NAIST-IS-DD0461034

Doctoral Dissertation

Learning and Decision-Planning in Partially Observable Environments

Hajime Fujita

January 30, 2007

Department of Bioinformatics and Genomics
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of Engineering

Hajime Fujita

Thesis Committee:

Professor Shin Ishii	Supervisor
Professor Kenji Sugimoto	Co-supervisor
Associate Professor Tomohiro Shibata	Member

Learning and Decision-Planning in Partially Observable Environments*1

Hajime Fujita

Abstract

In the real world, information about the environment depends on unreliable inputs received through sensory equipment. All living creatures and intelligent systems, therefore, must learn their policy and make decisions based on such imperfect information obtained by the interaction with the underlying environment. Learning and decision-making in partially observable situations are indispensable mechanisms in realistic environments, and problems with controlling computer systems under uncertainty, as well as experiments to reveal how such mechanisms operate in the brain, have received significant attention as imperative issues in diverse communities such as engineering, information science and cognitive psychology over the past few years.

This dissertation presents research results about learning and decision-planning in partially observable environments. As studies for controlling autonomous agents, we propose model-based reinforcement learning (RL) schemes for large-scale multi-agent problems with partial observability. Games constitute a challenging domain of RL for acquiring strategies, because most of them include multiple players and many unobservable variables in a large state space. The difficulty of solving such realistic multi-agent problems with partial observability arises mainly from the fact that the computational cost for the estimation and prediction in the whole state space, including unobservable variables, is too expensive. To overcome this intractability and enable an agent to learn in an unknown environment, an effective approximation method is required with explicit learning of the environmental model. This dissertation applies our methods in particular to the card game of "Hearts." This game is a well-defined example of an imperfect information game, and can be approximately formulated as a partially observable Markov decision process (POMDP) for a single learning agent. To reduce the computational cost, we use effective approximation techniques in which the heavy integration required for the estimation and prediction can be approximated by an averaged state or a plausible number of samples. Com-

*1 Doctoral Dissertation, Department of Bioinformatics and Genomics, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0461034, January 30, 2007

puter simulation results show that our methods are effective in solving such a difficult partially observable multi-agent problem.

As studies for modeling human behavior, we demonstrate the performance of a probabilistic model for human decision-making and estimation processes in a partially observable environment. Since theoretical POMDP studies have indicated that optimal decision-planning under partially observable situations has computational difficulties, it is speculated that the human brain avoids the difficulties with some effective approximation for estimating unobservable states and making decisions. To reveal how the decision-making mechanism with the approximate estimation process operates in the brain, we explore the possibility that our model-based RL framework is implemented in the human brain. Behavioral experiments using a virtual maze were carried out to thirteen normal subjects. Because the subjects could not observe their real position in the maze but obtain partial observation, it is important for solving the problem effectively to estimate the unobservable real positions based on the observation sequence. Model-based analyses show that our model can reproduce the subjects' behaviors with high accuracy, and indicate that humans estimate unobservable states based on the framework of the incremental Bayes estimation.

Keywords: Partially observable Markov decision process (POMDP), Reinforcement learning (RL), Card game Hearts, Behavior model.

部分観測環境における学習と意思決定に関する研究[*2]

藤田 肇

内容梗概

実世界では，周囲を取りまく環境に関する情報は感覚器官を通して知覚できる不完全な入力に依存するため，生物や知的システムは，環境との相互作用を通して得られた部分的な観測系列に基づいて学習と意思決定を行わなければならない．したがって，部分観測状況における学習と意思決定を担う機構は実環境で不可欠であり，近年様々な研究分野でこの本質的な機構を理論的・実験的に解き明かす試みがなされている．

本学位論文では，部分観測環境での方策学習と意思決定に関する研究結果について述べる．まず，部分観測性を持つ大規模な未知環境におけるモデル同定型強化学習法を提案する．本手法に基づく強化学習エージェントは，部分観測性，マルチエージェント系，大規模な状態空間という 3 つの困難な条件を持つ環境において意思決定と方策の学習を行う．想定した条件を満たす応用課題としてカードゲーム “Hearts” を取り上げ，提案手法に基づいてこの実環境における戦略の獲得を目指す．このゲームでは，相手が所持するカードは観測できないため，各プレイヤーにとって部分観測状況となる．そのため，各エージェントは観測できないカードの分布を，明にあるいは暗に推定しなければならない．また，4 人対戦用の競合型ゲームであるため，複雑なマルチエージェント系となる．したがって，環境の一部である相手プレイヤーのモデルを同定した上で，効果的に行動予測を行う必要がある．さらに，通常の 52 枚のカードを用いるこのゲームは，巨大な状態空間を持つ．したがって，方策を決定するために必要な，予測と推定に伴う計算困難性を回避するために，効果的な近似解法が必要になる．本研究では，この予測と推定に関わる 2 種類の近似アルゴリズムを提案する．第一の手法は，平均場近似法の考え方を応用したものであり，第二の手法は，サンプリング法を利用したものである．これらの手法に基づく学習エージェントの性能を計算機シミュレーションにより評価した．その結果，上級レベルの強さを持つルールベースエージェントよりも優れた戦略を獲得でき，本手法が複雑な部分観測環境における強化学習法として有用であることが示された．

次に，部分観測環境におけるヒトの意思決定モデルを提案する．部分観測環境での最適意思決定問題は計算論的に困難であることが分かっており，ヒトは実世界で非観測状態の推定や方策決定などの計算に近似的な処理を行っていると考えられる．本研究では，部分観測状況におけるヒトの意思決定過程を解明するために，先の研究で提案した計算モデルの脳内実現の可能性を探る．ここでは，13 人の被験者に対して部分観測迷路課題を課し

[*2] 奈良先端科学技術大学院大学 情報科学研究科 情報生命科学専攻 博士論文，NAIST-IS-DD0461034，2007 年 12 月 14 日

た．被験者は迷路内において自身の周囲に関する限られた観測を得ることができるが，真の絶対位置を知ることはできない．したがって，課題を効率よく達成するためには，部分的な観測情報から迷路内における自身の位置と頭の向きを推定する必要がある．提案モデルは，部分観測迷路課題に対する被験者の意思決定過程を適切に説明できると同時に，逐次ベイズ推定に基づく非観測状態の推定を，ヒトが脳内で実現している可能性を示唆する．

キーワード: 部分観測マルコフ決定過程，強化学習，カードゲーム Hearts，行動モデル

Acknowledgement

修士学生として本学へ入学した当初は他講座への配属を希望していましたが，配属面談で不勉強な性格であることを看破されて落選し，当てが外れて途方に暮れていたところを，面談もなしに希望調査票に書いた貧弱な志望動機だけで迎え入れてくださったのが石井先生でした．その後は，興味深くやりがいのある研究テーマを与えてくださり，正確に個性を把握してそれを尊重しながら，怠け者を根気よく指導してくださいました．先生の適切な導きにより，海外発表をはじめとして貴重な経験にも恵まれ，論文として研究成果を残すことができ，こうして学位論文を提出することができました．研究全般にわたって熱心にご指導くださった石井先生に，まず第一に深く感謝したいと思います．ありがとうございました．

必修だった「英語ライティング法」の授業で，論文のレビューを書くという宿題に真剣に取り組まない様子を見ると，スミス先生は不真面目な学生を部屋に呼んで粛々と諭し，その後は英会話の機会を与えてくださるなど，熱心に英語の指導をしてくださいました．この学位論文を書き上げるにあたっても辛抱強く表現を直してくださり，そのお蔭で満足のいく文章に仕上げることができました．いくつかの研究成果についても，スミス先生の助力なくしては成し得なかったと思います．ここに改めて感謝致します．

論理生命学講座の皆様には，日常から研究に関して様々な助言をして頂きました．特に，中村さんは机の位置も研究分野も近かったためお世話になることが多く，一時期は頼り切りになることが自分でも問題に感じられるほどでした．転勤後は最も身近に相談できる人がいなくなり不安でしたが，それでもどうにか研究を進められているのは，中村さんのお蔭によるところが大きいと思います．吉田さんとは同郷のせいもあって普段から親しくさせて頂いたのですが，その分ご迷惑をおかけすることも多かったようで反省しています．最後に立派な研究テーマで共同研究に取り組めたことを嬉しく思います．吉本さんには，特に研究を始めたばかりの時期に大変お世話になりました．沖縄へ移られた後も，時々研究のことについて親身に相談に乗ってくださるばかりでなく，ソフトボール大会でも大活躍で，とても心強い存在でした．大羽さんには，もちろん今も適切なアドバイスを頂いていますが，何よりも入学当初に輪講でお世話になりました．当時は内容が難しすぎてほとんど何も理解できませんでしたが，この研究分野の面白さについては肌に浸みるように理解できました．また，普段から親しく接してくださり，この謝辞も大羽さんたちと一緒に夜食を食べに行った後に書いています．前田さんとは，つい先ほど夜中のエレベータの中で会いました．夜遅くまで学生の研究指導をしてくださるほどの熱心さに自身も大

きく与れたことは，実に幸運だったと思います．そして，研究を始めた初期の頃に，専門分野に関する基礎知識を教えてくださったのが玉越さんでした．計算機の使いかたを始めとして，いまだに強く影響を受けている部分が多く，お世話になった様々なことが忘れられません．いまの自分が研究に携わり，学位論文を提出するまでに至れたのは，こうした偉大な先輩方の背中を見続けてきたからです．本当にありがとうございました．

長く一緒に頑張ってきた同級生や後輩にもお世話になりました．坂東君・川脇君とはよく映画や食事に出かけました．他愛もない話をしたり夜中にゲームをしたり，まるで足の引っ張り合いのようでしたが，そのお蔭で研究に励むことができました．平山君・森君とは一緒にカナダへ出張に行ったことが一番印象に残っています．今後も良き友達であると同時に，良きライバルとしていられるよう精進します．塚田君・本田君とは先ほど一緒にご飯を食べました．部屋も研究分野も違うので普段は顔を合わす機会があまりありませんが，たまに話をするたびに心強く感じました．大倉君には本学位論文の 5 章に記載した内容でとてもお世話になりました．論文を一貫したものに仕上げることができたのは彼のお蔭です．松原君とはよく研究の話をしました．お蔭で妙なロボットの知識が身につきました．そして，松山君とは阪大から一緒に本学へ移ってきて頑張りました．卒業後も親身に相談に乗ってくれたりなど，とても励みになりました．同級生に恵まれたことは，この上なく幸運なことだったと思います．本当にありがとうございました．

柴田さんには，研究室での生活全般に関わることはもちろん，毎回のゼミ発表でも貴重なご意見を頂きました．本学への入学と，柴田さんが助教授として本講座へ着任されたのがほぼ同時期だったため，同じだけの長い期間を同じ場所で努力し，そして熱心にご指導くださいました．ここに深く感謝致します．作村さんには，計算機のことについていつも我儘をきいて頂きました．クラスターを酷使するこの研究を華々しく締めくくることができたのも，蔭ながら助けて頂いたお蔭だと思います．ありがとうございました．杉本先生は，修士課程から一貫して副指導教官を担当してくださり，中間発表などの機会ではいつも貴重なご意見をくださいました．ありがとうございました．

お忙しい中，論文審査を引き受けて下さった石井信教授，杉本謙二教授，柴田智広助教授に感謝致します．ありがとうございました．

最後に，博士課程へ進学したいという「無謀な」願いを聞き入れてくれた両親と，それを支えてくれた家族に感謝します．

本学位論文と記載した研究成果は，多くの人々に支えられて達成することができました．ここに改めて，心より感謝を申し上げると同時に，今後もご指導くださいますよう厚くお願い申し上げます．

Contents

List of Figures

List of Tables

Notation

- $\mathcal{S}$: the state set
- s_t: a state at time t
- $\mathcal{A}$: the action set
- a_t: an action at time t
- $\mathcal{O}$: the observation set
- o_t: an observation at time t
- $R()$: the reward function
- r_t: a reward at time t
- $\mathcal{B}$: the belief space
- $\hat{\mathcal{B}}$: the approximated belief space
- b_t: a belief state at time t
- $\mathcal{X}$: the internal state set
- x_t: an internal state at time t, where $x_t = \{y_t, c_t\}$
- y_t: a state at time t, which is estimated by the subjects
- $\mathcal{C}$: the confidence set
- c_t: a confidence of the subjects for their state estimation

- γ: the discount factor
- $V()$: the value function
- $V_i()$: the value function at i-th iteration step
- $V^*()$: the optimal value function
- $\hat{V}()$: the approximated state value function
- $Q()$: the action value function
- $Q^*()$: the optimal action value function
- π: a policy
- π^*: an optimal policy
- T: the termination time

- $\rho(b_t, a_t)$: the reward function over the belief space, defined in equation (2.5)
- $\tau(b_t, a_t, o_{t+1})$: the state estimator function (the belief update function), defined in equation(2.6)

- α_i: an $|\mathcal{S}|$-dimensional hyper-plane
- Γ_i: the set of α_i hyper-planes at i-th iteration step
- Γ_i^*: the parsimonious set of hyper-planes, which is sufficient to represent the value function $V_i()$
- $\mathcal{M}$: the set of memory states
- m_t: a memory state at time t
- δ_t: an TD error at time t

- M: the number of agents in the common environment
- s_t^i: a state at time t for agent i ($i = 0, \cdots, M$)
- a_t^i: an action at time t for agent i
- o_t^i: an observation at time t for agent i
- ϕ^i: a strategy for agent i
- $\hat{\phi}^i$: a strategy for agent i, which is approximated by the learning agent
- H_t: a history at time t, representing $H_t = \{(o_t, -), (o_{t-1}, a_{t-1}), \cdots, (o_1, a_1)\}$
- h_t: a partial history at time t, representing $h_t = \{(o_t, -), (o_{t-1}, a_{t-1})\}$
- h_t^k: a truncated history with k-length at time t, representing $h_t^k = \{(o_t, -), (o_{t-1}, a_{t-1}), \cdots, (o_{t-k+1}, a_{t-k+1})\}$

- $U_t(H_t, a_t)$: the utility function at time t, defined in equation (4.2)
- N: the number of samples for the current state
- K: the number of samples for the next state
- $\hat{s}_t, \hat{s}_{t+1}$: a sampled state and a sampled next state, respectively
- $\hat{s}_t^{(j)}, \hat{s}_{t+1}^{(k)}$: a j-th sampled state and a k-th sampled next state, respectively
- $\hat{a}_t^{i,(k)}$: a constituent of the action sequence $\{\hat{a}^{1,(k)}, \cdots, \hat{a}^{M,(k)}\} \in \mathcal{A}^M(\hat{s}_t^{(j)}, a_t, \hat{s}_{t+1}^{(k)})$
- $\hat{o}_t^i$: a sampled observation
- $\hat{o}_t^{i,(j)}$: a j-th sampled observation
- T_m: a constant denoting the assumed action randomness
- T_m^i: a constant denoting the assumed action randomness of agent i's policy
- $\langle \cdot \rangle$: the expectation symbol
- $\mathcal{A}^M(s_t, a_t, o_{t+1})$: the set of possible sequences of opponents' actions $\{a_t^1, \cdots, a_t^M\}$ in which the state s_t reaches the next state whose observation is o_{t+1}, after the action a_t.
- $\hat{o}_t^i(a_t, H_t)$: an expected observation for agent i, defined in equation (3.7)
- $F^i(o_t^i, a_t^i; \hat{\phi}^i)$, $F^i(q_t^i, r_t^i; \hat{\phi}^i)$: an assumed utility of taking action a_t^i for an observation o_t^i
- p_t: an input of a value function at time t
- q_t^i: an input of i-th action predictor at time t

- r_t^i: an output of i-th action predictor at time t

- $\boldsymbol{s}$: a state sequence, representing $\boldsymbol{s} = \{s_1, \cdots, s_{T+1}\}$
- $\boldsymbol{a}$: an action sequence, representing $\boldsymbol{a} = \{a_1, \cdots, a_T\}$
- $\boldsymbol{o}$: an observation sequence, representing $\boldsymbol{o} = \{o_1, \cdots, o_T\}$
- $\boldsymbol{r}$: a reward sequence, representing $\boldsymbol{r} = \{r_1, \cdots, r_T\}$
- $\boldsymbol{x}$: an internal state sequence, representing $\boldsymbol{x} = \{x_1, \cdots, x_{T+1}\}$
- $\boldsymbol{y}$: an estimated state sequence, representing $\boldsymbol{y} = \{y_1, \cdots, y_{T+1}\}$
- $\boldsymbol{\theta}$: the internal state transition, representing $\boldsymbol{\theta} \equiv \{\theta_{ij} = P(x_{t+1} = j | x_t = i, a_t) \mid i = 1, \ldots, |\mathcal{X}|, j = 1, \ldots, |\mathcal{X}|\}$.
- $\boldsymbol{\sigma}$: the action selection probability of the subjects, representing $\boldsymbol{\sigma} \equiv \{\sigma_{ik} = P(a_t = k | x_t = i) \mid i = 1, \ldots, |\mathcal{X}|, k = 1, \ldots, |\mathcal{A}|\}$
- N_{s}: the number of real states s_{t+1} which do not violate the observation obtained at time $(t+1)$
- N_{a}: the number of executable actions
- N_{exp}: the number of exploratory actions
- N_{opt}: the number of optimal actions
- ϵ_{c}: a parameter defining the dynamics of the confidence
- ϵ_{exp}: a parameter defining the action selection probability in the exploratory mode
- ϵ_{opt}: a parameter defining the action selection probability in the exploitation mode

Chapter 1

Introduction

This dissertation presents research results about learning and decision-planning in partially observable environments. In the real world, environments often have partial observability: a decision-maker cannot directly access internal states of the environment, and can obtain only observations that contain partial information about the states. Learning and decision-making problems in such a situation can be theoretically formulated as partially observable Markov decision processes (POMDPs). When introducing this framework to realistic problems, however, serious computational difficulties arise because exact solutions require computing a policy over the entire belief space. Not only the estimation process for a large number of unobservable states, but also computing a policy depending on the estimation need too heavy computation even with effective approximation methods. Only a few of POMDP studies, therefore, assumed a large-scale partially observable environment, and how humans deal with such intractable problems in the real world is still almost unknown.

