**Project 3 Report (Video link)**
**Yogesh Luthra (yluthra3)**

**Summary**: Following algorithms were implemented and tested on a Soccer game environment (a multi-agent interaction environment) in order to replicate corresponding results in [1].

. utilitarian Correlated-Q learning (uCE-Q) using Linear Programming (LP)
. Foe-Q learning using Linear Programming (LP)
. Friend-Q learning
. Q-learning (each agent behaves independently from others trying to maximize his reward)

Obtained results, in terms of graphs and policies, match closely with published results in [1]

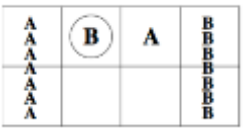

Fig 1

**Description Environment and its Implementation with assumptions**

As described in [1] and [2], a simplified version of Soccer game was implemented. This is zero-sum game with no deterministic equilibrium policies. However as mentioned in said papers and as will be shown in results below, there exist equilibrium with probabilistic action policies.

As per paper's description, Soccer field is a grid with 2 playable rows and 2 columns. The game has 5 possible actions N, S, E, W and stand in every state. Fig 1 shows a particular state of environment.

**Following rules of game were implemented** in code (Python). Mentioned along with them are the assumptions made, as it was found that not all rules of the game were sufficiently described.

1. If player with ball enters his goal, he gets +100 points and the other gets -100 (zero-sum game)
2. If player with ball enters opponents goal, he gets -100 and opponent gets +100.
3. (Assumption) If a player tried to go beyond field boundaries (or in goal positions without ball), his action would be considered 'stand'. E.g. if player A goes N from state s, he is considered 'stand'ing.
4. (Assumption) Both the players could move at the same time, with collision scenarios avoided as described below. [1] seemed to suggest only one player moved at a time. But since for CE-Q implementation, a joint action is sampled from the joint action probability distribution, it wasn't clear which player's action should be suppressed and what implication it could have.

**Collision scenarios** occurs when both players act to end up in same position. Corrective actions:

a. (Assumption) Both moved (N,S,W,E): a player is randomly selected to be assigned to new position.
b. Player with ball moves into a 'stand'ing player: ball changes possession.
c. (Assumption) Both players tried to swap positions: they would remain 'stand'ing.

**Implementation of Experiment**

For each of the implemented algorithms, following structure was followed:

- Top level algorithmic routine
  1. instantiates the Soccer Environment class having specifications on number of players, actions, states and transitions.
  2. instantiates a MultiQ agent (N players) class which decides how to explore and exploit in environment.
  3. defines constraints and optimization objective functions for equilibrium policies (uCE-Q, Foe-Q, Friend-Q or independent Q-learning). These are as per [1], [2]

MultiQ agent class and Constraint/Objective Optimization functions are implemented to generalize to N-player games. That is, not restricted to 2 players. Only Soccer environment is 2-player restricted.

MultiQ agent is implemented as described in section 3 of [1].

**Further specifics of the implementation are:**

1. uCE-Q (to obtain Correlated Equilibria solution concept)
   Number of constraints for LP $= \sum_i Nactions_i*(Nactions_i-1) + \prod_i Nactions_i + 1$ (where i: a player)
   $= 2*5*4$ (inequality) $+ 5*5$ (inequality) $+ 1$ (equality) $= 66$
2. Foe-Q (to obtain Adversarial Equilibria solution concept)
   Number of constraints for LP $= 2*\sum_i Nactions_i + 2 = 2*5 + 2 = 12$
3. Friend-Q (to obtained Coordinated Equilibria solution concept): Both players assume the other is a "friend".
4. Q-learning: Both players play individually to maximize their future discounted rewards, using vanilla Q-learning.

Every time environment is reset, agents are brought to state s as in Fig 1. Agent's actions are never forced and they decide to pick actions for next iteration according to e-greedy policy with eplison 0.2 and decaying alpha (initial=1 with decay=$0.001^{1/1000000}$). There are some assumptions here, as it is not explicitly mentioned how exploration is performed in [1] and if environment is reset to state s at beginning of an episode.

