CS7642: Project 3

Rory Michelen | rmichelen3@gatech.edu | Git Hash: e287415efe412abce8e1a65817c10595ac8f12f7

I. Introduction

This report will detail an attempt to replicate results from Greenwald and Hall's 2003 paper *Correlated Q-Learning.* In her paper, Greenwald discusses four algorithms for multi-agent Q-learning. She then runs an experiment to demonstrate the convergence and non-convergence of these four algorithms using a zero-sum game similar to American soccer.

This report will detail the author's efforts to replicate Green wald's results. It will provide an overview of multi-agent q-learning, including the four algorithms described by Greenwald. Additionally it will describe the process of replication including required assumptions as well as an analysis of the outcomes.

II. Problem Discussion

The objective of this project is to reproduce the results of Greenwald and Hall's 2003 paper, *Correlated Q-Learning*. The paper focuses on multi-agent reinforcement learning control problems, discusses traditional algorithms used in this space and introduces a new set of algorithms. Greenwald evaluates the performance of the algorithms described through two sets of experiments. The first set of experiments uses a trio of general-sum grid games while the second set focuses on a zero-sum game that is modeled after American soccer. This report will focus on the latter.

Note that the motivation for this replication is to demonstrate the utility of Greenwald's Correlated Equilibrium set of algorithms. This set of algorithms is a giant leap forward for multi-agent reinforcement learning since it (1) allows multi-agent reinforcement problems to reach an equilibrium easily using linear programming (not always possible for Nash Equilibriums) and (2) provides a framework for explicitly selecting among many equilibria (if multiple exist).

A. Soccer Environment: Description

The soccer environment described by Greenwald is a zero sum game in which two players occupy a 4x2 grid. This grid has two goals, one belonging to each of the players. One of the players is always in possession of a ball. At each time step, each of the two players chooses one of 5 actions: North, South, East, West, or Stick. The players' actions are executed in random order. If the set of actions results in no collision among the players, then both players' moves are executed successfully. However, if there is a collision, then the player whose move was executed first moves whereas the second player is forced to stick. Additionally, if the second player has the ball, then it is stolen and goes to player 1.

If the player with the ball moves into his or her own goal, he or she receives 100 points while his or her opponent receives -100 points (hence, zero-sum). However, if the player with the ball enters their opponent's goal, then the rewards issued are reversed.

B. Soccer Environment: Replication Details

Greenwald leaves many details of this environment to the imaginations of her readers. Firstly, she does not discuss whether players are allowed to select invalid moves. For example, if a player is in the northmost square, can he or she select to move north? In this replication players are allowed to select invalid moves. The result of an invalid move is the same as a Stick. This assumption was selected because it avoids having a different set of actions at each state (technically the action set could be the same at each state with some actions having probability zero).

The next detail not explicitly discussed by Greenwald is the definition of a collision. The first potential definition of a collision is that the two players' moves result in them occupying the same square in the grid. The second potential definition is that the two players' moves would require them to occupy the same square at least once during their moves (i.e. a swap). After trial and error, the second definition was assumed.

Thirdly, the authors did not thoroughly discuss the details of a ball switching possession. Greenwald states that "if the player with the ball moves second [during a collision], then the ball changes possession". However it is unclear if this is the only circumstance under which a ball switch may take place or one of many use cases. After experimentation, it was assumed that any time a player with the ball's move creates a collision, the ball changes possession.

III. Multi-Agent Q-Learning Mechanics

In traditional Q-learning, there is only a single agent interacting with the environment. Multi-agent Q-learning introduces the concept of multiple agents, each with their own agenda, interacting with the environment and, by association, with one another. Greenwald and Hall discuss many strategies to model this interaction. These models build upon the foundations of Q-Learning and can be split into three categories of strategies.

A. Strategy 1: Ignore your opponent

The first strategy is for each agent in the system to pretend that it is the only agent in the environment. By ignoring the other agents, their interactions become products of the environments. This strategy is equivalent to traditional Q-learning. It is easy to understand the issue with this strategy. From the agent's perspective, the same action can result in many future states, seemingly randomly. For example, consider an agent in the soccer environment. During one iteration, it moves West and receives 100 points. The next time the agent is in the same state, it attempts to move West, expecting a high reward, but instead it loses possession of the ball. In the second iteration, there was actually an opponent blocking the player. However, the agent only considered her space on the grid when evaluating her actions.

