Project 3: RLDM - CS 7642

GID: mmendiola3

Presentation: https://youtu.be/

Overview

This paper will review the reproduction of experiments done in "Correlated Q-Learning" (Greenwald and Hall 2003). We will explore the agents designed to learn policies for the multi-agent soccer environment outlined in the original paper, and compare the result. Finally, we will discuss the challenges and assumptions made in order to reproduce the graphs in Figure 3.

Soccer Environment

The soccer experiment rules were reproduced with a basic environment class built with an interface similar to those in OpenAI gym environments. It maintains state and models the actions and rewards outlined in the original paper. These include random move order, player losing ball if moving into an occupied cell, and a symetric reward of 100/-100 for the ball entering a goal. Experiments with various agents are made with calls to env.step(actions) and return next state, rewards, and the done flag. Actions, state, and rewards include data on both players in the soccer environment.

Experiment

Using the soccer environment model, experiments were run with four learning agents (uCE-Q, Foe-Q, Friend-Q, and Q-learning). Each experiment was run over 10e5 steps (env steps, not episodes). At the end of a game (player with ball in goal), the environment was reset to that shown in Figure 4 of the original paper. This starting state was not specified and experimentation was required to determine that this starting state was helpful in collecting enough observations of the specific player, state, action(s) where Q value changes were recorded in the original experiment. These values were recorded at the state in Figure 4 with action(s) of A=South, B=Stick (Q-learner tracked A's action only). Points are plotted only on steps where this state, action(s) pair is observed, which is another detail left out of the original paper.

Agents

Each learning agent used a discount factor γ of 0.9. They all used the same update procedure in Table 1 of (Greenwald and Hall 2003), including the Q value update formula. Each agent provided their own value function for $V_i(s')$.

Q-Learner

The Q-learning experiment used two separate agents and Q tables for each player. The value update function was $V^*(s) = \max_{a \in A(s)} Q^*(s, a)$, which is simply the max Q value in a given state across all actions. In Q-learning, the agents only track Q values for their own actions.

Initial α was set to 0.3, and decayed exponentially with a decay factor of $10^{log(0.001)/5e6}$. The decay factor was inspired by (Littman 2001) and both of these hyper-parameters required experimentation to find appropriate values. (Greenwald and Hall 2003) only specifies that α should decay to a minimum value of 0.001. The same decay schedule was used on ϵ. The Q-learning agent was the only one run on-policy, with ϵ-greedy actions. All other agents choose actions at random.

Multi-Agent Learners

Foe-Q, Friend-Q, and uCE-Q all used an initial α of 1.0. Friend-Q used a decay schedule explained in (Greenwald, Hall, and Zinkevich 2005) as $\alpha = 1/n(s, a)$. All other agents used the same exponential decay as Q-learning, with a decay rate of $10^{log(0.001)/4e6}$. These values and schedules were decided through experimentation and observation of convergence rates in the graphs. Explicit definition of these hyper-parameters would have made reproduction of the original experiment much more straightforward.

Foe-Q

The value function for Foe-Q was taken from equation 5 in (Greenwald and Hall 2003). Linear programming was used to find this value with probability constraints on the mixed strategy distribution (sum = 1, each ≥ 0) across the player's actions. Minimax constraints were modeled as $V - \sum_{a_1 \in A_1} \sigma_1(a_1) Q(s, a_1, a_2) \leq 0 \; \forall a_2 \in A_2(s)$, with the objective function to maximize V.

Friend-Q

The value function for Friend-Q was simply the max Q value across all joint actions in a given state.

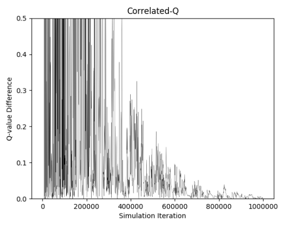
uCE-Q

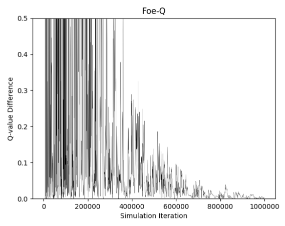
The correlated equilibrium was calculated similarly to Foe-Q, with the same probability constraints. Although CE strategy distributions are calculated across all joint actions. Rationality constraints were developed with the following code:

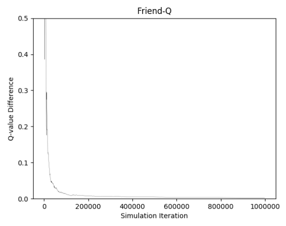
```
G_rationality = []
for i in np.eye(self.actions.shape[0], dtype=bool):
    for j in np.eye(self.actions.shape[0], dtype=bool):
        if np.array_equal(i, j):
            continue
        G0 = np.array(self.actions)
        G0[i] = Q0[j, :] - Q0[i, :]
        G1 = np.array(self.actions)
        G1[:, i] = np.vstack(Q1[:, j] - Q1[:, i])
        G_rationality.append(G0.flatten())
        G_rationality.append(G1.flatten())
```

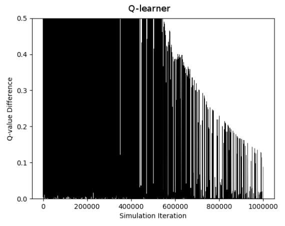
This implements equation 2 in (Greenwald, Hall, and Zinkevich 2005), which states that each player's recommended action is at least as good as any other action given that the other player also follows the recommended strategy. This is expressed more concisely in the following constraints derived from (Saberi 2009): $\sum_{\bar{s} \in S_{-p}} (u^p_{j,\bar{s}} - u^p_{i,\bar{s}}) x_{i,\bar{s}} \leq 0 \; \forall p$ and $\forall i, j \in S_p$. Note that the inequality is reversed to act as an upper bound for the LP solver. The objective function is taken from equation 9 in (Greenwald and Hall 2003), which maximizes the sum of all players' rewards. The value function uses the optimal mixed strategy distribution from LP to return $\sum_{\vec{a} \in A} \sigma(\vec{a}) Q_i(s, \vec{a})$. The greatest challenge in reproducing the original experiments was modeling the rationality constraints of CE for the LP solver.

Results









The plots of Q value changes for the specifically chosen player, state, action(s) closely resemble those in the original experiment. They are not perfectly aligned, as the hyper-parameters and decay schedules were not fully explained and could not be perfectly reproduced. Experimentation with these parameters was required for each learner to reproduce similar rates of convergence as the original paper. However, the overall behavior of the various agents were replicated completely. Correlated-Q and Foe-Q had identical Q values at each step (required static randomization seeds). The Q values converged in both cases. Friend-Q converged very quickly, though the policy learned was ill suited to an adversarial game. The Q-learner never converged, with Q value changes remaining as high as the learning rate throughout the trials.

References

Greenwald, Amy, and Keith Hall. 2003. "Correlated-Q Learning." In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, 242–49. ICML'03. Washington, DC, USA: AAAI Press. http://dl.acm.org/citation.cfm?id=3041838.3041869.

Greenwald, Amy, Keith Hall, and Martin Zinkevich. 2005. "Correlated Q-Learning."

Littman, Michael L. 2001. "Friend-or-Foe Q-Learning in General-Sum Games." In *Proceedings of the Eighteenth International Conference on Machine Learning*, 322–28. Morgan Kaufmann. http://jmvidal.cse.sc.edu/library/littman01a.pdf.

Saberi, Armin. 2009. "Lecture Notes in MS&E 334: Computation of Equilibria." Stanford University.