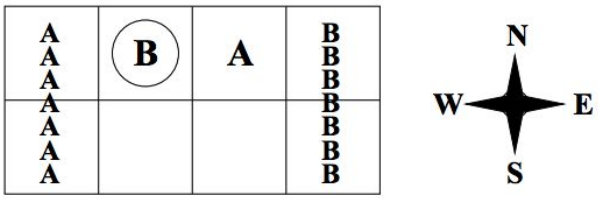
Correlated Q-Learning

Introduction

Purpose of this project is to reproduce and confirm the results presented by Amy Greenwald and Keith Hall in “Correlated Q-Learning”, 2003. In this project we’ll recreate Markov game from the paper and use it to test four Q-Learning algorithms.

Markov game environment

One of the games used by Greenwald and Hall in their experiments is so called “Soccer” game first introduced by Littman in “Markov games as a framework for multiagent reinforcement learning”. The game is taking place in a grid world of size 2x4.



There are two players. At each iteration both players choose one of 5 actions: move N, S, W, E or don’t move. Both players actions executed in random order with following rules: if player is trying to move to the cell occupied by another player then the movement doesn’t take place, but if the moving player had a ball at that moment then the ball is going to the standing player. If player that has ball moves to one of two targets, then player assigned to that target receives +100 points and other player loses 100 points, this makes it a zero-sum game. Original paper doesn’t specify what happens if player is attempting to move outside of the grid boundaries, so in our experiments we’re assuming that player doesn’t move in this case.

Q Learners

Greenwald and Hall introduced four different reinforcement learning algorithms for Markov games: standard Q-Learning, Friend Q-Learning, Foe Q-Learning and Correlated Equilibrium Q-Learning.

Q-Learning

Q-Learning uses Bellman update rule to update its state-action value table and later use it to calculate policy. Standard Q-Learning takes into consideration only its own actions. This

prevents Q-Learning from learning mixed strategies, moreover, its deterministic policies don't take into consideration opponent's actions.

Friend Q-Learning

Main difference of Friend Q-Learning from standard Q-Learning is modified update rule. Friend Q-Learning uses pair of actions (its own and opponents) instead of just its own action. This should allow it to adapt to opponents actions. But Friend Q-Learner only maximizes over action pairs, which is equivalent to expecting that opponent will actually help the player, which is usually not the case in zero-sum games.

Foe Q-Learning

In opposite to Friend Q-Learning Foe Q-Learner is trying to learn actions that will maximize minimal rewards, as if opponent's trying to minimize players rewards.
One way of implementing such policy is by using linear programming. By putting constraints on minimal value of expected rewards one can not only find maximum of this value but also calculate probability distribution for actions that can help achieve this expected value.

Correlated Equilibrium Q-Learning (CE Q-Learning)

Main interest of Greenwald and Hall in this paper was CE Q-Learning algorithms. These algorithms are introduced as an equivalent to Nash Q-Learning, which finds policies for players that are considered to be in Nash equilibrium, when neither player gains from changing the policy if opponent is sticking with its own. CE Q-Learning optimizes in respect to joint probability distributions of players actions. CE Q-Learning allows among action probabilities, so agents can optimize with respect to one another's probabilities, conditioned on their own.
Unlike Nash Equilibrium, Correlated equilibrium can be efficiently computed using Linear Programming. But there might be many numbers of correlated equilibria. Greenwald and Hall introduced four different methods of choosing one equilibria over another. We'll cover one of them: utilitarian CE Q-Learning, when total sum of expected payoffs of both players is maximized. This condition is used in linear programming as a target maximization function.

Experiments

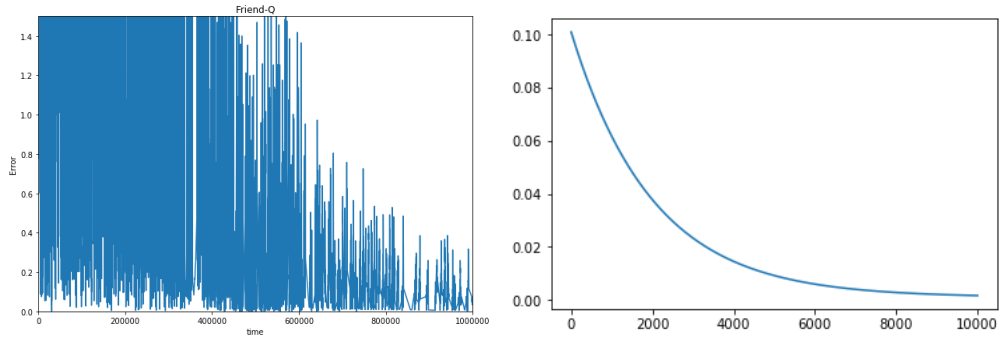
All four described algorithms were used to learn policies for Soccer game. After running algorithm in the environment we confirm that it converges to some policy by analyzing differences in Q values with time, by calculating "error" values equal to absolute value of the differences in Q values for initial state and for agent A taking action S and for agent B staying where it is.
But convergence doesn't guaranty that the learned strategy is optimal. To confirm that we also explore Q-table for initial state for all actions, which is enough to figure out what policy agent is going to use.

Implementation details

All learners during the experiments used the same discount rate as in the original paper $\gamma = 0.9$. Learning rate as well as epsilon value for epsilon-greedy policy for Q-Learner gradually decrease with time to 0.001. Greenwald and Hall didn't specify how those values decrease in their work, but here we used exponential decay to 0.001.
Friend-Q, Foe-Q and CE-Q use probability distributions to choose actions, so they don't require additional explorational strategy like Q-Learner.
Also we don't know how Q-values were initialized in original paper, so during this project all learners initialize their Q-tables using gaussian distribution.