This thesis is about the research results of policy learning and decision-theoretic planning in realistic partially observable environments. As studies for controlling autonomous agents, it presents model-based reinforcement learning (RL) schemes for large-scale multi-agent problems with partial observability, and the proposed RL methods are in particular applied to a card game, "Hearts," a well-defined example of a multi-player, competitive and partially observable game. As studies for modeling human behavior, this thesis demonstrates the performance of probabilistic models for human decision-making process in a partially observable environment, and indicates a possible implementation of the human brain.

1.1 Learning and decision-planning

In the real world, humans and other living creatures survive by selecting actions suitable for satisfying their physiologic cravings. Under the situation in which some

kind of reward, such as food or money proportional to the efficiency of an action, are provided from an environment as a resulting evaluation of the action, the simplest optimality can be defined as taking the action that yields the largest reward. Animals, therefore, try to learn the association among the rewarding effect from the discrepancy between what it predicted would happen and what actually happened, based on trial and error. In the research field of psychology, this associative learning is regarded as a fundamental principle for all living things, and the modification of voluntary behavior based on the reward is called operant conditioning. Thorndike (1911) observed the behavior of cats trying to escape from a home-made puzzle box; although the cats took a long time to escape when first constrained in the box, ineffective responses occurred less frequently and successful responses occurred more frequently with experience. This enabled the cats to escape in less time over successive trials. In general, of several responses made to the same situation, those which are accompanied or closely followed by satisfaction to the individual will, other things being equal, be more firmly connected with the situation, so that, when it recurs, they will be more likely to recur, and vice versa. Since the phenomenon describes the effect of reinforcing events on the tendency to select actions, it is called the "Law of Effect" (Thorndike, 1911); it is widely regarded as a basic principle underlying much behavior.

Many conditioning models under various conditions such as the Rescorla-Wagner model (Rescorla & Wagner, 1972), a famous model of classical conditioning in which animals are theorized to learn from the association of pairing stimuli, have been presented to describe reasonable ideas in terms of a learning and decision-making process for single trial tasks. Although these models could predict experimental results of the conditional learning according to simple mathematical equations, and have been regarded as effective computational models to explain learning processes of animals, they could not deal with decision-theoretic planning for delayed reward tasks in a multiple time scale; for example, considering the situation in which one achieves a great success due to a constant effort, even if the effort itself is nothing less than punishment, the total reward is maximized because the subsequent success is regarded as a large reward. That is, humans and other higher-order living creatures can determine an immediate action by considering a delayed reward expected in the future. The conventional conditional learning models for single trial tasks are not applicable to delayed-reward and multiple-step tasks. To deal computationally with learning and decision-making of higher order of living creatures, therefore, it is necessary to consider behavior models by assuming the general setting in which they maximize the cumulative reward that can be obtained by taking sequential actions over multiple steps. To analyze and interpret such behavior based on the reward sequence,

the learning theory based on trial and error has been studied in diverse fields in the psychology of learning.

In the research field of engineering, on the other hand, the optimal control problem of designing a controller to minimize a measure of a dynamical system's behavior over time, has been widely studied. One example is the problem of the optimal torque to control a robot arm. Such problems have not essentially included the concept of learning, but have been studied by solving analytically linear equations. As one of the approaches to this problem, Bellman (1957a) defined a functional equation, now often called the Bellman equation, by introducing the concepts of a dynamical system's state and of an optimal return function. The class of methods for solving optimal control problems by solving this equation came to be known as dynamic programming (DP), where the return function is a function which returns an expected cumulative return for a particular control in a certain state. DP algorithms solve optimal control problems based on iterative calculation of the optimal return function according to the Bellman equation: when sequential returns are provided for control signals under the situation that discrete states transit stochastically, which is formulated as a Markov decision process (MDP) (Bellman, 1957b), the optimal controller which maximizes the expected cumulative return can be obtained by iterative optimization between the controller and the return function. With the progress of learning theory based on trial and error in psychology, the DP devised in the field of engineering has been promoted to reinforcement learning (RL) (Sutton & Barto, 1998) as a new framework for solving problems of determining an optimal action sequence in unknown environments.

1.2 Reinforcement learning

RL is regarded as the first field to seriously address the computational issues that arise when learning from interaction with an environment to achieve long-term goals. In an MDP environment whose dynamics is unknown for an agent, problems whose solution is defined as acquiring the optimal policy that maximizes the expected cumulative reward can be solved by an RL method. In other words, in a situation that environmental states whose process satisfies the Markov property change according to an unknown probability distribution, a policy to maximize the expected accumulative reward can be obtained based on an RL algorithm. Since the policy is improved based on trial-and-error learning, two important aspects included in the Law of Effect apply to the learning scheme of RL: first, it is selectional, which means trying alternatives and selecting a better solution among them by comparing their results; and second, it is associative, which means the solution found by selection is associated with a

particular situation. By these two distinct aspects, RL is distinguished from other learning schemes in the machine learning field: supervised learning is associative but not selectional because the correct answer is given for learning, and natural selection in evolutionary methods is an example of a selectional process, but it is not associative. Based on the RL theory, through the selectional and associative process, that is, trying various alternatives in each situation and remembering which action yields the best performance in a particular situation, the optimal decision-making can be attained even in an unknown environment.

Various approaches have been proposed, and their performances and convergence conditions have been widely discussed in the theoretical research field of RL. In particular, temporal difference (TD) learning, developed by Sutton (1988) with inspiration from the concept of animal learning psychology and artificial intelligence research, has largely influenced the wide research area of learning and control. TD learning is an iterative stochastic approximation algorithm based on the prediction error of the expected cumulative reward, which does not require knowledge of the environmental model. Many powerful algorithms such as Q-learning (Watkins & Dayan, 1992) and Actor-Critic method (Barto, Sutton, & Anderson, 1983) have been proposed by applying the framework, and have been analyzed theoretically in terms of applicability, required computational time, converging proofs and relationship to other studies, under various conditions. In addition, many researches have attempted realistic problems as the application of such theoretical studies. Examples include robot controls with multiple degrees of freedom such as working robots (Mori, Nakamura, & Ishii, 2004; Collins, Ruina, Tedrake, & Wisse, 2005; Ueno, Nakamura, Takuma, Shibata, Hosoda, & Ishii, 2006), a stand up robot (Morimoto & Doya, 2001) and an acrobot (Nishimura, Yoshimoto, & Ishii, 2004), and high-dimensional problems in dynamic environments such as the elevator dispatch problem (Crites, 1996; Crites & Barto, 1996a, 1996b), a channel-allocation task for a cellular telephone system (Singh & Bertsekas, 1996), and playing Backgammon (Tesauro, 1994) have been studied as applications of the RL field.

In physiology fields such as neuroscience, on the other hand, the study of the neural substrates for reward-based learning has been strongly influenced by computational theories. In recent work, such theories have been increasingly integrated into experimental design and analysis; for example, Schultz et al. (1997) suggested that the phasic responses of midbrain dopamine neurons recorded from primates behaving for rewards resemble the prediction error used in TD learning for choosing advantageous actions, and Barraclough et al. (2004) suggested that the predicted mechanism of the prefrontal cortex, which receives projection from dopamine neurons, plays important

roles in solving complex decision-making problems. Such computational approaches make possible the study of the neural substrates of inherently subjective quantities that the models purport to quantify, such as the "value" or "utility" of an action, meaning the degree of reward that a subject expects to receive for executing that action (Daw & Doya, 2006). The theory of reinforcement learning, therefore, plays an important role as the theoretical basis of reward-based learning in diverse fields of research.

1.3 Partially observable problems

In an MDP formulation, an arbitrary state in an environment is observable for learning agents; the environmental state, which is an input to the agents' policy, is assumed to be available for computing an optimal policy. In general decision-planning problems, however, all information for making an optimal decision is not necessarily given to the agent; the agent cannot directly access the environmental state, and can obtain only observations which contain partial information about the state. For example, in localization problems in which an autonomous mobile robot localizes its actual location in various environments, the robot can obtain information around the current location just using input devices such as cameras placed at various points in the environment, but it is infeasible to prepare such external devices over its large moving domain in advance. The robot, therefore, should identify its location based only on the information obtained from mobile devices on the robot itself, but such information is often partial, since a mobile camera can provide not an absolute position in the environment but a limited scene around the robot, or such a sensory input may be unreliable due to background noise. If there is much uncertainty in the current location due to partial observability, it is difficult to determine an optimal moving direction for achieving the original goal. The problem, obtaining an optimal decision-making policy under the situation where only partial information about the real state of the environment is available to the agent, is called a partially observable problem, and many RL researchers in the machine learning community are now devoting much attention to this problematic field.

If the policy is determined only from an immediate observation based on conventional RL algorithms for MDP environments, without estimating an internal state explicitly or implicitly, it does not usually achieve the global policy (Chrisman & Littman, 1993; Singh, Jaakkola, & Jordan, 1994; Littman, 1994b), because the observation does not satisfy the Markov property; for example, in the same situation as the autonomous robot above, optimal control can no longer be expected using only

the immediate sensory input, which contains much uncertainty. One way to overcome this problem is to resolve the uncertainty by estimating the real state of the underlying system based on an assumed probabilistic model for an observation process. Such a framework which contains the estimation process of the unobservable state and learning the policy is called a partially observable Markov decision process (POMDP) (Astrom, 1965; Smallwood & Sondik, 1973; Sondik, 1978; Monahan, 1982; Lovejoy, 1991; White, 1991; Kaelbling, Littman, & Cassandra, 1998), and many algorithms based on this framework have been proposed (Peshkin, Meuleau, & Kaelbling, 1999; Loch & Singh, 1998; McCallum, 1993; Whitehead & Lin, 1995; Nikovski & Nourbakhsh, 2000). In particular, various approaches in which the posterior distribution of unobservable states, now often called a belief state, is updated by the Bayes rule have been studied (Littman, Cassandra, & Kaelbling, 1995; Brafman, 1997; Boutilier & Poole, 1996; Pineau, Gordon, & Thrun, 2003; Theocharous & Mahadevan, 2002; Thrun, 2000; Chrisman, 1992; Yoshimoto, Ishii, & Sato, 2003).

Since the belief state is a sufficient statistic which summarizes all information necessary for optimal action selection, this formulation gives rise to the standard approach to solving POMDPs, but several long-standing issues still remain. First, learning of the value function over the belief space is difficult even with an effective approximation (Hauskrecht, 2000); optimization of the value function requires heavy computation, because its input is a continuous probability distribution and usually has high dimensionality, meaning that the problem is transformed into an equivalent, completely observable MDP with a continuous state space consisting of all possible belief states. Second, an environmental model and the number of real states are required for explicitly estimating unobservable states; environmental information is used to calculate the belief state. These difficulties thus make dealing with various POMDP problems infeasible. Solving partially observable problems, therefore, has been regarded as a crucially difficult domain in this learning and decision-making research field.

Humans, on the other hand, are familiar with solving large-scale and complex partially observable problems; for example, in conversation, even if one cannot observe what opponents think or intend in their thoughts and their feelings or backgrounds, but can perceive only tone, expression, appearance and context of the conversation, it is possible to communicate with others smoothly by estimating such hidden information based on partial observations. That is, we cope with the computational difficulties ingeniously. How this mechanism operates in the human brain, however, is still almost unknown (Yoshida & Ishii, 2006); it is important to consider a possible hypothesis based on a reasonable computational model.

1.4 Contents of dissertation

This dissertation presents research results about learning and decision-planning in partially observable environments. As studies for controlling autonomous agents, it deals in particular with the card game "Hearts" (Perkins, 1998; Pfahringer, Kaindl, Kramer, & Furnkranz, 1999; Sturtevant & White, 2006), a four-player, competitive and partially observable game, and presents automatic strategy-acquisition schemes for this game based on the framework of model-based RL. Many card games have general properties: they are large-scale, multi-agent, and partially observable. Most cannot be played alone (that is, they are in a multi-agent setting), and cards in another player's hand or undealt cards are unobservable to each player (meaning a partially observable situation). Card games have, consequently, been studied as well-defined test-beds for strategy acquisition problems in the real world. In dealing with partially observable games (Chang, Ho, & Kaelbling, 2003; Dahl, 2002; Nair, Marsella, Tambe, Pynadath, & Yokoo, 2003; Hansen, Bernstein, & Zilberstein, 2004; Emery-Montemerlo, Gordon, Schneider, & Thrun, 2004), three difficulties arise: the first is to estimate the distribution of unobservable states based on the history of observations; the second is to predict the opponent agents' actions based on any acquired models; and the third is to cope with the computational intractability stemming from the huge state space. A card game with partial observability, therefore, is a challenging target to study. Computer simulation results show that our methods are effective in solving such a difficult partially observable multi-agent problem.

As studies for modeling human behavior, this dissertation demonstrates the performance of probabilistic models for human decision-making process in a partially observable environment. Since theoretical POMDP studies have indicated that optimal decision-planning under partially observable situations has computational difficulties, it can be speculated that the human brain avoids the difficulties with some effective approximation for estimating unobservable states and making decisions. To reveal how the decision-planning mechanism with the estimation process operates in the brain, these studies explore the possibility that our model-based RL framework presented above is implemented in the human brain. Behavioral experiments using a virtual maze were carried out to thirteen normal subjects. Because the subjects could not observe their real position in the maze but obtain partial observation, it is important for solving the problem effectively to estimate the unobservable real positions based on the observation sequence. Analysis results show that our model can reproduce the subjects' behaviors with high accuracy, and indicate that humans estimate

unobservable states based on the framework of the incremental Bayes estimation.

This dissertation is organized as follows: Chapter 2 provides a brief review of relevant background about POMDPs and previous works on various approaches. Chapters 3 and 4 describe two types of the model-based RL method. Chapter 5 presents probabilistic models for describing the human decision-making process, and Chapter 6 gives a summary of the dissertation with a brief discussion and an outline of future works.

Chapter 2

Preliminary

This chapter introduces the basic framework of partially observable Markov decision processes (POMDPs), and presents a brief summary of previous studies in this research area. POMDPs were first introduced to the control theory and operations research communities (Astrom, 1965; Smallwood & Sondik, 1973; Sondik, 1978; Monahan, 1982; Lovejoy, 1991; White, 1991) as a framework to make optimal decisions in stochastic dynamical systems with hidden variables. This framework was later considered by the artificial intelligence community as a principled approach to planning under uncertainty (Kaelbling et al., 1998). The last part of this chapter reveals the relationship between our studies and previous POMDP research.

2.1 Basic framework

This section establishes the basic terminology and essential concepts, and then introduces exact solutions exhibited by the value iteration algorithm, which have been proposed for POMDP planning problems.

2.1.1 Constituents

A standard POMDP framework consists of the following six constituents.

- **State space $\mathcal{S}$**
  The environment is defined by a set $\mathcal{S} = \{s^1, \cdots, s^{|\mathcal{S}|}\}$ of distinct states. A state $s \in \mathcal{S}$ should retain all relevant information about the environment compactly for optimal action selection. Although the number of states $|\mathcal{S}|$ may be finite, countably infinite or continuous, we focus on discrete state spaces with a finite number of states throughout this dissertation.
- **Action space $\mathcal{A}$**
  Agents existing in the environment have a set $\mathcal{A} = \{a^1, \cdots, a^{|\mathcal{A}|}\}$ of distinct

actions, and select an action $a \in \mathcal{A}$ according to the agents' policy π, in every time step. Although the number of actions $|\mathcal{A}|$ may be also finite, infinite or continuous, we assume that it is a discrete state space with a finite number of actions.

- **Observation space $\mathcal{O}$**
  The environment has a set $\mathcal{O} = \{o^1, \cdots, o^{|\mathcal{O}|}\}$ of distinct observations, and provides information about the underlying state to the agents, through an observation $o \in \mathcal{O}$ after every action a. The observation space $\mathcal{O}$, assumed to be discrete and finite, is the same as the state space $\mathcal{S}$ in MDPs, whereas it is not in POMDPs.

- **Transition function $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$**
  The state of the environment is influenced by the agents' actions, and changes with uncertainty according to the transition function T: such a stochastic dynamics is represented as the probability distribution $P(s_{t+1}|s_t, a_t)$, where t indicates a discrete time step. Note that this transition function satisfies the Markov property in which the next state s_{t+1} depends not on past states and actions but only on the pair of a current state s_t and action a_t.

- **Observation function $O : \mathcal{S} \times \mathcal{A} \times \mathcal{O} \to [0, 1]$**
  The agents perceive an observation with uncertainty according to the observation function O after taking an action in a certain state: such a stochastic process is modeled by the probability distribution $P(o_{t+1}|s_{t+1}, a_t)$.

- **Reward function $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$**
  A reward function R defines the goal of a learning problem, which maps the tuple of a state, an action and a next state into a numerical value $r \in \mathbb{R}$, indicating the intrinsic desirability of that state.

The POMDP framework provides a general learning and decision-planning model for acting optimally in partially observable domains. It often assumes a complete and correct world model, with stochastic state transitions, imperfect state tracking and a reward structure, but can also include learning of the model components (Shani, 2004).

These constituents, represented by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, T, O, R \rangle$, define the probabilistic environment. The POMDP framework assumes that the environmental dynamics T, O and R are stationary for the agent; these dynamics do not change over time. Because we assume problems in which there are multiple learning agents in a common environment, this stationary assumption cannot be applied due to the inter-

action among the agents. In this dissertation, however, the environment is regarded as stationary for the learning agent, under the assumption that there is a single learning agent, and our methods are then directly applied to a multi-agent problem; in the computer experiments described in Chapters 3.3 and 4.3, we show several acceptable results in dynamic environments.

2.1.2 Belief state and value functions

In POMDP problems, the objective of each agent is to acquire an optimal policy π^* maximizing an expected future reward; the objective function is defined as the expectation of the sum of the immediate reward over time:

$$E^{\pi}\left[\sum_{t=0}^{\infty}\gamma^{t}r_{t}\right], \tag{2.1}$$

where $0 \leq \gamma < 1$ is a discount factor which bounds the sum in a finite value, and E^{π} is the expectation symbol under the assumption that actions are selected according to the policy π. Although we focus in this chapter on the discounted infinite-horizon model, its formulation can be applied to the finite-horizon case by modifying the objective function; equation (2.1) can be written in this case as

$$E^{\pi}\left[\sum_{t=0}^{T}r_{t}\right], \tag{2.2}$$

where T denotes termination time. In later chapters, we use the finite-horizon model for our application problem.

