Project 3: RoboCup

Ran Chen
email: ranchen@gatech.edu

1 Game: Soccer

Game The soccer game is grid game of 2 rows and 8 columns, with two players A and B, as shown in **Fig** 1. The rules are determined according to the published description of the game, as well as some reasonable assumption, since a detailed list of rules is hardly presented in full in those publications. The rules are:

1. The game is always initiated or reset to state s, as shown in **Fig** 1, in which B holds the ball, occupying position $[0, 1]$ and A occupies position $[0, 2]$.
2. Each player has two possible actions [stick, N, S, E, W], and the action vector is represented in that order, e.g. $[0.3, 0, 0.7, 0, 0]$ means moving South with 0.7 chance and stick with 0.3 chance.
3. At each step of the game, A and B determines their moves simultaneously, but the execution of those two moves are always in random order.
4. When a player with the ball is trying to move into the position already occupied by the other player, then the ball changes possession, and neither player changes position.
5. When it ball is in a player's goal, then that player scores 100, its opponent scores -100, and game ends, no matter who has the ball at the time.
6. At a particular step of the game, game ending is only checked after the moves of both players are executed, i.e. even if the intermediate state of the game after the first move is executed includes the ball in the goal, the game does not end until the criteria is met after the second move is executed.

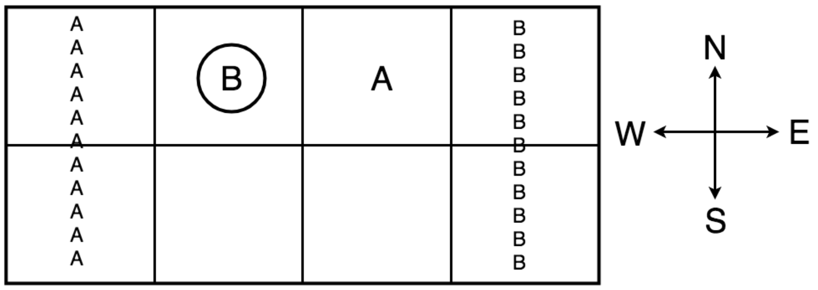


Figure 1: An illustration of the soccer grid game environment. This illustration is showing the initial state of the game, state s. The circle on player B indicates B is in possession of the ball. 4 directions of movement are demonstrated on the right. The lines of A and B indicates the position of goals of respective player.

Approach Different variants of multiagent Q-learning algorithms will be tested with soccer environment. The generic algorithm[1] is shown in **Algorithm** 1. Note: here the term "off-policy" refers to Sutton's definition, more clarification presented 2.

2 Clarifications: difficulties

Most of the difficulties I encountered in Project 3 were related to ambiguities in the descriptions of various learning algorithms and the game environment. After careful reading a few other related literature and consideration, the environment is re-defined in 1. A few additional clarifications regarding the algorithms need to be made before I can proceed.

Algorithm 1 Off-policy multiagent Q-Learning

```
input: $f$ - strategy selection function; $\gamma$ - discount; $\alpha$ - learning rate; $D$ - decay schedule; $T$ - total training steps
output: action-value function $Q_i^*$
initialize $s$ and $Q_1, ...Q_n$
for $t \in \{1, ..., T\}$ do
  choose and execute actions $a_1, ..., a_n$ at state $s$
  observe rewards $R_1, ...R_n$ and the next state $s'$
  for $i \in \{1, ..., n\}$ do
    $V(s') \leftarrow f_i(Q_1(s'), ..., Q_n(s'))$
    $Q_i(s, a_1, ..., a_n) \leftarrow (1-\alpha)Q_i(s, a_1, ..., a_n) + \alpha[(1-\gamma)R_i + \gamma V_i(s')]$
  end for
  $s \leftarrow s'$
  $\alpha \leftarrow D(\alpha)$
end for
```

On/off-policy In Table 1 in Greenwald's technical report "Correlated Q-Learning", published in 2005,[2] on/off-policy seem to only affect how actions are chosen, not how state value $V(s')$ is evaluated. This means even for on-policy Q-learning, state value $V(s')$ is still evaluated using greedy policy $\pi_{s'}$ calculated from $Q(s')$. Such notion of on/off-policy is inconsistent with Sutton's definition.[3] Here a distinction is made, since such notions are important for understanding how each variant of the algorithm performs in experiments.

