# CS 7642: Reinforcement Learning and Decision Making
# Project 3: ”Soccer”

**Thomas Kim**
tkim338
git hash: 580e11d7d6f06ed6b0018026c2e00b0b6f218548

## Problem

The objective of this project is to replicate a set of figures (Figure 1) presented in a paper by Amy Greenwald and Keith Hall that examines the behavior of various reinforcement learning algorithms: correlation-Q, foe-Q, friend-Q, and standard Q-learning. These algorithms are applied to agents of a simplified version of a ”soccer” game initially described by Littman in 1994 and to a specific state of the game in order to directly compare the behavior of each algorithm against each other.

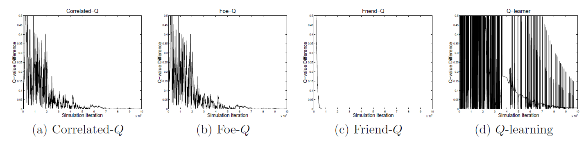


*Figure 3.* Convergence in the soccer game. All algorithms—except $Q$-learning—converge. As above, the CE-$Q$ algorithm shown is $u$CE-$Q$.

Figure 1: Performance of various reinforcement learning algorithms, (Greenwall and Hall, 2003)

## ”Soccer” Environment Development

The simplified ”soccer” game used here is made up of a 4 by 2 grid where the far left and far right columns are the goals for players A and B, who start in two tiles in the center of the grid. Each turn is processed as follows:

1. The state of the game is given to players A and B.
2. Players A and B decide on their own respective next moves, which can be north, south, east, west, or stick/stay.
3. Either player A or player B is randomly chosen to move first.
4. The first player to move makes their move.
5. The second player to move makes their move.

When a player makes a move, some constraints are checked before allowing a move to complete:

1. Players may not move outside of the bounds of the grid. A move that would place a player outside of the grid is instead processed as a no-op.
2. Players may not occupy the same space. A move that would place a player on the same tile as another player is instead processed as a no-op and each player loses 1 point from their score.
3. In cases where a player holds the soccer ball and attempts to move to a tile occupied by another player, no players change locations but the soccer ball is transferred to the possession of the player on the destination tile.
4. When a player with the ball moves onto a goal tile, the episode of the game is completed. 100 points are granted to the player whose goal has been scored on and 100 points are taken from the player who has been scored against. Goals can be scored by either player into either goal, meaning players can score a goal to earn points for themselves or they can score into their own goal and lose points as a result.

The initial state of the game at the beginning of each episode is constant, as shown in Figure 2.

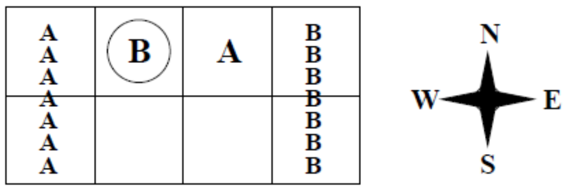


*Figure 4.* Soccer Game. State $s$.

Figure 2: Initial state of ”soccer” game.

The locations of players A and B are marked by their respective single letters. Each player’s goal is marked by a sequence of their respective letters. Player A scores positive points when the ball enters one of the two tiles marked with A’s and loses points when the ball enters one of the two tiles marked with B’s. The inverse is true for player B. The location of the ball is marked with a circle. The initial state always has the ball under the possession of player B. This specific characteristic is an assumption that is not explicitly stated in either the Greenwald and Hall paper or the Littman paper and was made to reduce the set of tested episodes to those relevant to this study. Another pitfall encountered when implementing this environment was initially implementing the larger 5 by 4 grid described in the original Littman paper. A second reading of the Greenwald and Hall paper revealed that a 4 by 2 grid was the appropriate grid size.

## Experiment Implementation

The following algorithms are based on the Nash-Q learning algorithm presented by Hu and Wellman in 1998 (Figure 7). The Nash function is substituted for the various algorithms for each experiment.

$$Q_i[s, a_1, \ldots, a_n] := (1 - \alpha_t) Q_i[s, a_1, \ldots, a_n] + \alpha_t \left( r_i + \gamma \mathsf{Nash}_i(s, Q_1, \ldots, Q_n) \right) \quad (6)$$

Figure 3: Nash-Q equilibrium (Hu and Wellman, 1998).

