Correlated-Q Learning: Project 3

KELLY “SCOTT” SIMS

I. Introduction

This article centers around a set of experiments as executed by Amy Greenwald and Keith Hall in their 2003 paper “Correlated-Q Learning”. In this experiment, which consisted of a multi-agent, zero-sum, markov decision process, the techniques of Q-learning, Friend-Learning, Foe-Learning, and Correlated-Q Learning were implemented in a simulated soccer environment. The environment itself, which will be discussed in detail later, creates a paradigm in which no deterministic policy exists. The four methods serve as different strategies to find equilibrium in an environment in which multiple equilibria can exist.

Q-learning, traditionally aims to maximize the expected future reward of a single agent interacting in it’s environment. A single agent in this method has no knowledge and bears no weight on the actions of any outside agents within said environment. It would be prudent for an agent that is implementing Q-learning in a multiagent environment to consider the other agent as an extension of the environment itself.

Friend-learning on the other hand is an abstraction of a paradigm in which muliti-agents cooperate. The Q-table involves the joint action space amongst all agents in the environment for every state representation. For disambiguation, consider the environment as proposed by Greenwald [1] in which two agents are allowed to move in the set of North, South, East, West, Stay. Agents operating under the Friend-learning objective would select the their own action as well as the other agents; E.g. “I go South and you go North”.

Foe-learning is the antithesis of Friend-learning. Instead of cooperating agents, agent A, assuming worst case response over opposing agent’s actions, would aim to maximize this worst case scenario. Conversely, agent B, assuming worst case response from A, would institute a strategy that minimizes the maximum possible response of A. This paradigm as proposed by Jon von Neumann is the Minimax Theorem. It is a strategy in zero-sum games that produces optimal rewards for all actors under rational play[2].

Finally, Correlated-Q looks to maximize both player’s rewards over joint distribution action space. Correlated equilibrium is said to be found if no agent can individually improve its expected utility by deviating from recommended actions[3]. This is very similar to Nash Equilibrium, which is an instance in which mutiple agents have found, and are acting in, a policy that nothing else gives them an incentive to switch. In fact, NE is a subset of CE. The main difference between the two is that CE incorporates a joint probability distribution wheras NE consists of vectors of independent probability distributions over actions.[4]

For concreteness, this article serves as the investigation, replication and discussion of the experiments proposed by Greenwald (2003). We institute different techniques of Q-learning and assess convergence in a zero sum MDP.

II. Rules of Play

The soccer environment as proposed for the experiment in Greenwald (2003), is actually a smaller version of the original experiment from Littman (1994) “Markov games as a framework for multi-agent reinforcement learning”[5]. The field is a 2x4 grid in which the first and last columns are goals. There are two agents, player A and player B. A always starts a game in the grid cell (1,3) whereas player B always starts a game in grid cell (1,2). Agent B’s personal goal is directly behind it in the first column (grid cells (1,1) and (2,1)). Conversely, Agent A’s personal goal is directly behind it as well in the last column (grid cells (1,4) and (2,4)).

Agent B always starts with the ball and attempts to score in agent A’s goal in the last column. The opposite applies for agent A when it has possession of the ball. The rules are simple in that each agent selects an action from the set of actions {North, South, East, West, Stay}. An agent is randomly selected to go first. That agent performs its randomly selected action, followed by the other agent executing its action. In the event both agents attempt to enter the same cell, whoever was randomly chosen to move first, is permitted occupation of the space. The other agent must remain in its previous position. In the event of said aforementioned collision, if the agent who moved first had possession of the ball, that agent maintains possession. Otherwise, any agent in possession of the ball involved in a collision in which they moved second, will relinquish control

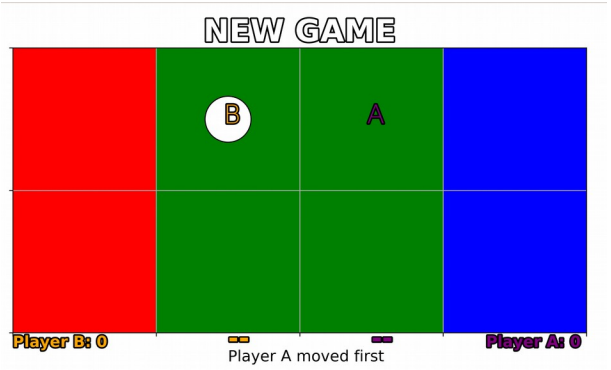

Figure 1: Soccer environment for multi-agent, zero sum, markov decision processes

of the ball to the opposing agent. Figure 1 shows a custom rendering of the environment as described.