First, define:

- $\pi(s)$ as strategy calculated from $Q(s)$ (either mixed strategy obtained from linear programming or pure strategy using argmax)
- $P\{\epsilon, \pi(s)\}$ as a ϵ-greedy policy where random action is chosen with probability ϵ, otherwise follow $\pi(s)$. In particular $P\{\epsilon \to 0.001, \pi(s)\}$ means ϵ-greedy with ϵ decaying to 0.001, and $P\{\epsilon = 1, \pi(s)\}$ means complete random action

According to Sutton's definition,[3] on/off policy affects how $V(s')$ is evaluated, not how action is chosen, then

- On-policy: evaluate $V(s')$ using $P\{\epsilon, \pi(s')\}$, choose action using $P\{\epsilon, \pi(s')\}$
- Off-policy: evaluate $V(s')$ using $\pi(s')$, choose action using $P\{\epsilon, \pi(s')\}$

According to Greenwald's definition,[2] on/off policy affects how action is chosen, not $V(s')$ evaluation,

- On-policy: evaluate $V(s')$ using $\pi(s')$, choose action using $P\{\epsilon \to 0.001, \pi(s')\}$
- Off-policy: evaluate $V(s')$ using $\pi(s')$, choose action using $P\{\epsilon = 1, \pi(s')\}$

In this report, we follow Sutton's definition, Greenwald's both on- and off-policy are actually off-policy, just with different exploration strategy, $\epsilon \to 0.001$ *vs.* $\epsilon = 1$.

Strategy selection function The strategy selection function f from the algorithm shown above is the determining factor for each variant of multiagent Q-Learning. In regular Q-Learning, the agents are unaware of the existence of other players, but can observe the state of the game in full, so every agent just learn its own Q function, thus an agent maximizes values of its own actions $\max Q(s, a)$. In friend-Q,[4] a particular agent A maximizes its own payoff at each state, and assumes the other players will coordinate their choices to let agent A receive the maximum payoff, thus an agent evaluate state value using $\max Q(s, \vec{a})$. Foe-Q[4] incorporate mixed strategies for individual actions at Nash equilibrium using maximin algorithm. Correlated-Q (CE-Q)[1] employs mixed strategies for joint actions at correlated equilibrium.

Iteration Greenwald's 2003 paper[1] is quite ambiguous on how a "iteration" is defined in their plots Q value differences during training. After carefully reading the paper, and experimenting with different assumptions, it is determined that one iteration means one step in the game, however the $Q(state = s, action = S)$ is not plotted at all steps, just the steps where it is updated. Later it is further discovered even that would make the figures too dense, only plotting the values when it is updated, and at a interval would generate figures as sparse as in the paper.

Identical plots Greenwald's 2003 paper[1] presented identical plots for foe-Q and ce-Q, and explained both algorithm seeks Nash equilibrium for a zero-sum game. This is true, however, in both algorithms, actions are chosen randomly with ϵ probability (either $\epsilon = 1$ or $\epsilon : 1 \rightarrow 0.001$, see explanation above), thus it is impossible to generate identical plots for two algorithms that both involve significant amount of stochasticity, unless a fixed random number seed is defined. In my experiments, I did not set random seed, since that would not be a good experiment design for random processes, and my plots for foe-Q and CE-Q are not identical but similar as a result.

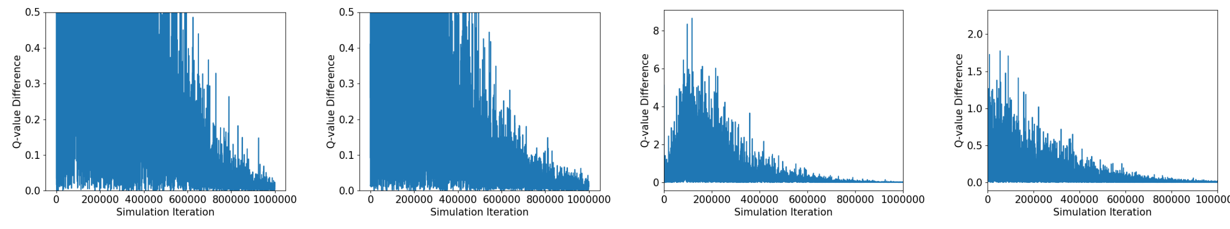
3 Experiments and Results

Experiments presented in Fig 3 in Greenwald paper[1] were replicated. Q, friend-Q, foe-Q, and CE-Q were tested on soccer environment using different parameters, and their results and performances were compared. The Q values at a particular state action pair for player A, $Q(state = s, action = S)$ is recorded during training iterations.

3.1 Q-Learning

First, regular Q-Learning is tested with off-policy ϵ-greedy with $\epsilon : 1 \rightarrow 0.001$ (off-policy by Sutton's definition,[3] or on-policy by Greenwald's[2]). The other training parameter includes: learning rate $\alpha \rightarrow 0.001$, discount $\gamma = 0.9$. In Q-Learning, the two agents are not aware of the actions taken by their opponents, only themselves, however they can observe the state of the game in full, i.e. player A can observe position of B, as well as possession of the ball, and vice versa. Here, since a player is only aware of its own action, it simply choose the action with the largest Q value at a particular state to evaluate the utility of that state.