Greenwall and Hall state that the set of figures in their paper shows the change in Q-value in player A's policy for a particular state and a particular set of actions. The state examined is the initial state given above in Figure 2 and the actions examined are S (south) for player A and stick (stay) for player B. For Q-learning, where each agent ignores the actions of the opponent, the actions of player B are ignored.

The experiments here also used agents with assumed values, decay rates, and minimum values for $\alpha$, $\gamma$, and $\epsilon$. $\alpha$ was set to an initial value of 0.9999 with a decay rate of 0.999 per time-step and a minimum value of 0.0001. $\gamma$ was set to an initial value of 0.9 with no decay. $\epsilon$ was set to an initial value of 0.999 with a decay rate of 0.9999 per time-step and a minimum value of 0.0001.

### Friend-Q Experiment

The friend-Q algorithm was first implemented and tested. This algorithm is presented by Littman in 2001 (Figure 4). This Nash function is substituted for the Nash function in

$$\mathsf{Nash}_1(s, Q_1, Q_2) = \max_{a_1 \in A_1, a_2 \in A_2} Q_1[s, a_1, a_2] \quad (7)$$

Figure 4: Friend-Q Nash equilibrium (Littman, 2001).

the Nash-Q algorithm by Hu and Wellman 1998. In this implementation, the agent governing player A's actions was trained by using an agent that chose actions for players A and B to maximize the reward for only player A.

Data from the experiment (changes in Q-value for player A in a particular state and set of actions) are collected at each time-step and output to a comma-delimited file at the end of the experiment, before plotting the collected data. The resulting figure matches the figure presented by Greenwall and Hall fairly well, with a high initial change in Q-value that drops quickly to close to zero, where it remains for the rest of the episodes. An interesting difference (that's also seen in the other experiments) is that the Q-value found here converges much more quickly than found by Greenwall and Hall. This is possibly explained by the aggressively high learning rate chosen in this experiment in order to train the agent more quickly. This difference in convergence speed does not seem to be critical, so no further testing was conducted to see if the agent could be trained more slowly with a reduced learning rate.

Another significant difference found between this experiment and that conducted by Greenwall and Hall is the final policy to which player B converged. Greenwall and Hall state that the friend-Q algorithm applied to this enviroment reaches a deterministic policy where (in the initial state), player B will choose to move east, passing the ball to player A, and expecting player A to score a goal for player B. In the experiment conducted here, when both players were governed by agents using the friend-Q algorithm, the final behavior the players in the initial state were player B sticking and player A moving south.

In order to reconcile this difference, additional experiments were run to explicitly train each agent to expect their opponent to help them score (i.e. act as a friend). Running a set of experiments to train players A and B to help score points for player B resulted in a policy that matched that of Greenwall and Hall's result. This was taken as proof that the friend-Q algorithm was correctly implemented. Then the same experiment was run, but modified to score points for player A. Q-value difference data was collected here and plotted in Figure 5. The original figure from Greenwall and Hall is shown in Figure

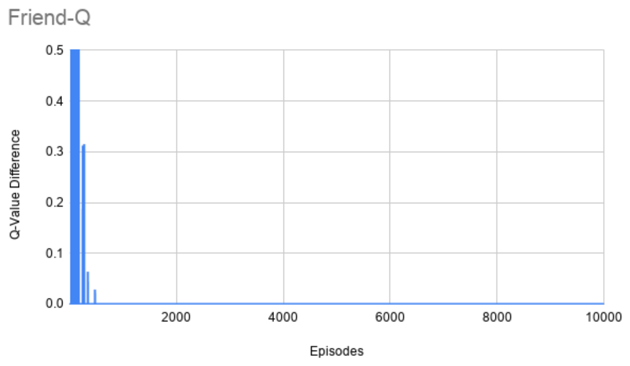
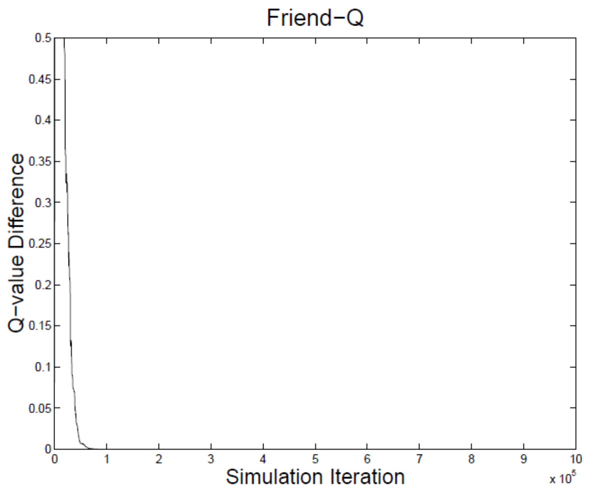