B. Strategy 2: Think Like Your Opponent

The second group of strategies enables agents to acknowledge that there are other agents with similar motives interacting with the same environment. This increases the state space to account for the positions of multiple agents (referred to as the joint action space). In this group of strategies, the agent must model the behavior of his opponent. In other words, does the agent think the other agents will help him score, is she trying to work against him, or something in between the two?

Once the agent has made an assumption about how the agents will all interact, it uses its own Q-table to estimate the behavior of the other agents. While this is an improvement of strategy one (at least we acknowledge that others exist!), it does not account for the fact that the opponents may have information that the agent does not. Therefore, the approximation of their behavior may be incorrect.

Greenwald discusses two variants of this strategy, Friend-Q and Foe-Q. In Friend-Q each agent assumes that her opponent is going to help her achieve her maximum reward. However, this should not be confused for cooperation. Since each agent thinks that the other is going to help her out, each agent takes a selfish action.

In Foe-Q, each agent assumes that the other is trying to minimize her reward. Therefore the agent identifies a policy that minimizes the damage that the other agent can do.

C. Strategy 3: Collude With Your Opponent

Similar to the second strategy, this strategy acknowledges that there are other agents interacting with the environment and assumes that the opponents motivations fall somewhere on the spectrum of coordination to competition. However, this group of strategies allows agents to either view or construct their opponents Q-tables. By doing so, agents can make better approximations of their opponents behavior.

Greenwald discusses four variants of this strategy, formally referred to as Correlated Q-Learning. Each of these four variants maximizes some function of all the agents' rewards. This paper will focus on Utilitarian Correlated Equilibrium Q Learning (uCE-Q) in which each agent takes the action that maximizes the sum of rewards for all agents.

At a glance, this seems nonsensical. The game is zero sum and therefore the sum of rewards will always be zero. However, the power of this algorithm lies in the fact that it achieves an equilibrium. In other words, each agent is satisfied with their policy and cannot achieve a better outcome given that their opponent does not change his policy.

D. A Modified Bellman Equation

The 3 strategies above have interesting implications for the original Bellman equation. In traditional Q-learning (as well as strategy 1) recall the update rule as:

$$Q(s,a) = Q(s,a) + \alpha(R_i + \gamma \underset{a`}{Max} Q(s`,a`) - Q(s,a))$$

Which can also be written as:

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha(R_i + \gamma \underset{a`}{Max} Q(s`,a`))$$

This update is a weighted average between the previous expected value of Q(s,a) and the new expected value based on our latest observation. This new expected value is composed of the observed immediate reward as well as the discounted expected value of the next state. However, the above equation assumes that the expected value of the next state is equal to

$$\underset{a`}{Max} Q(s`,a`)$$

or in plain english, the value of the action at the next state that maximizes the agent's value. However, when there are multiple agents, it is not guaranteed that the agent will be able to take the best action. There might be an opponent trying to minimize their value. To accommodate all of the potential scenarios, Greenwald proposes a generalization of the Bellman Equation described below:

$$Q(s,a) = (1-\alpha))Q(s,a) + \alpha(R_i + \gamma V(s'))$$

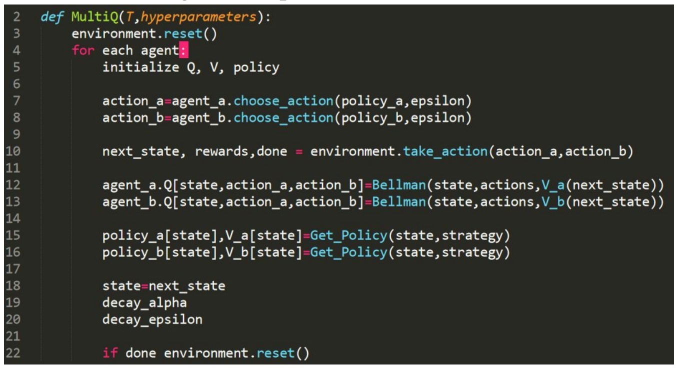
Here, the expected value of the next state is modified to include a function, V(s'). This general function empowers Q-learning to plug in a range of functions for the expected value at state s' to accommodate the motivations of other agents. Further, Greenwald makes another modification to the original Bellman equation and her result is the following:

$$Q(s,a) = (1-\alpha))Q(s,a) + \alpha((1-\gamma)R_i + \gamma V(s'))$$

By discounting the reward by $1-\gamma$, the impact of the immediate reward is diminished. Therefore less of the reward value is incorporated into the update. While Greenwald does not describe the intention of this modification, it is possible that this substitution provides bounds on the value of the update that are convenient for proving convergence.