The state s_t is not observable for each agent, and only the observation o_t, which contains partial information about the state, is available. If the policy is determined only from an immediate observation, without estimating a hidden state explicitly or implicitly, it does not usually converge to a global optimal policy (Singh et al., 1994; Kaelbling, Littman, & Moore, 1996), because the observation does not satisfy the Markov property. One way to overcome this problem is to use the history of the agent's experience $H_t = \{(o_t, -), (o_{t-1}, a_{t-1}), \cdots, (o_1, a_1)\}$. Because it is difficult, however, to maintain such a naive history with a limited memory capacity, a belief state $b(s_t)$ is often used (Astrom, 1965). Since the belief state summarizes the history as a probability distribution over $\mathcal{S}$, it is a sufficient statistic with the Markov property, which is updated upon every new observation according to the incremental Bayes

formula:

$$b(s_{t+1}) \equiv P(s_{t+1}|H_{t+1}) = \frac{P(o_{t+1}|s_{t+1}, a_t) \sum_{s_t \in \mathcal{S}} P(s_{t+1}|s_t, a_t) P(s_t|H_t)}{\sum_{s_{t+1} \in \mathcal{S}} P(o_{t+1}|s_{t+1}, a_t) \sum_{s_t \in \mathcal{S}} P(s_{t+1}|s_t, a_t) P(s_t|H_t)} \tag{2.3}$$

Note that the next belief state $b(s_{t+1})$ can be obtained from the current one $b(s_t)$ recursively. The optimal policy that maps a belief state into an action becomes a solution of a continuous space belief-state MDP.

The tiger problem

To demonstrate the concept of a belief state, we use a simple partially observable problem, "The tiger problem," introduced by Kaelbling et al. (1998). An agent is standing in front of two closed doors; behind one of the doors is a treasure box (reward), and behind the other is a tiger (penalty). The agent cannot perceive the real position of the tiger but can listen to sounds from behind the closed doors with incomplete accuracy; there is a chance that the agent hears the tiger's roar from the wrong door. $\mathcal{A} = \{\text{LEFT, RIGHT, LISTEN}\}$ are executable actions for the agent, and $\mathcal{O} = \{\text{LEFT, RIGHT}\}$ are observations obtained by taking the LISTEN action. The rewards r_t for opening the door with the treasure box, for opening the door with the tiger and for taking the LISTEN action are $+10$, -100 and -1, respectively. The LISTEN action does not change the tiger's position, but the LEFT and RIGHT actions, that is, opening either door, are followed by a state transition with a uniform probability; in other words, the episode is terminated by opening a door, and then another episode is initialized randomly. The probability that the agent can obtain the correct observation is $p = 0.85$.

Assuming that the tiger is equally likely to be behind either door, the initial belief state of the agent is $[0.5, 0.5]$, where the first dimension of the belief vector represents the probability of the tiger being behind the left door. If the agent takes the LISTEN action and observes LEFT, then the belief state is updated as $[0.85, 0.15]$. If it takes the LISTEN action and observes LEFT again, then the belief state is updated as $[0.97, 0.03]$; the agent confirms its conviction that the tiger is behind the left door by obtaining LEFT observations for the second consecutive time.

A belief-state MDP for the infinite-horizon discounted case satisfies the standard fixed-point equation, called the Bellman equation:

$$V^*(b_t) = \max_{a_t \in \mathcal{A}} Q^*(b_t, a_t), \tag{2.4a}$$

$$Q^*(b_t, a_t) = \rho(b_t, a_t) + \gamma \sum_{b_t \in \mathcal{B}} P(b_{t+1}|b_t, a_t) V^*(b_{t+1}), \tag{2.4b}$$

where b_t and b_{t+1} are abbreviations of the belief state $b(s_t)$ and $b(s_{t+1})$, respectively, and $\mathcal{B}$ denotes the belief space. $V^*(b_t)$ is the optimal value function of the belief state b_t, which represents the maximum of the expected cumulative reward in equation (2.1), and $Q^*(b_t, a_t)$ is the optimal action-value function, often called the optimal Q-function, which represents the maximum of the expected cumulative reward starting from a belief state b_t, taking the action a_t, and thereafter acting optimally. $\rho(b_t, a_t)$ is the expected one-step reward:

$$\rho(b_t, a_t) = \sum_{s_t \in \mathcal{S}} \rho(s_t, a_t) b(s_t) = \sum_{s_t \in \mathcal{S}} \sum_{s_{t+1} \in \mathcal{S}} R(s_t, a_t, s_{t+1}) P(s_{t+1}|s_t, a_t) b(s_t), \quad (2.5)$$

where $\rho(s_t, a_t)$ denotes an immediate reward expected with respect to the transition probability: $\rho(s_t, a_t) = \sum_{s_{t+1} \in \mathcal{S}} R(s_t, a_t, s_{t+1}) P(s_{t+1}|s_t, a_t)$. Since the next belief state b_{t+1} is calculated given the current belief state b_t, the new observation o_{t+1} and the action a_t according to equation (2.3), the probability $P(b_{t+1}|b_t, a_t, o_{t+1})$ is deterministic:

$$P(b_{t+1}|b_t, a_t, o_{t+1}) = \begin{cases} 1 & \text{for a certain state } b_{t+1}, \\ 0 & \text{otherwise,} \end{cases} \quad (2.6)$$

and the upper case in equation (2.6) is here defined as the state estimator function $\tau(b_t, a_t, o_{t+1})$. With this function, the optimal Q-function in equation (2.4) can be modified by taking a summation over all possible states and observations:

$$Q^*(b_t, a_t) = \rho(b_t, a_t) + \gamma \sum_{o_{t+1} \in \mathcal{O}} \sum_{s_t \in \mathcal{S}} P(o_{t+1}|s_t, a_t) b(s_t) V^*(\tau(b_t, a_t, o_{t+1})). \quad (2.7)$$

The optimal policy $\pi^* : \mathcal{B} \rightarrow \mathcal{A}$ is defined as the value-maximizing action:

$$\pi^*(b_t) = \underset{a_t \in \mathcal{A}}{\operatorname{argmax}}\ Q^*(b_t, a_t). \quad (2.8)$$

The objective of each agent is to acquire an optimal policy by calculating the optimal Q-function explicitly or implicitly.

2.1.3 Value iteration

The difficulty of obtaining an optimal policy by optimizing the value function arises mainly from the fact that an iterative optimization algorithm, such as value iteration (Bellman, 1957a), must be repeated infinitely over the continuous belief space. Smallwood and Sondik (1973), fortunately, found the piecewise-linear and convex properties of value functions: under the condition that an initial value function V_0 is piecewise

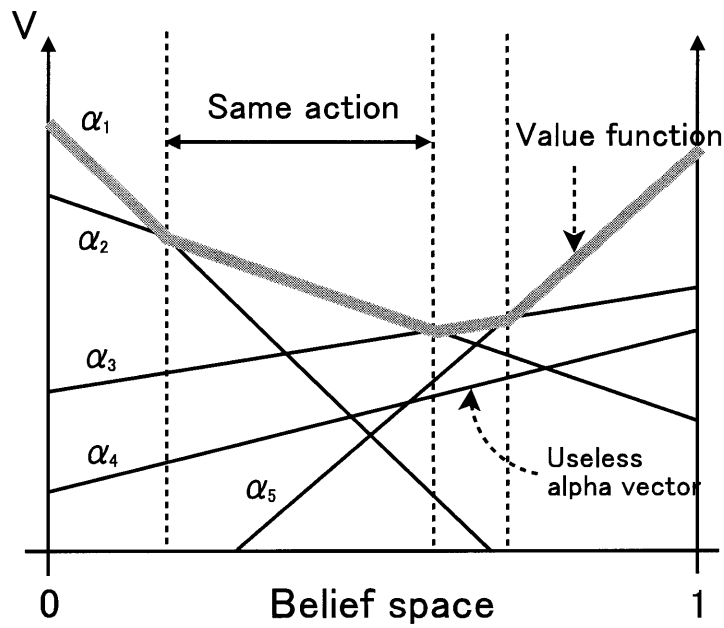

Fig. 2.1: Convex and piecewise-linear representation of a value function for a continuous belief state with $|\mathcal{S}| = 2$. The value function can be represented by the upper surface of the α-vectors associated with an action, defining the best immediate policy assuming optimal behavior for the following $(i-1)$ steps.

linear and convex, the i-th value function obtained after the i-th number of update steps is also a finite, piecewise-linear and convex function,

$$V_i(b_t) = \max_{\alpha_i \in \Gamma_i} \sum_{s_t \in \mathcal{S}} b(s_t)\alpha_i(s_t), \tag{2.9}$$

where α_i is a vector of size $|\mathcal{S}|$ and Γ_i is a finite set of vectors α_i. Figure 2.1 shows a graphical representation of these convex and piecewise-linear properties, for an example in which the number of real states $|\mathcal{S}|$ is two. Note that each α-vector is associated with an action which defines the immediate policy assuming optimal behavior for the following $(i-1)$ steps. Based on the properties, the i-th value function with the i-horizon finite solution set Γ_i can be computed from $(i-1)$-step vectors $\alpha_{i-1} \in \Gamma_{i-1}$:

$$V_i(b_t) = \max_{a_t \in \mathcal{A}} \left[\rho(b_t, a_t) + \gamma \sum_{o_{t+1} \in \mathcal{O}} \max_{\alpha_{i-1} \in \Gamma_{i-1}} \Psi(\alpha_{i-1}) \right], \tag{2.10a}$$

$$\Psi(\alpha_{i-1}) = \sum_{s_t \in \mathcal{S}} \sum_{s_{t+1} \in \mathcal{S}} P(s_{t+1}|s_t, a_t) P(o_{t+1}|s_t, a_t) \alpha_{i-1}(s_{t+1}) b(s_t). \tag{2.10b}$$

This computation process leads to a piecewise-linear and convex value function V_i represented by a finite set of linear functions $\alpha_i \in \Gamma_i$. Since the properties enable

us to obtain the next value function in a finite number of iterations, because a finite number of α-vectors is enumerated in every single update, the optimal value function in equation (2.4) or its approximation can be obtained using dynamic programming (DP) techniques such as the value iteration. Note, however, that the piecewise-linear and convex properties do not imply piecewise linearity of the optimal solution V^*.

Solving POMDP problems based on DP algorithms exhibits two kinds of computational difficulty. First, it is difficult to calculate the next belief state according to the Bayes rule in equation (2.3). Since the calculation includes two summations, the computational complexity of POMDP algorithms grows in proportion to $|\mathcal{S}|^2$. Second, expensive computation is required to update the value function over the continuous belief state. Since the set Γ_i may include useless α-vectors which are not constituents of a parsimonious set Γ_i^* (Zhang, 1995) consisting only of dominant vectors, such vectors should be removed from a redundant set Γ_i. In the simplest approach to keep the set parsimonious, called the enumeration algorithm (Sondik, 1971; Monahan, 1982), the pruning process is performed at every single update by the following three steps: the first step is to enumerate all possible vectors $\forall \alpha_{i-1} \in \Gamma_{i-1}$; the second is to evaluate their usefulness for representing the value function V_i; and the last is to select only useful vectors and construct a parsimonious set Γ_i^*. The computational complexity of a single update of this algorithm is $|\mathcal{A}||\Gamma_{i-1}|^{|\mathcal{O}|}$, because there are $|\mathcal{A}|$ distinct alternatives to take the action and permutations of each vector α_{i-1} of size $|\mathcal{O}|$. In the worst case, therefore, the computational cost to obtain the exact solution according to the belief-state MDP formulation can be evaluated as $O(|\mathcal{S}|^2|\mathcal{A}||\Gamma_{i-1}|^{|\mathcal{O}|})$; even for simple POMDPs, the problem of finding the optimal policy for the finite-horizon case is PSPACE-hard (Papadimitrious & Tsitsiklis, 1987) and for the discounted infinite-horizon case may not even be computable (Madani, Hanks, & Condon, 1999), due to the exponential growth with the increase in the number of observations.

Due to the heavy computational cost, the cause of the essential difficulty for solving POMDPs, it is important to improve efficiency by keeping the size of the linear function set as small as possible at every iteration step. There are two approaches for computing useful linear functions: the first is the extension of the enumeration algorithm (Sondik, 1971; Monahan, 1982) described above; and the second is to perform the value update on a single belief state (Sondik, 1971; Smallwood & Sondik, 1973).

The recent principal insight in the extensions of the first approach is that the pruning process of useless α-vectors can be interleaved directly with the enumeration process; the dominated vectors are early pruned in each set of partially constructed linear functions (Zhang & Liu, 1997; Cassandra, Littman, & Zhang, 1997; Zhang & Lee, 1998). The resulting value function is the same, but the algorithm is more

efficient because it discards useless vectors before constructing the full set of linear functions. Algorithms based on the second approach use the fact that each belief state has a region over the belief space in which its α-vector is maximal (Cheng, 1988; Pineau et al., 2003); the corresponding belief state of each region is called a witness point (Kaelbling et al., 1998). Such algorithms search in each value iteration step the complete belief simplex for a minimal set of witness points which generate the parsimonious set of α-vectors for the next horizon value function. These solutions typically require linear programming for finding the set of witness points, and are therefore costly in high dimensions. Zhang and Zhang (2001) argued that value iteration still converges to the optimal value function, if exact value iteration steps are interleaved with approximate value iteration steps in which the new value function is an upper bound to the previously computed value function. This results in a speedup of the total algorithm; however, linear programming is also required to ensure that the new value function is an upper bound to the previous one over the complete belief simplex.

Although various sophisticated algorithms exhibited by exact value iteration have been proposed as described above, the serious computational difficulty for obtaining the exact solution is inevitable even with the two types of devices, and this makes solving POMDP problems with only a few dozen states impractical; some effective approximations are essentially required. Practical solutions for solving POMDPs can be classified here into (I) value-function and belief-state approximation and (II) policy approximation. Existing solutions in case (I) obtain an approximate policy by optimizing the value function over the belief space with the explicit computation of the belief state, by using some heuristic approximations. Most algorithms in this case assume that the environmental model is known for the agents, after which the value iteration technique in equation (2.10) is applied. Solutions in case (II), on the other hand, do not calculate the belief state explicitly but resolve the partial observability by using certain kinds of memories, and then a policy is directly optimized. We introduce the algorithms of case (I) in Section 2.2 before describing those of case (II) in Section 2.3, and reveal the relationship of our studies to previous work in Section 2.4.

2.2 Value-function and belief-state approximation

To estimate the unobservable states, this class of algorithms attempts to optimize the value function over the belief space with the explicit belief update. Such algorithms can be further classified into: (I-a) solutions with value-function approxima-

tion; (I-b) solutions with belief-space approximation; and (I-c) solutions with both value-function and belief-space approximation. To avoid the computational difficulty, research in the POMDP area has focused on various heuristic approximation methods; practical algorithms use approximation methods for either the value function or the belief space or both, which are described in Section 2.2.1, Section 2.2.2, and Section 2.2.3, respectively. This class of algorithms is for the agents to require the environmental model; algorithms introduced in the following section assume the correct world model, unless otherwise stated.

2.2.1 Value-function approximation

The main idea used by this class of algorithms is to approximate the optimal value function $V : \mathcal{B} \rightarrow \mathbb{R}$ with a function $\hat{V} : \mathcal{B} \rightarrow \mathbb{R}$ defined over the complete belief state space $\mathcal{B}$. There have been many attempts to compute approximate value functions heuristically. Some popular algorithms are introduced here: completely observable MDP approximations; unobservable MDP approximations; curve-fitting approximations; and EM-based algorithms.

Completely observable MDP approximations

The simplest way to obtain a policy in partially observable environments is to learn the value function over the state space $\mathcal{S}$ under the assumption that the environment is completely observable; the approximation methods based on this assumption are called completely observable MDP approximations. According to the most likely state (MLS) heuristic method (Nourbakhsh, Powers, & Birchfield, 1995; Ishii, Yoshida, & Yoshimoto, 2002), the agents select an action with the maximum a posteriori (MAP) estimation:

$$\begin{aligned} \pi(b_t) &= \operatorname*{argmax}_{a_t \in \mathcal{A}} Q(s_t^*, a_t), \\ s_t^* &= \operatorname*{argmax}_{s_t \in \mathcal{S}} b(s_t). \end{aligned}$$

Since this approach ignores the possibility that the agent is in other states, the performance of the policy may deteriorate when the MAP estimation is incorrect. Other approaches with this type of approximation, therefore, calculate the expectation of the state value with respect to the current belief state:

$$\begin{aligned} \hat{V}(b_t) &= \sum_{s_t \in \mathcal{S}} b(s_t) V(s_t), \\ V(s_t) &= \max_{a_t \in \mathcal{A}} \left[\rho(s_t, a_t) + \gamma \sum_{s_{t+1} \in \mathcal{S}} P(s_{t+1} | s_t, a_t) V(s_{t+1}) \right], \end{aligned}$$

called MDP approximation (Lovejoy, 1993), or the expectation of the state-action value with respect to the current belief state:

$$\begin{aligned} \hat{V}(b_t) &= \max_{a_t \in \mathcal{A}} \sum_{s_t \in \mathcal{S}} b(s_t) Q(s_t, a_t), \\ Q(s_t, a_t) &= \rho(s_t, a_t) + \gamma \sum_{s_{t+1} \in \mathcal{S}} P(s_{t+1}|s_t, a_t) V(s_{t+1}), \end{aligned}$$

called QMDP approximation (Littman et al., 1995).

