EEE 448/548 - Project Report

Friend-or-Foe Q-learning in General-Sum Games

Baran Aşılıoğlu, Mustafa Mert Çetin

1 Strategic-Form Games

A strategic form game is defined by a set of n players, their action-choice sets $A_1, A_2, ..., A_n$, and their payoff functions $R_1, R_2, ..., R_n$. Each R_i maps the total action-choice tuple of the players to a scalar valued reward $\forall i \in 1 \leq i \leq n$. An n-player general-sum strategic-form game is a strategic-form game such that the sum of payoff for each player can be non-zero,

$$\sum_{i=1}^{n} R_i(A_1, A_2, ..., A_n) = c, \tag{1}$$

where c is a real number.

2 Nash Equilibria in Strategic-Form Games

Nash (1951) showed, every n-player general-sum strategic form game has a Nash Equilibrium. This is a set of action-choices for each player such that no player can improve its expected payoff by unilaterally changing its strategy:

$$R_i(\pi_1, ..., \pi_{i-1}, \pi_i, \pi_{i+1}, ..., \pi_n) \geq R_i(\pi_1, ..., \pi_{i-1}, \pi_i', \pi_{i+1}, ..., \pi_n), \tag{2}$$

for all action-choice selections $\pi_i^{'}$ and for all players $i \in 1 \leq i \leq n$. A game can have more than one Nash Equilibrium and the expected payoff of a player i can vary depending on the equilibrium considered.

The two kinds of Nash Equilibria that are crucial for this paper are Coordination Equilibrium and Adversarial Equilibrium. Coordination Equilibrium is defined as all players receiving their highest possible value,

$$R_i(\pi_1, ..., \pi_n) = \max_{a_1 \in A_1, ..., a_n \in A_n} R_i(a_1, ..., a_n) \tag{3}$$

Adversarial Equilibrium is defined as a Nash Equilibrium such that no player i can be hurt by any other unilateral change by any other player,

$$R_i(a_1, ..., a_{i-1}, a_i, a_{i+1}, ..., a_n) \leq R_i(a_1', ..., a_{i-1}', a_i, a_{i+1}', ..., a_n'), \tag{4}$$

for all combinations of action-choice selections $a_1, ..., a_n$.
A Strategic-Form Game can have both Coordination Equilibrium and Adversarial Equilibrium. However in this case, either the Coordination Equilibrium has a higher value than the Adversarial Equilibrium or all players have constant valued payoff functions, that is, $R_1 = c_1, R_2 = c_2, ..., R_n = c_n$. However, these special Nash Equilibria might not exist at all.

Proposition 1. *If a Strategic-Form Game has a coordination equilibrium, all of its coordination equilibria have the same value.*

Proof 1. *The proof is fairly direct since, in each equilibrium, players get their maximum payoff values which are unique.* ■

Proposition 2. *If a Strategic-Form Game has an adversarial equilibrium, all of its adversarial equilibria have the same value.*

Proof 2. *Let $a_1, ..., a_n$ and $b_1, ..., b_n$ be two adversarial equilibria for a strategic-from game with payoffs $R_1, ..., R_n$. If we compare the payoffs of player i under the a equilibrium to that of b equilibrium,*

$$\begin{aligned} R_i(a_1, ..., a_{i-1}, a_i, a_{i+1}, ..., a_n) &\geq R_i(a_1, ..., a_{i-1}, a'_i, a_{i+1}, ..., a_n) \\ &= R_i(b'_1, ..., b'_{i-1}, b_i, b'_{i+1}, ..., b'_n) \geq R_i(b_1, ..., b_{i-1}, b_i, b_{i+1}, ..., b_n) \quad (5) \end{aligned}$$

These statements are setting the adversarial equilibrium values to arbitrary values, and then to values of the other adversarial equilibrium. The inequalities follow from equation (2) and (4) respectively. The argument can also be repeated in reverse:

$$\begin{aligned} R_i(a_1, ..., a_{i-1}, a_i, a_{i+1}, ..., a_n) &\leq R_i(a_1, ..., a_{i-1}, a'_i, a_{i+1}, ..., a_n) \\ &= R_i(b'_1, ..., b'_{i-1}, b_i, b'_{i+1}, ..., b'_n) \leq R_i(b_1, ..., b_{i-1}, b_i, b_{i+1}, ..., b_n) \quad (6) \end{aligned}$$

Hence the equlibria are the same. ■

The uniqueness of the coordination and the adversarial equilibrium will play a key role in the paper.

