Project 3: RoboCup Correlated-Q Edition

Rodrigo De Luna Lara

November 26, 2018

Last commit hash: **7e0ffd99c80f4c8f1811498262749156292 4fa32**

The objective of this paper is discussing replication of the results for the soccer environment experiments in the paper *Correlated-Q Learning* (Greenwald and Hall 2003), specifically for the plots shown in Figure 3 of said paper for standard Q-learning, friend Q-learning, foe Q-learning and correlated Q-learning.

First, the soccer environment, based on Figure 1 (Greenwald and Hall 2003) was created. It consists of a 2 player game on a 4x2 grid, corresponding to 8 possible player positions (0-7), with 5 possible actions (moving North, West, South, East, or staying in place). The possession of the ball adds to the state. Each player has an end-zone, if the opposite player moves the ball into this zone they are awarded 100 points, and the opponent is penalized with the same amount of points (-100).

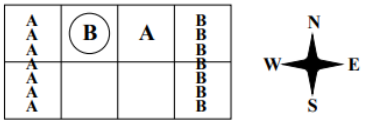


Figure 1: Soccer Environment

When an action is taken the position is updated by adding an offset to the current position based on the 4x2 grid, if a player is in state 1 (the second column in the top row) and he moves south his position is offset by 4 (ending up in state 5), this simplifies the description of the state. For each learner the same random seed is used when the game is initialized, so that conditions are normalized. Also, the environment is setup in such a way that respects the change of possession: "*if the player without the ball moves into the player with the ball, attempting to steal the ball, he cannot. But if the player with the ball moves into the player without the ball, the former loses the ball to the latter*" (Greenwald and Hall 2003)

Greenwald and Hall don't give much details on the exact implementation of their environment, details such as how randomness is considered, and what specific programming language was used for the experiments are not included. There are however some other specifications regarding the learners, namely that $\alpha \to 0.001$, $\lambda = 0.9$ and that $\epsilon \to 0.001$ for ϵ-greedy on-policy Q-learning. However, it is left to the imagination of the reader what are the starting values for α and ϵ and what's the decay schedule for both of them.

Regarding the starting values for α and ϵ, for α it was set to 0.90, which was just set to this value as most Reinforcement learning literature seems to veer off starting very high values of α. On the other hand, several values for ϵ were tested for the standard ϵ-greedy Q-learning, but the impact on the results was not deemed significant, so it was hardcoded at 0.90 too. What was very significant however was the choice of decay schedule for α, so 3 different schedules were derived from fitting a linear, logarithmic and exponential function on the extrema $[0, 0.9]$, $[1 \times 10^6, 0.001]$. The schedules for α and ϵ can be seen in Figure 2.

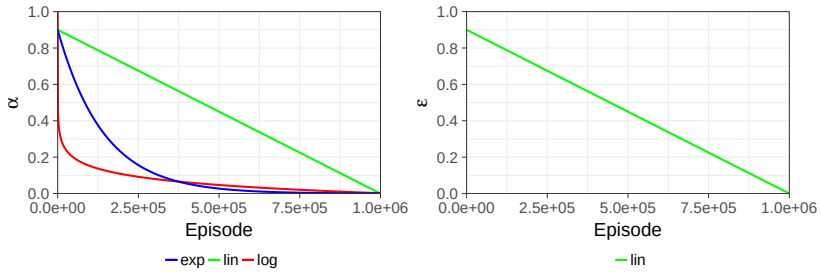


Figure 2: Decay schedules for α and ϵ

To replicate the results of Figure 3 from Greenwald and Hall, the change in the value for state s as described in the paper was tracked. This state is the one described in Figure 1, in which player 1 is in position 2, player 2 is in position 1, player 2 has the ball, with "*player A taking action S and player B sticking*" (Greenwald and Hall 2003). This state corresponds to the index [2, 1, 1, 2, 4] in the environment created for this analysis.

First, standard ϵ-greedy Q-learning was implemented, with the ϵ decay described in Figure 2, the Q-table in this case only considers the position of both players and the action of player 1, as well as the possession of the ball. At each episode an action is either taken at random or based on the Q-table depending on ϵ. Then the Q-table is updated based on the reward obtained. For friend Q-learning, both players take actions at random and the Q-table is expanded to consider player 2 actions, with the same update function.