Figure 5: Q-value progression of friend-Q agent policy.

### Foe-Q Experiment

The foe-Q algorithm was implemented next using the foe-Q equation presented by Littman in 2001 (Figure 7). As done in the friend-Q experiment, this equation was used in place of the Nash function provided in the Nash-Q algorithm presented by Hu and Wellman. The implementation of this



(c) Friend-$Q$

Figure 6: Original Q-value difference figure of friend-Q agent (Greenwall and Hall, 2003).

$$\text{Nash}_1(s, Q_1, Q_2) = \max_{\pi \in \Pi(A_1)} \min_{a_2 \in A_2} \sum_{a_1 \in A_1} \pi(a_1) Q_1[s, a_1, a_2] \quad (8)$$

Figure 7: Foe-Q Nash equilibrium (Littman, 2001).

Nash function was more complex than that of the friend-Q experiment. The minimax function from Homework 6 was adapted to work here. The optimal policy probabilities $\pi$ where calculated using the linear programming python package "pulp" using the Q-values of the current state of the game as constraints. This probability distribution was then used to randomly select an appropriate action for the agent to take.

The resulting plot matches the plot presented by Greenwall and Hall fairly well, except for the previously mentioned difference in learning rate between this experiment and that conducted by Greenwall and Hall. Another difference that may be significant is that Greenwall and Hall state that the foe-Q agent results should match exactly the results produced by the correlated-Q learning agent, which was not the case in this experiment.

## Correlated-Q Experiment

The algorithm implemented for the correlated-Q learning agent was taken from Greenwall and Hall 2003, shown in Figure 10.

This is a specific type of correlated-Q learning presented by Greenwall and Hall as the utilitarian correlated-Q learning algorithm, which maximizes the sum of the players' rewards. This algorithm was implemented in a similar way

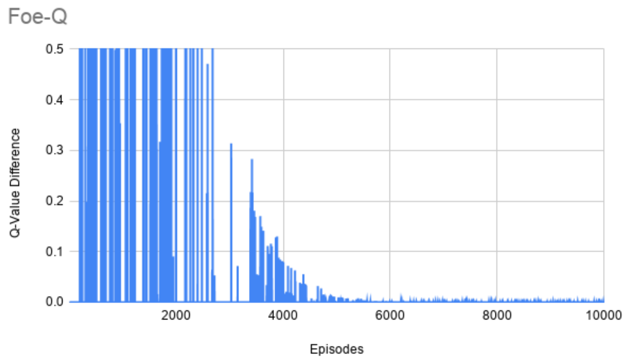
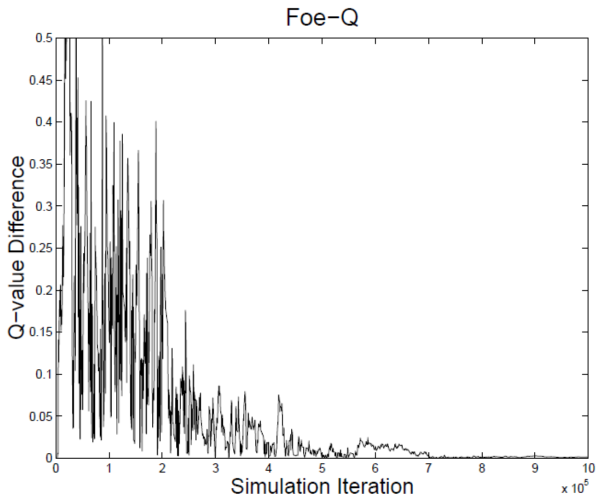


Figure 8: Q-value progression of foe-Q agent policy.