3 Markov Games

A One-Stage General-Sum n-player Game is defined by a strategic form game with a finite set of states S. Each state $s \in S$ has its own payoff function. A one-stage policy for a player i, π_i, is a probability distribution over its action-choice set A_i.

A Markov Games is a generalization of the one-stage game to multiple stages. It has a transition function that takes a state and the action-choices for all players and returns a probability distribution over next states. In this setting, a *policy* maps states to probability distributions over actions. The *value* for a player in a game, given discount factor $0 \leq \gamma \leq 1$, is the discounted sum of payoffs. In particular, let $\pi_1, ..., \pi_n$ be a set of policies for each player. The Q-function for player i is defined to be,

$$Q_i(a_1, ..., a_n) = R_i(a_1, ..., a_n) + \gamma \sum_{s' \in S} T(s, a_1, ..., a_n, s') \cdot Q_i(s', \pi_1, ..., \pi_n) \quad (7)$$

where $Q_i(s', \pi_1, ..., \pi_n)$ is the expected value of the $Q_i(s', a_1, ..., a_n)$ according to the policies $\pi_1, ..., \pi_n$.
Filar and Vrieze (1997) showed that when Q-functions are treated as the payoff functions for individual one-stage games, the policies at the individual states are in equilibrium if and only if the overall multistage policies are in equilibrium. Therefore, an equilibrium for a Markov Game can be found by finding a Q-function with the appropriate properties.

4 Nash-Q

Nash-Q is a method for learning a Q-function for a player from experience interacting with other players in the game. Experience is in the form of a trajectory $(s, a_1, ..., a_n, s'\prime, r_1, ..., r_n)$, where the game starts with state s, players choose the actions $a_1, ..., a_n$, a transition occurs to state s', and the payoffs $r_1, ..., r_n$ are received by the respective players.

Nash-Q learning works by maintaining a set of Q-functions and updating them by

$$Q_i(s, a_1, ..., a_n) := (1 - \alpha_t) \cdot Q_i(s, a_1, ..., a_n) + \alpha_t(r_i + \gamma \cdot Nash_i(s, Q_1, ..., Q_n)) \tag{8}$$

each time a new experience occurs. $Nash_i(s, Q_1, ..., Q_n)$ equals $Q_i(s, \pi_1, ..., \pi_n)$, where $\pi_1, ..., \pi_n$ are a Nash Equilibrium for the one-stage game defined by the Q-functions $Q_1, ..., Q_n$.
The values α_t are a sequence of learning rates and they are assumed to satisfy the standard stochastic approximation conditions for convergence,

$$\sum_{t=1}^{\infty} \alpha_t = \infty \tag{9}$$

$$\sum_{t=1}^{\infty} \alpha_t^2 < \infty \tag{10}$$

that is being square summable but not summable (Jaakkola et al. 1994).

It is a significant problem that the Nash function does not necessarily returns unique values, since a one-stage game can have multiple different Nash Equilibria. As a result, the Nash-Q learning method given in (8) cannot converge if the value returned by the Nash function uses different Nash Equilibria during updates.

One approach for a solution is to restrict the games such that the algorithm never encounters a one-stage game with multiple equilibria. However, this would greatly reduce the utility of the proposed algorithm.

Another approach can be to ensure that the learning algorithm knows precisely which type of equilibrium to use in value updates. This results in weaker restrictions and a more effective learning algorithm in general. Hence, the key idea presented in this paper will be to use Coordination and Adversarial Equilibria for Nash-Q updates, as they are unique as proven in Proposition 1 and 2, and let the algorithm know beforehand which type of Nash Equilibrium to consider.

