Prasad Vaidya **Project 3:- Correlated Q learning** CS 7642

Latest Github (git #) commit: 183f1c7deb195651c5ddf8849131000095938c20 April 14, 2021

Introduction & Theory

This paper attempts to develop strategies for two-player zero sum Markov game. A two-player zero-sum game is a specific kind of game in which the value received by one player is opposite to the value received by the opponent (sum of their values is zero, hence zero-sum), while a Markov game is stochastic game in which the probability transitions for different states and actions of games can be modeled by a Markov Process. Hence, a two player zero sum Markov game, is special kind of game wherein the game played between the two players is a zero sum game and the probability transitions for the game satisfy Markov property.

However, lets begin with a simple one player Markov game before attempting to learn for two players Markov game. Markov Decision Process (MDP) is a single player Markov game [1]. MDP is characterized by state value function (V(s)) & state-action value function (Q(s,a)) which are used to develop policy that dictates what action to take in which state. Q(s,a) can be learnt recursively based on the Bellman Optimality Equation as shown below:

$$Q(s,a) = r(s,a) + \gamma \times \max_a \times Q(s',a) \tag{1}$$

$$V(s) = \max_a Q(s',a) \tag{2}$$

$$\sigma(s) = \arg\max_a Q(s',a) \tag{3}$$

In the above equations, s is current state, a is the action taken, s' is next state and γ is discount factor while σ(s) represents the policy which maps state s to what action to take. The Bellman equation along with temporal difference learning (especially TD(0)) enables finding Q(s,a) iteratively which is called Q-learning and is given by (4).

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \times V(S_{t+1}) - Q(S_t, A_t)] \tag{4}$$

Q-learning is optimal for solving MDP which is one player game, and so it may not be sufficient to develop strategies for two-player zero sum Markov game. Nevertheless, the concepts of single player game (MDP and Q-learning) can be extended to two players game. Intuitively, both players will try to maximize their own gains. But, such actions may not exist in a general game settings. To account for such complications, the single player learnings can be enhanced with concepts from game theory. Game theory utilizes concept of equilibrium for solving general sum games. Basically, equilibrium is pair of strategies adopted by all the players playing game within which each players' strategy is optimal to herself when comparing the decisions of other players. This paper looks at 3 equilibrium.

The first equilibrium considered in this paper is Friend Equilibrium. Friend equilibrium is applicable to a special case of general sum game called coordination game wherein sets of action exist so that all players can achieve the same maximum reward [3].For such games, the equilibrium dictates that all players play in coordination, hence the name Friend Equilibrium. The Friend Equilibrium can be learnt using the same methodology as iterative Q-learning (4) by slightly modifying its value function (2) to (5) and its policy (3) to (6) [3].

$$V(s) = \max_{\overrightarrow{a} \in A(s)} Q_i(s', \overrightarrow{a}) \tag{5}$$

$$\sigma(s) = \arg\max_{\overrightarrow{a} \in A(s)} Q(s', \overrightarrow{a}) \tag{6}$$

The second equilibrium considered in this paper is called Foe Equilibrium. Foe equilibrium is also applicable to special case of general sum games called zero sum games wherein the players play with a goal to minimize the opponent's gain (opponent's maximum pay-off). Such types of games are solved using minimax algorithms [3]. Like friend equilibrium, foe equilibrium can also be learnt via the iterative Q-learning (4) by modifying its value function (2) to (7) and its policy (3) to (8) [3].

$$V_1(s) = \max_{\sigma_1 \in \Sigma_1(s)} \min_{a_2 \in A_2(s)} Q_1(s, \sigma_1, a_2) = -V_2(s) \tag{7}$$

$$\sigma(s) \in \arg\max_{\sigma \in foe} \min_{i \in I} \sum_{\overrightarrow{a} \in A} \sigma(\overrightarrow{a}) Q_i(s, \overrightarrow{a}) \tag{8}$$

The third equilibrium explored in this paper is correlated-equilibrium (CE) [1]. CE is probability distribution on joint action space in which all agents optimize their actions with respect to one another probabilities conditioned on their own. The equilibrium is set in such way that the expected reward for one player playing an action 'A' should always be at least the reward for playing any other action by same player. This condition applies to both players. To understand CE, lets consider chicken example as explained by Greenwald [1]:

P1/P2	L	R
T	6,6	2,7
B	7,2	0,0

The table on left shows the reward matrix with first value for P1 and second value for P2 for playing respective actions. For instance, when P1 plays T and P2 plays L than both players get reward of 6. The CE than dictates that the equilibrium that satisfies the condition that if P1 plays T its expected reward is at least that of playing B which is equivalent to:

$6\pi_{TL} + 2\pi_{TR} \geq 7\pi_{TL} + 0\pi_{TR} = -1\pi_{TL} + 2\pi_{TR} \geq 0$, where π_{TL} is the joint probability that P1 will play T P2 will play L. Similarly, based on other values in table including P2's values, CE satisfies other conditions as well:

$-1\pi_{TL} + 2\pi_{BL} \geq 0$, $1\pi_{BL} - 2\pi_{BR} \geq 0$ and $1\pi_{TL} - 2\pi_{BR} \geq 0$

These rationality conditions along with value functions gives solution (policy) to CE. This paper explores three CE's based on differences in value functions. 1) Utilitraian CE (uCE: Maximize sum of players' rewards) 2) egalitarian CE (eCE: Maximize the minimum of players' rewards) & 3) republican CE (rCE: Maximize the maximum of players' rewards). All equilibrium are learnt via modifying the Q-learning, the policy and value function equations as shown below. The equations 9,10 & 11 represent the equations for policy for uCE, eCE and rCE respectively and the same modification were applied to value function.

$$\sigma(s) \in \arg\max_{\sigma \in uCE} \sum_{i \in I} \sum_{\vec{a} \in A} \sigma(\vec{a}) Q_i(s, \vec{a}) \tag{9}$$

$$\sigma(s) \in \arg\max_{\sigma \in eCE} \min_{i \in I} \sum_{\vec{a} \in A} \sigma(\vec{a}) Q_i(s, \vec{a}) \tag{10}$$

$$\sigma(s) \in \arg\max_{\sigma \in rCE} \max_{i \in I} \sum_{\vec{a} \in A} \sigma(\vec{a}) Q_i(s, \vec{a}) \tag{11}$$

Methods

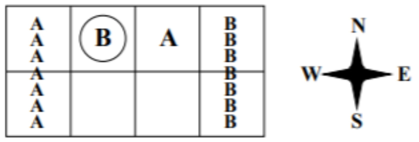

Figure 1: Soccer Field representing state s (Reproduced from Greenwald et al [1])

The paper mimics the experiments performed by Greenwald [1] for soccer and compares the results. As shown in Figure1, the soccer field is a 2 X 4 grid world with two players A & B (circle around B denotes B possesses ball). Within the soccer field, both the players cannot occupy the same grid location. Hence, there are total 56 (8 X 7) unique feasible combinations of grids that can be occupied by A & B and within each combination either A or B can possess the ball. Hence, for soccer field there are 112 unique states (56 X 2 - location combination X possession). Within each state, the agent can choose 5 actions (go north, east, west, south or stick) as shown in the Figure 1. The player's actions are chosen in random order. The rules for actions are: if executed action takes player out of the grid, than the player will stay at its location (For instance, if the players are in state s and the action recommended is North than the players will stay at the same grid location). On the other hand, if the sequence of actions causes collision than only the first player moves. However, during the collision if the player with ball moves second, than the ball changes its possession. Both the players will keep executing the moves recommended by their policies and governed by the above rules until a goal is scored (entering the goal state with ball possession is considered as scoring goal). As shown in Figure 1, the goal for player A are top and bottom leftmost grids while the goal for player B are top and bottom rightmost grids. If the player possessing ball scores the goal in her designated goal state, she receives a reward of + 100 points while the opponent receives reward of - 100 points. However, the if player scores goal in opponent's goal state, the player will receives a reward of - 100 and the opponent player will receive a reward of +100. Irrespective, of whether the goal is scored in self or opponent's goal state, the game terminates when the goal is scored. A new game can be restarted after termination. However, the new game is restarted in an arbitrary location for A & B (the locations excludes goal states) and the possession of the ball is also randomly assigned.

