

Online Reinforcement Learning in Multiagent Systems

Junling Hu

Simon School of Business
University of Rochester
Rochester, NY 14627
huju@simon.rochester.edu

Yilian Zhang

Department of Mathematics
University of Rochester
Rochester, NY 14627
zhyl@math.rochester.edu

Abstract

We propose a new multiagent Q-learning method, called best-response Q-learning that takes into account other agents' actual strategy. This method generalizes the Nash Q-learning method that is proposed before. We also provide an online implementation for this algorithm. Our experiments in a grid game show that this algorithm consistently performs better than the other multiagent learning methods and the traditional single-agent Q-learning method.

Introduction

Applying reinforcement learning (Kaelbling, Littman, & Moore 1996) to multiagent environment requires fundamental change on the theoretical framework and the learning algorithm itself. Hu and Wellman (Hu & Wellman 1998) showed that the framework for reinforcement learning had to change from a single-agent Markov Decision process to a general-sum stochastic game. It also requires re-defining the learning procedure. They proposed a Nash Q-learning algorithm that allows the agent to learn a Nash equilibrium strategy, and demonstrated good offline learning performance of this algorithm in two grid games (Hu & Wellman 2000).

Even though offline learning is interesting, we believe that a learning algorithm for multiagent systems has to be online. This is because agents constantly interact in a dynamic environment, and normally there is no free trial. This is the fundamental difference between online learning and offline learning, where there is a separate training period from testing phases. Our agent has to act while it learns.

One of the strong assumption made in Nash Q-learning is that another agent is rational, and also follows a Nash equilibrium strategy. This limits the use of Nash Q-learning in many situations, where the other agents may not be fully rational. We are looking for an online learning algorithm that allows our agent to receive its best performance in a multiagent system. Here we assume the system is fully decentralized, and each agent is self-interested.

We propose a new multiagent Q-learning method that does not make assumptions on the other agent's rationality.

Instead, our method directly estimates the other agent's actual strategy, and chooses a best response to that strategy. We call such learning method the Best-response Q-learning. Our learning method incorporates Nash Q-learning as a special case where the other agent happens to follow a Nash equilibrium strategy.

Our Best-response Q-learning method differs from previous research on best-response strategy in the following ways: First, Traditional research on best-response strategy is normally defined for one period, while our strategy covers the entire course of the game. Second, the criterion for our best response is Q-value, which is updated constantly in the game. Other researches on best response in games did not directly use the Q-values.

We test our new Q-learning algorithm in a grid game, adopted from (Hu & Wellman 2000). For performance comparison, we implement three other types of agents: the single-agent Q-learning agent, the Nash Q-learning agent, and the fictitious agent. We hold the type of agent 2 constant, and compare the performance of agent 1 while it takes on these four different types. Our experiments show that the best-response Q-learning agent consistently performs best among these four types. This result holds for every environment where agent 2 assumes a given type.

From our experiments, we also find that when the other agent is a Nash agent, our best response is to be a Nash agent. When the other agent is a single Q-learning agent, being a Nash Q-learning does not help. Instead, our agent has to directly estimate the action of the other agent, and choose a action accordingly.

Our research expands previous research on multiagent reinforcement learning, and provides an applicable method that can be used for online learning in multiagent systems.

Background Knowledge

Stochastic games (Filar & Vrieze 1997) are discrete time dynamic games where actions and states are finite. This model is an extension of Markov Decision Processes to the case of two or more agents. In such a game, agents choose their actions independently and simultaneously at each time period. They then receive their rewards and the state transits to the next state. The state transition follows a stationary probability distribution. Without confusion, we will refer to general-sum stochastic games by the term "stochastic

games” throughout this paper.