5 Friend-or-Foe Q-learning

Friend or Foe Q-learning is motivated by the idea that the Nash-Q learning algorithm can converge when the type of Nash Equilibrium to use is provided to the algorithm beforehand. To this purpose, FFQ algorithm is proposed and explained below:

In FFQ, the learner only maintains a Q-function for itself. Let $X_1, ..., X_k$ be the actions available to the k friends of player i, $Y_1, ..., Y_k$ be the actions available to its l foes. Then the the update performed by the Nash-Q algorithm in equation (8) is:

$$Nash_i(s, Q_1, ..., Q_n) =$$
$$\max_{\pi \in \prod(X_1 \times ... \times X_k)} \min_{y_1,...,y_l \in Y_1 \times ... \times Y_n} \sum_{x_1,...,x_k \in X_1 \times ... \times X_k} \pi(x_1)...\pi(x_k) \cdot Q_i(s, x_1, ..., x_k, y_1, ..., y_l) \quad (11)$$

which is just ordinary minimax-Q algorithm and can be implemented as a straightforward linear program. For simplicity, the two-player version of the FFQ is also explained.
When the opponent is considered a friend, the update performed is:

$$Nash_i(s, Q_1, Q_2) = \max_{a_1 \in A_1, a_2 \in A_2} Q_i(s, a_1, a_2) \quad (12)$$

This is just ordinary Q-learning in the combined action space of two players. When the opponent is considered a foe, calculation is:

$$Nash_i(s, Q_1, Q_2) = \max_{\pi \in \prod(X_1)} \min_{y_1 \in Y_1} \sum_{x_1 \in X_1} \pi(x_1) \cdot Q_i(s, x_1, y_1) \quad (13)$$

The idea is simply that player-i's friends are assumed to work together to maximize player-i's value, while player-i's foes are working together to minimize its value. Thus, n-player FFQ treats any game as a two-player zero-sum game with an extended action set.

6 Convergence of FFQ Algorithm

Let S be the set of states of a Markov Game, and the B(S) of bounded, real-valued functions over S be the set of value functions. Let $T : B(S) \to B(S)$ be an arbitrary contraction mapping with fixed point V^*. Having a direct access to the contraction mapping T would allow a successive approximation of V^*. However, T is not available and approximations of T must be constructed from experience interacting with the model. Consider a sequence of random operators $T_t : B(S) \to (B(S) \to B(S))$ and define $U_{t+1} = [T_t U_t]V$ where V and $U_0 \in B(S)$ are arbitrary value functions. We say T_t approximates T at V, if U_t converges to TV with probability 1 uniformly over X[1]. The idea is that T_t is a randomized version of T that uses U_t as a sort of memory to converge to TV. The following theorem states that this sequence of T_t can be used to estimate the fixed point V^* of T.

Theorem 1. *Let T be an arbitrary mapping with fixed point V^*, and let T_t approximate T at V^*. Let V_0 be an arbitrary value function, and define $V_{t+1} = [T_t V_t]V_t$. If there exists functions $0 \leq F_t(s) \leq 1$ and $0 \leq G_t(s) \leq 1$ satisfying the conditions below with probability one, then V_t converges to V^* with probability 1 uniformly over S:*

Condition 1. *For all U_1 and $U_2 \in B(S)$, and all $s \in S$,*

$$|[T_t U_1]V^*(s) - [T_t U_2]V^*(s)| \leq G_t(s)|U_1(s) - U_2(s)| \quad (14)$$

Condition 2. *For all U and $V \in B(S)$, and all $s \in S$,*

$$|[T_t U]V^*(s) - [T_t U]V(s)| \leq F_t(s) \sup_{x'} |V^*(x') - V(x')| \quad (15)$$

Condition 3. *For all $k > 0$, $\prod_{t=k}^{n} G_t(s)$ converges to zero uniformly in s as n increase,*

Condition 4. *There exists $0 \leq \gamma < 1$ such that for all $s \in S$ and large enough t,*

$$F_t(s) \leq \gamma(1 - G_t(s)) \quad (16)$$

6.1 Convergence of Friend-Q

The fixed-point V^* is as follows:

$$V^*(s) = max_a(R(s,a) + \gamma \sum_y P(s,a,y) \cdot V^*(s)) \tag{17}$$

The learning-model is as follows:

$$Q_{t+1}(s_t,a_t) := (1-\alpha_t(s_t,a_t)) \cdot Q_t(s_t,a_t) + \alpha_t(s_t,a_t) \cdot (r_t + \gamma \max_a Q_t(s_t,a_t)) \tag{18}$$