Utilizing the above game dynamics, four agents (Q-learning, Friend-Q, Foe-Q and Correlated Equilibrium-Q (CE-Q)) were trained to determine the optimum policy for each state. All the agents were trained against themselves (i.e. if agent A is playing Foe, Agent B will also play Foe) for 1,000,000 games. During training for all the agents, the initial learning rate was set to 1 and the learning rate was decayed at rate of 0.999993 per game till the minimum learning rate of 0.001 was reached and than the learning rate was held constant for remaining games. During each step of the game, what action to take was determined by an epsilon-greedy policy, wherein the initial epsilon was set to 1 with a decay rate of 0.9999954 per game, till the minimum value of 0.01 was reached after which this value was held constant for rest of the games. The discount factor (γ) of 0.99 was chosen for training all the four agents. Finally each agent was trained according to its functional representations (Q-learning was learnt according to (2) & (3), Friend-Q was learnt by (5) & (6) , Foe-Q was learnt by (7) & (8) and CEQ was learnt by 9-11).

The training for Q-learning and Friend-Q required taking max and argmax at each training step which was relatively easy. However, given that Foe equilibrium is solved by minimax algo, Foe-Q was trained at each step using linear programming. Similarly, all the three CEQs were also trained using linear programming. Given that the variables of linear program represent probability of what action to take in what state, both implementations (Foe-Q and CEQ) had some common constraints such as sum of all variables should be 1 (sum of probabilities = 1) and all variables ≥ 0 (probability can't be negative). However, apart from common constraints, each implementation had constraints specific to them which were dictated by what they were trying to optimize (in other words dictated by their value function as explained above). The Foe-Q used the rationality constraints recommended by Littman [2] for rock, paper & scissors (RPS) game (except the rationality constraints were extended to 5 variables from 3 variables and the reward matrix was replaced by Q-values from previous iteration). Similarly, the rationality constraints for CEQ, were similar as above for "Chicken Two Player Game" (except the rationality constraints were extended for 25 variables from 4 variables and the reward matrix was replaced by Q-values from previous iteration). In addition, each CE-Q had few more constraints specific to their value functions. Once, the agents were trained for 1,000,000 games, the Q-tables were considered as final Q-values and policy for each state for each learner was determined using their respective functional representations. After obtaining the final policies for each agent for both players, 10,000 soccer games were played with these agents against each other. During the games, it was observed that in some games, the player movement was restricted to certain grids and the game was not proceeding ahead. Hence, during evaluation phase, the maximum number of steps allowed for both players combined together were restricted to 1000. If no result was obtained in 1000 total steps, the game was terminated and scored as "Tie" game.

Results & Discussions

The paper tries to mimic the soccer experiment performed by Greenwald. Four different agents (as shown in figure 2) were trained on the soccer environment for 1,000,000 games. Figure 2 shows the difference in Q-values (error) after each game for player A. The error in Figure 2 (Friend-Q, Foe-Q and uCE-Q) are difference in Q-values for player A for being in state s and taking action South while player B taking action Stick. For Q-learning, the error represents the difference in Q values for agent A being in state s and taking an action South.

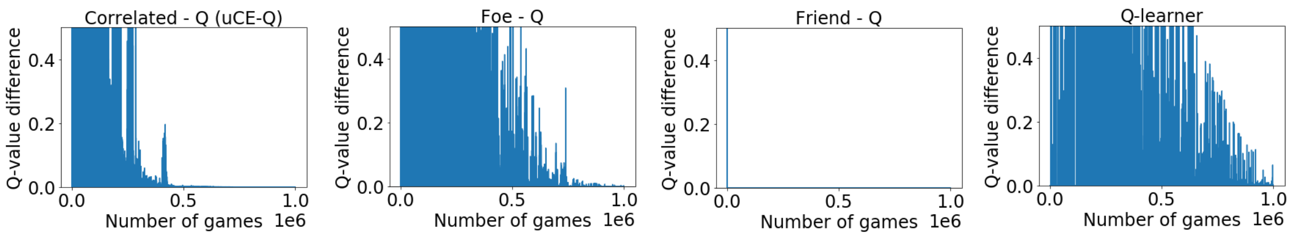

Figure 2: Change in Q-values of state s per game while training for different agents