Strategy in Stochastic Games

An agent’s *strategy* is a plan for playing a game. A strategy $\pi = (\pi_0, \dots, \pi_t, \dots)$ is defined over the entire course of the game, where π_t is called the *decision rule* at time t . A decision rule is a function $\pi_t : \mathbf{H}_t \rightarrow \sigma(A^i)$, where \mathbf{H}_t is the space of possible histories at time t , with each element $H_t = (s_0, a_0^1, \dots, a_0^n, \dots, s_{t-1}, a_{t-1}^1, \dots, a_{t-1}^n, s_t)$, and $\sigma(A^i)$ is the space of probability distributions over agent i ’s actions. π is called a *stationary strategy* if $\pi_t = \bar{\pi}$ for all t , that is, the decision rule is independent of time. π is called a *non-stationary strategy* if $\pi_t \neq \bar{\pi}$. An interesting type of non-stationary strategy is the one that has its decision rule conditional on the history of the game play.

In this paper, we focus on stationary strategies. Particularly, we are interested in the stationary strategy whose decision rule is $\bar{\pi} = (\bar{\pi}(s^1), \dots, \bar{\pi}(s^m))$, which assigns a probability distribution over available actions for each state s^j , $j = 1, \dots, m$, where m is the number of states. For each state s^j , $\bar{\pi}(s^j) = \{P(a_1), \dots, P(a_M)\}$, where $P(a_k)$ is the probability of taking action a_k , satisfying $\sum_{k=1}^M P(a_k) = 1$, where M is the total number of actions available to this agent.

Nash Q-learning

This learning method was proposed by Hu and Wellman (Hu & Wellman 1998) in 1998. The learning method is defined in the framework of an n -player stochastic game. The goal of this learning is *Nash equilibrium Q-values*, $Q_*(s, a^1, \dots, a^n)$, which is defined as the agent’s total discounted rewards when all agents execute their joint actions (a^1, \dots, a^n) in state s and then follow their equilibrium strategies π_*^1, \dots, π_*^n thereafter. A set of strategies $(\pi_*^1, \dots, \pi_*^n)$ is a Nash equilibrium if for every agent k , the strategy π_*^k is the best response to all the others’ strategies.

The learning agent, say agent k , maintains n Q-tables, one for itself and one for every other agent. Agent k updates the entries in each table Q^i , *foreveryagent* $i = 1, \dots, n$, according to the following rule:

$$Q_{t+1}^i(s, a^1, \dots, a^n) = (1 - \alpha_t)Q_t^i(s, a^1, \dots, a^n) + \alpha_t[r_t^i + \beta \text{Nash}Q_t^i(s')], \quad (1)$$

where $\text{Nash}Q_t^i(s') = \pi^1(s') \dots \pi^n(s')Q_t^i(s')$ and $(\pi^1(s'), \dots, \pi^n(s'))$ is a mixed-strategy Nash equilibrium of the stage game $(Q_t^1(s'), \dots, Q_t^n(s'))$.

The convergence of this algorithm in offline training environment does not depend on the agents’ actions in the game.

Generalized Multiagent Q-learning

Limitation of Nash Q-learning

The Nash-Q learning method makes strong assumption about the other players’ rationality. If the other players do not follow Nash equilibrium strategies, then our player is stuck with wrong strategies.

Best-response Q-learning

A best response Q-learning agent maintains a belief on its Q-function, and uses the following equation to update the Q-values, assume there are only two agents in the system:

$$Q_{t+1}^i(s, a^1, a^2) = (1 - \alpha_t)Q_t^i(s, a^1, a^2) + \alpha_t[r_t^i + \beta \max_{\pi^i(s')} \pi^i(s')Q_t^i(s')\hat{\pi}^j(s')], \quad (2)$$

where $\hat{\pi}^j(s')$ is agent i ’s belief on agent j ’s mixed strategy in state s' . Agent i will chooses its best-response strategy $\pi^i(s')$ for the next state.

In this learning, there is no assumption on the behavior of the other agent. The agent maintains belief on its own Q-table. However, there is an extra level of learning, that is the learning of the other agent’s actual strategy $\pi^j(s')$.

Nash equilibrium is a special case of this learning method. If agent 1 believes that agent 2’s strategy belongs to a Nash equilibrium, then agent 1’s best response is to choose a strategy in that equilibrium. Then the solution will be equivalent to solving the Nash equilibrium directly.