The dynamic programming-operator defining the optimal Q-function is as follows:

$$[TQ](s,a) = R(s,a) + \gamma \sum_y P(s,a,y) \cdot max_{a'} Q(y,a') \tag{19}$$

The randomized approximate dynamic-programming operator that gives rise to the Q-learning rule is as follows:

$$[T_tQ'](s,a) = \begin{cases} (1-\alpha_t(s,a)) \cdot Q'_t(s,a) + \alpha_t(s,a) \cdot (r_t + \gamma \max_a Q_t(s,a)), & \text{if s = } s_t \text{ and a = } a_t \\ Q'(s,a), & \text{otherwise} \end{cases} \tag{20}$$

Littman and Szepesvari (1996) shows that the Friend-Q algorithm formalized as above satisfy the conditions 1 and 2. Furthermore, $G_t(s,a)$ and $F_t(s,a)$ is defined as follows:

$$\begin{aligned} G_t(s,a) &= \begin{cases} (1-\alpha_t(s,a)), & \text{if s = } s_t \text{ and a = } a_t \\ 1, & \text{otherwise} \end{cases} \\ F_t(s,a) &= \begin{cases} \gamma\alpha_t(s,a), & \text{if s = } s_t \text{ and a = } a_t \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{21}$$

Littman and Szepesvari (1996) also shows that these definitions of $G_t(s,a)$ and $F_t(s,a)$ satisfy conditions 3 and 4, provided that the learning rate is decayed in accordance with stochastic approximation criterion. Hence, it is concluded by Theorem 1 that the Friend-Q algorithm converges to its fixed-point V^*, over $S \times A$, with probability 1. $S \times A$ is the combined space of all the states and actions.

6.2 Convergence of Foe-Q

Convergence of Foe-Q follows fairly directly after the convergence of the Friend-Q. The fixed point for Foe-Q is as follows:

$$V^*(s) = max_{\pi \in \prod(A)} \min_{b \in B} \sum_{a \in A} \pi(a) \cdot (R(s,a,b) + \gamma \sum_{y \in S} P(s,a,b,y) \cdot V^*(y)) \tag{22}$$

The Q-learning rule for Foe-Q is as follows:

$$\begin{aligned} Q_{t+1}(s_t,a_t,b_t) := (1-\alpha_t(s_t,a_t,b_t)) \cdot Q_t(s_t,a_t,b_t) \\ + \alpha_t(s_t,a_t,b_t) \cdot (r_t + \gamma \cdot max_{\pi \in \prod(A)} \min_{b \in B}(\pi(a) \cdot Q_t(s_t,a_t,b_t))) \end{aligned} \tag{23}$$

After similar definitions and argument as the previous section, it can be concluded that, by Theorem 1, Foe-Q algorithm converges to its fixed-point V^*, over $S \times A \times B$, with probability 1.

Although FFQ converges, it does not always converge to a Nash equilibrium policy. As a preliminary, define Friend-Q as FFQ assuming all opponents are friends and define Foe-Q as FFQ assuming all opponents are foes. Then, in addtion to Theorem 4, there should be also special cases as depicted in the following theorem.

Theorem 2. *Friend-Q learns values for a coordination equilibrium if there exists one and Foe-Q learns values for an adversarial equilibrium if there exists. This is true regardless of opponent behavior.*

Proof. It is sufficient to show that the value of a coordination equilibrium in a one-stage game is the maximum reward and that the value of adversarial equilibrium in a one-stage game is the minimax value. The first part of the proof is trivial since the it is true by the definition of the coordination equilibrium. The second part of the proof is shown below.
Let R_1, ..., R_n be the rewards in a one-stage game. Let $\pi, ..., \pi$ be one-stage policies that achieve the minimax value for player i and $\rho_i, ..., \rho_n$ be a set of one-stage policies in adversarial equilibrium. Thus,

$$R_i(\pi_1, ..., \pi_n) \geq R_i(\pi'_1, ..., \pi_n) = R_i(p_1, p'_2, ..., p'_n) \geq R_i(p_1, p_2, ..., p_n) \tag{24}$$