It can be observed from Figure 2 that the Q-learning did not converge. Given that Q-learning optimizes single player game, it doesn't account for the opponent's actions and hence it's seeking to learn deterministic policies causing non-convergence. Table 1A shows the policy for Q-learning for both players in state s. Both players try to take deterministic actions with player A choosing to Stick & player B choosing to go South. This is anticipated as both players are trying to maximize effect of their own actions without considering opponents actions. And so from A's point of view (pov), she should Stick to block B and from B's pov she should go South to avoid being blocked as both are not considering opponent's actions. While stochastically taking both these actions are good for state s as will be observed in subsequent sections, these two agents are taking these actions in deterministic manner. Also, it can be observed from Figure 2, that the difference in Q-values is decreasing as the number of games is increasing. This observation is due to decline in the learning rate as opposed to actual convergence.

On the other hand, Figure 2 shows that the Friend-Q converges within very small number of games. Similar to Q-learning, Friend-Q also learnt deterministic policies. Furthermore, due to Friend-Q's assumption of coordination game, it learnt incorrect policies. As seen from Table 1A, in state s player B has a deterministic policy of going East. This is because player B falsely assumes that player A is friend and will score goal on her behalf, and so B's policy is to go east and thus will end up giving the possession of the ball. On the other hand, for state s, player A has a policy to take action North which results in player A staying where she was because the game dynamics dictate that whenever an action causes player to move out of grid, player stays back. In reality the Q-values for all A's actions were very close for state s and so it didn't matter A's action but it chose North due to argmax. Here, A also learnt incorrect action because she assumed that B will score goal on her behalf and so she was indifferent to the action.

Figure 2 also denotes that like Friend-Q, Foe-Q also converged. However, Foe-Q took more than 600K games to converged. As opposed to Friend-Q & Q-learning, Foe-Q learnt stochastic policies. It can be seen from Table 1A that in state s, player A chooses action to Stick (North = sticking for state s, because of the way game dynamics are set up and so A can't go out of grid and will stick back to the same location) with probability of 0.51 while decides to take action South with probability of 0.49. Player B decides to stick with probability 0.53 while decides South action with probability 0.47. Thus the policy that both players learn for state s under Foe-setting is stochastic. This policy makes sense because at state s if player B undertakes deterministic actions than it can lead to forever blocking. Hence, taking stochastic policy in this state, can give B a chance to pass player A and vice versa. Also, as stated in theory, soccer is a zero-sum two player game, which is typically solved using minimax algorithm. Hence, the stochastic policy learnt by Foe-Q algorithm should be optimal equilibrium policy as it uses minimax too.

Table 1A Probabilities of actions in state s

Algorithm	Players	P(North)	P(East)	P(West)	P(South)	P(Stick)
Q learning	A	0	0	0	0	1
	B	0	0	0	1	0
Friend Q	A	0	1	0	0	0
	B	0	1	0	0	0
Foe Q	A	0.51	0	0	0.49	0
	B	0	0	0	0.47	0.53
uCE-Q	A	0.24	0	0	0.54	0.22
	B	0.27	0	0	0.45	0.28
eCE-Q	A	0.22	0	0	0.55	0.24
	B	0.29	0	0	0.44	0.27
rCE-Q	A	0.25	0	0	0.52	0.24
	B	0.27	0	0	0.47	0.27

Table 1B Number of Wins for Player A in 10K games

PlayerA ↓ /*PlayerB* →	Random	Q-learning	Friend-Q	Foe-Q	UCE-Q
Random	5011	1729	2215	1849	1804
Q-learning	8147	3503	3671	3746	5081
Friend-Q	7622	4019	5606	4201	4145
Foe-Q	8203	4968	5771	4671	5005
UCE-Q	8276	4962	6040	4897	4992