We define the optimal Q-value to be Q_*^i for agent i whose elements satisfy the following equation for every s, a^1 and a^2 :

$$Q_*^i(s, a^1, a^2) = r^i(s, a^1, a^2) + \beta \sum_{s' \in S} p(s'|s, a^1, a^2)v^i(s', \pi_*^i, \pi^j), \quad (3)$$

where $v^i(s', \pi_*^i, \pi^j)$ is agent i ’s total discounted reward over infinite periods starting from state s' given that the other agent follows the strategy π^j during the game, and π_*^i is agent i ’s best response to π^j .

We have the following two propositions related to the convergence of this learning algorithm.

Proposition 1 *In a two-agent stochastic game, if $\hat{\pi}^j$ is the same as π^j , or in other words if agent i has correct belief about agent j ’s actual strategy, then $\{Q_t^i\}$, updated by the process in (2), converges to Q_*^i defined in (3).*

Proof. Let the learning agent be agent 1. We can construct a new state $S = (s, a^2)$. Then the tuple (s, a^1, a^2) becomes (S, a^1) . Since $\hat{\pi}^j$ is the same as π^j , we get the following equation:

$$\begin{aligned} Q^1(s')\hat{\pi}^2(s') &= Q^1(s')\pi^2(s') \\ &= \left[\sum_{a^2} Q^1(s', a^1, a^2)\pi^2(s, a^2) \right]_{a^1} \\ &= Q^1(S') \end{aligned}$$

where $Q^1(S')$ is an $M \times 1$ matrix with M to be the number of actions for agent 1.

It is easy to show that

$$\max_{\sigma^1} \sigma^1 Q^1(S) = \max_{a^1} Q^1(S, a^1)$$

where σ^1 is the probability distribution over agent 1’s actions. Therefore equation (2) can be rewritten as

$$Q_{t+1}^1(S, a^1) = (1 - \alpha_t)Q_t^1(S, a^1) + \alpha_t[r_t^1 + \beta \max_{a^1} Q^1(S', a^1)], \quad (4)$$

which is identical to single-agent Q-learning. Therefore this process would converge. \square

Most of the time, an agent will never know the actual strategy of the other player. However, our agent can keep learning, and eventually learn the right strategy of the other if the learning succeeds. Therefore we are more interested in the following proposition:

Proposition 2 *In a two-player stochastic game, if $\hat{\pi}^j(t)$ converges to π^j , or in other words, $\|\hat{\pi}^j(t) - \pi^j\| \rightarrow 0$ as $t \rightarrow \infty$, then $\{Q_t^i\}$, updated by the process in (2), converges to Q_*^i defined in (3).*

The essence of this proposition is similar to that of Proposition 1, and it should not be difficult to have a formal proof. But we will leave it out here.

Implementation

The Best-response Q-learning algorithm

Let our learning agent be indexed by 1. The algorithm follows the following steps;

1. Initialize $Q_t^1(s, a^1, a^2) = 0$ for all s, a^1, a^2 .
Initialize belief on the other agent's strategy $\hat{\pi}^2$.
Choose an action
2. Observe rewards, new state, and actions taken by all agents. Update belief on $\hat{\pi}^2$.
3. Update Q-values according to the following process:

$$Q_{t+1}^1(s, a^1, a^2) = (1 - \alpha_t)Q_t^1(s, a^1, a^2) + \alpha_t[r_t + \beta \max_{\pi^1(s')} \pi^1(s')Q_t^1(s', \hat{\pi}^2(s'))], \quad (5)$$
4. choose action a_t^1
5. Return to Step 2.

An important feature of this learning algorithm is in step 2, where the agent learns about the other agent's strategy, in addition to learning its own Q-tables. By separating these two phases, we can allow the agent to recognize the changing strategies played by the other agent. It is also easy to see that Nash Q-learning is a special case here, where the Nash agent updates its belief on agent 2's Q-tables in step 2, and derives a Nash equilibrium strategy.

Online Learning Strategy

The implementation of Best-response Q-learning to actual games relies on the action choice. Here we propose a method to handle action choices.