The first inequality follows from the fact that pi˙1 is minimax. The equality is obtained by taking $\pi'_1 = \rho_1$ and $\rho'_2, ..., \rho'_n = \pi'_2, ..., \pi'_n$. The last inequality follows from the definition of adversarial equilibrium. Same procedure can be followed for ρs instead of πs. Therefore the value of an adversarial equilibrium for a player can be found by minimax algorithm. ■

7 Examples

A multi-player soccer game is conducted to analyze the behavior of FFQ algorithm. In this game, there are two teams and each player (agent) views the other players as either friend (same team) or foe (opponent team). Each player has the actions N, S, E, W and stay in the same position. The game structure is depicted in Figure 1. All players and the ball are initialized randomly. If a player takes the ball to his goal, he scores +100, and the opponent scores -100. If the takes the ball to the opponents goal, rewards are reversed. In both cases, the game ends.

The game has conducted between a Q-learning agent, a Foe-Q agent, and a Friend-Q agent. Each agent has trained against himself and played against the other ones with the addition of a random agent. Both the convergence and the performance of the agents are analyzed. In general, the performance of Q-learning agent was very similar to Foe-Q agent and they were better than Friend-Q agent. Every agent has performed well against random agent. Friend-Q performed poorly against the other agents because the 'judgment' of Friend-Q about the opponents moves was relatively optimistic, therefore his choice of actions were to his disadvantage in some cases. Whereas Q-learning and Foe-Q agents did not suffer from misjudgment of their opponents. The results are summarized as win-rates in Table 1.

On the convergence aspect of the algorithms, Foe-Q and Friend-Q proved themselves to be convergent. However, Q-learning did not converge. The reason is that the Nash-Q algorithm diverges for an arbitrary Nash function. Although that Q-learning algorithm converges, the Q-learning agent fails in 'guessing' the opponents moves and choosing different Nash equilibrium points for each step. This problem does not rise in the cases of Foe-Q and Friend-Q since they specifically choose either adversarial equilibrium or coordination equilibrium, respectively.

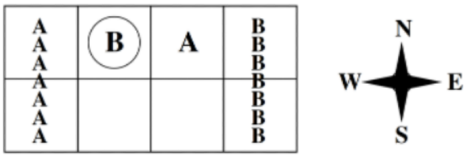

Figure 1: Soccer game setup.

vs	Foe-Q	Friend-Q	Random
Q-learning	0.4973	0.4416	0.8585
Foe-Q		0.5697	0.8329
Friend-Q	0.4056		0.7796

Table 1: Results of soccer game simulation.

8 Conclusion

Friend-or-foe Q-learning (FFQ) provides slightly different approach to reinforcement learning in general-sum games. It requires either a coordination or an adversarial equilibrium to converge. However, it offers some specifications to Nash-Q. As seen in our example, choosing random (Q-learning algorithm) Nash equilibrium points and Nash functions does not necessarily converge, in fact, in general, it would not converge. In addition to this advantage, FFQ also does not require to calculate each players value function. FFQ is also successful at multi-player games by classifying other players as friends or foes. On the other hand, Nash-Q does not require that kind of labeling.

Between Friend-Q and Foe-Q, Foe-Q provides highly trustable results since it is minimax value. The values learned by Foe-Q is independent from opponents actions. Friend-Q does not provide such guarantee and sometimes it tends to provide incompatible policies, therefore, resulting in worse results than Foe-Q. The advantage of Friend-Q is the possibility to achieve the highest reward depending on actions of the opponent whereas Foe-Q acts cautiously to prevent bad situations and it learns from pessimistic assumptions.

Finally, both FFQ and Nash-Q fail to find equilibria when both coordination and adversarial equilibria do not exist. Which is a problem that any method mentioned in this paper could not address.

9 References

Filar, J., & Vrieze, K. (1997). *Competitive Markov Decision Process*. Springer-Verlag.

Jaakola, T., Jordan, M. I., & Singh, S. P. (1994). On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6, 1185-1201.

Littman, M. L., & Szepesvari, C. (1996). A generalized reinforcement-learning model: Convergence and applications. *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 310-318)

Nash, J. (1951). Non-Cooperative Games. *Annals of Mathematics*, 54, 286-295.