Although the agents with this class of approximation exhibit good performance, even in such a large-scale problem, they do not take any action to resolve the partial observability because no uncertainty is considered on subsequent future steps due to the completely observable assumption. Since it is important to take exploratory actions when the variance of the belief state is large, the performance based on these heuristics may deteriorate. This problem, called certainty equivalence in control theory (Doucet, Freitas, & Gordon, 2001) or dual control in adaptive control literature (Cassandra, 1998), can be solved by the following two approaches: the first is to perform a deep look-ahead prediction (Murphy, 2000); and the second is to use the entropy of the belief state (Cassandra, 1998; Ishii et al., 2002). Cassandra (1998) demonstrated a switching algorithm between an exploration and exploitative policies by calculating the entropy of the belief state. Ishii et al. (2002) proposed a control method of exploitation-exploration meta-parameters in a model-based RL formulation for switching between these trade-off policies, according to the entropy of the belief state, with an exploration bonus. This latter idea is promising not only for making the agents effective at control but also for considering a model of the human brain.

Hauskrecht (2000) proposed a similar method based on the completely observable MDP approximation, called fast-informed bound approximation. A different point from other approximations is that it partly incorporates observation effects into the estimation of action values by calculating expected observation:

$$\begin{aligned} \hat{V}(b_t) &= \max_{a_t \in \mathcal{A}} \sum_{s_t \in \mathcal{S}} b(s_t) Q(s_t, a_t), \\ Q(s_t, a_t) &= \rho(s_t, a_t) + \gamma \sum_{o_{t+1} \in \mathcal{O}} \max_{a_{t+1} \in \mathcal{A}} \sum_{s_{t+1} \in \mathcal{S}} P(s_{t+1}, o_{t+1}|s_t, a_t) Q(s_{t+1}, a_{t+1}). \end{aligned}$$

Since this method changes the order of summations (it calculates the summation over the observation before calculating the summation over the current state), the action selection depends on the expected observation probabilities. Although the computational cost of the fast-informed bound method is more expensive than that of other

MDP approximation methods due to the computation over the observation space, its control quality is better than that of others; Hauskrecht (2000) showed promising experimental results in a reasonably-sized problem. This method can be interpreted as a hybrid model between the exact solution and the MDP approximation.

Unobservable MDP approximations

The completely observable MDP approximation assumes complete observability of underlying states to obtain simpler value updates. The other extreme is to ignore all observations available to the agents; an MDP with no observation is called unobservable MDP. Hauskrecht (2000) proposed a value-function approximation method with the idea of the unobservable MDP. In this method, the value function is modified to

$$\hat{V}_i(b_t) = \max_{a_t \in \mathcal{A}} \left\{ \rho(b_t, a_t) + \gamma \max_{a_{t+1} \in \mathcal{A}} \sum_{s_t \in \mathcal{S}} \sum_{s_{t+1} \in \mathcal{S}} P(s_{t+1}|s_t, a_t) b(s_t) Q(s_{t+1}, a_{t+1}) \right\}.$$

The resulting $\hat{V}_i$ retains the piecewise-linear and convex properties. Note that the summation over the observation space is removed by the unobservable MDP assumption. Although the computational cost of the exact value update increases exponentially as the number of the observation increases, the number of α-vectors consisting of the approximate value function $\hat{V}_i$ is at most $|\mathcal{A}||\Gamma_{i-1}|$ and the complexity of the approximate update grows linearly. The computation time for a single update can be evaluated as $O(|S|^2|\mathcal{A}||\Gamma_{i-1}|)$. Since the problem of finding an optimal solution for the unobservable MDP remains intractable, however, this approximation technique is usually not so useful (Hauskrecht, 2000).

Curve-fitting approximations

The exhaustive value update in equation (2.10) may be inefficient, because it is performed over the entire belief space under the implicit assumption that each belief state is regarded as equal. To solve this problem, each value update should be performed in proportion to the frequency of the encountered belief state - the idea of simulation-based algorithms in which the value functions are optimized from real experiences obtained by the interaction with the environment is beneficial. Geffner and Bonet (1998) applied this idea to learning in finite-horizon POMDP problems, by monitoring the current belief state and performing the value iteration for the current point in belief state space.

The idea of this on-line approach can be applied to learning a parametric model for the value function; a differentiable continuous function $f(b_t; \theta)$ with parameter θ approximates the values of the corresponding belief state, instead of remembering all belief-value pairs, so that the parametric function can fit the real value function in

terms of error-minimizing. Littman et al. (1995) proposed a simple approach in which the Q-function represented by linear functions $Q(b_t, a_t) = \phi(\theta) \cdot b_t$ is optimized so as to minimize a least-squares error using stochastic gradient ascent. This approach is similar to the LSTD algorithm in the RL research area for MDP environments (Bradtke & Barto, 1997). Parr and Russell (1995) proposed a similar algorithm, called SPOVA, using simulation to estimate gradients for updating the approximate function:

$$\hat{V}(b_t) = \left[\sum_{\alpha \in \Gamma} (b_t \cdot \alpha)^k\right]^{\frac{1}{k}}, \tag{2.15}$$

where k is a smoothing parameter. Artificial neural networks can also be used to approximate the value function with a factored belief representation (Rodriguez, Parr, & Koller, 1999), described in Section 2.2.3.

EM-based algorithms

Although algorithms based on the belief-state MDP formulation require the environmental model for the explicit computation of the belief update in equation (2.3), it may be impractical in real world settings. Problems in which the environmental model is unknown for the agent should be solved with system identification. A natural approach is to use a hidden Markov model (HMM), which is a well-known stochastic model for learning discrete-state dynamics of the environment with hidden state variables. Chrisman (1992) examined the simple approach using an HMM, called utile distinction memory algorithm, with the estimation of dimension required for problem-solving. Using the belief state calculated by the HMM, the value function is approximated by using the QMDP method (Littman et al., 1995). Yoshimoto et al. (2003) applied this idea to algorithms for solving problems with continuous spaces by using the Kalman filter. Although both methods gave good results in several experiments, learning models in maximizing an observation likelihood is not necessarily related to the policy performance. We presented, on the other hand, a similar HMM-based method using a likelihood function defined over a reward sequence (Fujita, Nakamura, & Ishii, 2006); the environmental model is estimated, given reward sequences obtained by interactions with the environment while acquiring its policy, such that rewards increase. The details of our approach are described in Section 2.3.1.

2.2.2 Belief-space approximation

There has been a class of algorithms which find lower dimensional representations of the belief space; they approximate the continuous belief space $\mathcal{B}$ with some tractable

space $\hat{\mathcal{B}}$. These approaches deal with POMDP problems by finding an appropriate sub-dimensional manifold of the belief space before learning the value function over the sub-dimensional space, $V : \hat{\mathcal{B}} \rightarrow \mathbb{R}$. Some popular algorithms are introduced here.

Factored representation

To find a tractable subspace, most studies exploit certain types of problem structure to make the belief state representation more compactly by using a Bayesian network. The Bayesian network can represent the state of the system in a set of random variables. A two time-slice dynamic Bayesian network (DBN) represents the system at two time steps, and the conditional dependencies between random variables from time t to time $(t + 1)$ can be represented by edges in a directed acyclic graph (Dean & Kanazawa, 1989). If the target problem has an available structure as the conditional dependencies, the belief state can, therefore, be represented as a distribution over a certain subset of the state variables.

Boyen and Koller (1998) developed an approximation scheme for representing the belief state by using DBN. They found that such DBN representation may cause error accumulation, which leads to the belief state approximation diverging. Projections of the DBN which produce independent sets of state variables result in converging belief states, which allows recovery from errors. These projections can be estimated to construct DBNs which perform well under a specified reward criterion. Poupart and Boutilier (2000) proposed the value-directed compression method, which considers a sequence of linear projections to find the smallest linear sub-dimensional manifold that is both consistent with the reward function and invariant with respect to transition and observation parameters. Since the algorithm finds a linear projection of the belief space, exact POMDP planning can be achieved directly in the projected space, and the full value function recovered through inverse projection.

Using principal component analysis

Roy and Gordon (2003) proposed an alternative algorithm which uses exponential-family principal component analysis to project high-dimensional beliefs to a low-dimensional non-linear subspace. This approach can generally achieve more compact compression than linear compression techniques, due to its non-linear projection. Although the grid-based algorithms may be suitable to planning over a non-linear subspace, such planning is much more complicated than that over the linear compression subspace.

2.2.3 Value-function and belief-space approximation

The last idea is to approximate both the value function $V : \mathcal{B} \rightarrow \mathbb{R}$ and the continuous belief space $\mathcal{B}$ with a function $\hat{V}$ and some tractable space $\hat{\mathcal{B}}$, respectively, and to learn the approximate value function $\hat{V} : \hat{\mathcal{B}} \rightarrow \mathbb{R}$ over the reduced space $\hat{\mathcal{B}}$. There have been several studies to achieve this.

Grid-based approximations

The value function can be approximated by values associated with each belief point $\hat{b}_t^i$ $(i = 1, \cdots, |G|)$ in a finite set $G = \{\hat{b}_t^1, \cdots, \hat{b}_t^{|G|}\}$, in which the points are distributed according to an arbitrary grid pattern over the continuous belief space. Given a set of grid points G, the value at each $\hat{b}_t \in G$ can be defined as

$$V(\hat{b}_t) = \max_{a_t \in \mathcal{A}} \left[\rho(\hat{b}_t, a_t) + \gamma \sum_{o_{t+1} \in \mathcal{O}} P(o_{t+1}|\hat{b}_t, a_t) V(\tau(\hat{b}_t, a_t, o_{t+1})) \right]. \tag{2.16}$$

If $\tau(\hat{b}_t, a_t, o_{t+1})$ is a constituent of the set G, then $V(\tau(\hat{b}_t, a_t, o_{t+1}))$ is defined by the value backups, otherwise $V(\tau(\hat{b}_t, a_t, o_{t+1}))$ is approximated using an interpolation rule such as

$$V(\tau(\hat{b}_t, a_t, o_{t+1})) = \sum_{i=1}^{|G|} \lambda(i) V(\hat{b}_t^i), \tag{2.17}$$

where $\lambda(i) \geq 0$ and $\sum_{i=1}^{|G|} \lambda(i) = 1$. The calculation for the interpolation function in equation (2.17) can be achieved by linear programming techniques, and this process produces a convex combination over grid points. The approximated value function $\hat{V}(b_t)$ can be obtained by performing the value iteration only for a finite set of grid points.

The most effective way of constructing a grid set is controversial. Lovejoy (1991) used a fixed-resolution regular grid over the entire belief state. Although value interpolations based on a triangulation concept can be calculated quickly by considering only neighboring grid points, the number of grid points grows exponentially with the dimensionality of the belief, and hence this fails to scale for large state spaces. Hauskrecht (1997) and Brafman (1997) used variable-resolution non-regular grids, which can increase resolution in areas of poor accuracy by adding new grid points based on a simulation technique. Although this reduces the number of grid points with similar accuracy, the computational cost for calculating the interpolation function increases, because grid points diverge due to biased placement. Zhou and Hansen

(2001) proposed a grid-based algorithm using variable-resolution regular grids which achieve both fast interpolation and increased resolution in the necessary areas. This algorithm, however, also fails to scale well for large state spaces due to the exponential growth of the number of grid points.

Sampling-based representation

Thrun (2000) demonstrated that the sequential Monte Carlo technique (Doucet et al., 2001) is beneficial for performing approximate belief state tracking in environments with continuous state spaces and action spaces. The algorithm has a trade-off between approximate accuracy and running time, depending on the number of samples; the approximate belief state converges to the true belief state as the number of samples increases, for an arbitrary model. The value function is approximated by a function approximator with a nearest neighbor method in which the distance metric is the Kullback-Leibler divergence between sets of belief states smoothed by a Gaussian kernel. Poupart et al. (2001) analyzed the approximation error of the value function incurred by using a sequential Monte Carlo technique, and developed an adaptive scheme for determining the number of samples according to the probability of selecting an optimal action.

Curve-fitting approximation with factored representation

Rodriguez et al. (1999) used the factored belief representation described in Section 2.2.2 for belief state tracking with various approximators of the value function. They demonstrated that the SPOVA method, described in Section 2.2.1, and an artificial neural network including the factored representation can work well in several problems.

2.3 Policy approximation

Most methods described so far in this chapter focus on estimating the value function over the continuous belief state before extracting a policy. An alternative approach is to search a policy space directly without optimizing the value function. Because it may be necessary for an optimal action selection over an infinite horizon to retain complete sequences of past experiences, the agents need an infinite memory to store information accumulated over infinite time, instead of calculating the belief state. An arbitrary policy, therefore, can be represented as an infinite policy graph. Since it is infeasible, however, to deal with such an infinite space due to computational limitation, we may then reduce the search to policies with a finite memory. With this approximation, every policy for an arbitrary POMDP problem with discrete finite

spaces can be represented as a finite policy graph, or finite state machine (FSM) model (Platzman, 1980). The idea of this model is the most general one in policy approximation approaches; other models can be interpreted as its special cases. We therefore first introduce algorithms using an FSM model. An advantage of this class of algorithms is that the agents require no environmental model; the algorithms introduced in the following section do not assume the correct world model, unless otherwise stated.

2.3.1 Finite state machine

A finite state machine (FSM) model consists of a finite set of memory states $\mathcal{M} = \{m^1, \cdots, m^{|\mathcal{M}|}\}$, a finite set of observations $\mathcal{O} = \{o^1, \cdots, o^{|\mathcal{O}|}\}$, a finite set of actions $\mathcal{A} = \{a^1, \cdots, a^{|\mathcal{A}|}\}$, a transition function $T : \mathcal{M} \times \mathcal{O} \rightarrow \mathcal{M}$ mapping a current memory state to the next memory state given an observation, and an output function $\pi : \mathcal{M} \rightarrow \mathcal{A}$ mapping a current memory state to the action. An FSM controller can be evaluated by constructing the value function. Platzman (1980) found that the value function for the FSM controller from a memory state m_t is linear, and the controller can be evaluated by solving a set of linear equations. After the evaluation, the FSM controller is improved by modifying the structure of the state machine. Hansen (1998) proposed an effective policy iteration algorithm using FSM controllers with the computation of the belief state under the assumption that the environmental model is known for the agent. Empirical results showed that this approach converges faster than exact value iteration in large-scale domains, because it often requires fewer iterations.

The FSM model can be generalized to a stochastic model: the transition function T is represented as a probability distribution $P(m_{t+1}|m_t, o_t)$, and the output function π is also represented as a probability $\pi(o_t, m_t, a_t) \equiv P(a_t|m_t, o_t)$. In the stochastic FSM model, the problem can be formulated as the maximization of the objective function in equation (2.1) with respect to the joint distribution:

$$
\begin{aligned}
&P^{\pi}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{o}, \boldsymbol{m}) \\
&= P(s_1)P(m_1)P(o_1)\prod_{t=1}^{T} P(o_{t+1}|s_t, a_t)P(s_{t+1}|s_t, a_t)P(m_{t+1}|m_t, o_t)\pi(o_t, m_t, a_t) \\
&= \left[P(s_1)P(o_1)\prod_{t=1}^{T} P(o_{t+1}|s_t, a_t)P(s_{t+1}|s_t, a_t)\right]\left[P(m_1)\prod_{t=1}^{T} P(m_{t+1}|m_t, o_t)\pi(o_t, m_t, a_t)\right] \\
&= W(\boldsymbol{s}, \boldsymbol{o})A^{\pi}(\boldsymbol{a}, \boldsymbol{o}, \boldsymbol{y}),
\end{aligned}
$$

where $\boldsymbol{s} = \{s_1, \ldots, s_{T+1}\}$, $\boldsymbol{m} = \{m_1, \ldots, m_{T+1}\}$, $\boldsymbol{a} = \{a_1, \ldots, a_T\}$, and

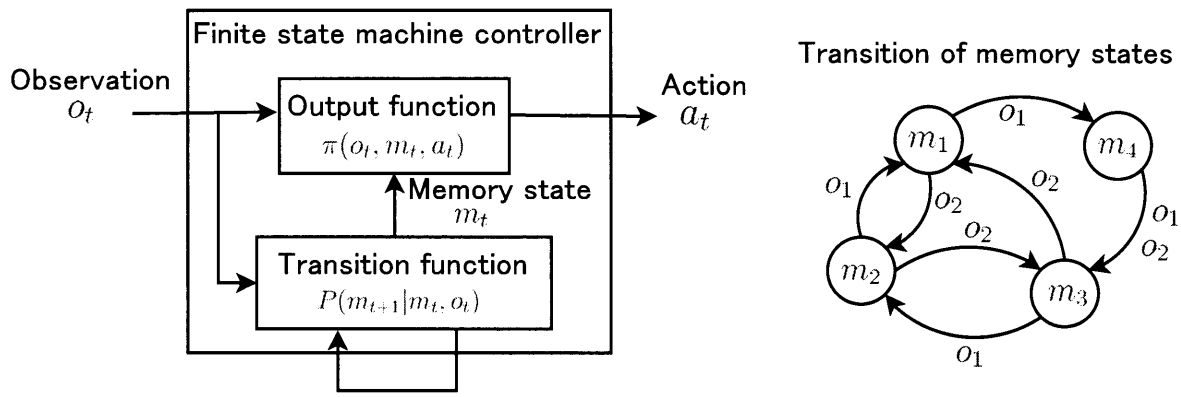

Fig. 2.2: Architecture of the finite state machine (FSM) controller.

left: The architecture of the FSM controller. The current memory state is provided to the output function with an immediate observation.

right: An example of a four-state FSM transition with two observations. Nodes and arcs represent memory states and transitions conditioned on observations, respectively. Since a current memory state is provided to the output function as input information, there are stochastic associations between each memory state and control action.

