Comparing Q-Learning, Friend or Foe Q and Correlated Q in Soccer Game

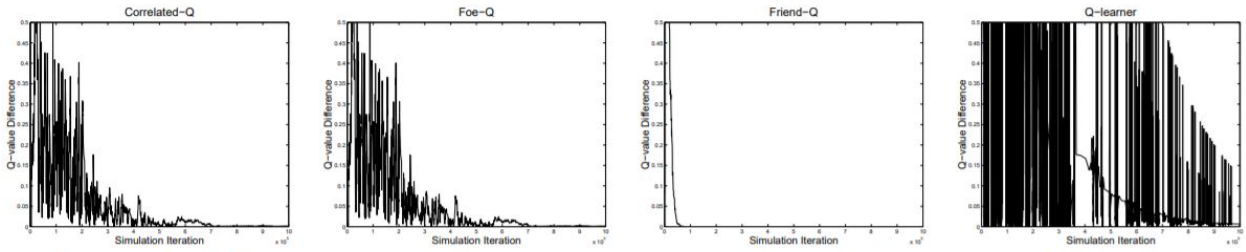
CS 7642 Reinforcement Learning

Safin Salih

Hash: 95efa606348411ae3bc3789c1da9df2cb854be30

Introduction

Game theory provides a solution in decision making problems where there is more than one agent acting on the environment. Similar to the real world, where exists many other players (e.g countries, firms) with various goals , and based on their motives and actions, we adapt to it . We will look at four different techniques on solving a 2 player, zero-sum game. They are as follows; Q-learning, Friend-Q, Foe-Q and CEQ. We'll begin with exploring the rules of the game, then see how each of the algorithms solves the game with our intent being to match Greenwald's[1] graph.

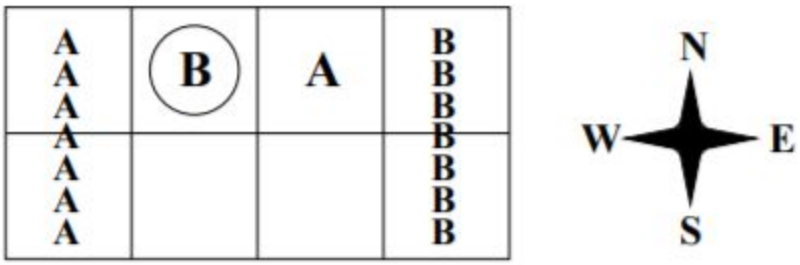


(a) Correlated-Q (b) Foe-Q (c) Friend-Q (d) Q-learning

[1]

Soccer Game

The game is a 2 (row) by 4 (column) zero-sum soccer grid game where two players are encouraged to score by going to the appropriate square with the ball. Each square can be occupied by one player, and only one of the two players can have the ball, depicted by a circle, at each state (see the graph below). There are 112 possibles states and actions are executed simultaneously. When a player with a ball moves to another square that is already occupied, neither player moves, but the ball changes possession. The possible action that can be taken at any given state are, Up, Left, Right, Down, or Stay. However these actions are not deterministic, but stochastic . If a player scores by moving to the appropriate square, that player receives +100 points and the other player recieve -100.



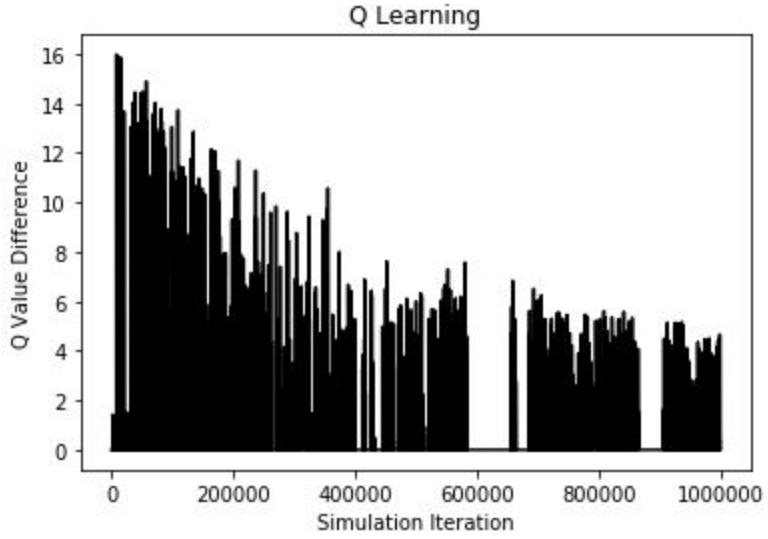
[1]

Q-learning

In the classical Q-learning that was defined by Sutton, we have to create a Q-table , this is an off-policy learning algorithm, where we update the Q-table by using the Bellman equation.

$$Q^{new}(S_t, a_t) = (1 - \alpha) * Q(S_t, a_t) + \alpha * (Rew_t + \gamma * max_a Q(s_{t+1}, a))$$ [2]