Regarding foe Q-learning, both players take actions and random and the Q-table is updated by the solution of the minimax Q function (given that the game is zero-sum) using the glpk linear programming solver in *cvxopt*, automatically creating and filling the necessary matrices, considering that player 2 is acting against player 1 actions, as defined by Littman (2001).

Finally, correlated Q-learning was implemented in a similar fashion, building the matrices for the linear programming by hand and filling the necessary values, considering the constraints for each player. The matrices are created considering the combination of both players' actions to define their respective constraints. An equilibrium can be reached with maximizes both players' rewards. Each strategy for each player is compared to every other strategy, for both players, and all probabilities are constrained to be >0.

Figure 3 shows the output for Q-learning, friend Q-learning and foe Q-learning for the three α decay schedules.

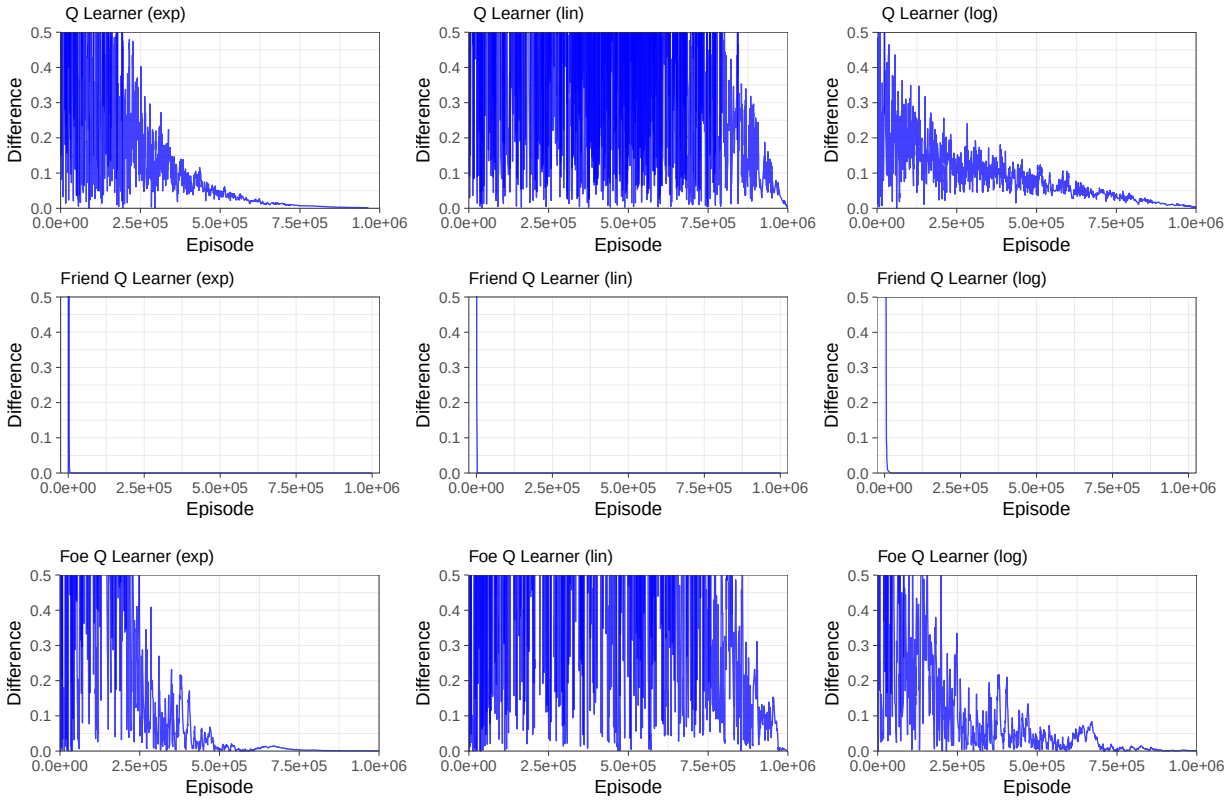
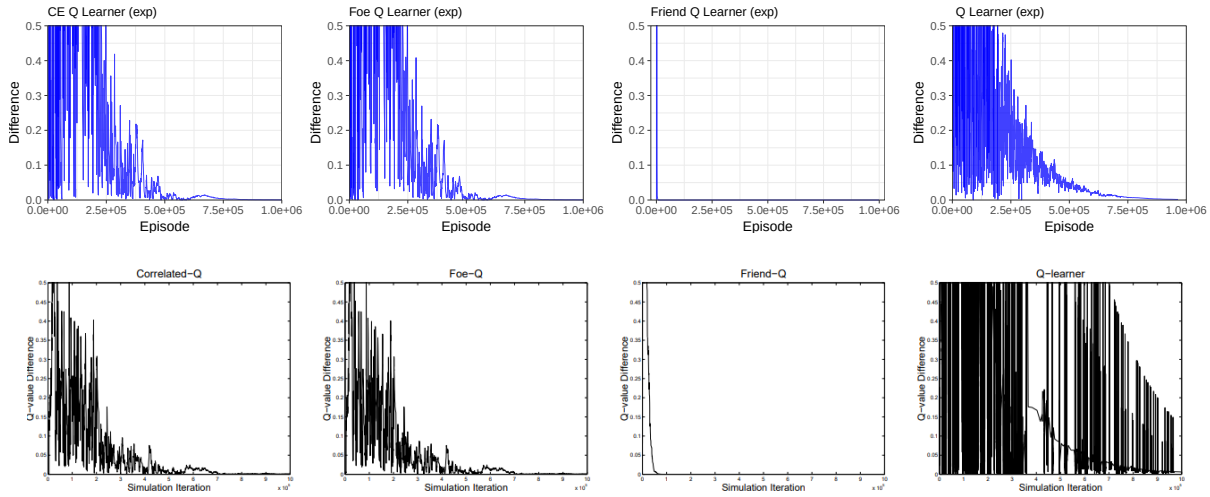