A game is considered over when an agent in possession of the ball crosses into either their own goal, or the opponent's goal. An agent who scores in their own goal receives -100 points and their opponent receives 100 points. Conversely, an agent that scores in the opponents goal will receive 100 points and their opponent receives the negation.

Some ambiguities in the description from the original paper are those in which fail to outline the little details or the environment. It was assumed in our implementation that an agent in possession of the ball attempting to move out of the playing field would remain in the same position. Possession would follow the aforementioned logic. Another assumption made from initial environment descriptions was that there are no restriction on a player entering into the goals on either side without the ball. By all intents and purposes, the goals are merely an extension of the normal playing field for the agent without possession.

III. THE EXPERIMENT

For all four experiments, the environment was represented by 112 different encoded states. Given that there are eight positions for agent A to exist in, eight positions for agent B to exist in, and one of two possesion scenarios (either agent A has the ball or agent B), then there are 8 x 8 x 2 = 128 different state representations. However, given that both agents can not exist in the same cell, 8 states are not accessible when agent A has the ball and 8 states are not accessible when agent B has the ball. Thus the final state representation of the environment was always constructed from a 112 column vector. Each row of the vector being a proxy to a state representation. All implementation was done in Python 3.7.

Greenwald (2003) failed to specify hyperparameters chosen to conduct the experiment for all four scenarios. Thus a manual gridsearch was performed across pertinent hyperparameters. Each experiment was ran for one million iterations. It wasn't immediately clear if iterations was equivalent to time steps or to episodes, where an episodes ends after an agent with the ball crosses either goal line. It was assumed that the former was the intended implementation. Each algorithm was implemented as a full "off policy" learning session except for Q-learning. This simply means that at each time step, an action was randomly choosen for both agents from the action set {North, South, East, West, Stay}. Q-learning followed an ϵ-greedy approach. All updates on Q-tables followed from equation 1.

$$Q_i^*(s,a) = (1-\gamma)R(s,a) + \gamma \sum_{s'} P[s'|s,a]V_i^*(s') \quad (1)$$

Convergence was gauged against a base line state representation and action criteria. The base line state was always the initial starting state where B is located at (1,2) with possession and A is located at (1,3). The base action for agent A was always South, and for B, stay. For example, in Q-learning, agent A's q-table is a 112x5 matrix (there are 5 action choices). In our state encoding algorithm, the intial state of a game was index 71. At the start of an iteration, initial q value was stored for qA[71, 1] , where action 1 is equivalent to south. Actions were randomly chosen for each agent, the environment was advanced based off of actions, the q-tables were updated, and the q-value at qA[71, 1] before time step advancement was taken as the difference from qA[71, 1] after. Logic was implemented so that this calculation would only happen in the event that the advancement of the environment was due to a result of the random actions being chosen involved agent A being "South" from environment state 71.

A. Q-learning

For Q-learning, each agent had a 112x5 Q-table randomly initialized between zero and one. Convergence was checked only on player A using the logic mentioned in the previous section. With no formal proof, the value and objective function for Q-learning:

$$\text{Value function: } V_i^*(s,\vec{a}) = \max_{\vec{a} \in A(s)} Q_i^*(s,\vec{a}) \quad (2)$$

$$\text{Objective function: } \sigma^i \in arg\max_{\sigma \in CE} \sum_{\vec{a} \in A} \sigma(\vec{a}) Q_i(s,\vec{a}) \quad (3)$$

In Greenwald's original experiment, it was shown that Q-learning never converged [Figure 2]. It only gave the impression of convergence due to the learning rate decay schedule.