(b) Foe-$Q$

Figure 9: Original Q-value difference figure of foe-Q agent (Greenwall and Hall, 2003).

$$\sigma \in \arg\max_{\sigma \in \text{CE}} \sum_{i \in I} \sum_{\vec{a} \in A} \sigma(\vec{a}) Q_i(s, \vec{a}) \quad (9)$$

Figure 10: Utilitarian correlated equilibrium Q-learning (Greenwall and Hall, 2004).

to the friend-Q agent, but in this case, the optimal agent was found by computing the element-wise sum of the two player's Q-value matrices. The results are shown below in Figure 11, with the original Greenwall and Hall plot shown in Figure 12.

The figures match fairly well with the common exception being the learning or convergence rate. It's also noted that this figure does not exactly match that of the foe-Q agent,

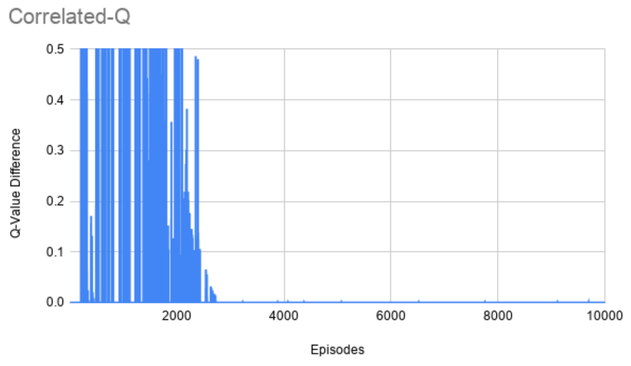


Figure 11: Q-value progression of correlated-Q agent policy.

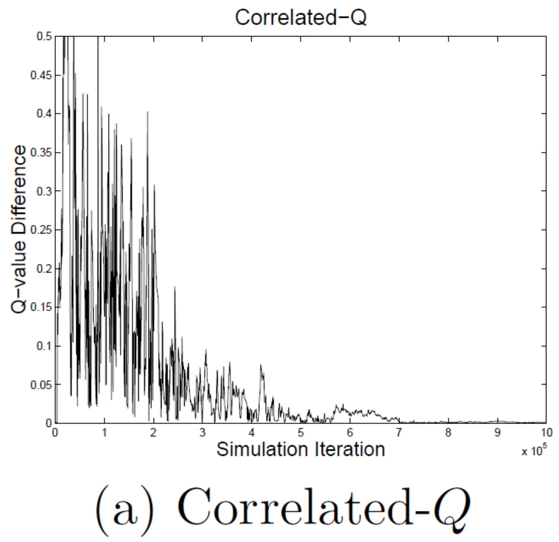


Figure 12: Original Q-value difference figure of correlated-Q agent (Greenwall and Hall, 2003).

which Greenwall and Hall state should be the case. The overall trends are the same, but differences from episode to episode exist. These differences may be explained by a few different factors. A seed for the random number generator used for a handful of different purposes in the experiments was not set and each experiment was run independently, resulting in variations in behavior between experimental agents. Setting a common seed could eliminate these differences, however, there are still differences in the order of operations between the correlated-Q experiment and the foe-Q experiment, which would mix up the order in which random numbers were generated, negating the benefit of using a common seed. Code changes could be made so that these random numbers where generated in the same order for each experiment and thus exactly replicated the random numbers drawn, but this would necessitate a complete rewriting of the project code and was therefore not explored further for the scope of this project, as matching general trends was seen as more critical than matching Q-value differences from episode to episode.

## Q-Learning Experiment

The Q-learning algorithm was adapted from the SARSA model first implemented in Homework 3. This implementation is similar to that of the friend-Q model, where the maximum Q-value is found and the associated action taken. However, in this experiment, as described and conducted by Greenwall and Hall, each agent's policy was oblivious to the actions of the opponent, reducing the space of the policy by a dimension.