E. Multi-Agent Q-Learning Algorithm

Based on the above discussion, Greenwald proposes a general algorithm for n-agent Q-learning. Figure 1is a paraphrased version of this algorithm specific to the case n=2.



```
def MultiQ(T,hyperparameters):
    environment.reset()
    for each agent:
        initialize Q, V, policy

        action_a=agent_a.choose_action(policy_a,epsilon)
        action_b=agent_b.choose_action(policy_b,epsilon)

        next_state, rewards,done = environment.take_action(action_a,action_b)

        agent_a.Q[state,action_a,action_b]=Bellman(state,actions,V_a(next_state))
        agent_b.Q[state,action_a,action_b]=Bellman(state,actions,V_b(next_state))

        policy_a[state],V_a[state]=Get_Policy(state,strategy)
        policy_b[state],V_b[state]=Get_Policy(state,strategy)

        state=next_state
        decay_alpha
        decay_epsilon

        if done environment.reset()
```

Figure 1: Pseudo-code for the author's implementation of 2-agent Q-learning.

IV. Implementation

A. Experiment Design

This paper replicates the four experiments that Greenwald implemented on the soccer environment in her 2003 paper. For each experiment, 1,000,000 iterations were ran on the

soccer environment with two agents using one of the four strategies discussed above: Q-learning, Friend-Q, Foe-Q, uCE-Q. For each experiment, a graph is generated demonstrating the errors, or temporal differences in Q-values at a particular state. The error formula provided for state s and joint action a at time step t and agent i is shown below:

$$err_i^t = \mid Q_i^t(s,a) - Q_i^{t-1}(s,a) \mid$$

B. Q-Learning

Replicating Greenwald's results for traditional Q-learning was the easiest of the four tasks since there is very little ambiguity in the mechanics of standard Q-Learning. Determining the hyperparameters was the only non-trivial component.

Greenwald states in her paper that $\alpha \rightarrow 0.001$. This is assumed to mean that alpha is initialized at some value and decays to 0.001 The details of the decay schedule are left to the imaginations of the readers. However, Greenwald provides a very interesting hint. She states "at all times, the amplitude of the oscillations in error terms are as great as the envelope of the learning rate". The envelope, or window, created by the error terms, provides us with the general shape of alpha. If epsilon were constant, we could use this information to interpolate but Greenwald explicitly states that epsilon decays from 1 to 0.001. Therefore, visual inspection of figure 3(d) indicates that the envelope (and therefore alpha) decays exponentially. Therefore the same assumption was made for epsilon.

Figure 2 shows the replication results and should be compared with Greenwald's figure 3(d):

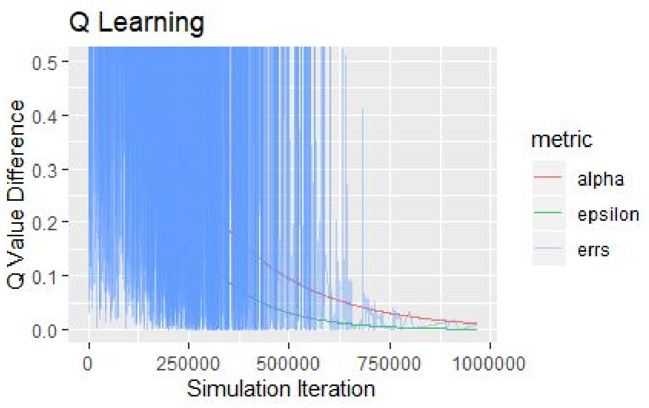


Figure 2: Simulation results for standard Q-Learning. This chart should be compared to Greenwald's 3(d). Epsilon and alpha are added for reference.

While this chart does not precisely match Greenwald's, the general trend and conclusion is the same. Q-learning does not converge in this multi-agent environment. Therefore a strategy that explicitly models the behavior of other agents is needed.

In Greenwald's original chart, it is difficult to be convinced of non-convergence since a decaying alpha forces the errors to decrease exponentially. Therefore, this chart provides alpha and epsilon as a reference. From here it is clearer that the amplitude of oscillations in the errors is proportional to alpha at all points.