$\boldsymbol{o} = \{o_1, \ldots, o_T\}$ denote a sequence of real states, memory states, actions taken by the agent and observations obtained from the environment, respectively. $W(\boldsymbol{s}, \boldsymbol{o})$ is the environmental model, and $A^{\pi}(\boldsymbol{a}, \boldsymbol{o}, \boldsymbol{m})$ is the agent's policy represented by an FSM controller. Figure 2.2 illustrates the architecture of the FSM model. Given the fixed number of memory states $|\mathcal{M}|$, parameters of the stochastic FSM model, $P(m_{t+1}|m_t, o_t)$ and $\pi(o_t, m_t, a_t)$, can be directly learned by calculating a gradient ascent; Meuleau et al. (1999a, 1999b) proposed a direct learning algorithm of FSM controllers based on a gradient method. Aberdeen and Baxter (2002) used a similar approach; they called a memory state of the FSM controller an internal state, to emphasize a mechanism in which the agent modifies the dynamics of the memory system. Shelton (2001) applied an importance sampling technique to train an FSM controller in off-line manner. Poupart and Boutilier (2004) proposed a hybrid algorithm, called the bounded policy iteration, which combines insights from both exact policy search and gradient search.

An alternative approach for training FSM controllers is to use the maximum likelihood (ML) inference (Hartley, 1958; Dempster, Laird, & Rubin, 1977). We proposed a learning method according to the ML inference (Fujita et al., 2006) whose likelihood

function is defined as

$$P(\boldsymbol{r}|\boldsymbol{a},\boldsymbol{o};\boldsymbol{\theta},\boldsymbol{\sigma}) = \sum_{\boldsymbol{m}} P(\boldsymbol{m},\boldsymbol{r}|\boldsymbol{a},\boldsymbol{o};\boldsymbol{\theta},\boldsymbol{\sigma})$$
$$= \sum_{\boldsymbol{m}} P(m_1) \prod_{t=1}^{T} P(r_t|m_t,a_t) P(m_{t+1}|m_t,a_t,o_t), \tag{2.18}$$

where $\boldsymbol{r} = \{r_1, \cdots, r_T\}$ is a sequence of rewards. Our aim was to obtain the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\sigma}$, related to the memory state transition and observation processes, respectively, based on the maximum likelihood inference so that the likelihood function in equation (2.18) is maximized. More concretely, $\boldsymbol{\theta} \equiv \{\theta_{ij} = P(m_{t+1} = j|m_t = i, a_t, o_t)|i = 1,\ldots,|\mathcal{Y}|, j = 1,\ldots,|\mathcal{Y}|\}, \boldsymbol{\sigma} \equiv \{\sigma_{ik} = P(r_t = k|m_t = i, a_t)|i = 1,\ldots,|\mathcal{Y}|, k = 1,\ldots,|\mathcal{R}|\}$. Since all the agent must do to solve problems is to estimate the Markov chain over a statistic variable related to rewards, we assumed that the rewards are dependent not on the pair of a real state and an action but on the pair of a memory state and an action. The agent, therefore, estimated the dynamics of the memory state $P(m_{t+1}|m_t, a_t, o_t)$ based on the ML inference, given reward sequences obtained by interactions with the environment; the reward sequence is regarded as the series of observations, and the expectation-maximization (EM) inference was carried out to estimate the dynamics. By assuming that all variables take discrete values, we used the multinomial model; the EM inference was then performed by the input/output hidden Markov model (HMM) (Bengio & Frasconi, 1996). After the estimation process for the dynamics of the memory state, the agent learned its policy $P(a_t|m_t)$ based on a conventional RL algorithm for MDP environments.

To evaluate the performance of our method, we applied it to a partially observable problem, "The Tiger Problem", described in Section 2.1.2. In this experiment, the agent estimated a memory state model using the data generated from 500 episodes in an environment where an action was selected randomly at each time step. To improve the policy, the REINFORCE algorithm (Williams, 1992), which is a type of policy-gradient-based RL algorithm, was used. We restricted the maximum number of actions to ten so that the agent could not take the LISTEN action for all time, and assumed that the number of internal states was five: $|\mathcal{M}| = 5$. Figure 2.3 represents a memory state model estimated by the agent. Arrows represent transitions with a maximum probability when the agent selects the LISTEN action; each arrow is pointing toward $m_{t+1} = \text{argmax}_{m_{t+1}} P(m_{t+1}|m_t, a_t = \text{LISTEN}, o_t)$. LEFT and RIGHT in the figure represent observations obtained by taking the LISTEN action. Table 2.1 shows action selection probabilities acquired by the RL algorithm for the estimated model. Note that the performance of the agent has two factors: validity of the internal state model and performance of the policy acquired for the model.

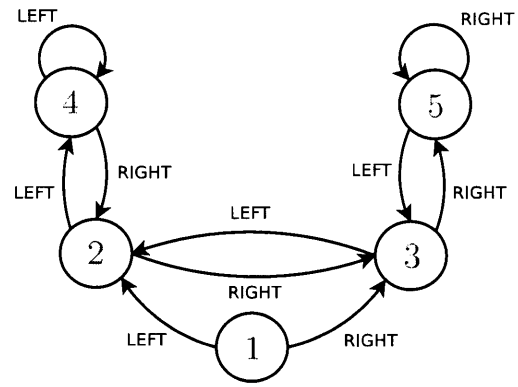

Fig. 2.3: The state transition when the agent takes the LISTEN action, $m_{t+1} = \text{argmax}_{m_{t+1}} P(m_{t+1}|m_t, a_t = \text{LISTEN}, o_t)$, in the Tiger Problem with $p = 0.85$.

m_t \ a_t	LISTEN	LEFT	RIGHT
1	0.999992	0.000003	0.000005
2	0.997126	0.000003	0.002871
3	0.999746	0.000000	0.000254
4	0.009600	0.000000	0.990400
5	0.003291	0.996709	0.000000

Table. 2.1: Action selection probability acquired by the agent in the Tiger Problem with $p = 0.85$.

This Tiger Problem with $p = 0.85$ can be optimally solved by using 4-length history. Since the transition in Fig. 2.3 has the same representation ability as 2-length history, the agent based on our method could not achieve the exact optimal solution but a sub-optimal one with faster learning speed than a naive history-based approach. We also applied our method to the Tiger Problem with $p = 0.80$, after assuming that the number of memory states is seven: $|\mathcal{M}| = 7$. Note that, in this setting, the problem can be optimally solved by using 7-length history. Figure 2.4 represents a memory state model estimated by the agent, which has the same representation ability as 4-length history. Table 2.2 shows action selection probabilities acquired by the RL algorithm for the estimated model. In both settings, the agent based on our method can acquire an appropriate policy in this reasonably-sized partially observable environment.

Two issues, however, still remain in this method. First, the number of memory states should be estimated; in this work, the number of memory states $|\mathcal{M}|$ is given to the agent in advance, but it should be possible to estimate with the model estima-

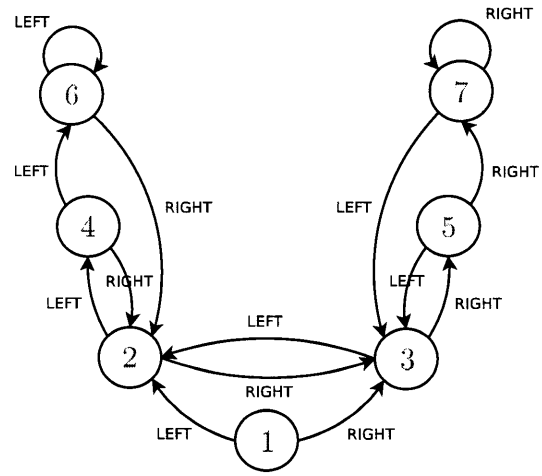

Fig. 2.4: The state transition when the agent takes the LISTEN action, $m_{t+1} = \text{argmax}_{m_{t+1}} P(m_{t+1}|m_t, a_t = \text{LISTEN}, o_t)$, in the Tiger Problem with $p = 0.80$.

m_t \ a_t	LISTEN	LEFT	RIGHT
1	0.999896	0.000002	0.000102
2	0.999417	0.000568	0.000015
3	0.999985	0.000011	0.000004
4	1.000000	0.000000	0.000000
5	0.999992	0.000008	0.000000
6	0.000539	0.000000	0.999461
7	0.006082	0.993917	0.000000

Table. 2.2: Action selection probability acquired by the agent in the Tiger Problem with $p = 0.80$.

tion process. An appropriate number of memory states can be gradually estimated by increases or decreases in the state number with decision-making processes (Poupart & Boutilier, 2004), or alternatively mixture models such as a Dirichlet process, non-parametric Bayesian inference, may be beneficial for this issue. Second, the two separate phases, the model estimation phase by the maximum likelihood inference and the policy acquisition phase by an RL algorithm, should be unified; there are two objective functions optimized separately. Since this separation may prevent acquisition of an optimal solution, the memory model should be estimated with reinforcement learning of a policy on-line.

Each node of an FSM is associated with a corresponding distinct linear function in the equivalent value function representation. In the case exemplified as Figure 2.1, for example, a policy defined by the value function is equivalent to that represented by

the FSM controller with four memory states, because there are four dominant linear functions. Kaelbling et al. (1998) showed that an FSM controller can be obtained by solving the belief state MDP before constructing an FSM model by converting each linear function of a convergent value function into a memory state with a corresponding immediate action.

2.3.2 Reactive memoryless policy

A special case of an FSM model, which is the simplest alternative in other models, is to learn a policy over the observation space instead of the state space, by applying methods for completely observable MDPs such as the classical Q-learning (Watkins & Dayan, 1992) or TD learning (Sutton, 1988), computing $Q(o_t, a_t)$ rather than $Q(s_t, a_t)$. These policies are known as memoryless or reactive, as they react to the latest observation solely without keeping track of past observations; the memoryless policy is the same as an FSM controller with one memory state. Since the observation space does not satisfy the Markov property, direct application of the conventional MDP algorithms results in poor performance. In these methods, however, there is no need to estimate the unobservable states, making it easy to define the policy in MDP formulation. Many early studies, therefore, analyzed the detailed properties of the memoryless policy in theoretical research. Littman (1994a) provided a theoretical proof that the general problem of finding the optimal deterministic memoryless policy is NP-hard and presented a simple branch-and-bound heuristics for computing the policy. He also suggested that it is important to use an external memory architecture; this idea was promoted for approaches with an external memory bit by Peshkin et al. (1999), described in the next section.

Singh et al. (1994) showed that a stationary stochastic memoryless policy can perform arbitrarily better than a stationary deterministic memoryless policy[*1], with satisfactory examples being constructed, such as a McCallum's (1993) maze. They also indicated that direct application of the Q-learning and TD learning over the observation space may suffer from the perceptual aliasing property in partially observable environments, because these algorithms do not learn a suitable observation value but converge to different types of solutions. That is, the observation value $V(o_t)$ obtained by the TD learning, whose TD error is defined as $\delta_t = R(o_t, a_t) + \gamma V(o_{t+1}) - V(o_t)$, diverges from the optimal value $V^*(o_t)$, where $R(o_t, a_t)$ is an immediate reward ob-

[*1] They also argued an interesting point about game theory, namely that the reason why the optimal strategies in zero-sum games are stochastic is the same as that for stochastic policies in POMDPs: the lack of knowledge of the opponent's action constitutes the unobservable states.

tained by taking an action a_t given the observation o_t.

Loch and Singh (1998) demonstrated empirically that eligibility traces (Singh & Sutton, 1996) over the observation space can work well in several environments, because they allow an observation-action pair to access what happens many time steps later, bridging the divergence to certain information about the quality of an action. They did not, however, compare the performance of the eligibility traces with other algorithms such as the Monte Carlo method (Sutton & Barto, 1998). Baird and Moore (1998) showed that the hybrid memoryless algorithm incorporating both searches in the space of policies and of value function, called the VAPS algorithm, converges for partially observable environments in which a memoryless policy exists. They too did not, however, compare their method with other algorithms.

2.3.3 Memory-based algorithms

Searching a memoryless policy space often yields a poor controller, because the value function whose representation is an equivalent to the corresponding FSM controller is approximated by a single linear function; it is too loose an approximation for most partially observable problems. Studies of the reactive memoryless policy provided a theoretical limitation for such algorithms. Researchers now focus on approaches in which the estimation of the unobservable states is based on some kind of memory. Since any memory mechanism can be interpreted as memory states of an FSM model, all memory-based approaches are essentially identical to the FSM method.

Finite memory

The simplest approach other than the reactive memoryless policy is to use the last k observations as input to the policy. The size of the observation history in such a naive approach is fixed for all observation sequences. Lin and Mitchell (1992) proposed a finite history-based method, called Window-Q, to learn the Q-function $Q(h_t^k, a_t)$ by using an artificial neural network, where the input of the function is the k-step history $h_t^k = \{(o_t, -), (o_{t-1}, a_{t-1}), \cdots, (o_{t-k+1}, a_{t-k+1})\}$. McCallum (1996) proposed a sophisticated history-based method using the k nearest neighbor algorithm, called nearest sequence memory (NSM). An agent based on the NSM method remembers past information by using an instance T_t, where the instance is defined as a tuple: $T_t = \langle T_{t-1}, a_t, r_t, o_{t+1} \rangle$. An action value on an instance T_t is calculated as the average of action values on k instances nearest to the instance T_t, in which the similarity metric between instances is defined in a recursive manner. Estelle (2003) examined the NSM approach to learn faster than Q-learning with a fixed memory window as the problem size increases.

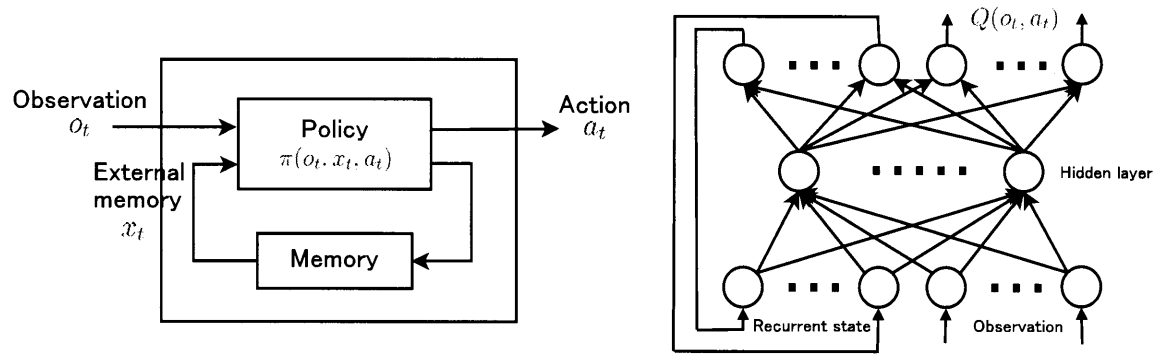

Fig. 2.5: Architectures of the external memory and recurrent neural network.

left: The architecture of the external memory. The output of the agent's policy is stored to the external memory x_t, and is then fed back in the next time step to the policy as the input with the new observation.

right: The architecture of the recurrent neural network. Since the output of the neural network is fed back to a previous layer of the network in the next time step, it has an external memory mechanism.

The finite memory with a flat structure can be arranged to a finite suffix memory with a tree structure. Ron et al. (1994) proposed a finite history-based method using probabilistic suffix automata with an adaptive length, in which the history is represented as a path through the tree starting at the root, with the latest information mapped to the root. McCallum (1996) promoted this idea to the utile suffix memory algorithm, which incorporates the data structure of the suffix tree into the NSM method, and further developed the UTREE algorithm where the observation is treated as a vector, and different branches can be created depending on the value of specific components of the history.

External memory

Littman (1994a) argued that the agent should have some external memories to keep past events, where the term 'external' indicates a mechanism in which the agent cannot modify its dynamics, whereas the FSM controller modifies the transition probability of the memory state; the external memory exhibits in the same way as a working memory. A point of difference between algorithms using the external memory and the history-based method is that the output of the agent's policy is stored to the external memory x_t, after which it is fed back in the next time step to the policy as the input with the new observation; the agent acquires a policy π given an immediate observation o_t and an external memory x_t. The external memory is then expected to compensate for the partial observation. Figure 2.5 (left) represents the architec-

ture of this external memory approach. Peshkin et al. (1999) demonstrated that the SARSA (Sutton & Barto, 1998) and VAPS (Baird & Moore, 1998) algorithms with the external memory architecture work well in several partially observable environments.

Lin and Mitchell (1992) proposed similar algorithm, called Recurrent-Q, learning action values for an immediate observation by using a recurrent neural network. Since the output of the neural network is fed back to a previous layer of the network, it has an external memory mechanism within its hidden layer. Figure 2.5 (right) represents the architecture of the Recurrent-Q approach. A different point from the simple external memory approach is that, since a neural network acquires connections between layers based on the back-propagation algorithm, an effective feedback system which can compensate for the partial observation can be acquired so that the error between the true action value and an estimated value is minimized.

2.4 Discussion

The major targets of the studies described in Chapters 3 and 4 are large-scale and multi-agent problems with partial observability; there are multiple agents in a common large-scale, partially observable environment, and they are in a cooperative or competitive situation. For solving such difficult realistic problems, no solution exhibited by exact value iteration can be applied, because the value function over the entire belief space has an intractable complexity in large-scale problems.

In the RL method presented in Chapter 3, the value function learns over an expected observation; this is similar to the reactive memoryless approach in which the agent determines its action based on an immediate observation. As described in Section 2.3.2, however, learning the value function without estimating unobservable states may lead to the difficulty of obtaining an accurate value due to the perceptual aliasing property, and the resulting policy may exhibit only poor performance. The expected observation, therefore, includes the estimation process within its representation to compensate for the unobservable information. Such an expected observation is an approximate representation of the belief state, which is calculated by changing the order of summations; it calculates the summation over the observation before calculating the summation over the next state. The action selection, therefore, depends on an averaged point; this approximation idea is similar to the fast-informed bound approximation, described in Section 2.2.1.