We divide the game into two stages. Stage 1: Explore each action at least once. Stage 2: Start Exploitation vs. Exploration phase.

At the beginning of the game, the agent will try to visit state-action tuple (s, a^1, a^2) at least once. In single-agent Q-learning, such an action is very easy to be realized. In order to try (s, a) at least once, the agent can choose action that has not been chosen under state s . However, this method does not work for agent in a multiagent environment. Even if agent 1 can take each action in state s at least once, this can

still leaves many cell (s, a^1, a^2) in Q-table un-visited. This is because agent 1 cannot control the action of agent 2, and cannot beforehand predict which action agent 2 will pick. If agent 1 predict that agent 2 will choose a^2 , but instead agent 2 picks \tilde{a}^2 . the actual tuple that has been visited is (s, a^1, \tilde{a}^2) .

In order to overcome this problem, the agent will keep track of the number of times that the tuple (s, a^1, a^2) has been visited. Let $n(s, a^1, a^2)$ be the number of times that tuple has been visit. In state s , if there are cell that have not been visited, the agent gives equal opportunities to actions associated with that cell. For example, if

$$\begin{matrix} & a_1^2 & a_2^2 & a_3^2 \\ \begin{matrix} a_1^1 \\ a_2^1 \\ a_3^1 \end{matrix} & \begin{pmatrix} 3 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{matrix}$$

In this case, agent 1 will choose its three actions with probability $(\frac{2}{7}, \frac{2}{7}, \frac{3}{7})$.

Once all the actions have been visited at least once, the agent enters the second stage. The agent then explores (choose the non-optimal actions) with probably p and exploits with probability $1 - p$. Under state s , the single Q-learning agent sets $p = \frac{1}{1+n(s,a)}$, where $n(s, a)$ is the number of times the action a has been visited in state s . For Best-response Q-learning agent, assuming the agent is indexed by 1, $p = \frac{1}{1+n(s,a^1)}$ where $n(s, a^1) = \sum_{a^2} n(s, a^1, a^2)$. Such an exploration-exploitation strategy is called ϵ -Greedy strategy and satisfies the GLIE (Greedy in the Limit with Infinite Exploration) property defined by (Singh *et al.* 2000).

Experiments

We implement the Best-response Q-learning agent in a 4x4 grid game, which is identical to Game 1 in (Hu & Wellman 2000) except the difference in the number of grids.

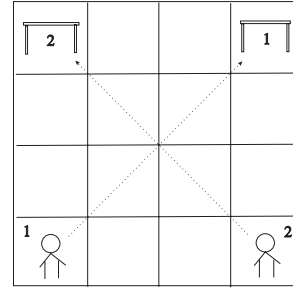


Figure 1: The 4 x 4 grid game

Basic assumptions

We assume that agents do not know the locations of their goals at the beginning of the learning period. Furthermore, agents do not know their own and the other agent's reward functions. Agents choose their actions simultaneously. They can observe the previous actions of both agents and the current state (the joint position of both agents). They also observe the immediate rewards after both agents choose their actions.

In this game, two agents start from respective lower corners, trying to reach their goal cells in the top row. An agent can move only one cell a time, and in four possible directions: *Left, Right, Up, Down*. If two agents attempt to move into the same cell (excluding a goal cell), they are bounced back to their previous cells. The game ends as soon as one agent reaches its goal. The agent who reaches its goal gets a positive payoff, while the one who does not gets nothing. In case both agents reach their goal cells at the same time, both are rewarded with positive payoffs.

The objective of an agent in this game is therefore to reach its goal with a minimum number of steps. A path (plan) is a sequence of actions from the starting position to the final destination. In the context of a game, a plan is called a *strategy*, or policy. A shortest path that allows an agent to reach its goal as soon as possible represents an optimal strategy. Two shortest paths that do not interfere with each other constitute a Nash equilibrium, since each path (strategy) is a best response to the other.

A state s represents the agents' joint location. Each location is a cell in the grid, and they are indexed from 0 to 15, starting from lower left corner and ending at the upper right corner. For example, the state (0, 3) indicates that Agent 1 is at lower left corner and agent 2 is at lower right corner; state (12, 15) indicates that Agent 1 is at upper left corner and agent 2 is at upper right corner. The total number of possible states is $15 \times 15 = 225$, excluding the goal positions.