To construct Q-value difference graphs, Q-value of player A for state s with joint action as 'A' taking 'S' and 'B' 'stand'ing is monitored in every iteration. Player A's Q-value difference is calculated as = $| Q^t_a(s, a) - Q^{t-1}_a(s, a) |$

**Project 3 Report (Video link)**
**Yogesh Luthra (yluthra3)**

# Results

## Graphs

**uCE-Q Results**

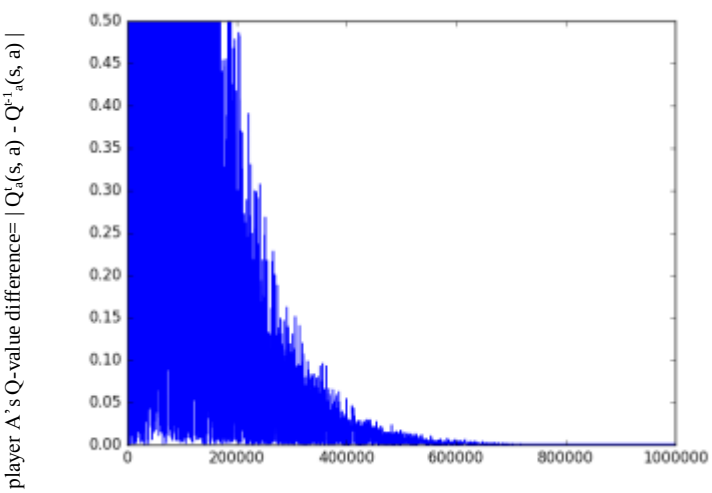


Fig 2 Simulation Iteration (t) →

**Foe-Q Results**

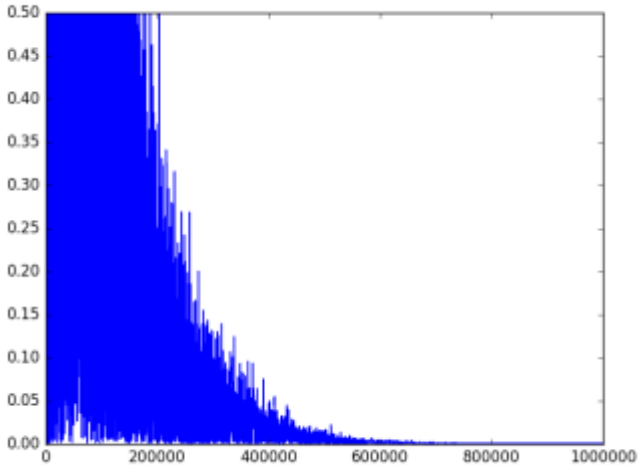


Fig 3 Simulation Iteration (t) →

**Friend-Q Results**

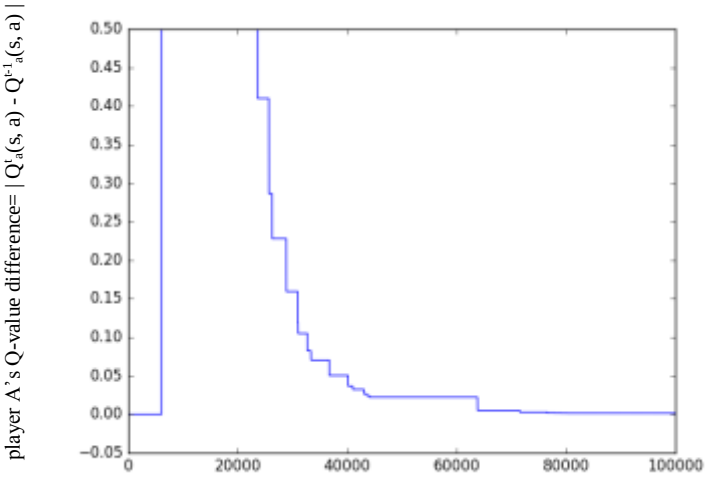


Fig 4 Simulation Iteration (t) →

**Q-learning Results**

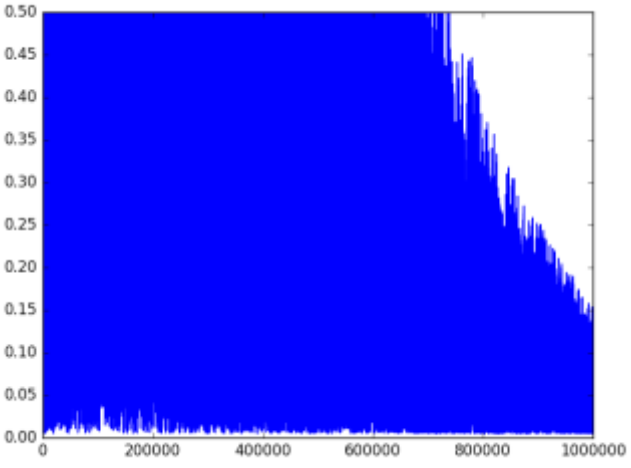


Fig 5 Simulation Iteration (t) →

Please note scale of Fig 4 = 1-to-100k iterations and not 1 million. Since Friend-Q converged much faster than other algorithms, the x-axis was zoomed-in to show convergence behavior well. All figures 2-5 show close match with published results in [1]. However graphs here look busier than in [1] for Fig 2,3,5 possibly because of differences in how environment was reset and whether actions were forced in [1] on agents at start of each episode if starting in state s. It is not clear what was done in [1]. In my case, although environment was reset to state s, but joint action A→'S' B→'stand' was never forced. So the monitored (s, jointAction) tuple was reached few times per 100 steps. This caused many zeros in Q-value difference and thus busier graphs.

**Project 3 Report (Video link)**
**Yogesh Luthra (yluthra3)**

**Obtained Policies at end of 1 million iterations:**
Following corresponding equilibrium policies were obtained:

1. uCE-Q: Joint Action strategy in state s.

