GIT HASH for code: `27b6404c8dc17174f1bb74ea61153cb8dc976642`

Correlated-Q for a Grid Based Soccer Game

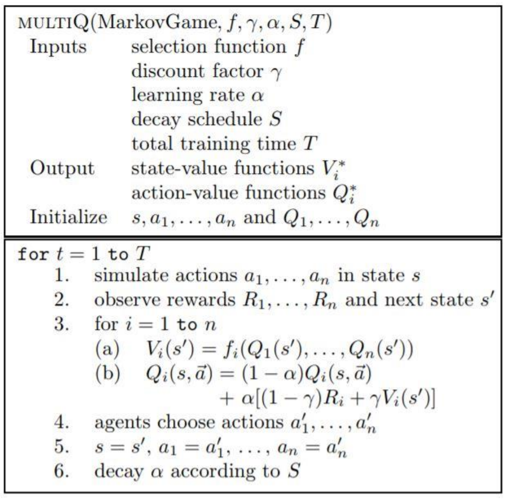
Tharun Saranga
College of Computing
Georgia Institute of Technology
Atlanta, USA
tarun.saranga116@gmail.com

***Abstract*— In this paper the Soccer Problem from the paper Correlated-Q [1] (CE-Q) learning is replicated as the results shown in the Figure 3 of the paper.**

***Keywords*— Friend-and-Foe-Q (FF-Q), Nash-Q**

I. Introduction

In order to tackle the problem of learning in multiagent environments and multiagent learning algorithms this paper discusses the Correlated-Q learning algorithm and its behavior in the Soccer Environment. Nash-Q learning algorithms are proposed which converge to the Nash equilibrium policies under restrictive conditions and Friend-and-Foe-Q learning which always converges but only in some classes of games. Correlated-Q learning is a multiagent Q-learning algorithm that generalizes both Nash-Q and FF-Q. Nash-Q uses Nash equilibrium conditions. Friend-Q is a Q learning algorithm in a two player grid game that always believes that the opposite player will choose actions that would be beneficial to it. Foe-Q believes that the opposite player will always choose opposite actions that are detrimental to it. CE can be computed with linear programming. CE can achieve higher rewards than NE. In general-sum Markov games one of the difficulties is selecting from multiple equilibria with multiple reward values. CE allows the possibility of dependencies in agent's randomization. It is a probability distribution over the joint space of actions, the agents optimize with respect to one another's probabilities, considering their own interests. Nash equilibrium has no efficient computation method, but CE can be computed easily with linear programming. A multiagent Q-learning implementation is presented below in Table 1.
It takes inputs of selection function which determines which



```
MULTIQ(MarkovGame, $f, \gamma, \alpha, S, T$)
  Inputs       selection function $f$
               discount factor $\gamma$
               learning rate $\alpha$
               decay schedule $S$
               total training time $T$
  Output       state-value functions $V_i^*$
               action-value functions $Q_i^*$
  Initialize   $s, a_1, \dots, a_n$ and $Q_1, \dots, Q_n$

for $t = 1$ to $T$
    1. simulate actions $a_1, \dots, a_n$ in state $s$
    2. observe rewards $R_1, \dots, R_n$ and next state $s'$
    3. for $i = 1$ to $n$
         (a) $V_i(s') = f_i(Q_1(s'), \dots, Q_n(s'))$
         (b) $Q_i(s, \vec{a}) = (1-\alpha)Q_i(s, \vec{a})$
                 $+ \alpha[(1-\gamma)R_i + \gamma V_i(s')]$
    4. agents choose actions $a'_1, \dots, a'_n$
    5. $s = s'$, $a_1 = a'_1, \dots, a_n = a'_n$
    6. decay $\alpha$ according to $S$
```

Table 1

action to select based on the logic defined. Discount factor, is the variable used in Q-learning to discount the future expected reward. Learning rate is the variable that tells how fast to learn the environment. Decay is the amount of decay in the learning rate to gradually decrease it over time. Total training time is the simulation steps in which the algorithm learns the environment.
CE-Q has four variants which has different objectives. We will use the Utilitarian CE-Q which maximizes the sum of player's rewards as:

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

Equation 1

II. Problem Description

The Soccer Game is assumed to be a zero-sum grid game for which there is no deterministic equilibrium policies. The soccer field is a grid with a circle representing the ball. The two players have action space of N, S, E, W and stick. If both the players collide only first one moves. If the player with the ball moves second the ball changes the position. If the player takes the ball to the goal area he scores +100, if it is his goal and other player receives -100 and vice-versa ending the game.