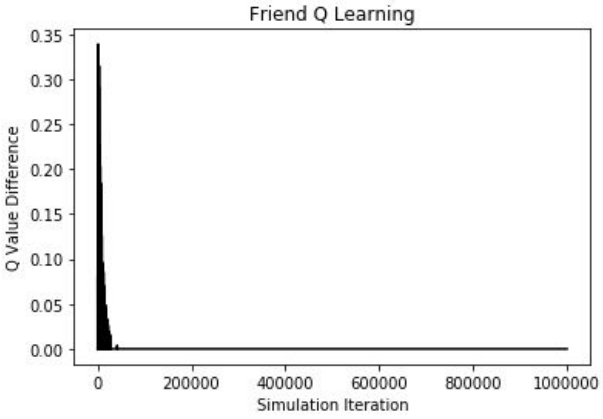
For Q-learning, the opponent's choice of action will not be taken into account even though this is a multi-agent problem; the agent sees the other agents as part of the environment . While in Friend-Q, Foe-Q and CEQ, the agent takes into account what the opponent's action are before making a decision. The best action for each player is implemented using ϵ- greedy approach, starting at some fixed value, 1.0, then we slowly decay the value of epsilon by multiplying it by 0.9999 at each iteration. We also did this for alpha, but our initial value was 0.9. As we decay the value of epsilon we go from taking random action to taking actions that give us maximum discounted future rewards. What was missing from the paper, was what was the initial alpha and epsilon value, and how it was decreased after iteration. For our implementation,we did non-linear reduction in alpha and epsilon instead of linear reduction. In result Q-learning did not guarantee convergence, and if it does it takes longer than the other methods. The reduction in the Q-difference graph comes from the reduction in alpha value. Large initial alpha values equals larger Q value difference, however this didn't help convergence. The large Q value differnce in the paper were not shown in our graph, but what we did capture was the oscillating Q value difference.

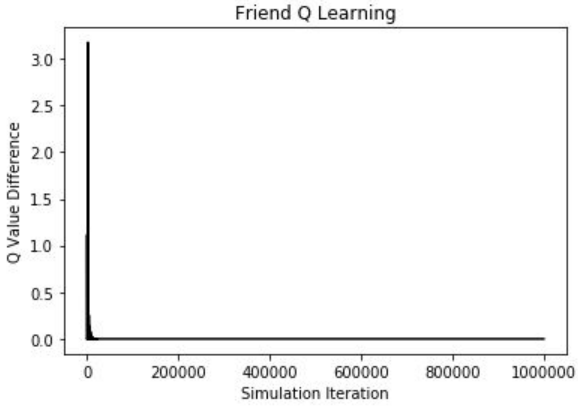


Friend-Q learning

$$Nash_1(s, Q_1, Q_2) = \max_{a_1 \in A_1, a_2 \in A_2} Q_1[s, a_1, a_2]$$ [2]

Similar to Foe-Q learning, Friend-Q is an extension of Nash-Q. We know beforehand that Friend and Foe Q algorithm converges to a Nash equilibrium given a a two player, constant-sum game. In Friend-Q, players are aware of all other players actions as well as the full state of the game. When it comes to evaluating the utility of a state, the opponent will choose joint actions that yield the highest Q value, and assumes other players will cooperate with it. This include even passing the ball, so players can score or getting out of the way. Safe to assume out of all the methods, Friend-Q converges the quickest. In a sense, this is an optimistic algorithm,because players are not competing , but cooperating towards a common goal.The way we approach this problem was similar to Q learning. We ignore the opponent's action and use the most optimistic estimation of the opponent's at a given state. Hence Friend-Q converges to a deterministic policy . We let γ = 0.9, α start at 0.5 and decay non-linearly by multiplying it by 0.9999. Different initial alpha yield different results. Here are 5 different initial alpha values and results. The left starts at α = 0.1 and the right is 0.4.





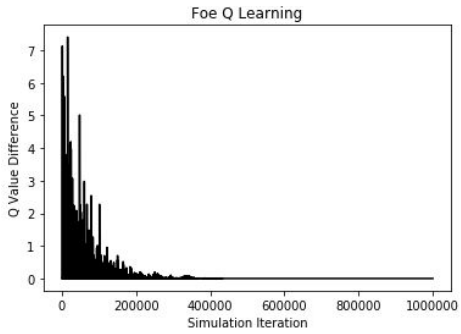
Foe-Q learning

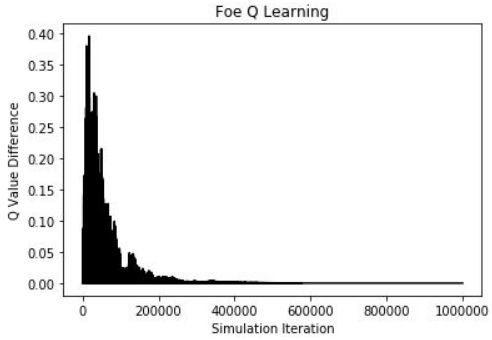
Both Foe-Q and Friend Q are an extension of Nash-Q, but Foe-Q is a pessimistic agent.