The differences from Greenwald's chart are also clear. Firstly, the error terms decrease much more quickly. This is likely a result of the decay schedule and initial values for alpha and epsilon, but could also be due to Q-table initial values or the 3 assumptions discussed in section II B. The chart in this report is also much denser in the first 250,000 iterations than

Greenwald's figure 3(d). This could be due to the plotting logic used by Greenwald. In the plot above, error terms of zero were excluded. Note that there are many zero error terms because not every iteration visits the particular state of interest. Nevertheless, it appears that Greenwald removed additional points and potentially interpolated.

C. Friend-Q

The Friend-Q algorithm adds an additional layer of complexity to standard q-learning. In this algorithm, the agent must consider the joint-action space, that is, all pairwise combinations of actions between him and his opponent. Then, the agent selects the action pair that maximizes his own rewards. Unfortunately for this agent, his opponent is doing the exact same thing. Perhaps a better name for this algorithm is 'Frenemy-Q".

The Friend-Q implementation requires a few assumptions regarding Greenwald's implementation. Firstly, there is no mention of epsilon decay schedule in the 2003 paper. However, after a review of the literature, it was discovered that Greenwald's definition of "off-policy" is such that epsilon=1. In other words, a random action is taken at each step. However, evaluation of a given state, s is determined using V(s).

Another assumption required is the handling of terminal states. Greenwald's Multi-agent Q-learning algorithm does not mention any special handling of the terminal state update. When V(s`) is ignored for terminal state updates, the Friend-Q algorithm converges much quicker than Greenwald's. Therefore, since the purpose of this project is to replicate Greenwald's results, V(s`) was included in terminal state updates.

Identifying the initial value of alpha was handled using a sweep of values. If alpha was too close to 1, the algorithm converged slower than Greenwald's results since too much randomness was learned at the beginning. With alpha too low, the initial error terms were too heavily discounted and never reached 0.5 as seen in figure 3(c).

Figure 3 demonstrates the replication results for Friend Q. In order to replicate Greenwald's chart as closely as possible, a 100-point moving average is plotted instead of the raw error values.

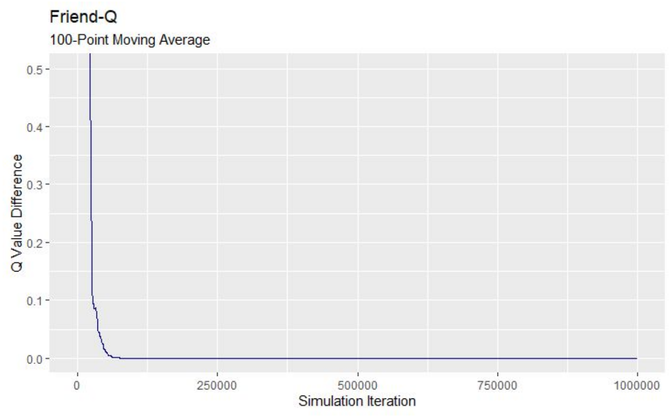


Figure 3: Simulation results for standard Friend-Q. This chart should be compared to Greenwald's 3(C). Epsilon and alpha are added for reference.

D. Foe-Q

Foe-Q was implemented and ran using similar assumptions as in the Friend-Q implementation: $\epsilon = 1$, $\alpha = 0.9$, and $\alpha \rightarrow 0.001$ exponentially. Similarly, there was no special handling of terminal states.

Figures 4 and 5 share the results of this replication. Figure 4 should be compared with Greenwald's 3(b).

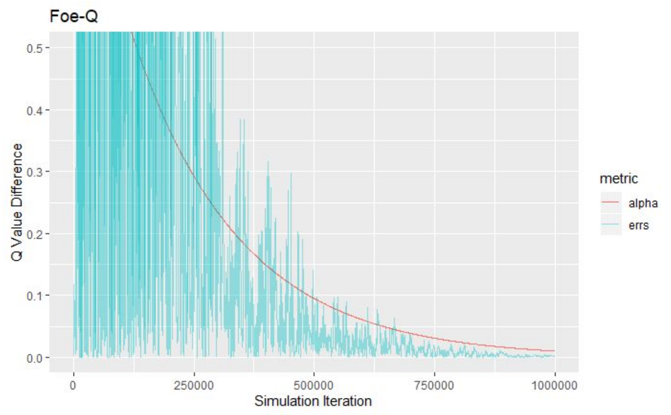


Figure 4: Simulation results for Foe-Q This chart should be compared to Greenwald's 3(b). Alpha is added for reference.