Q-Learner

Following graph (on the left) shows how Q-values for particular state and actions change with time. X-axis represents training time iterations and y-values are errors calculated following way: $Err_t = |Q_i^t(s,\ a)\ - Q_i^{t-1}(s,\ a)|$.

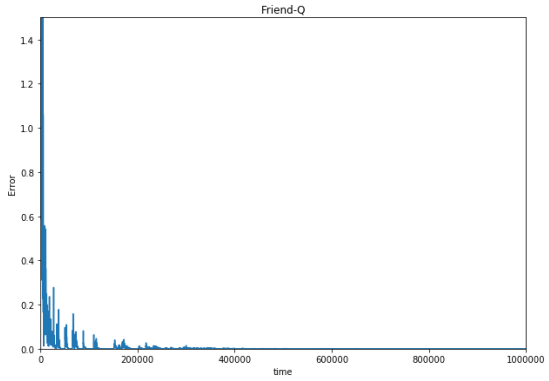


Left: errors in Q-values in time; Right: learning rate decay with time

Even though errors decrease with time its mostly due to decreasing learning rate (graph on the right). Even after a million iterations Q-values continue to change. And if we look at Q values themselves in different moments of time we'll see that agent constantly trying to change strategies looking for optimal one. But as it was mentioned not only its difficult for Q Learner to do this since it isn't even aware that it has an opponent, but it can't learn mixed strategy. And all optimal strategies for Soccer game are mixed strategies as we'll see in other experiments.

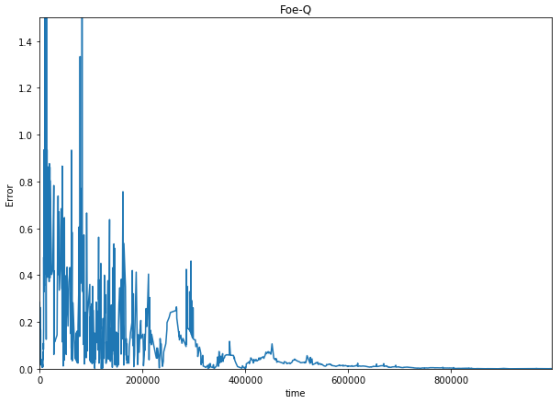
Friend Q-Learner

Friend Q-Learner converged extremely fast. But better analysis of Q-values showed that one player (without a ball) didn't have any preferences in actions at all, since it probably expected for its opponent to score a goal for it and another player wanted to go East assuming that other player will move aside. Even though these policies align with learner's assumptions it's not an optimal strategy.



Foe Q-Leaner

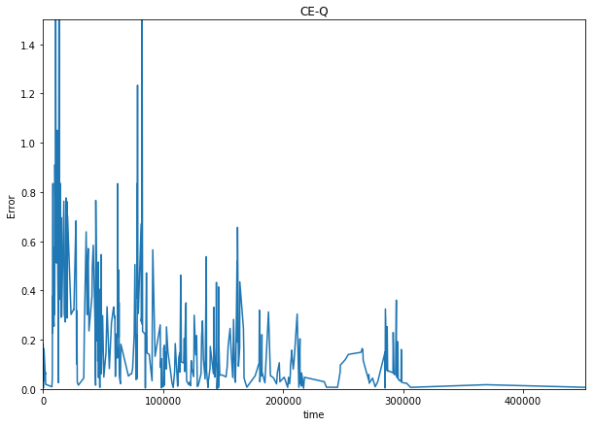
Foe Q-Learner is the first one that converged to optimal strategy. Both players were trying to choose between primarily staying (this also includes trying to leave the field, which leads to staying) and moving south trying to bypass each other, sometimes player with a ball might also choose to move straight to the goal hoping that opponent will choose to move south.



It's also worth noting that Q-values errors decrease gradually unlike Friend Q-Learner, but not as slowly as in case of Q-Learner, so its decrease can't be explained by decrease of the learning rate, also it experiences less spikes than Q-Learner, which confirms stable convergence.

CE Q-Learner

Finally CE Q-Learner showed similar results to Foe Q-Learner. It also converged to nondeterministic policies for both players, where each one randomizes between sticking and heading south. But it took more computational time to do it, since it had much more constraints for linear programming and was calculating joint probabilities for all action combinations and not just the probabilities for one player's actions. So in the end CE Q-Learner was able to learn minimax policy for zero-sum game. Although in non-zero sum games CE Q-Learner can actually outperform Foe Q-learner, since players can estimate conditional probabilities for all actions pairs and they might decide to cooperate with each other to achieve better results.



Pitfalls

Since we made an assumption that player can take actions that would lead it outside the field and still stay on the same spot, learners didn't see any differences between these actions and action to stick to the same spot. These lead in many cases for learners to ignore latter, which resulted in flat Q-Values update graphs. But since Q-values were initialized randomly some learning attempts were quite consistent with original paper and learned similar strategies.
Also some fast linear programming implementations have proven to be unstable in case of CE Q-Learning, perhaps because of sometimes unpredictable changes in Q-Values and strict restrictions in those implementation. But other implementations showed just as good results even though they didn't run as many training iterations as other q-learners due to limited computational time.

Overview

In the end project was successful in repeating Greenwald and Hall experiments and achieved similar results, which once again confirmed efficiency of Q-Learners in multiagent systems. Just like in the original paper choice of proper Q-Learner algorithm should depend on task's goal and type of the task. Also it was shown that Foe Q-Learning and CE Q-Learning can be efficiently computed using linear programming giving advantages of such algorithms as Nash Q-learning and at the same time making it computationally efficient.