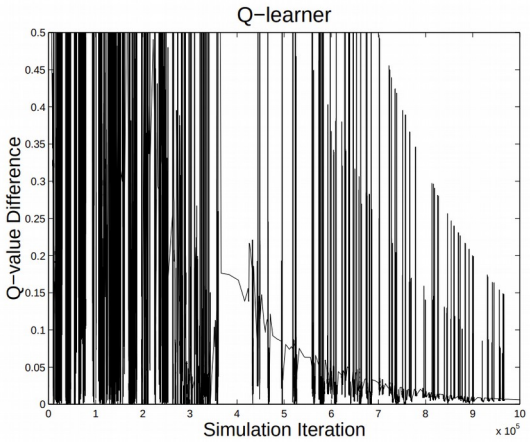

Figure 2: Original Q-learning convergence results from Greenwald (2003)

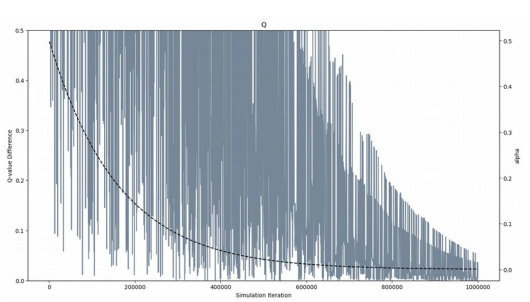

Figure 3: Replication of results of Greenwald's (2003) Q-learning multi-agent algorithm with learning rate alpha decay schedule superimposed

In our replication of the experiment, we achieved the same results, graph over time iterations that reveals a down sloping error. The only difference between the graphs would be due to data density itself. We rendered more data points than the original. The pseudo convergence was due to the learning rate decay schedule of :

$$alpha = alpha * \left(e^{-\ln(500)/10e6}\right)$$

The schedule was chosen so that the alpha value would be 0.001 from 0.9 by the end of 1MM iterations. Figure 3 shows our results of the Q-learning algorithm.

We believe this paradigm is explained by the simple fact that Q-learning is intended to optimize the rewards of a single agent acting in either a stochastic or deterministic environment. Because the added element of another intelligent agent, shapes the environment's response in multi-agent MDP, the environment is neither deterministic nor stochastic. The Q-learning objective in essence is chasing a dynamic environment. This is an environment that is always changing "intelligently" off of the agent's own move.

B. Friend-learning

In contrast to normal Q-learning, Friend-learning did converge in both the original experiment, and our own. Again, a q-table was initialized randomly between zero and one. This time, the action space was taken over the joint action space between agent A and agent B. For example, at any given time, the set actions chosen among players could be {(N,N), (N,S), (N,E), (N,W), (N,Stay), (S, N), (S, S) …..}. There are a total of 25 unique joint combinations of agent actions. Therefore a q-table in Friend-learning was 112x25. The objective and value function for Friend-learning without formal proof is:

$$\text{Value function: } V_i^*(s,\vec{a}) = \max_{\vec{a}\in A(s)} Q_i^*(s,\vec{a}) \quad (4)$$

$$\text{Objective function: } \sigma \in \arg\max_{\sigma\in CE} \max_{i\in I} \sum_{\vec{a}\in A} \sigma(\vec{a}) Q_i(s,\vec{a}) \quad (5)$$

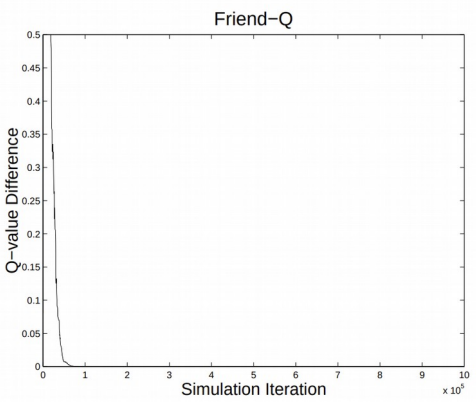

Figure 4: Original Friend-learner results from Greenwald (2003)

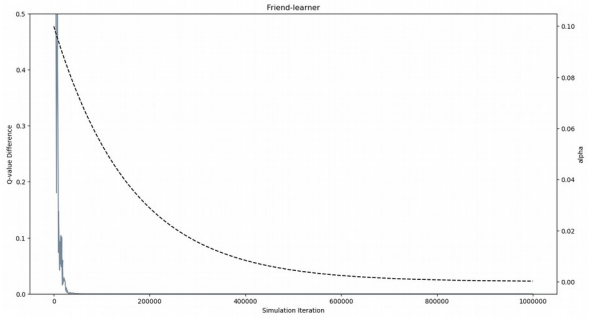

Figure 5: Our replication of Friend-learner with learning rate alpha superimposed

The decay schedule of alpha followed that of traditional Q-learning. The only difference was the initial value of alpha. We had to start a much lower alpha, 0.1, to replicate the results of the original experiment. Figure 4 and figure 5 show the results of Greenwald's and our experiment respectively. Our experiment converged after approximately 50,000 iterations. This was in alignment with the original.