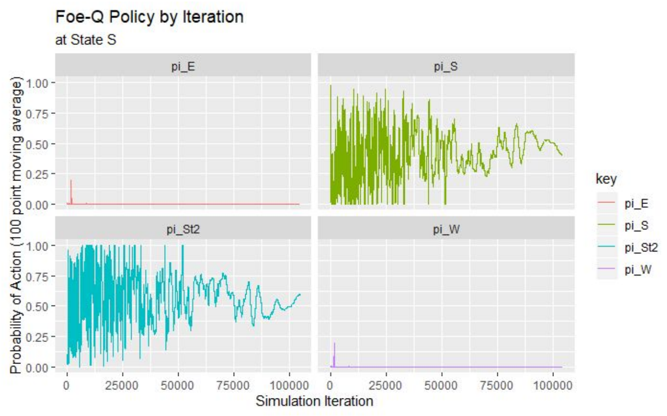


Figure 5: Foe-Q Policy by iteration. Probability of each action by timestep. North and Stick are summed together..

Figure 5 displays the convergence of agent A at state S to a non-deterministic policy of 50% sticking and 50% south. Note that since invalid moves were allowed, moving north from state S was equivalent to sticking. Therefore, the agent oscillated between the two actions. This is accounted for in figure 5 by summing the probabilities of agent A choosing action North and Stick at state S.

This stochastic strategy makes sense intuitively. To understand why, consider the agent's other options. Moving east is counter productive since this enables her opponent to move towards his goal. Moving west is a poor strategy because the agent will either collide with agent B (and not steal) or agent B will head south, making A even further from a steal than their previous state. This leaves Stick (by means of Stick or North) and South. If either of these strategies is chosen deterministically, then agent B can take advantage of this information. By the same logic, agent B's action will be non-deterministic. Therefore, A cannot predict if Stick or South is better and will ultimately split the difference.

E. uCE-Q

Figure 6 displays the result of replicating uCE-Q, which required a linear program with 65 inequality constraints. The same hyperparameters as Foe-Q and Friend-Q were used. Note that setting $\epsilon = 1$ avoids a difficult question to answer. Consider a scenario where epsilon is not equal to one. In this case, at each time step, both agents are required to select an action from a non-deterministic policy. However, this policy specifies probabilities from the joint action space. In other words, both agent A and agent B would have to choose actions for themselves as well as their opponents. This can be handled easily by a centralized system that chooses 1 joint action pair (for example, stop lights). However the solution for a decentralized system is not clear and opens many more questions.

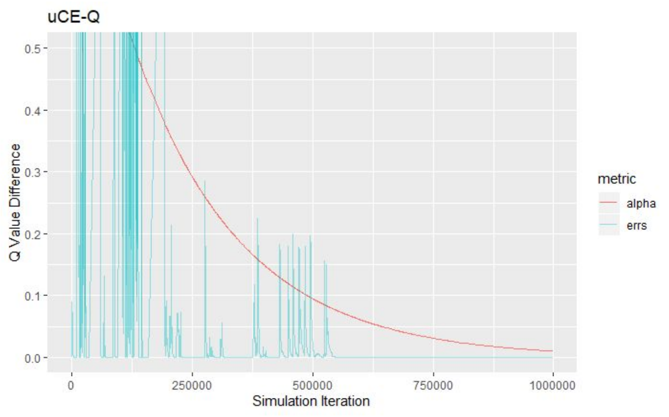


Figure 6: Simulation results for uCE-Q. This chart should be compared to Greenwald's 3(A). Epsilon and alpha are added for reference.

V. Conclusions

This report has detailed the process of replicating Greenwald and Hall's results for 4 multi-agent reinforcement learning algorithms in a zero-sum environment. The results of the replication are generally consistent with Greenwald's and can be summarized as follows: (1) Traditional q learning did not converge. In other words, the strategy of ignoring other agents and assuming their behavior is purely due to the environment does not necessarily converge. (2) The Friend-Q algorithm converges to a deterministic policy for both agents. However the strategy of either agent is poor because each assumes the other is trying to help her. (3) Foe-Q and uCE-Q converge to non-deterministic strategies in which each agent is able to successfully minimize the damage done by their opponent.

VI. References

[1] Amy Greenwald, Keith Hall, and Roberto Serrano. "Correlated Q-learning". In: ICML. Vol. 20. 1. 2003, p. 242.

[2] A. Greenwald, K. Hall, and M. Zinkevich. (2005). Correlated Q-learning. Retrieved from Brown University Library.

[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

[4] Sutton R. S., & Barto A. (2018) Reinforcement learning: An introduction, Cambridge MA: The MIT Press