In Proceedings of the 5th International Joint Conference on Autonomus Agents and Multiagent Systems (AAMAS'06), 789-791, May 2006.

Multi-agent Reinforcement Learning Algorithm to Handle Beliefs of Other Agents' Policies and Embedded Beliefs

Takaki Makino
Dept. of Frontier Sciences and Science Integration, the University of Tokyo
5-1-5 Kashiwa-no-ha,
Kashiwa, Chiba, 277-8568 Japan
mak@cb.k.u-tokyo.ac.jp

Kazuyuki Aihara
Institute of Industrial Science, the University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo 153-8505 Japan
ERATO Aihara Complexity Modelling Project, JST
3-23-5 Uehara, Shibuya-ku, Tokyo 151-0064 Japan
aihara@sat.t.u-tokyo.ac.jp

ABSTRACT

We have developed a new series of multi-agent reinforcement learning algorithms that choose a policy based on beliefs about co-players' policies. The algorithms are applicable to situations where a state is fully observable by the agents, but there is no limit on the number of players. Some of the algorithms employ embedded beliefs to handle the cases that co-players are also choosing a policy based on their beliefs of others' policies. Simulation experiments on Iterated Prisoners' Dilemma games show that the algorithms using on policy-based belief converge to highly mutually-cooperative behavior, unlike the existing algorithms based on action-based belief.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

Algorithms, Theory

Keywords

multi-agent reinforcement learning, policy-based belief, embedded belief, Iterated Prisoners' Dilemma

1. INTRODUCTION

Each member's effort to maximize its own benefit constitutes social behaviors, such as competition and cooperation. One promising way to implement such behaviors on computers is to extend a reinforcement learning algorithm into multi-agent environments, in order to provide independent trial-and-error learning for each agent. If the agents are smart enough, such social behaviors would emerge spontaneously.

Nevertheless, past studies on N-agent reinforcement learning have limited to specific structure of games [1, 2, 3]. Such limitations are required due to the way to extend reinforcement learning framework. These studies rely on the following update rule, which is the same as the classical TD-learning except action is replaced by a vector of actions:

$$Q_i(s(t),\vec{a}(t)) \leftarrow (1-\alpha)Q_i(s(t),\vec{a}(t)) + \alpha\big[r_i(t)+\gamma V_i(s(t+1))\big], \quad (1)$$

where $s(t)$ is the state at time t, $\vec{a}(t) = (a_1(t),\ldots,a_N(t))$ is the vector of actions for all players taken at time t, $Q_i(s,\vec{a})$ and $V_i(s)$ are the expected action-state value and state value for agent i ($1 \le i \le N$), $r_i(t)$ is the reward given to agent i as a result of actions $\vec{a}(t)$, α is the learning rate, and γ is the discount factor.

A Major problem of this formulation is that it cannot deal with learning co-player agents. Equation 1 assumes that the expected value $V_i(s)$ for agent i only depends on the current state s [6]. This is equivalent to assume stateless co-players, i.e., the states of other agents are independent of the expected value. Learning agents change its behavior based on the learned information; it is hard to find the best policy against such learning co-players under the assumption of statelessness.

Recently, Weinberg and Rosenschein have tried to break this situation by proposing the NSCP-learner [7] that finds the best-response policy to other players' policies. In their algorithm, agent i maintains a model $\langle\pi_j\rangle_i$ of another agent j ($1 \le j \le N, j \ne i$)[1], and calculates V_i by using the models:

$$V_i^{\pi_i,\langle\vec{\pi}_i\rangle_i}(s) = \max_{\pi_i} \sum_{a_1,\ldots,a_N} \pi_i(s,a_i) \cdot \prod_{1\le j\le N, j\ne i} \langle\pi_j\rangle_i(s,a_j) \cdot Q_i(s,\vec{a}), \quad (2)$$

where the policy $\pi_i(s,a_i)$ is the probability of agent i's taking action a_i on condition of state s and $\langle\vec{\pi}_i\rangle_i$ is the vector of beliefs about policies of the co-player agents for agent i, i.e., $\langle\vec{\pi}_i\rangle_i = (\langle\pi_1\rangle_i,\ldots,\langle\pi_{i-1}\rangle_i,\langle\pi_{i+1}\rangle_i,\ldots,\langle\pi_N\rangle_i)$. In short, the agent uses models to anticipate the actions of the co-players and calculates the state value using the belief on their expected actions.