In this experiment's initial implementation, the space of the policy was not reduced, as it was shared among all of the experiments, but instead, in calculating the optimal action for an agent, the sum of expected rewards for all opponent actions for each self action was considered. This resulted in an agent that approximated the behavior of an agent that ignored the opponent's actions, however, since the Q-value update function was left untouched, the sum of Q-values used here learned at a slower rate than expected and was skewed by Q-values with large magnitudes that were not frequently visited.

The remedy to this issue was to leave the action calculating function as-is for the Q-learning agent and instead modify the update function. The Q-value update function was modified to update and copy the newly computed Q-value across all opponent actions in the matrix, thus resulting in a Q-policy space that was reduced by a dimension, instead copied across that dimension, resulting in a policy functionally identical to one that ignored opponent actions. The final results of this experiment are shown in Figure 13 below with the original Greenwall and Hall plot in Figure 14.

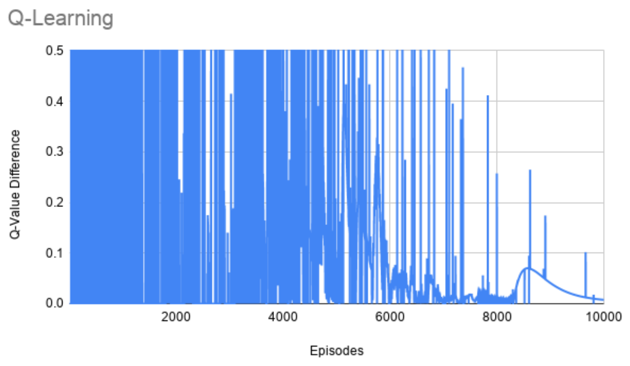
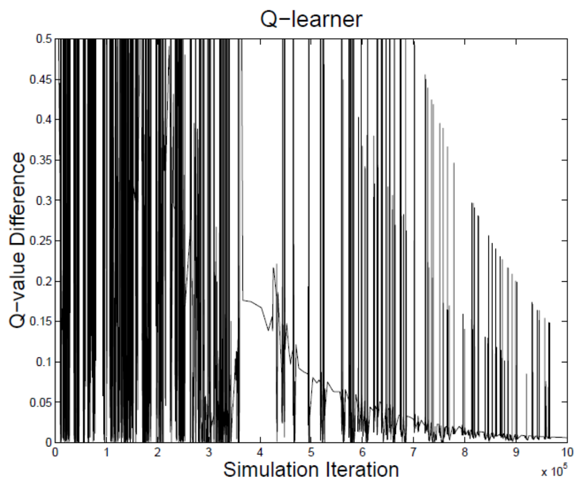


Figure 13: Q-value progression of Q-learning agent policy.

These figures match well in the characteristics highlighted by Greenwall and Hall, in that average Q-value difference does gradually decrease, but does not converge, as Q-values spike intermittently, with errors greater than the actual value itself. These characteristics are shown in the plot generated here. There is still the difference in learning rate exhibited in the previous figures.



(d) $Q$-learning

Figure 14: Original Q-value difference figure of Q-learning agent (Greenwall and Hall, 2003).

## Evaluation of Results

The set of figures generated by the experiments outlined here match fairly well to the figures presented by Greenwall and Hall in their 2003 paper. There is an overall difference in the number of episodes required to reach convergence, but this was attributed to possible differences in initial learning rate $\alpha$ and its decay, although the minimum $\alpha$ was replicated.

There was also a difference between the results of the foe-Q agent and the correlated-Q agent implemented in this set of experiments. Greenwall and Hall state that these results should be identical and present identical plots for their results. The differences in the plots generated here may be explained by differences in random numbers given by the random number generator used in a handful of places through the project code. Had the experiments been run concurrently or a random number generator seed been used, these plots may match episode to episode.

## References

Amy Greenwald and Keith Hall. ”Correlated Q-learning”. In: *ICML* Vol. 20. 1. 2003, p. 242.

Junling Hu and Michael P. Wellman. ”Multiagent reinforcement learning: Theoretical framework and an algorithm.” In: *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 242-250, 1998.

Michael L. Littman. ”Markov games as a framework for multi-agent reinforcement learning”. In: *Proceedings of the Eleventh International Conference on Machine Learning*, pages 157–163, July 1994.

Michael L. 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.