$$Nash_1(s, Q_1, Q_2) = \max_{\pi \in \Pi(A_1)} \min_{a_2 \in A_2} \sum_{a_1 \in A_1} Q_1[s, a_1, a_2]$$ [1]

Meaning each player assumes the other player wants the worst for him/her .The goal being to find an optimal mixed policy that maximizing own reward, while minimizing its opponent. For this zero-sum game, we implement a single Q table instead of two Q tables, and in our update rule, we constructed a minimax algorithm(modeling adversarial player) via linear programming. To solve the probabilities of actions, we have to create a system of linear equations and constraints to be solved by linear programming. Unlike pure strategies, minimax strategies for a zero-sum game lets players to predicted probabilities and then multiplied by the Q-values of future actions . Then we update the current Q value is updating with respect to the alpha value and gamma value. Because of this Foe-Q is more robust than Friend-Q and has less limitation and

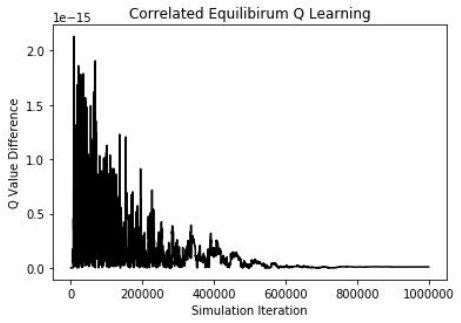
both known to converge as proven by proven in Theorem 5 [3] . In the left we initialized α= 0.9, and the right, α= 0.5 and we decay non-linearly at each iteration with γ= 0.9. And from the graph, we can conclude that we see convergence around 350,000.

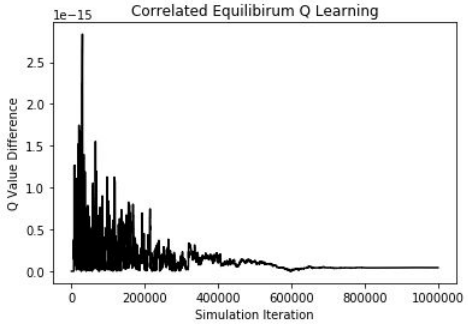


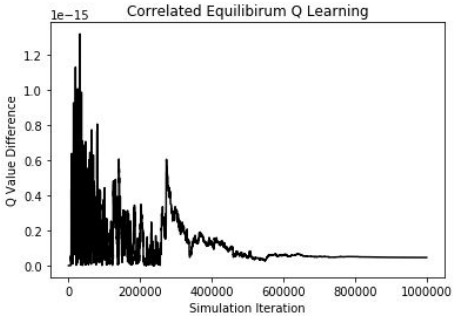


Correlated-Q learning

To find a solution a correlated equilibrium, first we must create a list of linear equations that represent constraints for both players. Purpose being that we want either players not be compelled to choose other actions not proposed by the arbiter. And ultimately get to a point where each player's best interest is to follow the advice that was given. CE-Q strategy is similar to Foe-Q in using LP in the update , but CE-Q finds the utility of a state using mixed strategy and get to correlated equilibrium. Doing so requires a set of linear equations for the probabilities for each joint action pair allowing for optimal distribution for actions. Hence, we must use LP to solve the joint distribution at each state. Here are the results we obtain from various α and we can conclude that it converges around 600,000 step.







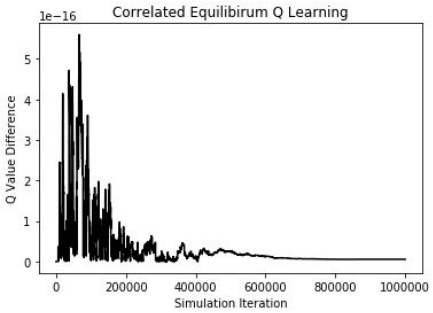


Figure $\alpha = 1.0$ (Top Left), $\alpha = 0.8$ (Top Right) $\alpha = 0.4$(Bottom Left) $\alpha = 0.1$ (Bottom Right)

Discussion

Choosing how to decay epsilon and alpha posed a difficulty when replicating graphs, since the values and methods were unknown.I did not experiment with any other values for gamma values. Also, the initial choice of having two Q-table would make plots not correspond to the graph[1], so it was best to stick to one Q table instead. Another issue that arises when trying to replicate Q-learning, is when to input the data to be graphed, since this made a difference in the plot. Overall, our graphs looked similar to Greenwad's paper, with some variation

Bibliography

[1]Amy Greenwald, Keith Hall, and Martin Zinkevich. Correlated Q-Learning. Technical Report CS-05-08, Brown University, 2005.

[2]Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. MIT Press, 1998

[3]M. Littman. Friend or foe Q-learning in general-sum Markov games. In Proceedings of Eighteenth International Conference on Machine Learning, pages 322– 328, June 2001
CEQ