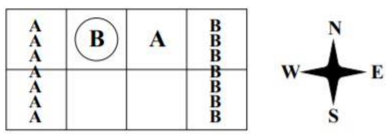


Fig. 1

Fig 1 shows a State in the Soccer Game for which there is no deterministic equilibria. This Soccer problem is solved using the Correlated-Q learning algorithm maximizing the sum of each player's rewards.

III. Experimental Setup

The Soccer environment has been modeled as a grid of 2x4 array in python development environment. With Sates representing the 3 values (Position of player A, position of player B and who has the ball). Accounting for some constraints such as two players cannot occupy same space, there are a total of 112 states in total. The game is simulated with four different learners: General Q-learner, Friend-Q, Foe-Q and Correlated-Q learner. The errors from each learners simulation is collected and graphed to show the differences between each learner.

IV. Test Run

It is observed that the random seed value has a lot of effect on the result of the experiments. Each simulation run took about 5 mins to run for 1000 iterations. Finding the right seed value to replicate the results same as that of the paper is found to be difficult with computing and time constraint. The following experiments have been run with the best seed value I could find. There is a high possibility that other seed values will give more similar results as in the paper. The four Q-learners are implemented.

A. *Q-learner:*

A general Q-learner was implemented for this. The actions are selected from the joint action space and are updated based on that. The Q-tables for each player are isolated such that the one player never learns about the actions of the other player. This approach makes the Q-learning algorithm never converge.

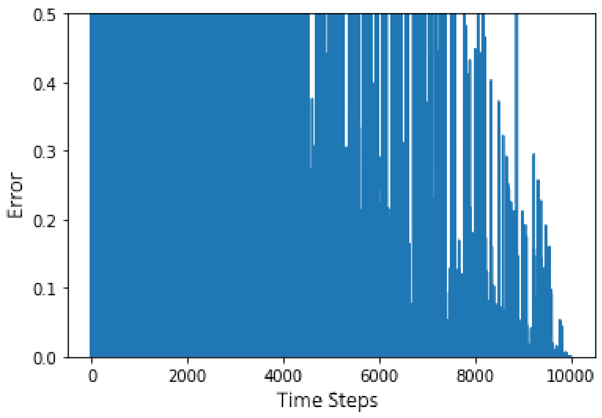


Fig. 2: Q-learner

From the graph it can be seen that the general Q-leaner never really converges. The gradual decrease in error can be accounted from the decrease in the learning rate of the algorithm. This graph is consistent with the graph from the original paper. The Q-learner has been trained for 10,000 simulation steps as the learning rate reaches to zero. This non-convergence occurs for Q-learner because it looks for the deterministic equilibria which do not exist in such two player grid games. This would lead to non-convergence of the normal Q-learner.

B. *Friend-Q learner:*

Friend-Q learner was implemented using the joint action space of both agents and running regular Q-learner. This would give each player the information about the other player. It assumes that the other player would take an action to maximize its reward. The following graph shows the error