These results are to be expected in Friend-learning given that the premise behind it is the agent's cooperate with each other. The q-update is based on the joint actions, therefore the agents are basically "sharing" the same q-table during training. This allowed for the rapid convergence.

C. Foe-learning

Foe learning is where difficulty in replciation really started to arise. It utilizes the minimax theorem as described in the introduction. Most of the challenge stemmed from properly utilizing the cvxopt python library. To start with, Q-tables were intialized for both agents as well as V-tables, V being the value of a state from the minimax approach. Therefore, the q-table was of dimension 112x5x5 and the V-table was a column vector of 112x1 – a representation for each state. The Q-table followed this shape in order to optimize an agent's choice for a given state from the minimax optimization.

In taditional Q-learning, Q is updated as a function of the next action from s'. In minimax optimization, the value of the next state stems from solving the LP of the 5x5 joint action space at given state. Q[s] = A(5x5). This joint action space optimized under the notion of (6) and the objective function is (7).

$$\text{Value function: } V_1(s) = \max_{\sigma_1\in\sum_1(s)} \min_{a_2\in A_2(s)} Q_1(s,\sigma_1,a_2) = -V_2(s) \quad (6)$$

$$\text{Objective function: } \sigma \in \arg\max_{\sigma\in CE} \min_{i\in I} \sum_{\vec{a}\in A} \sigma(\vec{a}) Q_i(s,\vec{a}) \quad (7)$$

Upon advancing the environment each step, the linear program was computed to solve the minimax optimization for (s'). We were able to successfully utilize SciPy's linalg library to solve

the LP, but it was orders of magnitude slower than CVXOPT (1,000,000 iterations took 2 hours to run).

After about 3 days of tweaking and white board level discussion with peers, the error in cvxopt implementation was determined to be the incorrect orientation of the state matrix into the LP. In order to correctly setup the LP, the 5x5 state action matrix had to be transposed. This one oversight caused hours of downtime. Figure 6 and 7 shows the original and replicated result respectively.

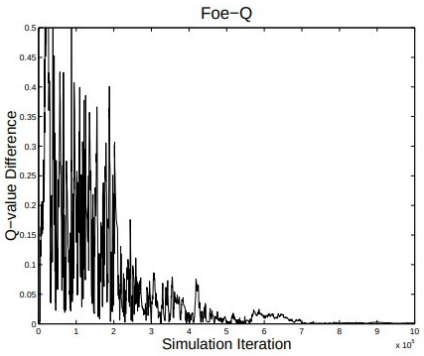

Figure 6: Foe-Learning as executed in Greenwald's (2003) original experiment

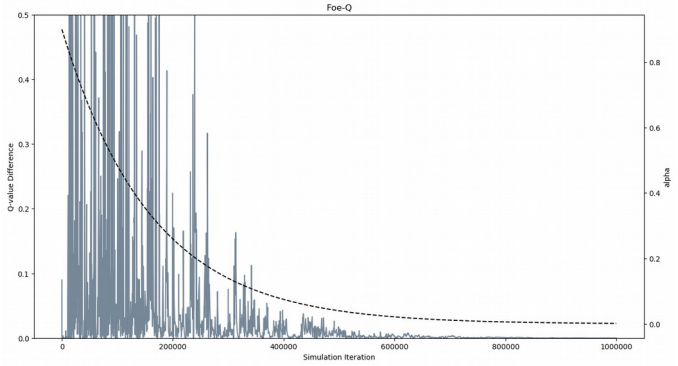

Figure 7: Replication of Foe-learning with alpha decay schedule superimposed

Overall, our results strongly resemble the original. Deviations between the two probably stem from the differences in random actions taken during the one million iterations. Also, we have incoroporated a higher data density in the renderings.