However, we'd like to point out that Weinberg's extension is not sufficient. Suppose two possible beliefs $\langle\pi_j\rangle_i, \langle\pi'_j\rangle_i$ about policies of agent j, which choose the same action a_j for the immediate response to state s but different actions afterwards. Then the state values for these policies, $V_i^{(\ldots,\langle\pi_j\rangle_i,\ldots)}(s)$ and $V_i^{(\ldots,\langle\pi'_j\rangle_i,\ldots)}(s)$ should be able to be different, whereas eq. 2 cannot.

In this paper, we propose a new way to extend the action-value function Q with the vector of *policies* to solve this problem. Although an agent cannot know the current policies of the co-player agents, one can develop a *belief* of policies of the co-player agents and use it for the extended action-value function. This extension also lets us to introduce an embedded policy belief, i.e., a belief

[1]Weinberg *et al.* used the notation $\hat{\pi}^j$, but in this paper, we use the angle-bracket notation to clarify whose model is on whose policy.

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.

Initialize $Q_i(s,a)$
Repeat (for each episode):
  Observe the initial state s
  Repeat (for each step):
    Derive π_i from Q_i
    Choose a_i from π_i and s
    Take action a_i, observe $\vec{a}, r, s'$
    $Q_i(s,a) \leftarrow (1-\alpha)Q_i(s,a) + \alpha\left[r + \gamma \max_{a_i'} Q_i(s',a_i')\right]$
    $s \leftarrow s'$;
  until s reaches the end of the episode

Figure 1: Algorithm of a Level-0 agent of index i (equivalent to the original Q-learning algorithm).

Initialize $Q_i(\langle\vec{\pi_i}\rangle_i, s, a)$ and $\langle\vec{\pi_i}\rangle_i$
Repeat (for each episode):
  Observe the initial state s
  Repeat (for each step of episode):
    Derive Π_i from Q_i
    $\pi_i = \Pi_i\left(\langle\vec{\pi_i}\rangle_i\right)$
    Choose a_i from π_i and s
    Take action a_i, observe $\vec{a}, r, s'$
    $\langle\vec{\pi_i}\rangle_i' \leftarrow$ New belief, obtained by updating $\langle\vec{\pi_i}\rangle_i$ with s and $\vec{a}$
    $Q_i(\langle\vec{\pi_i}\rangle_i, s, a) \leftarrow (1-\alpha)Q_i(\langle\vec{\pi_i}\rangle_i, s, a) + \alpha\left[r + \gamma \max_{a_i'} Q_i(\langle\vec{\pi_i}\rangle_i', s', a_i')\right]$
    $s \leftarrow s'$; $\langle\vec{\pi_i}\rangle_i \leftarrow \langle\vec{\pi_i}\rangle_i'$;
  until s reaches the end of the episode

Figure 2: Algorithm of a Level-1 agent of index i.

about the co-player's policy belief, for better estimation of the co-player's policy. The simulation results on Iterated Prisoners' Dilemma (IPD) games show that our algorithm show better results than agents based on existing algorithms. We expect that such an algorithm would lead us to study mathematical structure of communication, such as mutual-estimation structure [4].

2. ALGORITHMS

As a starting point, we suppose a simple agent called Level-0, which uses a single-agent Q-learning algorithm (Fig. 1). This agent treats the other agents as a part of an environment and pays no attention to them. As a result, such an agent can only derive a very primitive learning in a multi-agent environment.

In the following, we discuss agents that use information of other agent's policies. Since agent i cannot directly see policy π_j of another agent j, agent i needs to obtain an estimate $\langle\pi_j\rangle_i$ of j's policy. If agent j has a stationary policy, some kind of statistical method can accurately estimate the policy of agent j by collecting a sufficiently large sample of the co-player's actions. Such a statistical method can be also applied to an co-player agent whose policy changes but converges to a limit [7].

2.1 Level-1: Using Policy-based Beliefs

We can extend the algorithm of a Level-0 agent to choose actions with regard to the policy beliefs $\langle\pi_i\rangle_i$ of other agents. One simple way to do so is to regard the tuple of policy belief and the original state $(\langle\vec{\pi_i}\rangle_i, s)$ as an extended state. Accordingly, agent i has an extended policy $\pi_i^+(\langle\vec{\pi_i}\rangle_i, s; a)$ and extended action-value function $Q_i^{\pi_i^+}(\langle\vec{\pi_i}\rangle_i, s; a)$. Assuming that the belief is accurate and that the change to the policies of the other agents has the Markov property, the agent can learn to choose an optimal policy. Thus we can design a Level-1 agent, which obtains $\langle\pi_i\rangle_i$ by using some statistical estimation and use $Q_i^{\pi_i^+}$ to choose an action.