Figure 3: Output for first 3 learners on different α schedules

It can be seen that Greenwald and Hall most likely used an exponential decay schedule for α, given how it more closely resembles the results in their paper. A linear decay schedule seems to give significantly different results, with convergence only starting to show near the end of the 1 million episodes. For friend Q-learning there doesn't seem to be a very significant difference, as the learning rate seems to not matter much when both players are "*cooperating*".

Figure 4 shows the comparison of the output for this analysis using the exponential decay function for *alpha* alongside the results from Greenwald and Hall.



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

Figure 4: Comparison of Results

It can be seen that the results match the behavior of the implemented algorithms very closely, although with some differences. Both correlated Q-learning and foe Q-learning have very similar results, but they have larger difference values before convergence in comparison with Greenwald and Hall's. Nonetheless they seem to converge near the same episodes as their implementation, after 500 thousand episodes. Friend Q-learning seems to match the results too, converging very quickly before reaching 100 thousand episodes.

The largest difference, surprisingly, is in Q-learning. The implementation in this analysis is standard ϵ-greedy Q-learning, which is what Greenwald and Hall state to be using in their paper, but their results have significant spikes in the Q-value difference. However, if these spikes are ignored, it can be seen behind them that the difference seems to be converging in a fashion similar to what was produced for this analysis. Given that Greenwald and Hall don't provide enough details on how they're handling an ϵ-greedy policy and how ϵ is decaying, it is difficult to assess why the results are so different.

The plots in Greenwald and Hall have a y-axis limit of 0.50, but the results obtained show values well over that limit, as seen in Figure 5, the reason why they decided to do this is unclear and not discussed in the paper at all. Both methods still seem to have very similar results however, so this may not be very significant.