As shown in **Fig** 2a and 2b, Q-Learning does not converge after 1×10^6 iterations. The reason for decreasing Q-value difference is the same as what is stated in the paper: decreasing learning rate α. **Fig** 2c and 2d are a zoomed out view to show the entire range of Q-value differences, larger initial value of α produced higher Q value differences in early stages of the learning, but does not help with convergence. Note that with α initialized at both 0.9 and 0.15, the learning curve, with y-axis capped at 0.5, look very similar to the one presented in the paper.[1] And the only difference of using different α seems to be the scale of Q-value difference, both converges within 1×10^6 steps and to the same policy, this behavior is also observed in cases of foe-Q and CE-Q, however in friend-Q, on optimal α is required to converge as fast as possible.



(a) $\alpha : 0.9 \rightarrow 0.001$ (b) $\alpha : 0.15 \rightarrow 0.001$ (c) $\alpha : 0.9 \rightarrow 0.001$ (d) $\alpha : 0.15 \rightarrow 0.001$

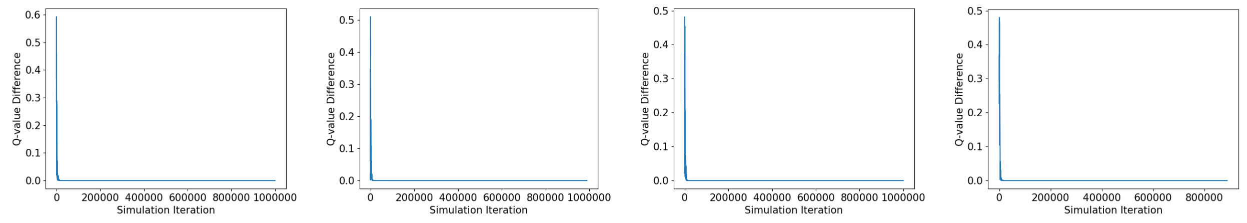
Figure 2: Q-value differences during training, at different initial values of α, for Q-Learning, zoomed-in (a and b), and zoomed-out (c and d). Here (a) corresponds to Fig 3(d) in Greenwald2003 paper.[1]

3.2 Friend-Q

In friend-Q,[1,2,4] a player, A, is aware of actions of all players, as well as the full state of game, however in evaluating the utility of a particular state, it chooses the joint action with the highest Q value, and assumes its opponent will cooperate and let it choose that joint action. The opponent behaves the same, assuming A will cooperate with it.

Here, friend-Q is tested with $\alpha : 0.2 \rightarrow 0.001$, **Fig** 3a and 3b show the result from friend-Q learning two Q tables for both players, using off-policy learning with $\epsilon = 1$ and $\epsilon : 1 \rightarrow 0.001$ respectively (or off- and on-policy respectively defined by Greenwald,[2] detailed explanation see 2). Similar to the figure presented in the paper,[1] friend-Q converges very fast, with $\pi_A(s)$ converging to a particular action randomly, and $\pi_B(s) = E$.

Results described above were obtained by learning Q tables for A and B separately. Learning only one Q table is also feasible, by assuming the payoff matrix at any particular state is always opposite for the two players. The result obtained from learning a single Q table is shown in **Fig** 3c and 3d, also comparable to the results shown in the paper.[1] Same as the paper, $\pi_A(s)$ converges to a particular action randomly, and $\pi_B(s) = E$.



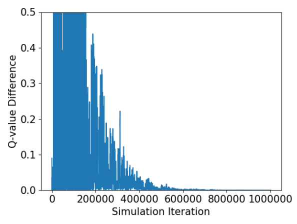
(a) 2 Q tables ($\epsilon = 1$) (b) 2 Q tables ($\epsilon : 1 \rightarrow 0.001$) (c) 1 Q table ($\epsilon = 1$) (d) 1 Q table ($\epsilon : 1 \rightarrow 0.001$)

Figure 3: Q-value differences from friend-Q, $\alpha : 0.22 \rightarrow 0.001$. Here (a) corresponds to Fig 3(c) in Greenwald2003 paper.[1]

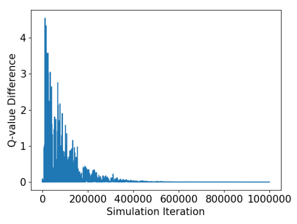
3.3 Foe-Q

In foe-Q, each agent tries to maximize its own reward and minimize its opponents, and the utility of a particular state $V(s)$ is evaluated to be opposite for the two players. For a zero-sum game like the soccer game discussed in this report, learning two Q tables or a single Q table should both work.