We can have another viewpoint on this algorithm, in which a Level-1 agent i chooses a policy, instead of just an action, by using its belief about the other agents' policies. To choose a policy, the agent uses a *metapolicy* Π_i, which maps the policy belief of the other agents $\langle\vec{\pi_i}\rangle_i$ to a policy π_i. Interestingly, a metapolicy Π_i is just a notational variant of an extended policy π_i^+, since we can obtain a policy from π_i^+ by fixing $\langle\vec{\pi_i}\rangle_i$. Figure 2 shows the algorithm of a Level-1 agent in the new notation.

Initialize $Q_i(\langle\vec{\pi_i}\rangle_i, s, a)$, $\langle\vec{\Pi_i}\rangle_i$, and $\langle\langle\vec{\pi}\rangle\rangle_i$
Repeat (for each episode):
  Observe the initial state s
  Repeat (for each step of episode):
    Derive Π_i from Q_i
    $\langle\vec{\pi_i}\rangle_i \leftarrow \langle\vec{\Pi_i}\rangle_i\left(\langle\langle\vec{\pi}\rangle\rangle_i\right)$; $\pi_i \leftarrow \Pi_i\left(\langle\vec{\pi_i}\rangle_i\right)$
    Choose a_i from π_i and s
    Take action a_i, observe $\vec{a}, r, s'$
    $\langle\vec{\Pi_i}\rangle_i' \leftarrow$ New belief, obtained by updating $\langle\vec{\Pi_i}\rangle_i$ with $\langle\langle\vec{\pi}\rangle\rangle_i$, s and $\vec{a}$
    $\langle\langle\vec{\pi}\rangle\rangle_i' \leftarrow$ New embedded belief, obtained by updating $\langle\langle\vec{\pi}\rangle\rangle_i$ with s and $\vec{a}$
    $\langle\vec{\pi_i}\rangle_i' \leftarrow \langle\vec{\Pi_i}\rangle_i'(\langle\langle\vec{\pi}\rangle\rangle_i)$
    $Q_i(\langle\vec{\pi_i}\rangle_i, s, a) \leftarrow (1-\alpha)Q_i(\langle\vec{\pi_i}\rangle_i, s, a) + \alpha\left[r + \gamma \max_{a_i'} Q_i(\langle\vec{\pi_i}\rangle_i', s', a_i')\right]$
    $s \leftarrow s'$; $\langle\vec{\Pi_i}\rangle_i \leftarrow \langle\vec{\Pi_i}\rangle_i'$; $\langle\langle\vec{\pi}\rangle\rangle_i \leftarrow \langle\langle\vec{\pi}\rangle\rangle_i'$
  until s reaches the end of the episode

Figure 3: Algorithm of a Level-2 agent of index i.

Assuming that the policy beliefs are accurate and their change has the Markov property, agent i can learn a metapolicy, which is optimal in the sense of choosing the best policy in response to the policies of the other agents. However, in playing with other Level-1 agents, this assumption becomes unrealistic, since the policy selected by a metapolicy changes rapidly.

2.2 Level-2: Using Embedded Beliefs

A more elaborated agent, which we call Level-2, develops a belief about metapolicies of other agents. As mentioned before, a metapolicy is a map from policy beliefs to a policy; to estimate the policy of agent j using a metapolicy, agent i needs to have an embedded belief within agent j. We denote $\langle\langle\pi_k\rangle_j\rangle_i$ to represent agent i's belief about agent j's belief about agent k's policy. Agent i can obtain a belief of agent j's policy, given metapolicy belief $\langle\Pi_j\rangle_i$ and embedded policy belief $\langle\langle\vec{\pi_j}\rangle_j\rangle_i$ about agent j:

$$\langle\pi_j\rangle_i = \langle\Pi_j\rangle_i\left(\langle\langle\vec{\pi_j}\rangle_j\rangle_i\right) \quad . \tag{3}$$

We denote the agent i's embedded beliefs of all the other agents as

$$\left\langle\overrightarrow{\langle\vec{\pi}_{\cdot}\rangle_{\bar{i}}}\right\rangle_i = \left(\left\langle\langle\vec{\pi_1}\rangle_1\right\rangle_i, \ldots, \left\langle\langle\vec{\pi}_{\overline{i-1}}\rangle_{i-1}\right\rangle_i, \left\langle\langle\vec{\pi}_{\overline{i+1}}\rangle_{i+1}\right\rangle_i, \ldots, \left\langle\langle\vec{\pi}_{\overline{N}}\rangle_N\right\rangle_i\right). \tag{4}$$

In a perfect information game, every agent shares the same information, and every agent knows that the information is shared. Thus, agent i has no reason to distinguish the embedded belief $\langle\langle\pi_k\rangle_j\rangle_i$ for agent j from the one $\langle\langle\pi_k\rangle_{j'}\rangle_i$ for agent j', and as a result, we can denote them as $\langle\langle\pi_k\rangle\rangle_i$. This reduces the representation of embedded beliefs from $\left\langle\overrightarrow{\langle\vec{\pi}_{\cdot}\rangle_{\bar{i}}}\right\rangle_i$ to $\langle\langle\vec{\pi}\rangle\rangle_i$.