In the RL method presented in Chapter 4, on the other hand, we use completely observable approximation, described in Section 2.2.1; the agent tries to approximate the value function in the underlying state space, instead of the observation or belief state

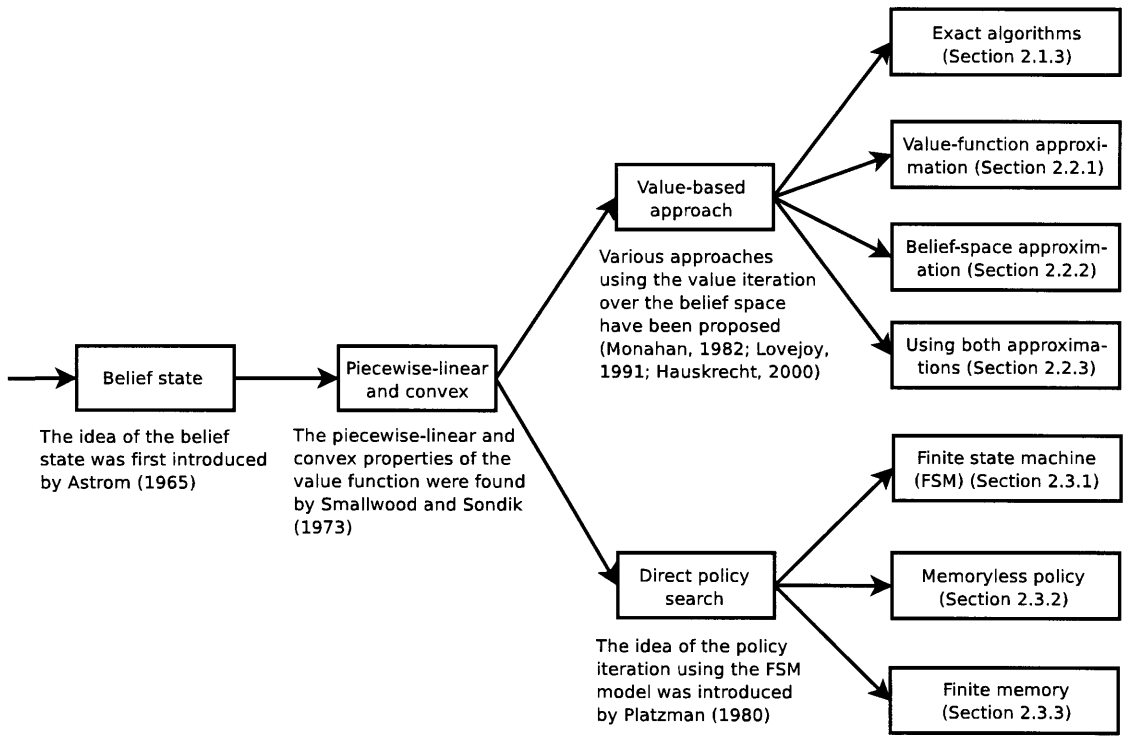

Fig. 2.6: History of POMDP research. The algorithms for POMDP problems can be classified broadly into two categories: first, the value-function and belief-state approximation (including exact algorithms); and second, the policy approximation. Solutions in the first category learn the value function over the continuous belief space with the explicit computation of the belief state, and this formulation is generally called belief-state MDP. Solutions in the second category search the policy space directly with the finite space assumption, and any algorithm can be generalized as an FSM model. Our POMDP-RL methods, presented in Chapters 3 and 4, can be classified mainly into the first category. Several ideas used by solutions in the second category, however, are also applied to our methods.

space. POMDP problems are then solved based on expectation of the value function with respect to the belief state. This is, therefore, similar to QMDP (Littman et al., 1995), except that our method does not maintain the Q-function explicitly but calculates the utility function by one-step-ahead prediction based on the acquired model. To calculate the utility function, estimation and prediction must be performed; the estimation is to calculate an expectation over possible unobservable current states whose probabilities are proportional to the belief $p(s_t|H_t)$, and the prediction is to calculate an expectation over possible next states and observations whose probabilities are proportional to the transition probability $p(s_{t+1}|s_t, a_t)$ and observation probability $p(o_{t+1}|s_t, a_t)$, respectively. Integration in these expectations is almost impossible because most challenging problems have a large number of states; for example, card

games using an ordinary 52-card deck have $52!/(13!)^4 \simeq 10^{28}$ states if every combination of 52 cards is considered. In our method, this heavy integration is approximated by a sampling technique. This methodology is then similar to Thrun's (2000) method in which approximate belief state tracking is performed by using a sequential Monte Carlo technique, or to the grid-based approximation methods, both described in Section 2.2.3.

Since the environmental model is unknown for the agent, the learning agent based on our RL methods acquires the model directly by multiple action predictors; the state transition of usual multi-agent games depends on opponent agents' actions. The agent learns the policy of each opponent agent explicitly by the corresponding predictor from past experiences, and determines its action based on the acquired opponent agents' policies. Our method, therefore, is a model-based approach (Sutton, 1990; Moore & Atkeson, 1993; Doya, Samejima, Katagiri, & Kawato, 2002), described in Section 2.2.1. Throughout this dissertation, the environment is regarded as stationary for the learning agent, under the assumption that there is a single learning agent, and the proposed POMDP-RL methods are then applied to multi-agent problems: most of this dissertation, it is assumed that there is only one learning agent in the environment. When there are multiple learning agents, the environmental dynamics can be influenced by learning of the other agents. The environment, therefore, does not primarily satisfy a stationary Markov assumption even for underlying states, and so cannot be properly formulated as a POMDP. In an ordinary approach, the multi-agent problems should be dealt with by other appropriate frameworks such as game theory (Fudenberg & Tirole, 1991), which aims at providing solutions to problems of selecting optimal actions in non-stationary multi-agent environments; the optimal solution for all agents is known as a Nash equilibrium. Although it is not difficult to obtain the equilibrium solution for zero-sum two-player games such as Poker, it is intractable to do so in general-sum multi-player games with partial observability such as the game Hearts, because the highly complicated cooperative or competitive relationship among the multiple agents causes difficulty in calculating an analytic solution. Most studies about acquiring an optimal policy in multi-agent environments, therefore, have restricted their targets to simple problems with two players (Stone & Veloso, 2000; Bowling & Veloso, 2000); a comprehensive examination of multi-agent learning techniques applicable to general situations has not yet been undertaken. It is thus assumed that the POMDP assumption is approximately valid if the environmental model can be learned faster than it changes. To achieve this fast learning, the learning agent makes the action predictors learn independently to predict unknown behaviors of the opponent agent. When one opponent agent changes

its policy, this is enough to make the corresponding predictor adapt to the change. Since it enables the agent to adapt to environmental changes quickly, our method, formulated as a single-agent system, can be applied to multi-agent systems.

In Chapters 3 and 4, we use the following notations and assumptions. t indicates an action turn of the learning agent. The variables (state, observation and action) for agent i ($i = 0, \cdots, M$) are denoted by s_t^i, o_t^i and a_t^i,[*2] where M is the number of opponent agents and $i = 0$ signifies the learning agent; s_t, o_t and a_t are the same as s_t^0, o_t^0 and a_t^0, respectively, and s_{t+1} is the same as s_t^{M+1}. A strategy of agent i ($i = 1, \cdots, M$) is denoted by ϕ^i, which corresponds to the action selection policy or the policy parameters of the agent. We make two assumptions: first, agent i selects its action a_t^i based on an immediate observation o_t^i according to its action selection probability $P(a_t^i|o_t^i, \phi^i)$; and second, the other agents' strategies ϕ^i are fixed for the time being, that is, there is only one learning agent in the environment, and hence the environment is stationary. In the computer experiments described in Section 4.3, however, we will relax the second assumption and show several acceptable results in dynamic environments with multiple learning agents.

[*2] Some games have partial observability in opponent players' actions a_t^i ($i = 1, \cdots, M$); for example, in a game where players discard cards face down, each player cannot observe the actions. Such unobservable actions, however, can be estimated if the cards discarded by the actions are included in unobservable state s_t; the agent can select an action based on the estimated actions. Since the agent can calculate likelihood as long as the actions contain some partial observation, our method, which estimates unobservable states based on likelihood information, is applicable. Although we do not, in this dissertation, assume any partial observability in opponent agents' actions, our method can thus be extended to deal with such situations.

Chapter 3

Model-based Reinforcement Learning for Partially Observable Games with Mean-field State Estimation

This chapter presents a model-based reinforcement learning scheme for large-scale, multi-agent and partially observable environments with mean-field state estimation.[*1] To overcome several computational difficulties, we use a mean-field-like analog approximation; the distribution of unobservable states is approximated by an expected observation which includes the estimation process under the assumption that the unobservable states are distributed with uniform probability. To adapt dynamic multi-agent environments, the learning agent has multiple forward models for predicting environmental behavior, and learns them independently. Computer simulation results show that our method is effective at solving a difficult partially observable multi-agent problem.

3.1 Model

In our RL method, the learning agent selects an action according to the soft-max policy:

$$P(a_t|o_t) = \frac{\exp(\langle\delta_t\rangle(a_t)/T_m)}{\sum_{a_t \in \mathcal{A}} \exp(\langle\delta_t\rangle(a_t)/T_m)}, \tag{3.1}$$

where T_m is a constant which denotes the assumed action randomness. $\langle\delta_t\rangle(a_t)$ denotes an expected TD error, which is defined as a residual error between a value of the current observation and the sum of an expected immediate reward and a value of

[*1] The contents of this chapter appear in our papers (Ishii, Fujita, Mitsutake, Yamazaki, Matsuda, & Matsuno, 2005; Fujita, Matsuno, & Ishii, 2003).

the next observation with respect to the predictive distribution:

$$\langle \delta_t \rangle(a_t) = \langle R(o_{t+1}) \rangle(a_t) + \gamma \langle V(o_{t+1}) \rangle(a_t) - V(o_t), \tag{3.2}$$

where

$$\langle f(o_{t+1}) \rangle(a_t) \equiv \sum_{o_{t+1} \in \mathcal{O}} P(o_{t+1}|a_t, H_t) f(o_{t+1}). \tag{3.3}$$

$R(o_{t+1})$ denotes an immediate reward at time step t, γ denotes a discount factor[*2], and V is the value function over the observation space. Learning the value function without estimating unobservable states may cause difficulty in obtaining an accurate value due to the perceptual aliasing property in partially observable environments; the same observation may come from different states whose state values should be different, but the value function defined on the observation space cannot detect the difference between the values, and this intractable property may prevent the value function from converging to a global optimum (Singh et al., 1994; Kaelbling et al., 1996). In our method, therefore, we use a feature extraction technique so that the observation o_t contains partial information about unobservable states within low dimensionality; the details of this technique are described in Section 3.2. Although optimization of the value function, whose input is a probability distribution (that is, belief state) over the high-dimensional state space, is too complex and requires heavy computation, learning the value function over the observation space with the estimated information of the unobservable states is an effective method, especially for large-scale POMDP problems.

The predictive distribution of the next observation $P(o_{t+1}|a_t, H_t)$ in equation (3.3) can be given by

$$P(o_{t+1}|a_t, H_t) = \sum_{s_{t+1} \in \mathcal{S}} P(o_{t+1}|s_{t+1}) \sum_{s_t \in \mathcal{S}} P(s_{t+1}|s_t, a_t) P(s_t|H_t). \tag{3.4}$$

Note here that this calculation in equation (3.4) includes two summations; in large-scale partially observable problems, this causes an important difficulty in addition to the optimization of the value function. Since there are M opponents' actions within the transition from a state s_t to the next state s_{t+1} in multi-agent problems, the transition probability $P(s_{t+1}|s_t, a_t)$ in equation (3.4) is represented by using the real

[*2] Our study mainly aims at dealing with finite-horizon problems, but could be applied to infinite-horizon problems whose objective function is defined in equation (2.1), because our method is basically an extension of the classical value iteration algorithm.

action selection probability $P(a_t^i|o_t^i, \phi^i)$ of agent i:

$$P(s_{t+1}|s_t, a_t)$$
$$= \sum_{\{s_t^1,\cdots,s_t^M\}\in\mathcal{S}^M} \sum_{\{a_t^1,\cdots,a_t^M\}\in\mathcal{A}^M} \prod_{j=0}^{M} P(s_t^{j+1}|s_t^j, a_t^j) \prod_{i=1}^{M} \sum_{o_t^i\in\mathcal{O}} P(a_t^i|o_t^i, \phi^i) P(o_t^i|s_t^i). \quad (3.5)$$

In many multi-agent games with partial observability, the environmental dynamics are deterministic and have two properties:

(i) the state s_t^{i+1} can be uniquely determined given a state s_t^i and an action a_t^i, that is, $P(s_t^{i+1}|s_t^i, a_t^i)$ is 1 for a certain state s_t^{i+1}, otherwise 0; and
(ii) the observation o_t^i can also be determined without any ambiguity given a state s_t^i, that is, $P(o_t^i|s_t^i)$ is 1 for a certain observation o_t^i, otherwise 0.

Under these properties and equation (3.5), equation (3.4) is simplified to

$$P(o_{t+1}|a_t, H_t) = \sum_{s_t\in\mathcal{S}} P(s_t|H_t) \sum_{\{a_t^1,\cdots,a_t^M\}\in\mathcal{A}^M(s_t,a_t,o_{t+1})} \prod_{i=1}^{M} P(a_t^i|o_t^i, \phi^i). \quad (3.6)$$

where $\mathcal{A}^M(s_t, a_t, o_{t+1})$ denotes the set of possible sequences of opponents' actions $\{a_t^1, \cdots, a_t^M\}$ in which the state s_t reaches the next state whose observation is o_{t+1}, after the action a_t. Once the predictive distribution of the next observation in equation (3.6) is calculated, the action selection based on the TD error expected with respect to the distribution, which contains the estimation process of the unobservable states and the prediction process of the environmental dynamics, can be achieved by the soft-max selection rule in equation (3.1).

Calculating the predictive distribution, however, presents three problems:

(a) the summations over possible current states s_t and possible opponents' actions $a_t^1, \cdots, a_t^M$ have computational intractability because there are so many candidates in a large-scale problem;
(b) the computation for constructing the belief state $P(s_t|H_t)$ over possible current states is intractable due to the large state space and high dimensionality; and
(c) the prediction of opponent agents' actions is difficult because the action selection probability of opponent agent $P(a_t^i|o_t^i, \phi^i)$, which constitutes a part of the environmental dynamics, is unknown for the learning agent and may change in a multi-agent setting.

To solve the computational intractability problems (a) and (b), we use a mean-field-like analog approximation. Since the observation of each opponent agent o_t^i

depends on the unobservable state s_t, in the straightforward calculation without any approximation, the action selection probability $P(a_t^i|o_t^i, \phi^i)$ should be calculated for all possible observations o_t^i expected with respect to the posterior probability of all possible unobservable states $P(s_t|H_t)$.

To avoid such exhaustive enumeration of the current state s_t, in our method, the computation of equation (3.6) is approximated using the expected observation $\hat{o}_t^i(a_t, H_t)$, by the following three steps: the first step is to estimate the probability $P(o_t^i|a_t, H_t)$; the second step is to calculate the expected observation $\hat{o}_t^i(a_t, H_t)$ for agent i:

$$\hat{o}_t^i(a_t, H_t) \equiv \sum_{o_t^i \in \mathcal{O}} o_t^i P(o_t^i|a_t, H_t), \tag{3.7}$$

with details of these two steps described in Section 3.2; and the last step is to calculate equation (3.6) with the following expected observation:

$$P(o_{t+1}|a_t, H_t) \approx \sum_{\{a_t^1, \cdots, a_t^M\} \in \mathcal{A}^M(s_t, a_t, o_{t+1})} \prod_{i=1}^{M} P(a_t^i|\hat{o}_t^i(a_t, H_t), \phi^i). \tag{3.8}$$

The expected observation $\hat{o}_t^i(a_t, H_t)$ contains the history information H_t, and it includes the estimation process of the unobservable states, as in calculating the belief state $P(s_t|H_t)$; this approximation can simultaneously solve problem (b). Note, however, that the approximation process does not retain the complete propagation of the belief state, because the expected observation in equation (3.7) is calculated under the assumption that the unobservable states are distributed with uniform probability in each time step. According to deterministic property (ii) described above, an observation for opponent agents o_t^i is determined without any ambiguity given a possible state s_t^i, and the stochastic process of the environment depends on the opponents' observations. In our method, the process is approximated as an alternative stochastic process which depends on an analog observation in equation (3.7). This approximation may cause bias; the environmental dynamics estimated by equation (3.8) deviates from the real environmental one due to the estimation error between the real observation o_t^i and the expected observation $\hat{o}_t^i(a_t, H_t)$. The summation over the current state, however, is no longer required; since all possible observations are summarized to the expected observation as a central value, the calculation of the summation can be removed. To solve the latter difficulty in problem (a), that is, the computational intractability for summation over possible opponents' actions $a_t^1, \cdots, a_t^M$, we use a pruning technique, the details of which are described in Section 3.2.

To solve problem (c), the learning agent uses action predictors. Since there are M opponents' actions within the one-step transition in multi-agent problems, the

environmental dynamics is approximated by M action predictors; i-th action predictor ($i = 1, \cdots, M$) approximates the action selection probability of the i-th opponent agent $P(a_t^i|o_t^i, \phi^i)$:

$$P(a_t^i|o_t^i, \phi^i) \approx P(a_t^i|o_t^i, \hat{\phi}^i) = \frac{\exp(F^i(o_t^i, a_t^i; \hat{\phi}^i)/T_m^i)}{\sum_{a_t^j \in \mathcal{A}} \exp(F^i(o_t^i, a_t^j; \hat{\phi}^i)/T_m^i)}, \tag{3.9}$$

where $F^i(o_t^i, a_t^i; \hat{\phi}^i)$ denotes an assumed utility of taking action a_t^i for an observation o_t^i, and T_m^i is a constant denoting the assumed randomness of agent i's policy. $\hat{\phi}^i$ is not the real strategy ϕ^i by agent i but a strategy approximated by the i-th action predictor. The predictive distribution in equation (3.6) can be calculated by using the action predictor in equation (3.9), but in the actual computation process of our approximation method, equation (3.9) is not calculated but the action selection probability of the i-th opponent agent, whose input is the expected observation $P(a_t^i|\hat{o}_t^i(a_t, H_t), \phi^i)$, is:

$$P(a_t^i|\hat{o}_t^i(a_t, H_t), \phi^i) = \frac{\exp(F^i(\hat{o}_t^i(a_t, H_t), a_t^i; \hat{\phi}^i)/T_m^i)}{\sum_{a_t^j \in \mathcal{A}} \exp(F^i(\hat{o}_t^i(a_t, H_t), a_t^j; \hat{\phi}^i)/T_m^i)}. \tag{3.10}$$

Once the action selection probability is estimated by the corresponding action predictor in equation (3.10), the decision-making based on the TD error expected with respect to the predictive distribution can be attained with the mean-field-like analog approximation.