Four types of agents

We have implemented four types of agents. The first type uses the traditional single-agent Q-learning method. We choose this type for the purpose of comparison. This is also an agent who ignores the actions of others. The other three types all take into the other agents' actions into account. In this sense, they are all joint action learners (Claus & Boutillier 1998). Our Best-response Q-learning agent is called NashSwitch agent here, because it makes an initial assumption that the other agent is a joint action learner, and therefore its best response strategy is a Nash strategy. This agent behaves exactly like a Nash Q-learning agent at the beginning of the game.

Single-Q agent A Single-Q agent uses the standard single-agent Q-learning method (Watkins & Dayan 1992). It learns about the Q-values for the state and its own actions, ignoring the actions of the other agents. A Single-Q agent updates its Q-values based on the following equation:

$$Q_{t+1}(s, a) = (1 - \alpha_t)Q_t(s, a) + \alpha_t[r_t + \beta \max_b Q_t(s', b)], \quad (6)$$

where $\alpha_t \in [0, 1)$ is the learning rate.

The Single-Q agent's optimal action is the one that satisfies $\max_a Q(s, a)$.

Nash-Q agent The second type of agent uses Nash Q-learning method. It learns about the Q-values for the state and agents' joint action (a^1, a^2). Its updating rule is defined in equation (1).

The Nash-Q agent determines its optimal action based on Nash equilibrium solution for current stage game,

$$(Q^1(s), Q^2(s)).$$

Fictitious agent A fictitious agent (Fudenberg & Levine 1998) chooses the other players play stationary strategies. Based on such assumptions, This agent counts the frequency of the other player playing certain actions in each state, and believes that such frequency is the probability, or in other words the mixed strategy that the other agent plays. Our agent then find its best response policy to play against this mixed strategy. The mathematical process is as the following:

In state s , if detecting agent 2's frequency to be $\{\frac{n_1}{n_1+n_2}, \frac{n_2}{n_1+n_2}\} = \{p_1, p_2\}$, The fictitious agent looks at its reward table for the current state s , and then solve the following equation:

$$\max_{\{q^1, q^2\}} E(r) = q^1(p^1 r_{11} + p^2 r_{12}) + q^2(p^1 r_{21} + p^2 r_{22})$$

where $q^1 + q^2 = 1$.

NashSwitch agent This is a best-response Q-learning agent who adopts Nash belief as its initial belief. Over time, it checks to see if the other agent is a Nash agent or not. If not, it switches away from Nash Q-learning.

Since it has little information about the other agents, it will take the default type of a Nash-Q agent. However, the other agent might not be a Nash-Q agent. Therefore, after certain periods, our agent then checks to see if the other agent is actually a joint action learner. A joint action learner (include both Nash-Q and Fictitious agent) should keep updating its beliefs on the joint payoffs. In that case, our agent should be able to visit almost all the state-action pairs over time. If it is not the case, then our agent will change both its learning method and action choice model.

There are several ways to detect the other agent's type. One is through performance measure. If for 1000 steps after step 50,000, the agent receives very bad performance by performing its Nash actions, then it knows that other agent is not a Nash one. Another criterion is the convergence results checking internally. If most cells have converged, then the other player is a joint action learner. If not, then the other agent is a single-agent learner.

The NashSwitch agent has its optimal action determined by the following solution: $\max_{\pi^1(s)} \pi^1(s) Q_t^1(s) \hat{\pi}^2(s)$.

The experimental process

A game starts from the initial state (0, 3). After observing the current state, agents choose their actions simultaneously. They then observe the new state, both agents' rewards and the action taken by the other agent. In the new state, agents repeat the process above. Agents' accumulated rewards are recorded for every step. Each step is counted as one period in the game. When at least one agent moves into its goal position, the game restarts. In the new episode, each agent is randomly assigned a new position (except its goal cell). The learning agent keeps the information learned from previous episodes, including its own accumulated rewards Each episode on average takes about 12 steps.