Similar to Foe-Q, the utilitarian CE-Q also converged. As seen from Table 1A, uCE-Q learnt stochastic policies for state s. Specifically, for player A, uCE-Q learnt to take action of Sticking with probability 0.46 (P(North) + P(stick)), and probability of action of going South was 0.54. Similarly, for player B, uCE-Q learnt to take action with probability of sticking (0.55) and probability of going South (0.45). The policy for uCE-Q in state s is very similar for Foe-Q. This statement can be further extended to all states as the policies & Q-values for Foe-Q and uCE-Q are very close to each other. This implies that uCE-Q learnt the same policy as Foe-Q (which is the optimal solution for zero-sum game). Also, in their paper Greenwald [1] demonstrated that in case of coordination game (GG1) CE-Q learned the same Q-values as Friend-Q (which should be correct as its a coordination game) and also showed in the game, the CE-Q agents were able to collaborate well and get max rewards. This observation indicates that CE-Q is a really powerful algorithm and can learn both adversarial as well as coordination policies efficiently.

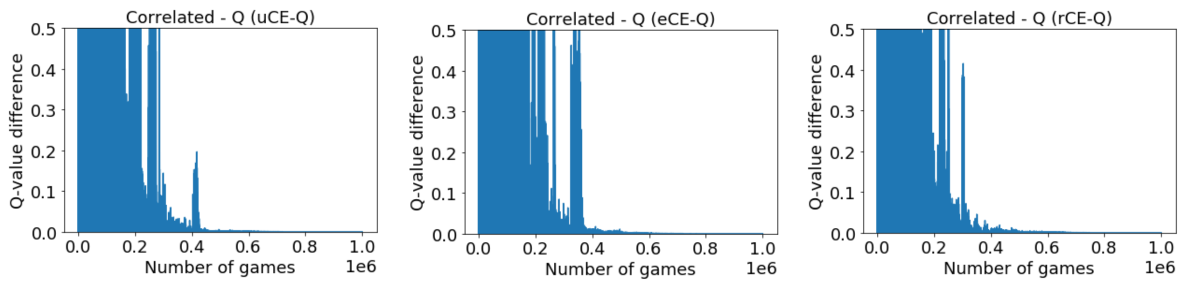

Figure 3: Change in Q-values of state s per game while training for different CE-Q agents

In this paper, the Q-values & policies learnt by Foe-Q and uCE-Q were close, but the convergence pattern between the two was little different. In comparison, Greenwald observed that the Q-values, policies & convergence pattern for these two strategies coincided. This difference could be attributed to couple of reasons. The biggest reason could be that in this paper, while implementing Foe-Q there were very few numerical instabilities but while implementing CE-Q there were couple of instabilities which were dealt by a) normalizing the probabilities b) not updating policy during numerical instabilities. Such issues may not have arisen or dealt differently by Greenwald. Another reason, could be while this paper did not set any random seeds during training both models, Greenwald could have used fixed random seeds for both experiments. The other reason could be that while the starting state ball possession was randomly chosen for this paper, it was not clear how they were chosen by Greenwald. Another reason could be the discrepancy with the term iteration. Greenwald ran their experiments for a 1M iterations (the term iteration was not elicited in the paper). However, for the current paper, one game was assumed as one iteration. And so if Greenwald's iteration represented step instead of a game, than the training in this paper was done for 1M games (approx 7 to 10M steps) which could lead to some over-fitting resulting in small difference in Q-values and policies. Also, the different discount factor (0.99 vs 0.9) chosen for this study might have lead to differences in Q-values.

Apart, from uCEQ, eCE-Q and rCE-Q as explained above in theory were also implemented. Figure 3 shows that both eCE-Q and rCE-Q converged. However, rCE-Q was initially giving unbounded solution. This issue can be anticipated as rCE-Q tries to maximize the maximum value. Hence an additional rationality constraint was added to rCE-Q that limited the objective value to 100 for all steps (since at no step can players' reward exceed 100 which is the reward for scoring goal). Adding this constraint ensured rCE-Q became bounded and converged to almost same Q-values as uCE-Q and eCE-Q. Similar to uCE-Q, both eCE-Q and rCE-Q also learnt stochastic policies and take similar actions with similar probabilities of sticking or going south as shown in Table 1A. SO like uCE-Q, these two CEQ's also learnt Q-values and policies as Foe-Q (the optimal policies). Also, similar to uCE-Q, Greenwald demonstrated that eCE-Q and rCE-Q learned the same Q-values as Friend-Q and gave optimum solution for coordination game. These observations indicate that all the 3 variants of CE-Q are really powerful in learning optimal equilibrium policies in both adversarial and coordination game settings.

