} 1 & Q_{NN} & Q_{NS} & Q_{NE} & Q_{NW} & Q_{NS_t} \\ 1 & Q_{SN} & Q_{SS} & Q_{SE} & Q_{SW} & Q_{SS_t} \\ 1 & Q_{EN} & Q_{ES} & Q_{EE} & Q_{EW} & Q_{ES_t} \\ 1 & Q_{WN} & Q_{WS} & Q_{WE} & Q_{WW} & Q_{WS_t} \\ 1 & Q_{S_tN} & Q_{S_tS} & Q_{S_tE} & Q_{S_tW} & Q_{S_tS_t} \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & -1 & -1 & -1 & -1 & -1 \end{pmatrix} b = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ -1 \end{bmatrix} c = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{12}$$

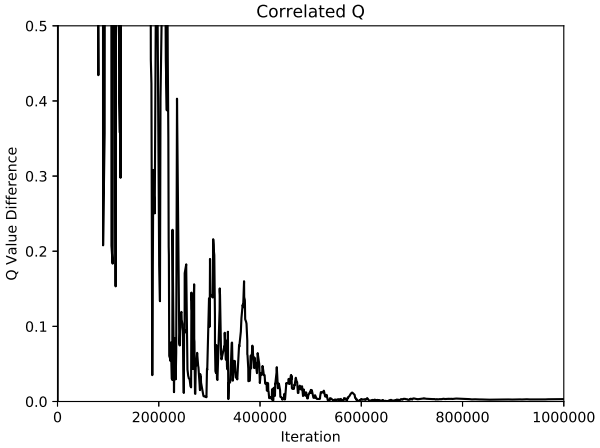
The solution x, if converged, would be the probabilities of taking a particular action. This implies that FoeQ generates a *Non-Deterministic* policy at state S.



Foe-Q converges and produces a mixed strategy for the agent. The mixed strategy is such that the opponent has the best strategy.

VIII. CE Q

CE Q is very similar to Foe Q except that we now maintain 2 tables one for each player and the set of constraints is far higher. The paper mentions that all the equilibria are equivalent and went with the utilitarian CE. We are now optimizing the joint expectation of the sum of Q values of both the players. Following in the lines of traffic game we'll have 20 constraints per player and a 25 positivity constraints. Overall the matrix A is now of shape $67x26$. The c vector is simply -1 with negative of sum of individual Q values for every action A, action B pair.



The correlated equilibrium will again generate a nondeterministic policy in case of convergence with a probability distribution over both the agents actions a $(5x5 =)25$ dimensional vector.

In some sense, CE Q provides a *rational* distribution by placing a constraint the current actions expected value is always more of equal to the conditional expectation of the value of other actions given that this action is taken. This increases the probability of an action to be taken conditional the given action.

Both Correlated-Q and FoeQ are supposed to be the same, as all equilibria are same for this problem. However, due to random initializations, it seems that both of them vary a bit in the beginning while they are diverging. But after 200K iterations, when the convergence is inevitable, both the algorithms produce the same updates and the errors converge and produce the same mixed policy in the end.

It is important the both the algorithms have the same initial α and same decay rate to produce matching graphs. This is because they produce the same update but the rate it which it is applied is on us and it should match for the differences to match.

IX. CONCLUSION

This is one of those experiments where reproduction is particular challenging due to vague description of hyperparameters. To add to the misery the 2005 version of the paper does things differently and produces entirely different set of results although the general trend remains same.

X. REFERENCES

[1]M. L. Littman, in *ICML*, Vol. 1 (2001) pp. 322–328.
[2]A. Greenwald, K. Hall, and R. Serrano, in *ICML*, Vol. 3 (2003) pp. 242–249.