The learning rate is defined as the inverse of the time of visits. More specifically, for multiagent learning agent,

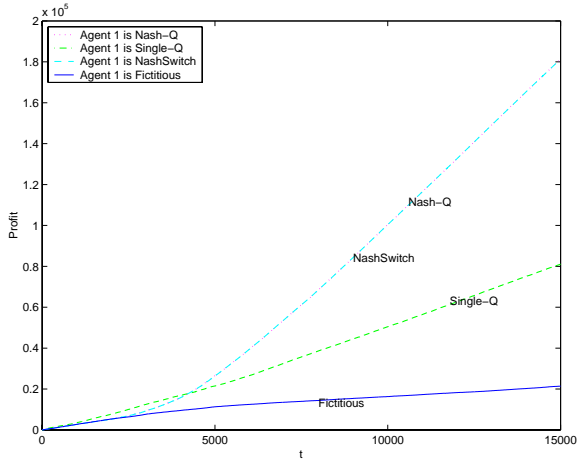


Figure 2: Agent 1's accumulated rewards when Agent 2 is Nash-Q

$\alpha_t(s, a^1, a^2) = \frac{1}{n_t(s, a^1, a^2)}$, where $n_t(s, a^1, a^2)$ is the number of times the tuple (s, a^1, a^2) has been visited. For single-Q agent, $\alpha_t(s, a) = \frac{1}{n_t(s, a)}$. When $\alpha_t = \frac{1}{95} = 0.01$, the results from new visits hardly changes the Q-values already learned.

Experimental Results

For each case, we run 20 experiments, and get their average results. This is to prevent the random factor in the game.

When agent 2 assumes the type of Nash-Q agent, agent 1 has different performance when it adopts four different types of learning. The results are shown in Figure 2, Interestingly, the graph shows that at beginning a Nash-Q agent does not perform as good as other Q-learning and fictitious agents. But over time, Nash-Q agent quickly catches up. By 15,000 steps, Nash-Q agent for outperforms the other two types, and Fictitious agent performs worst among all the types.

Figure 3 shows the performance of agent 1 when Agent 2 assumes the type of fictitious agent. In this case, Nash-Q is still the one that performs best, and Fictitious is the worst. However, the performance of Nash-Q is slightly lower than previous case, where agent 2 is a Nash agent.

Figure 4 shows agent 1's performance under different types when Agent 2 is a Single-Q agent. In this case, Single-Q agent outperforms the other two types and Fictitious is still the worst among all types.

As we can observe from Figure 2 to 4, the best performance that agent 1 can receive is when agent 2 plays a Nash strategy, and agent 1 responds with a Nash-Q strategy. As Figure 2 shows, when both agents plays Nash-Q strategy, agent 1's accumulated reward reaches to 18×10^4 by step 15,000, When both agents plays Single-Q strategy, shown in Figure 4, agent 1's accumulated reward can only reach 7×10^4 , which is less than half of the payoff when both plays Nash. This suggests that the strategy that ignores other agent would never lead to a good performance for the whole system.