| Player | | B | | | | |
|---|---|---|---|---|---|---|
| | | 'S' | 'E' | 'stand' | 'W' | 'N' |
| A | 'S' | [[**0.251** | 0.000 | **0.232** | -0.000 | 0.000] |
| | 'E' | [ 0.000 | -0.000 | 0.000 | -0.000 | 0.000] |
| | 'stand' | [ **0.268** | 0.000 | **0.249** | -0.000 | 0.000] |
| | 'W' | [-0.000 | -0.000 | -0.000 | -0.000 | -0.000] |
| | 'N' | [ 0.000 | -0.000 | 0.000 | -0.000 | 0.000]] |

This shows players A and B randomize between 'S' and 'stand' as equilibrium policies, which is same as mentioned in [1]

2. Foe-Q: Joint Action strategy in state s.

| Player | 'S' | 'E' | 'stand' | 'W' | 'N' |
|---|---|---|---|---|---|
| A | [[ **0.498** | 0.000 | **0.502** | -0.000 | 0.000] |
| B | [ **0.501** | -0.000 | 0.000 | -0.000 | **0.499**]] |

This shows player A randomizes between 'S' and 'stand', and player B randomizes between 'S' and 'N'. But since in state s, taking 'N' is same as 'stand', player B also randomizes between 'S' and 'stand'. So converged policies between players and among uCE-A and Foe-Q exactly match as in [1].

3. Friend-Q: strategy in state s

| | 'S' | 'E' | 'stand' | 'W' | 'N' |
|---|---|---|---|---|---|
| A [[ | 1. | 0. | 0. | 0. | 0.] |
| B [ | 0. | 1. | 0. | 0. | 0.]] |

As mentioned in the paper, player B simply decides to hand over ball to player A, considering him a friend and assuming A will score for him, which is fallacious for zero-sum game. But Q-diff convergence is very fast, also as in [1]

4. Q-learning never converged as seen in Fig 5

GitHub Repo:

https://github.gatech.edu/yluthra3/GeneralSumGames

**References:**

[1] https://www.cs.duke.edu/courses/spring07/cps296.3/correlated_q.pdf

[2] M. 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.