A Level-2 agent obtains an embedded belief $\langle\langle\pi_k\rangle\rangle_i$ by using the same statistical estimation method as the one for Level-1's belief policy; the only difference is that Level-2 needs $\langle\langle\pi_i\rangle\rangle_i$, the embedded estimation of the policy of agent i itself within another agent.

Using $\langle\langle\vec{\pi_j}\rangle_j\rangle_i$, Level-2 agent i is able to develop a belief about the metapolicy $\langle\Pi_j\rangle_i$ of agent j. Assuming the space of the embedded policy is finite, one can perform statistical estimations for every possible $\langle\langle\vec{\pi_j}\rangle_j\rangle_i$. By using the embedded policy belief $\langle\langle\vec{\pi}_{\cdot}\rangle_{\bar{i}}\rangle_i$ and the metapolicy belief $\langle\Pi_j\rangle_i$, a Level-2 agent can derive a policy belief $\langle\vec{\pi_i}\rangle_i$ and perform reinforcement learning on $Q_i(\langle\vec{\pi_i}\rangle_i, s; a_i)$. Note that this extended action-value function is the same as the one in the Level-1 agent.

Figure 3 shows the algorithm of the Level-2 agent. The algorithm is the same as that of the Level-1 agent (Fig. 2), except that $\langle\vec{\pi_i}\rangle_i$ is now calculated from $\langle\langle\vec{\pi}\rangle\rangle_i$ and $\langle\vec{\Pi_i}\rangle_i$.

2.3 Level-n: Using Deeper Embedded Beliefs

We can go along this line to create higher-lever agents successively. For instance, a Level-3 agent maintains $\langle\langle\langle\vec{\pi_l}\rangle_k\rangle_j\rangle_i$, $\langle\langle\Pi_k\rangle_j\rangle_i$,

Table 1: Pay-off matrix for agents X and Y

X \ Y	C	D
C	3 \ 3	5 \ 0
D	0 \ 5	1 \ 1

and $\langle \Pi_j \rangle_i$; A Level-4 agent maintains $\langle\langle\langle\langle \vec{\pi}_m \rangle_l \rangle_k \rangle_j \rangle_i$, $\langle\langle\langle \Pi_l \rangle_k \rangle_j \rangle_i$, $\langle\langle \Pi_k \rangle_j \rangle_i$, and $\langle \Pi_j \rangle_i$; and so on. In any case, the innermost embedded policy belief is obtained by a statistical estimation, and applied to metapolicy beliefs to obtain a, presumably better, policy belief.

Although the number of embedded policy beliefs and metapolicy beliefs seems to increase exponentially, we can reduce it by assuming a perfect information game as in the Level-2 agent's case. Since agent i has no reason to distinguish $\langle\langle \Pi_k \rangle_j \rangle_i$ from $\langle\langle \Pi_k \rangle_{j'} \rangle_i$, we can combine them into $\langle\langle \Pi_k \rangle\rangle_i$. Similarly, $\langle\langle\langle \vec{\pi}_l \rangle_k \rangle_j \rangle_i$ for any k and j can be reduced to $\langle\langle\langle \vec{\pi}_l \rangle\rangle\rangle_i$. This sort of reduction limits the number of beliefs to a linear increase.

3. SIMULATION EXPERIMENTS

We conducted a series of simulation experiments, in which the proposed agents play the Iterated Prisoners' Dilemma [**?**], a simple two-player game. In every turn, each agent would independently choose an action, either cooperation (C) or defection (D). The agents would then observe the result, the choice of the other agent and the pay-off given as in Table 1. Each agent would strive to maximize the sum of the pay-offs for itself over many turns. Noise in action is introduced with the probability of ε ; thus an agent chooses unwanted action with a probability of $\varepsilon/2$.

In the experiments, we applied reinforcement-learning algorithms to the game. A state refers to the last step of the action history of the two agents, i.e., $s(t) = (a_{\text{other}}(t-1), a_{\text{self}}(t-1))$; thus there are four possible states. Since we only used a deterministic policy, a policy becomes a four-dimensional vector of C or D. In other words, the policy space has only $2^4 = 16$ elements.