The Q-tables obtained after training for 1M games, were considered as final reward matrices for each learner and were used to determine the equilibrium policies for each state for each player. The agents were than played against each other for 10K games and the number of wins for player A were as recorded in Table 1B. It can be seen that when an agent played against itself (across table diagonal) specifically for random, Foe-Q and uCE-Q, the number of games A won was ≈ 5000 (i.e. 50% wins). This is expected, since its a zero-sum game and so when two players play against each other with same strategy, they should win 50% of games. However, for Q-learning, number of games A won was ≈ 3500. But it was also observed that for Q-learning ≈ 2700 games were "Tie" owing to its deterministic policies. Hence, if Tie games are assigned equally to both players, than it can be seen for Q-learning number of games A won will be ≈ 4800 (inline with the expected 50% wins). Friend-Q however, had slightly higher than expected number of wins when played against herself (and this could be due to the artifact of the states it might have seen while training). Table 1B also shows that all the agents did extremely well against Random agent (Row 1 and Column1). However, Friend-Q performed little poorly (76%) as compared to Q,Foe-Q and uCE-Q ($\approx$ 82% wins). Also, when considering trained learners, Friend-Q did poorly against all the other learners (Row 3, Table 1B). This observation is as expected because Friend-Q is making incorrect assumption that the opponent is Friend. So it will be easier for all the other trained agents to defeat Friend-Q. Q-learner on the other hand, performed similar ($\approx$ 50%) to Foe-Q and uCE-Q. This was somewhat surprising, given that Q-learning did not converge and also learnt deterministic policies. However, the policies it learnt led her to behave and compete closely with agents that learnt correct equilibrium policies (Foe-Q and CE-Q). This implies that although Q-learning learnt deterministic policies in non-convergent fashion it is learning in the correct direction. Foe-Q and uCE-Q when played against each other, also like Q-learning compared very closely ($\approx$ 50%) to each other. This is as expected because Foe-Q and uCE-Q learnt very similar stochastic policies with their Q-tables being very close to each other. This observation indicates that Foe-Q v/s. uCE-Q is equivalent to playing against the player with same strategy.

Conclusions

Greenwald's paper introduced a new concept of correlated-Q learning (based on correlated equilibrium and Q-learning) an algorithm for determining equilibrium in general sum Markov games. This paper attempted to recreate some of the experiments that Greenwald used to establish the efficacy of Correlated-Q learning. This article demonstrated that for a 2 X 4 grid world mimicking Soccer game; Q-learning never converged and learnt deterministic policies; Friend-Q learning converged, however learnt incorrect deterministic policies; Foe-Q converged and learnt stochastic policies while uCE-Q converged and learnt stochastic policies similar to Foe-Q, which is correct policy for Soccer as its zero sum two player game. As recommended by Greenwald, this paper also showed that the other two types of correlated Q-learning (eCE-Q and rCE-Q) also converged and produced stochastic policies and Q-values very similar to uCE-Q. Furthermore, an additional experiment was performed in this paper by pitching various strategies against each other and it was shown that all agents did better than random agent indicating all agents had learned better than taking random actions. For the rest of the group of trained agents, all agents did better than Friend-Q learning because of Friend-Q's incorrect assumption. Although, Foe-Q and CE-Q performed similar to each other, Q-learning was comparable to these two. Furthermore, although not demonstrated in this paper, but demonstrated by Greenwald, CE's also learn correct equilibrium in coordination game as well. Hence, it can be concluded that Correlated Equilibirum Q-learning is really powerful algorithm to learn for two player general sum game under both adversarial and coordination types of settings.

References

[1] Amy Greenwald, Keith Hall, and Roberto Serrano. Correlated q-learning. In *ICML*, volume 3, pages 242–249, 2003.

[2] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

[3] Michael L Littman. Friend-or-foe q-learning in general-sum games. In *ICML*, volume 1, pages 322–328, 2001.