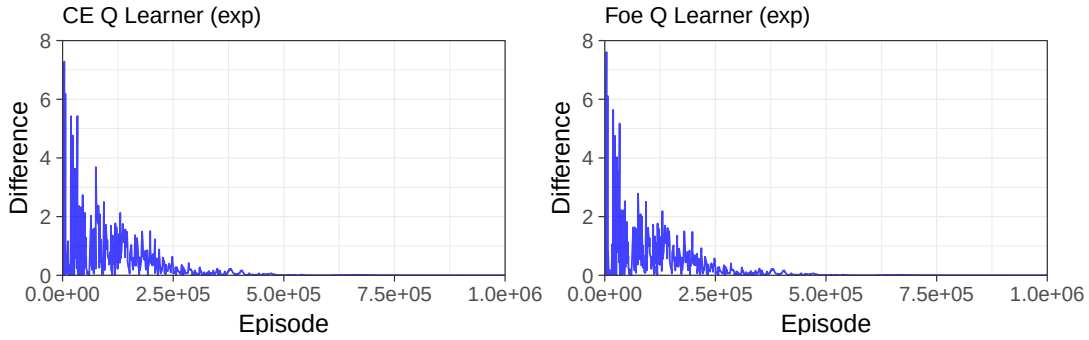


Figure 5: Raw Output for Foe and Correlated Q-Learning

Figure 6 shows the difference between updates for foe and correlated Q-learning, which despite the fact that the plots seem nearly identical shows that the Q-values are updated in different magnitudes by each method, although they both converge to similar values.

There were several difficulties in implementing the learners and trying to replicate the results by Greenwald and Hall. Surprisingly one came from the random functions in numpy, for some reason some methods to get random numbers, although intuitevely the "same" gave different results when simulating the environments, particularly

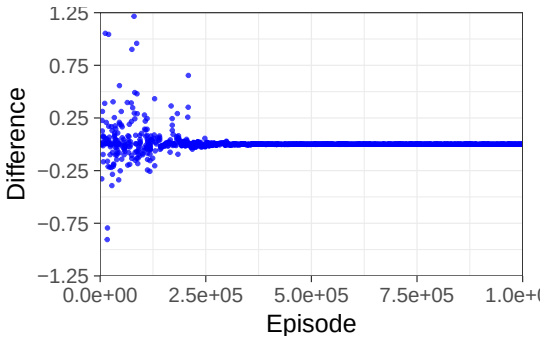


Figure 6: Δ Difference between Foe and Correlated Q-Learning

numpy.random.choice and *numpy.random.randint*. Additionally the setting of the random seed was not very straightforward, placing it in different parts of the code seemed to have various effects.

Randomness is not to be minimized when dealing with reinforcement learning, and despite this authors and researchers seem reluctant to better document their experiments and ensure reproducibility, which should be an essential part of the scientific method but seems overlooked by most researchers in ML and AI. The issue seems to be so common in science in general that there exists the term *replicability crisis*, "*an ongoing methodological crisis in the social and soft sciences in which scholars have found that the results of many scientific studies are difficult to impossible to replicate or reproduce*" (Wikipedia contributors 2018).

Given how difficult it seems to exactly replicate Reinforcement Learning papers, perhaps this concept should be extended to include it. And there certainly is no excuse in this modern era, researchers could provide full replicable cases in Python for example, setting random seeds to improve reproducibility or at least providing more thorough descriptions of their implementations (such as the decay function for α instead of just stating that it tends to 0.001)

Given more time, it'd be good to improve how the correlated equilibrium is calculated with linear programming and the minimax function is calculated. Using matrices makes it very difficult to interpret what is going on, and almost requires setting them in paper first and then figuring out how they can be constructed from available data, which was the approach followed in this analysis. Using symbolic programming may be more interpretable and perhaps would have made it easier to implement both foe and correlated Q-learning.

Nonetheless, regardless of the lack of information for replicablity provided by Greenwald and Hall, the results mostly match their theoretical interpretation of the algorithms. Friend, foe, and correlated Q-learning all show convergence in accord with the paper, but Q-learning has contradictory results with respect to the paper. The vague description of the *on-policy Q-learning (i.e., ϵ-greedy, with $\epsilon \to 0.001$, $\alpha \to 0.001$, and $\gamma = 0.9$)* (Greenwald and Hall 2003), was followed to implement the learner but it still failed to produce their results.

Therefore, the algorithms show to work as intended given a zero-sum game and generally match the paper's results (except Q-learning), and due to the lack of information exact replicability is practically impossible for the reasons discussed above.

References

Greenwald, A., and K. Hall. 2003. "Correlated-Q Learning." *In Proceedings of Eighteenth International Conference on Machine Learning.*

Littman, M. 2001. "Friend-or-Foe Q-Learning in General-Sum Games." *In Proceedings of Eighteenth International Conference on Machine Learning.*

Wikipedia contributors. 2018. "Replication Crisis — Wikipedia, the Free Encyclopedia." https://en.wikipedia.org/w/index.php?title=Replication_crisis&oldid=868455840.