The approximation techniques described above enable us to solve large-scale and partially observable problems. Our method provides, for example, an effective solution to multi-agent problems whose underlying state space is discrete, including various multi-agent games.

3.2 Function approximators and feature extraction

In the previous preliminary study, the input and the output of the value function were the current naive observation o_t and the corresponding numerical value $V(o_t)$, respectively (Matsuno, Yamazaki, Matsuda, & Ishii, 2001). This implementation, however, makes the input dimension equal to or larger than the number of cards, a high dimensionality that causes the performance of the RL agent to deteriorate due to the large and redundant representation. To achieve effective learning in a realistic problem, therefore, we use a feature extraction technique; for our specific application, the game Hearts, the high-dimensional observation o_t is converted to a 25-dimensional input p_t according to the following representation:

- $p_t[1]$: the number of club cards which have been already played in the current game, or in the learning agent's hand,
- $p_t[2]$: same as $p_t[1]$, but the suit is diamonds,
- $p_t[3]$: same as $p_t[2]$, but the suit is spades excepting ♠Q, ♠K and ♠A,
- $p_t[4]$, $p_t[5]$ and $p_t[6]$: the probability where the opponent agent i ($i = 1, 2, 3$) has ♠Q, respectively,
- $p_t[7]$: a status for ♠K,
- $p_t[8]$: a status for ♠A,
- $p_t[9]$ to $p_t[21]$: a status for each of heart cards, and
- $p_t[22]$ to $p_t[25]$: a bit sequence.

Since the most important card is ♠Q in the game Hearts, we allocate three dimensions to represent its predictive distribution. The game's rules provide us with the following information:

1. If agent i did not play a spade card when a leading card was a spade card in a past trick of the current game, $p_t[i+3]$ is zero.
2. $p_t[4] + p_t[5] + p_t[6] = 1$.

Under limitation from these two kinds of information, the probability that agent i has ♠Q, $p_t[i+3]$, is calculated as a uniform probability. The statuses of ♠K, ♠A, or heart cards, take -1, 0 or 1, which represent the cases when the card has already been played in the current game, when it is in the opponent agent's hand, or when it is in the learning agent's hand, respectively. The bit sequence represents the playing order in the current trick; for example, when the learning agent is the second player in the current trick (the t-th playing turn of the learning agent), $\{p_t[22], p_t[23], p_t[24], p_t[25]\} = \{0, 1, 0, 0\}$. Note that the value function depends not only on the immediate observation o_t but partly on the estimation of the unobservable state; $p_t[4]$, $p_t[5]$ and $p_t[6]$ represent an estimated distribution of the unobservable card, ♠Q, and $p_t[7]$ to $p_t[21]$ represent an estimated status of the important cards. In the actual process, $V(o_t)$ and $V(o_{t+1})$ in equation (3.2) are replaced with $V(p_t)$ and $V(p_{t+1})$, respectively, according to the above feature extraction.

The expected observation $\hat{o}_t^i(a_t, H_t)$ for agent i, which is an input of the action predictor, has also high dimensionality. To reduce the dimensionality and achieve effective learning in the realistic problem, the expected observation $\hat{o}_t^i(a_t, H_t)$ is converted to a 26-dimensional input q_t^i according to the following representation:

- $q_t^i[1]$: the expected number of club cards in agent i's hand which are weaker than the strongest card already played in the current trick (only if the leading

card is a club, otherwise zero),

- $q_t^i[2]$: the expected number of club cards in agent i's hand which are stronger than the strongest card already played in the current trick (only if the leading card is a club, otherwise the expected number of club cards in the agent i's hand),
- $q_t^i[3]$: same as $q_t^i[1]$, but the suit is diamonds,
- $q_t^i[4]$: same as $q_t^i[2]$, but the suit is diamonds,
- $q_t^i[5]$: same as $q_t^i[1]$, but the suit is spades excepting ♠Q, ♠K and ♠A,
- $q_t^i[6]$: same as $q_t^i[2]$, but the suit is spades excepting ♠Q, ♠K and ♠A,
- $q_t^i[7]$: an expectation value in which ♠Q is in agent i's hand.
- $q_t^i[8]$: an expectation value in which ♠K is in agent i's hand.
- $q_t^i[9]$: an expectation value in which ♠A is in agent i's hand.
- $q_t^i[10]$ to $q_t^i[22]$: an expectation value in which each heart card is in agent i's hand, and
- $q_t^i[23]$ to $q_t^i[26]$: a bit sequence.

Let $C_t^i(\spadesuit\mathrm{Q})$ be a binary value, taking either 1 or 0, which represents whether agent i has, for example, ♠Q or not just before its t-th turn, respectively. An expectation value of the binomial variable $C_t^i(\spadesuit\mathrm{Q})$ is the same as the probability that agent i has ♠Q:

$$\hat{C}_t^i(\spadesuit\mathrm{Q}|a_t, H_t) = P(C_t^i(\spadesuit\mathrm{Q}) = 1|a_t, H_t). \tag{3.11}$$

The game's rules provide us with the following information:

1. Agent i does not have a card of the suit, if agent i did not play a card whose suit is the same as that of a leading card in a past trick of the current game.
2. Agent i may have all possible cards (excepting cards in the learning agent's hand and cards that have already been played in the current game).

Under limitation from these two kinds of information, we assume that the unobservable cards are distributed with uniform probability to each opponent agent's hand. $\hat{C}_t^i(\text{a-card}|a_t, H_t) \in [0, 1]$, representing an expectation value for which "a-card" is in agent i's hand, is then calculated with respect to the distribution. $q_t^i[1], \ldots, q_t^i[6]$ are calculated using $\hat{C}_t^i(\clubsuit 2|a_t, H_t), \ldots, \hat{C}_t^i(\spadesuit\mathrm{J}|a_t, H_t)$, and $q_t^i[7], \ldots, q_t^i[22]$ correspond to $\hat{C}_t^i(\spadesuit\mathrm{Q}|a_t, H_t), \ldots, \hat{C}_t^i(\heartsuit\mathrm{A}|a_t, H_t)$, respectively; q_t^i can be obtained by calculating the estimated card probability $\hat{C}_t^i$. The bit sequence is the same as p_t.

Given an extracted input q_t^i, the utility function F^i of agent i returns a 26-dimensional output vector r_t^i each of whose dimensions represents:

- $r_t^i[1]$: the merit value where agent i plays an arbitrary club card which is weaker than the strongest card in the current trick,
- $r_t^i[2]$: the merit value where agent i plays the weakest club card (among the remaining club cards) which is stronger than the strongest card in the current trick,
- $r_t^i[3]$: the merit value where agent i plays a club card (neither the weakest nor the strongest among the remaining club cards) which is stronger than the strongest card in the current trick,
- $r_t^i[4]$: the merit value where agent i plays the strongest card (in remaining club cards) which is stronger than the strongest card in the current trick,
- $r_t^i[5]$: same as $r_t^i[1]$, but the suit is diamonds,
- $r_t^i[6]$: same as $r_t^i[2]$, but the suit is diamonds,
- $r_t^i[7]$: same as $r_t^i[3]$, but the suit is diamonds,
- $r_t^i[8]$: same as $r_t^i[4]$, but the suit is diamonds,
- $r_t^i[9]$: the merit value where agent i plays an arbitrary spade card (except ♠Q, ♠K and ♠A) which is weaker than the strongest card in the current trick,
- $r_t^i[10]$: the merit value where agent i plays an arbitrary spade card (except ♠Q, ♠K and ♠A) which is stronger than the strongest card in the current trick,
- $r_t^i[11]$: the merit value where agent i plays ♠Q,
- $r_t^i[12]$: the merit value where agent i plays ♠K,
- $r_t^i[13]$: the merit value where agent i plays ♠A, and
- $r_t^i[14]$ to $r_t^i[26]$: the merit value where agent i plays each heart card.

The merit value represents a tendency where agent i plays the corresponding card; the larger the merit value, the more likely the agent is to play the card.

According to the above feature extraction technique, the action selection probability of agent i is calculated by the following four steps: first, the expected observation $\hat{o}_t^i(a_t, H_t)$ is obtained by calculating the estimated card probability $\hat{C}_t^i$ of each card for agent i; second, the expected observation $\hat{o}_t^i(a_t, H_t)$ is converted to the extracted and hence compressed 26-dimensional input $q_t^i(a_t, H_t)$ according to the above representation, but note that $q_t^i(a_t, H_t)$ is directly obtained without calculating $\hat{C}_t^i$ in the actual implementation, and this integrated calculation corresponds to the computation in equation (3.7); third, the utility function F^i returns the compressed 26-dimensional output r_t^i given the input $q_t^i(a_t, H_t)$; and fourth, the i-th action predictor calculates the action selection probability:

$$P(r_t^i | q_t^i(a_t, H_t), \hat{\phi}^i) = \frac{\exp(F^i(q_t^i(a_t, H_t), r_t^i; \hat{\phi}^i)/T_m^i)}{\sum_{a_t^j \in \mathcal{A}^j} \exp(F^i(q_t^i(a_t, H_t), r_t^j; \hat{\phi}^i)/T_m^i)}, \tag{3.12}$$

where $\mathcal{A}^j$ denotes the set of possible actions for agent j. Note that the utility $F^i(\hat{o}_t^i(a_t, H_t), a_t^i; \hat{\phi}^i)$ in equation (3.10) is replaced with $F^i(q_t^i(a_t, H_t), r_t^i; \hat{\phi}^i)$ by the feature extraction. The action selection probability $P(r_t^i|q_t^i(a_t, H_t), \hat{\phi}^i)$ in equation (3.12) consists of a 26-dimensional vector each of whose dimensions represents an occurrence probability r_t^i, which is different from the primitive action a_t^i. The action selection probability for a compressed action r_t^i is, therefore, converted to that of a primitive action a_t^i.

When there are so many possible opponents' actions, the computational difficulty described in the latter part of problem (a) arises. To overcome this problem, we use a pruning technique whereby an action a_t^i whose merit value is smaller than the difference of the mean and the standard deviation (s.d.) of the merit values over the possible actions, (mean)−(s.d.), is pruned, and further prediction from the action is eliminated. In other words, a state transition due to an action whose merit value is fairly small is ignored; this pruning technique enables us to avoid the computational difficulty and to obtain an efficient computation of the summation in equations (3.3) and (3.8).

Feature extraction conducted by considering the properties of the target problem allows us to reduce the dimensionality and improve the learning efficiency. Large-scale realistic problems, however, still have huge state spaces and high dimensionality; for example, there are about 10^{12} possible inputs q_t^i for the action predictor. This causes difficulty in learning the action selection model directly from a limited number of learning samples, by using a simple table lookup approach such as a multinomial model, due to the large number of effective parameters in the lookup table. To overcome this difficulty, we use function approximators for the value function and action predictors. A function approximator can approximate an input-output relation as a non-linear regression model with a reasonable number of parameters, and it enables the agent to use its generalization ability for unknown situations.

The value function is approximated by a normalized Gaussian network (NGnet) (Sato & Ishii, 2000) with the feature extraction technique for its input. In this study, the NGnet $V(p_t)$ is trained so as to learn the relationship between a compressed input p_t and the expected cumulative reward according to the same learning method as that of the critic learning in the actor-critic algorithm:

$$V(p_t) \leftarrow V(p_t) + \eta_c \delta_t, \tag{3.13}$$

where η_c denotes a learning rates, and δ_t is the TD error defined as

$$\delta_t = R(p_{t+1}) + \gamma V(p_{t+1}) - V(p_t), \tag{3.14}$$

where the immediate reward is defined as $R(p_t) = n$ when the agent gets n penalty points (n may be 0) between the t-th and $(t+1)$-th play, and the discount factor γ is 1.0 in our application.

The utility function F^i in equation (3.12) is also represented by the NGnet with the feature extraction for its input and output. The NGnets are trained according to four steps: first, the actual observation $\bar{o}_t^i$ for agent i is available after one game has finished; second, the 52-dimensional observation vector $\bar{o}_t^i$ is converted to the compressed 26-dimensional input $\vec{q}_t^i$; third, the 26-dimensional output $\vec{r}_t^i$ is calculated from the actual action $\bar{a}_t^i$; and fourth, these two vectors, the input $\vec{q}_t^i$ and output $\vec{r}_t^i$, are given to the NGnet for the supervised learning. The output vector $\vec{r}_t^i$ consists of elements of 1 or 0; the target value of the dimension corresponding to the actually taken action $\bar{a}_t^i$ is 1, otherwise 0. Although the parameters of the NGnet are tuned so that the output represents non-negative values (from 0 to 1) in each dimension, the utility function does not directly represent the probability. In other words, the summation over possible actions of agent i is not always 1 because the set of possible actions $\mathcal{A}^j$ may depend on the state s_t. We thus use the soft-max normalization in equation (3.12) after removing impossible actions.

3.3 Computer simulations

We applied our RL method to the card game Hearts,[*3] which is a well-defined example of large-scale and multi-agent problems with partial observability. To evaluate our method, we carried out computer simulations where an agent trained by our RL method played against rule-based agents which have 50 general rules for playing cards from their hands.[*4] The performance of an agent can be evaluated by the acquired penalty ratio, which is the ratio of the penalty points acquired by the agent to the total penalty points of the four agents. If the four agents have equal strength, their penalty ratio averages 0.25. The rule-based agent is an experienced-level player of the game Hearts. The acquired penalty ratio was 0.41 when the three rule-based agent played against one random agent, which is a reference agent for the absolute

[*3] Throughout this dissertation, we assumed that there are M opponent agents intervening between the t-th and $(t+1)$-th action turns of the learning agent. In the game Hearts, however, if the leading players of the t-th turn and of the $(t+1)$-th turn are different, the number of intervening agents is not $M = 3$. Nonetheless, the above explanation can easily be extended to such a case. The details of the game's rules used in our experiments are described in Appendix B.

[*4] The observation process of the rule-based agents is deterministic, and our action predictors use this fact (see equation (3.8)). Although the action selection process of the rule-based agents is also deterministic, our action predictors assume that it is probabilistic and the stochastic process is approximated as the soft-max policy in equation (3.12).

evaluation of the agents' strength, where it played cards from its hand at random; in other words, the random agent acquired about 2.1-fold penalty points of rule-based agents on average.

3.3.1 Single agent learning in stationary environment

Figure 3.1 shows the result when the agent trained by our method challenged the three rule-based agents. Each point represents the average of the penalty ratio for the 2,000 games over 20 learning runs, each consisting of 120,000 training games. After about 80,000 games playing with the three rule-based agents, our RL agent came to acquire a smaller penalty ratio than the rule-based agents; the RL agent became stronger than the rule-based agents, which is statistically significant as shown in the upper panel in Fig. 3.1. By observing the results of each learning run (data not shown), we found that the automatic strategy acquisition was robustly achieved by our RL method. In our preliminary study, an agent trained by our model-based RL method could not beat the rule-based agents after 5,000 training games (Matsuno et al., 2001). The present RL method is similar to our previous preliminary model-based RL method in principle, but includes the feature extraction techniques used in the value function and the three action predictors. Since the techniques reduced the dimensionality of the function approximators, the learning process was largely accelerated, and then sufficient training games could be carried out so that learning of the agent converged.

In the first experiment described above, our RL method was applied to the problem under the POMDP assumption that there is only one learning agent in the stationary environment. In the following, we apply our method directly to multi-agent environments where there are multiple learning agents making the environment dynamic.

3.3.2 Multi-agent learning in dynamic environments

Figure 3.2 shows the result when one learning agent trained by our RL method, one learning agent based on the actor-critic algorithm, and two rule-based agents played against each other. Since performance may be influenced by seat position (that is, an agent may have an advantage/disadvantage based on its seat position if the agents have different strengths), the seat position of the four agents was fixed throughout the training run, because its influence is negligible in 2,000 averaged games. To clarify the advantage of our RL method, an actor-critic agent also uses the same feature extraction techniques for its actor and critic. As a result, the actor-critic agent with the feature extraction technique learns much faster than an actor-critic agent without

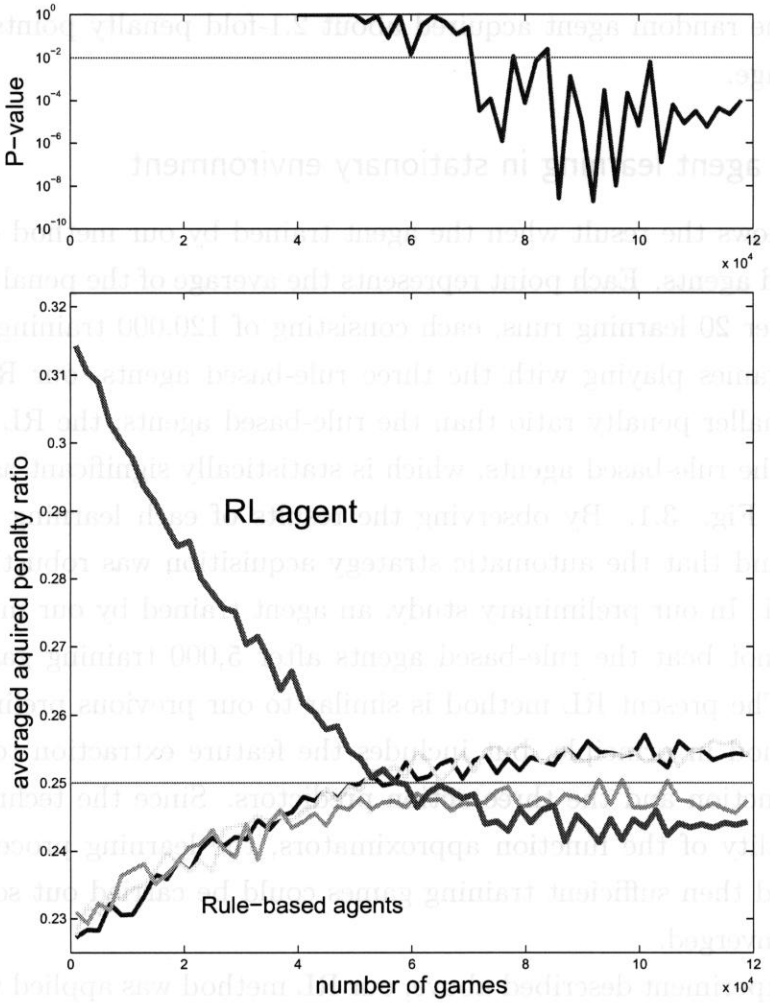

Fig. 3.1: Computer simulation result in an environment consisting of one learning agent trained by our RL method and three rule-based agents.

upper panel: P-values of the t-test where the null hypothesis is "the RL agent has the same strength as the rule-based agents" and the alternative hypothesis is "the RL agent is stronger than the rule-based agents." The test was done independently at each point on the abscissa. The horizontal line denotes the significance level of 1%. After about 80,000 training games, the RL agent was significantly stronger ($p < 0.01$) than the rule-based agents. The non-parametric Wilcoxon's rank-sum test also showed a similar result (not shown).

lower panel: The abscissa denotes the number of training games and the ordinate denotes the penalty ratio acquired by each agent. We executed 20 learning runs, each consisting of 120,000 training games. Each point represents the average for the 2,000 games over the 20 runs. The discount factors γ in equations (3.2) and (3.14) were both 1.0, the constants T_m and T_m^i in equations (3.1) and (3.12) were both 1.0, and the learning rate η_c in equation (3.13) was 1.0. These parameters were heuristically determined.

the feature extraction (Matsuno et al., 2001) (data not shown). Although the average penalty ratio of our RL agent became smaller than those of the rule-based agents after about 50,000 training games, the learning agent trained by the actor-critic algorithm improved little and its performance remained much worse than that of other agents. This result implies that our model-based approach based on the POMDP formulation is more efficient than a model-free approach, that is, the actor-critic algorithm.