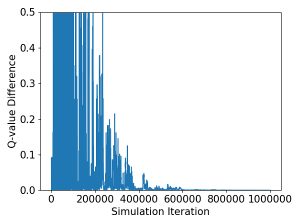
Fig 4a and 4b shows the Q-value differences during training for foe-Q, learning two Q tables, with off-policy ϵ-greedy and $\epsilon = 1$ through out learning period (same as on-policy defined by Greenwald, detailed clarification see 2).[2] Different α only changes the scales of Q-value differences, not the result of convergences (data not shown). Learning a single Q table and calculate strategies at maximin equilibrium for both players from the same Q table produces almost identical Q-value difference plot with $\epsilon = 1$ (**Fig** 4c). With $\epsilon : 1 \rightarrow 0.001$, the convergence appears to take slightly more iterations (**Fig** 4d), but the result after convergence is unchanged. Foe-Q with all different variations in parameters converged to the same policy for state s, as described in the paper,[1] $\pi_A(s) \approx [0.5, 0, 0.5, 0, 0)]$ and $\pi_B(s) \approx [0.5, 0, 0.5, 0, 0)]$. Note that at state s, actions N and stick are equivalent for both A and B, sometimes the learning algorithm converges to N, or partially to N, instead of stick, such results should be considered valid.



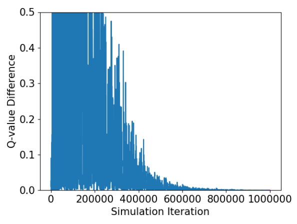
(a) $\epsilon = 1$, zoomed-in



(b) $\epsilon = 1$, zoomed-out



(c) $\epsilon = 1$

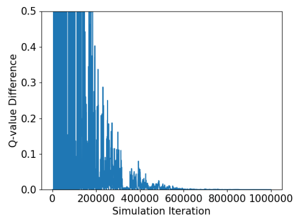


(d) $\epsilon : 1 \to 0.001$

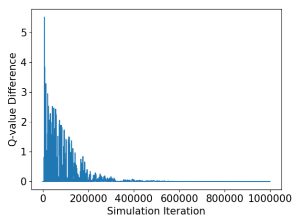
Figure 4: Q-value differences from foe-Q, by learning 2 Q tables (a and b) and a single Q table (c and d), $\alpha : 0.9 \to 0.001$. Here (a) corresponds to Fig 3(b) in Greenwald2003 paper.[1]

3.4 CE-Q

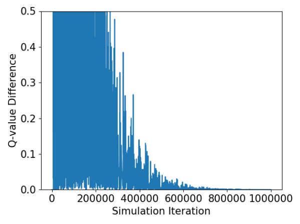
CE-Q evaluates the utility at a particulate state $V(s)$ using the mixed strategy of joint actions obtained at a correlated equilibrium. Here the CE-Q learner learns two separate Q tables for A and B, and using Q values from both tables to generate a strategy of joint actions using linear programming. Again, off-policy with both $\epsilon = 1$ and $\epsilon : 1 \to 0.001$ (or off- and on-policy as Greenwald defined, detailed clarification discussed in 2)[2] are tested. As shown in **Fig** 5a and 5b, the algorithm converges in the similar manner to foe-Q. By comparing **Fig** 5a and 5c, the algorithm seems to be able to converge faster by always choosing action randomly ($\epsilon = 1$), as opposed to ϵ-greedy with $\epsilon : 1 \to 0.001$. Different α values are also tested, they do not seem make any difference other than changing the scale of Q-value differences, similar to the observations in foe-Q and Q-Learning. CE-Q with all different variations in parameters converged to the same policy for state s, as described in the paper,[1] $\pi_A(s) \approx [0.5, 0, 0.5, 0, 0)]$ and $\pi_B(s) \approx [0.5, 0, 0.5, 0, 0)]$. Similar to what was discussed in foe-Q (3.3), at state s, actions N and stick are equivalent for both A and B, sometimes the learning algorithm converges to N, or partially to N, instead of stick, such results should be considered valid.



(a) $\epsilon = 1$, zoomed-in



(b) $\epsilon = 1$, zoomed-out



(c) $\epsilon : 1 \to 0.001$, zoomed-in

Figure 5: Q-value differences with CE-Q, $\alpha : 0.9 \to 0.001$. Here (a) corresponds to Fig 3(a) in Greenwald2003 paper.[1]

References

[1] Amy Greenwald and Keith Hall. Correlated-Q Learning. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pages 242–249, Washington, DC, USA, 2003. AAAI Press.

[2] Amy Greenwald, Keith Hall, and Martin Zinkevich. Correlated Q-Learning. Technical Report CS-05-08, Brown University, 2005.

[3] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[4] Michael L. Littman. Friend-or-Foe Q-learning in General-Sum Games. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 322–328, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.