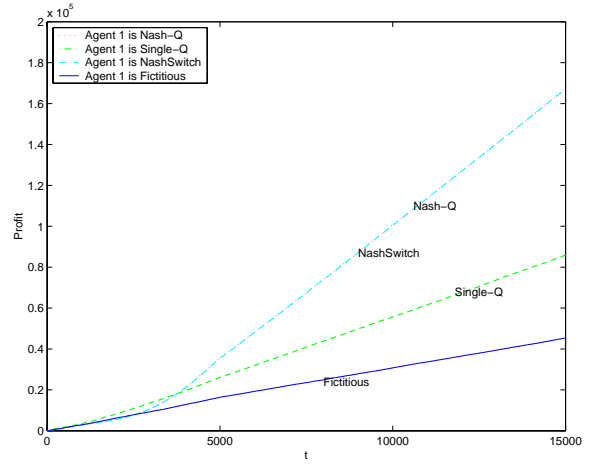


Figure 3: Agent 1's accumulated rewards when Agent 2 is Fictitious

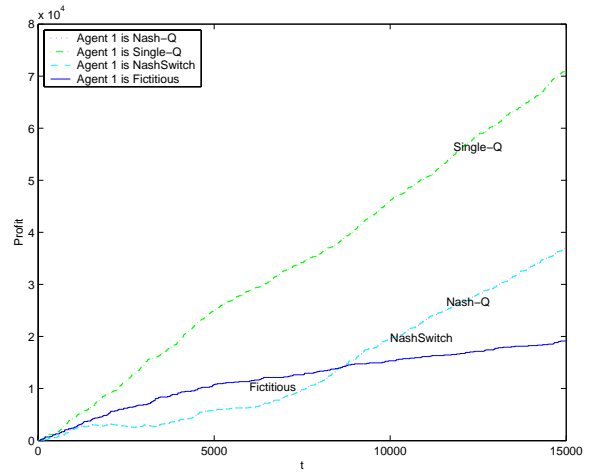


Figure 4: Agent 1's accumulated rewards when Agent 2 is Single-Q

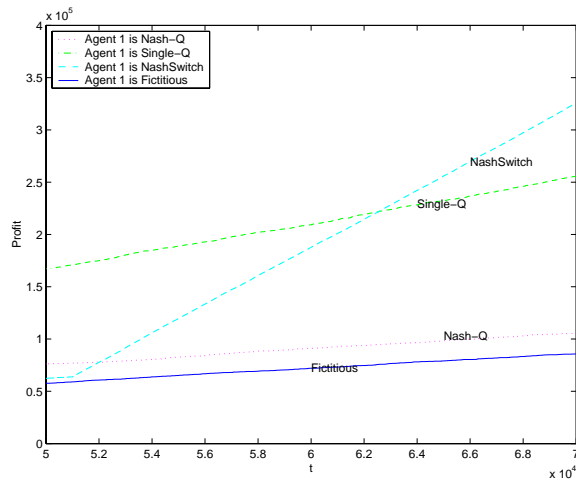


Figure 5: Agent 1's accumulated rewards when Agent 2 is Single-Q

However, NashSwitch Agent picks up its performance after it realizes that the other agent is not a joint action learner. Figure 5 shows the performance for the later steps, between step 50,000 and 70,000.

Conclusion and Future work

We propose a new multiagent Q-learning method that takes into account of other agents' actual strategy. This method is defined in general-sum stochastic games, and it is more general than the Nash Q-learning method proposed by Hu and Wellman (Hu & Wellman 1998). By removing away the assumption of the other agent's rationality, we allow the agent to learn the other agent's actual strategy. Based on the belief on the other agent's strategy, our agent updates its Q-values by assuming that it will take the best response strategy in the next state.

Our experiments show consistent best performance of this Q-learning algorithm, in comparison with several known Q-learning algorithms.

In the future, we will test other exploration and exploitation strategies. For example, test the Boltzmann function. We would also like to study non-stationary strategy, which has not been explored in this paper.

References

- Claus, C., and Boutilier, C. 1998. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 746–752.
- Filar, J., and Vrieze, K. 1997. *Competitive Markov Decision Processes*. Springer-Verlag.
- Fudenberg, D., and Levine, D. K. 1998. *The Theory of Learning in Games*. The MIT Press.
- Hu, J., and Wellman, M. P. 1998. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning*, 242–250. Madison, WI: AAAI Press.
- Hu, J., and Wellman, M. P. 2000. Experimental results on Q-learning for general-sum stochastic games. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 407–414. Stanford, CA: Morgan Kaufmann Publishers.
- Kaelbling, L.; Littman, M. L.; and Moore, A. W. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* 4:237–285.
- Singh, S.; Jaakkola, T.; Littman, M. L.; and Szepesvári, C. 2000. Convergence results for single-step on-policy reinforcement learning algorithms. *Machine Learning* 38(3):287–308.
- Watkins, C. J. C. H., and Dayan, P. 1992. Q-learning. *Machine Learning* 3:279–292.