D. CEQ-learning

Lastly, we [attempted] to replicate CEQ-learning. This one posed the greatest challenge. Understanding the notation in the original paper proved difficult with transcrbing from theorem to code. Q-tables for both agents were intialized to one at a dimension of 112x5x5. Also, two V-tables were intialized again as 112x1 column vectors for both agents. Actions were randomly chosen and the environment was advanced. Upon advancing, the joint action 5x5 matrices from both agent's Q-tables were used to solved the LP for correlated equilibrium. The result from the LP served as the value V[s']. (8) and (9) are representative of the value and objective function for CEQ.

$$\text{Value function: } V_i(s) \in CE_i(Q_1(s), \ldots, Q_n(s)) \quad (8)$$

$$\text{Objective function: } \sigma \in \underset{\sigma \in CE}{arg\max} \sum_{i \in I} \sum_{\vec{a} \in A} \sigma(\vec{a}) Q_i(s, \vec{a}) \quad (9)$$

It was said that CEQ should equal Foe-learning results not only for Q-values at the baseline example, but also in the convergence graphs. Ultimately, we were able to reconstruct something that converges, but not to equality of Foe-learning. Figure 8 shows replication results of CEQ-learning. The original results are omitted here as they are exactly the same as Foe-Learning.

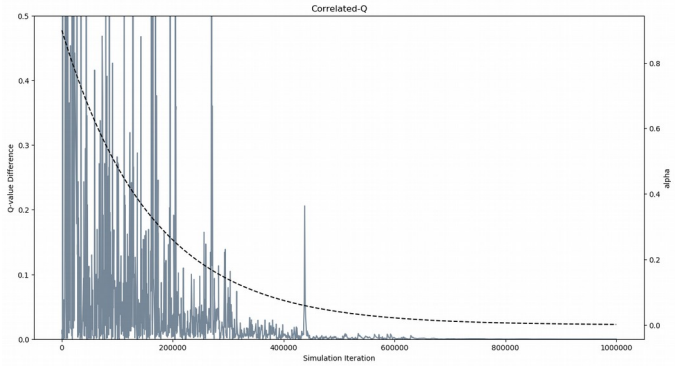

Figure 8: Replication of CEQ-learning after 1MM iterations

Convergence happened around approximately 600k iterations as in the original, but there exists a massive spike after 400k that didn't exist in the original experiment nor in Foe-learning. Starting off with an alpha of 0.9, again, the decay schedule was set to equal that of all other algorithms. We believe the failure in replication stems from incorrect construction of constraints in the LP for each agent. All signs point to this being the root cause due to the fact of the smallest tweak in our implementation led to wildly differen results. Due to time constraints, futher tweaking had to be abandoned.

IV. Conclusion

Reproducing results from this paper proved extremely difficult. Due to the ambiguities in the original paper, several other sources had to be researched, different hyperparameters had to be gridsearched, and multiple whiteboard discussions had to be had with fellow peers. As stated earlier, when it came to Foe and CEQ learning, it was rather difficult converting theory to algorithm. We managed to reproduce Q, Friend, and Foe learning almost exactly, but fell short on CEQ due to time constraints and overall weakness in a background in linear programming. This project alone has inspired myself to enroll in a course that offers LP next semester.

References

[1] Greenwald, A. and Hall, K. (2003). Correlated-Q learning. AAAI Spring Symposium, 242–249.

[2] Shah, Nisarg "Game Theory: Zero-Sum Games, The Minimax Theorem", http://www.cs.toronto.edu/~nisarg/teaching/304f17/slides/CSC304-L5.pdf

[3] Albrech, Stefano (2017) "Multiagent Learning", http://www.cs.utexas.edu/~larg/ijcai17_tutorial/multiagent_learning.pdf

[4] Zhang, Haoqi, "Correlated-Q Learning and Cyclic Equilibria in Markov games", http://www.eecs.harvard.edu/~parkes/cs286r/spring06/presentations/zhang-presentation-0313.pdf

[5] Littman, M L. (1994). Markov games as a framework for multi-agent reinforcement learning Proceedings of the eleventh international conference on machine learning. Vol. 157, 242–249