We specified an agent's belief space of another's policy to be equal to the policy space of the agent itself, that is, a four-dimensional vector of C or D. The statistical estimation method of the policy belief is simple; whenever the agent observes an action different from the policy belief, the agent replaces the corresponding element of the belief vector to make the belief match the observation.

We also provide two series of variants for comparison with our algorithm. One is Level-0(H3) agent, which is the same as Level-0 agent except that it uses three steps of action history as the state of reinforcement learning so that this agent has the same table size as the Level-n agents ($n \geq 1$) without using beliefs. The other is Level-n(A) agents, which uses action beliefs $\langle \vec{a}_j \rangle_i$ instead of policy beliefs $\langle \vec{\pi}_j \rangle_i$. Metapolicy reduces to a map from expected opponent's action $\langle \vec{a}_j \rangle_i$ to a policy, and so do metapolicy beliefs. Note that the algorithm of the Level-1(A) agent is equivalent to the NSCP-learner algorithm [7] (see eq. 2). To distinguish from these variants, we denote agents based on our proposed algorithms as Level-n(P).

In all of the simulations described below, we fixed $\alpha = 0.03$ and $\gamma = 0.99$. Since our interest is in the ability of agents playing against peer agents, we focused on simulations that cover only the case that two equivalent agents play IPD games.

3.1 Results

Figure 4 shows averaged pay-off over 2,000 trials of the IPD game by two same-level agents. After the initial learning time, the proposed Level-n(P) agents ($n \geq 1$) obtained average pay-offs very close to 3, the best pay-off that can be obtained only by mutual cooperation. This shows clear contrast to the pay-offs obtained by Level-0, Level-0(H3), and Level-n(A) agents; in fact, in the experiments of each of these agents, the agents failed learning to act cooperatively in more than 30% of the trials. These results indicate that our Level-n(P) agents have better ability to recognize the situation with peer learning agent and to learn the optimal policy.

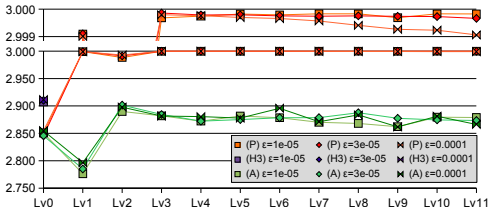

Figure 4: Average pay-off after learning with various ε values. Pay-off is averaged for 5×10^8 steps after 5×10^8 learning. The upper graph enlarges the top part of the lower graph.

Effects of having deeper embedded beliefs are not clear, except slight loss of pay-off is observed in case of $\varepsilon = 0.0001$. This indicates that there is some penalty to keep deeply embedded beliefs in noisy environments. However, since the penalty is very small, we can say that our algorithm is robust to the noise.

4. SUMMARY

We developed a new series of multi-agent reinforcement-learning algorithms that choose a policy based on beliefs about co-players' policies. The algorithms are applicable when a state is fully observable, and there is no limit on the number of players or a particular configuration of pay-offs. By assuming that the co-players also use policy-based beliefs to choose a policy, we also presented algorithms that use various depths of embedded policy beliefs.

We compared the algorithms of different embedding depths and algorithm variants through a series of simulation experiments on the IPD game. The results showed that the algorithms using policy-based belief converged to mutually cooperative behavior, unlike the existing algorithms based on action-based beliefs.

Acknowledgments

This study is partially supported by the Advanced and Innovational Research program in Life Sciences and Grant-in-Aid for Scientific Research on Priority Areas 17022012 from MEXT of Japan.

5. REFERENCES

[1] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proc. of AAAI-98*, pages 746–752. 1998.

[2] J. Hu and M. P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proc. of ICML 1998*, pages 242–250. 1998.

[3] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proc. of ICML 1994*, pages 157–163. 1994.

[4] T. Makino and K. Aihara. Self-observation principle for estimating the other's internal state. Mathematical Engineering Technical Reports METR 2003–36, the University of Tokyo, Oct. 2003.

[5] L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11:387–434, 2005.

[6] Y. Shoham, R. Powers, and T. Grenager. On the agenda(s) of research on multi-agent learning. In *Proc. of Artificial Multiagent Learning: Papers from the 2004 AAAI Fall Symposium, Technical Report FS-04-02*. 2004.

[7] M. Weinberg and J. S. Rosenschein. Best-response multiagent learning in non-stationary environments. In *Proc. of AAMAS'04*, pages 506–513, 2004.