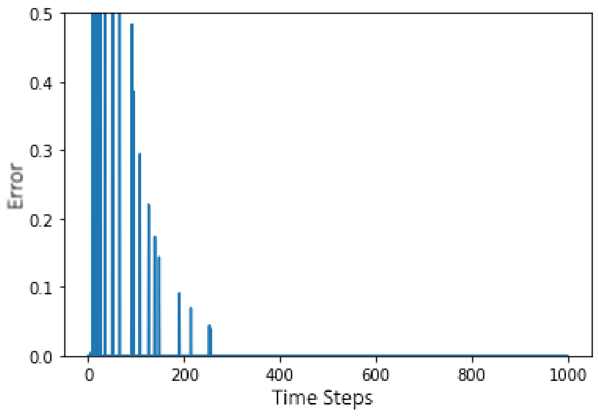


Fig. 4: Friend-Q learner

output of the Friend-Q learner after 1000 iterations of the simulation. The Friend-Q learner learnt the environment within 300 simulation steps. The graph is also consistent with the graph from the original paper. It converges faster because the two opposite player would help each other in scoring thinking that the other will do the same, thereby increasing their reward. But, the Friend-Q converges to irrational and deterministic policies.

C. *Foe-Q learner:*

This learner picks a strategy that would minimize the opponents reward even though it is not the best action for him. This would use a minimization and maximization operations using cvxopt library. The cvxopt library would minimize the rewards and the maximum of this distribution is selected by the agent. The Q- table is updated accordingly. The following graph shows the error of the Foe-Q learner.
The Foe-Q learner even though not exactly similar to the

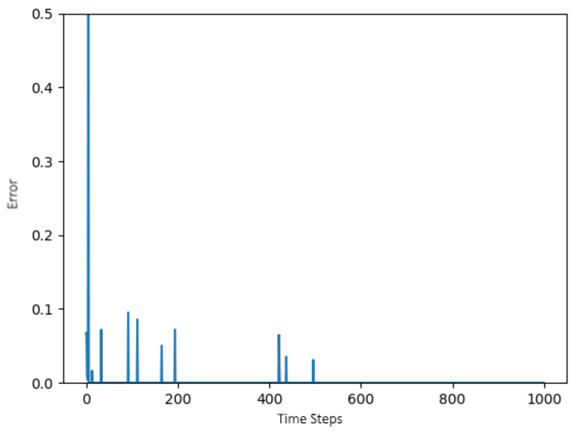


Fig. 3: Foe-Q learner

actual graph from the paper, it has the same envelope as the graph from the original paper. The small errors in the paper are completely absent in this simulation. This might be because of the random learning and exploring strategy implemented in the Q-learning. It learnt the environment within 600 simulation iterations on average.

D. *Correlated-Q learner:*

This learner uses the correlated equilibria. Linear programming gives the CE at each state. Out of the four CEs, the paper described the graph shows the result of the utilitarian CE-Q which maximizes both agent’s rewards. Using the Probabilistic action space and Q-table the optimal action is found and the new states are updated in the Q-table. The following graph shows the error of the CE-Q.

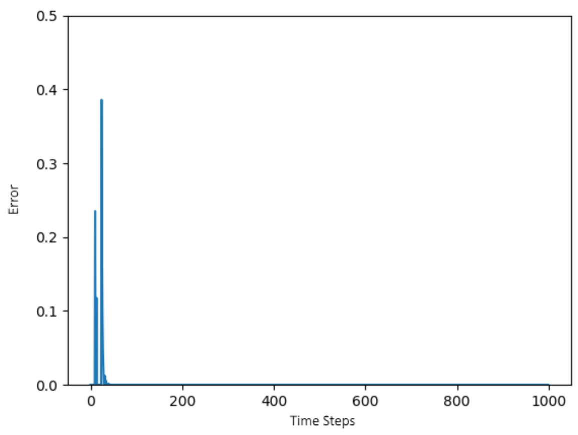


Fig. 5: CE-Q learner

Correlated-Q learner has similarities with the Foe-Q. The Q-values and the policies match each other. CE-Q generalizes Nash-Q and FF-Q providing optimal policy even in non-deterministic cases. The graph shows that it converges within 200 steps.

V. CONCLUSION

The general Q-learner fails to converge to a solution for the two player non-deterministic grid based game. This situation is resolved by augmenting the Q-learner with Nash-Q, FF-Q and CE-Q. CE-Q would generalize Nash-Q and FF-Q to work in all the situations. CE-Q generalizes for all situations and Friend-Q would converge with an irrational policy.