Figure 3.3 shows the result when two learning agents trained by our RL method and two rule-based agents played against each other. The seat position of the four agents was fixed throughout the training run. After about 50,000 training games, both of the two learning agents became stronger than the rule-based agents, which is statistically significant as the upper panel in Fig. 3.3, even in this difficult multi-agent setting.

These two simulation results, Figs. 3.2 and 3.3, show that our RL method can be applied to the concurrent learning of multiple agents in a multi-agent environment. This applicability is partly attributed to the fast learning by the efficient function approximators. In our RL method, the learning agent has action predictors which approximate the policy of each opponent agent. When one opponent agent changes its policy, it is enough to make the corresponding predictor adapt to the change. Since this enables the agent to adapt quickly, our RL method becomes applicable to complex multi-agent settings where the Markov property fails.

3.3.3 Validation match against human player

Although the learning agents trained by our RL method became stronger than the rule-based agents, the experiments above cannot eliminate the possibility that the RL agents got a specific strategy to the rule-based agents; the RL agents may adapt just a certain situation. To validate the general strength of the learning agent, we carried out evaluation games with the human player, who is the designer of the rule-based agent. Figure 3.4 shows the result when two RL agents trained by our method, one rule-based agents and the human player played together, with fixed seat positions. One hundred evaluation games in both learning runs were played before learning and after 10,000, 30,000, 50,000, 70,000 and 90,000 training games, and these training games were carried out with three rule-based agents; one rule-based agent was then replaced by the human player in each evaluation games. The learning run was executed twice, as depicted by each point in Figure 3.4, representing a mean over 200 games. This result shows that the learning agents successfully acquired a general strategy to become as strong as the human player.

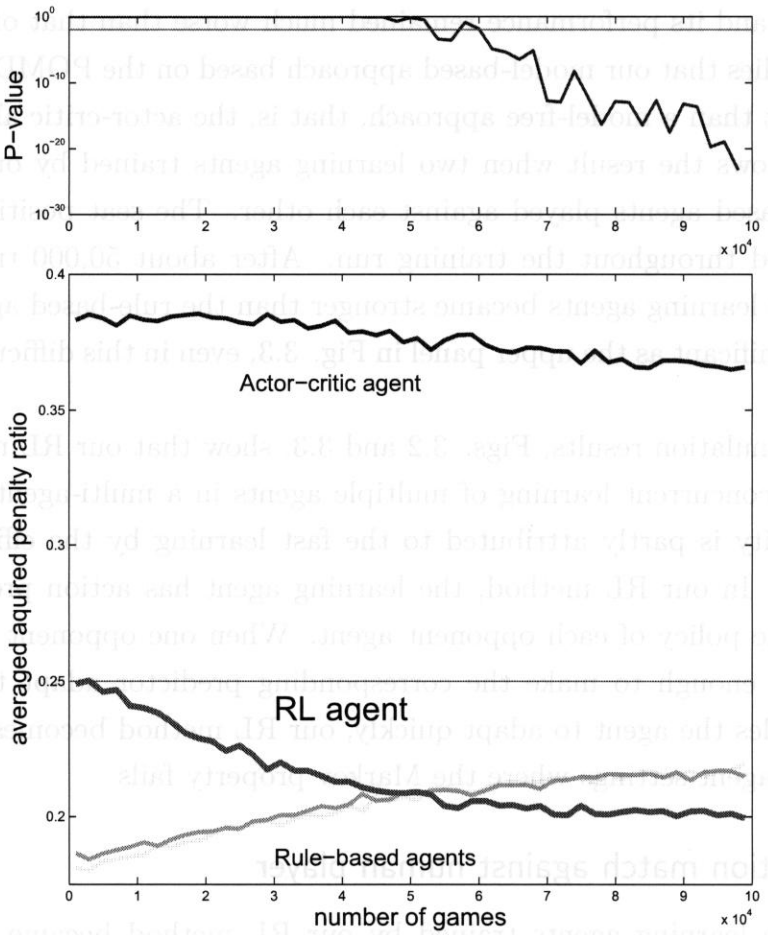

Fig. 3.2: Computer simulation result in an environment consisting of one learning agent trained by our RL method, one learning agent trained by the actor-critic algorithm, and two rule-based agents.

upper panel: P-values of the t-test where the null and alternative hypotheses are the same as in the previous experiment (Fig. 3.1). After 60,000 training games, the RL agent became significantly stronger than the rule-based agents, but the agent trained by the actor-critic algorithm did not (P-values are not shown because they remained around 1).

lower panel: The abscissa and the ordinate denote the same as in Fig. 3.1, but the scale of the ordinate is larger here. We executed 20 learning runs, each consisting of 100,000 training games. The parameter values and other experimental setup were also the same.

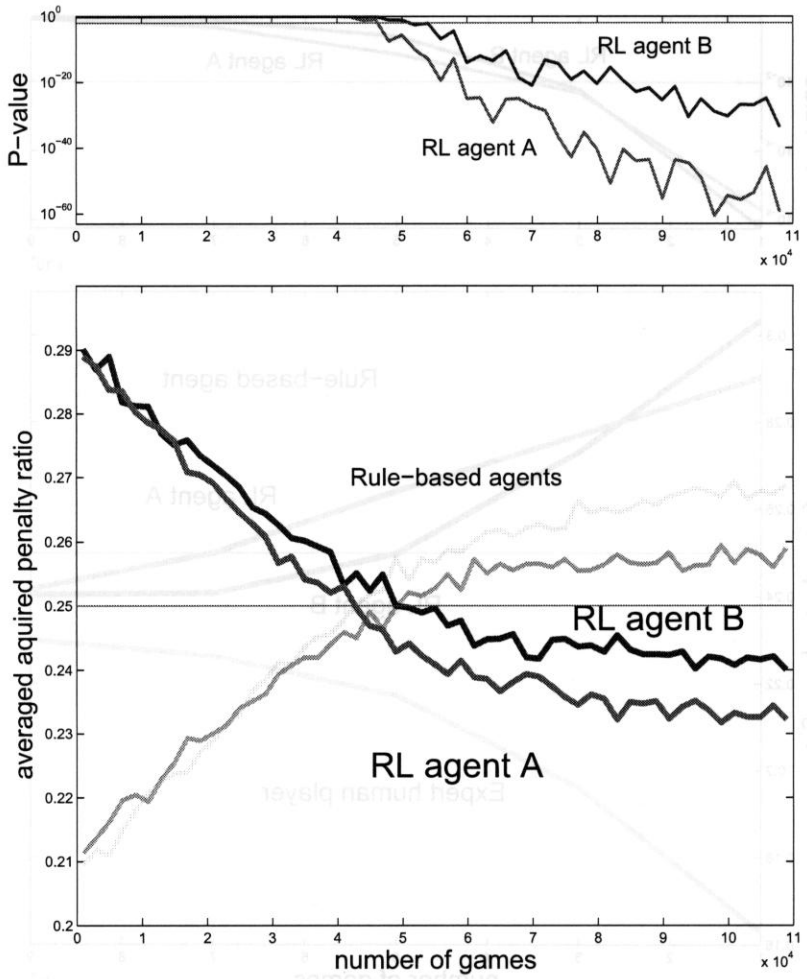

Fig. 3.3: Computer simulation result in an environment consisting of two learning agents trained by our RL method and two rule-based agents.

upper panel: P-values of the t-test where the null and alternative hypotheses are the same as in the previous experiment (Fig. 3.1). After 50,000 training games, the two RL agents were significantly stronger than the rule-based agents.

lower panel: The abscissa and the ordinate denote the same as in Fig. 3.1. We executed 20 learning runs, each consisting of 80,000 training games. The parameter values and other experimental setup were also the same.

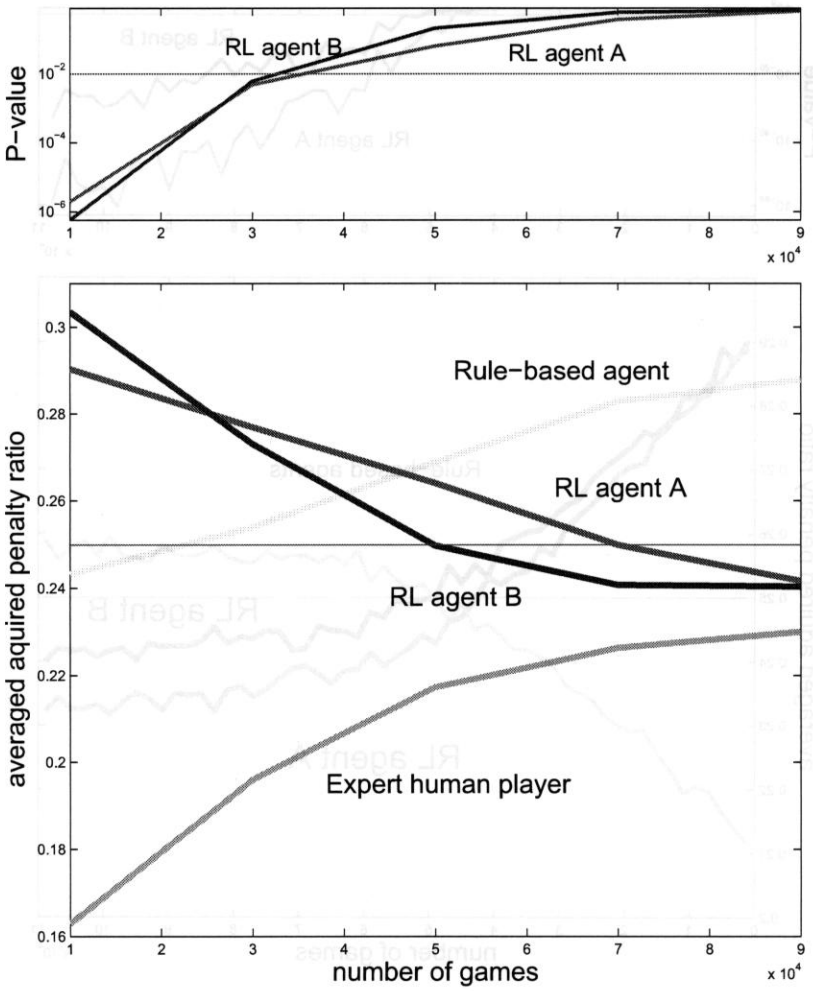

Fig. 3.4: Computer simulation result when two learning agents trained by our RL method, one human player and one rule-based agents, played together. One hundred evaluation games were carried out before learning and after 10,000, 30,000, 50,000, 70,000 and 90,000 training games (with three rule-based agents). We repeated the training and evaluation runs twice. The abscissa and the ordinate denote the same as in Fig. 3.1. Each point denotes the average of 200 (2 × 100) evaluation games.

3.4 Discussion

In our RL method, calculating the expectations to cope with uncertainty is not required for obtaining the expected TD error, because such calculation is contained in the action prediction by using the action predictors with the expected observation $\hat{o}_t^i(a_t, H_t)$ which partly incorporates the estimation of unobservable states through the feature extraction technique. Since the belief state is summarized as the expected observation, which can be interpreted as an averaged state, our approach enables us to reduce the computation time; this idea in which a probability distribution is approximated as an expectation value by changing the order of summations is similar to the fast-informed bound approximation, described in Section 2.2.1. This method may produce a bias in estimating the expected TD error due to the assumption that the opponent agent determines its action, based not on its real observation o_t^i, but on the expected observation $\hat{o}_t^i(a_t, H_t)$ estimated from the observation history of the learning agent according to equation (3.7). Since, however, our target is to deal with realistic POMDP problems consisting of a huge number of states, reducing the computation time is crucial. The computer simulation results showed that our RL method is applicable to such a realistic problem and also to more difficult problems with multiple agents.

Existing algorithms have not attained the level of human players, unlike in perfect information games like Tesauro's TD-gammon (Tesauro, 1994). A possible reason is that the transition over the observation space of imperfect information games does not satisfy the Markov property, and the conventional RL methods developed for MDP problems are not suitable for such non-Markovian problems, which is demonstrated in Fig. 3.2 by using the actor-critic algorithm.

There have been many multi-agent RL studies applied to simplified problems (Littman, 1994a; Hu & Wellman, 1998; Nagayuki, Ishii, & Doya, 2000; Salustowicz, Wiering, & Schmidhuber, 1998; Sandholm & Crites, 1995; Sen, Sekaran, & Hale, 1994; Tan, 1993), and some studies attempted to solve realistic problems like the elevator dispatch problem (Crites, 1996; Crites & Barto, 1996a, 1996b), which, however, suggested that the performance was not good when there was unobservable information. Our study aimed at presenting an RL method applicable to large-scale partially observable problems with multiple agents, and we have successfully created an experienced-level player of the game Hearts. To overcome the partial observability which occurs inevitably in multi-agent problems, the learning agent estimates the unobservable state variables with the several approximation techniques, and to

cope with the multi-agent property, it has multiple forward models to predict the environmental behavior, and they are trained individually. When one opponent agent changes its policy, this is enough to make the corresponding predictor adapt to the change. Since our method enables the agent to adapt to environmental changes quickly, when formulated as a single-agent POMDP problem, it can be applied to multi-agent problems. The experimental results demonstrated that our RL method can be applied to a realistic multi-agent problems with partial observability.

Although the expected observation $\hat{o}_t^i(a_t, H_t)$ contains the estimation of the unobservable states based on the history information, it does not propagate the belief information, because the expected observation in equation (3.8) is calculated under the assumption that the unobservable states are distributed with uniform probability. In addition, this mean-field-like analog approximation may cause the deviation to approximate the real distribution of the unobservable states, especially when the distribution has multiple local peaks in a sparse space. This indicates less uncertainty in the environment. Such an analog approximation with no belief propagation may cause the estimation accuracy of the utility value to deteriorate; consequently, we plan to explore more effective techniques in our next work.

Chapter 4

Model-based Reinforcement Learning for Partially Observable Game with Sampling-based State Estimation

This chapter presents an alternative approach to a model-based reinforcement learning for the same target as that of the previous chapter.*[1] As discussed in Section 3.4, there are two crucial problems in our approach: first, the estimated distribution of unobservable states may deviate from the real distribution due to the estimation error incurred by using the analog approximation; and second, the estimation process does not propagate belief information over time due to the uniform assumption. They may cause estimation error of the action values, and the performance of the agent may deteriorate even with less uncertainty in the environment. To solve these problems, we use a sampling technique in which the heavy integration required for estimation and prediction can be approximated by using a plausible number of samples. Computer simulation results show that the new RL method based on a sampling method can solve the remaining problems and attain a dramatic improvement over the previous one.

4.1 Model

In our RL method, the agent selects an action according to the greedy policy:

$$\pi(H_t) = \underset{a_t}{\text{argmax}}\ U(H_t, a_t), \tag{4.1}$$

where $U(H_t, a_t)$ is the utility function at time step t. This function is defined as an expectation of a one-step-ahead future value with respect to the belief state and

*[1] The contents of this chapter appears in our papers (Fujita & Ishii, 2005, 2006, 2007).

transition probability:

$$U(H_t, a_t) = \sum_{s_t \in \mathcal{S}} P(s_t|H_t) \sum_{s_{t+1} \in \mathcal{S}} P(s_{t+1}|s_t, a_t)[R(s_t, a_t, s_{t+1}) + \gamma V(s_{t+1})], \quad (4.2)$$

where $R(s_t, a_t, s_{t+1})$ denotes an immediate reward at time step t, and $V(s_{t+1})$ denotes the state value function of the next state s_{t+1}. γ denotes a discount factor. In large-scale problems, it is difficult to learn the value function over the belief space, because optimization of the value function, whose input is a probability distribution over the high-dimensional state space, is too complex and requires heavy computation. We then use the completely-observable approximation (Littman et al., 1995); the agent maintains the state value function so that the self-consistency equation holds on the underlying MDP, and calculates the state-action value by a one-step-ahead prediction (the second summation in equation (4.2)). After that, it calculates the history-action utility as an expectation of the state-action utility with respect to the belief state (the first summation in equation (4.2)), under the knowledge that the optimal value function for the belief space can be approximated well by a piecewise linear and convex function (Smallwood & Sondik, 1973).

The calculation of the utility function, however, includes three difficulties:

(a) the summations in equation (4.2) over possible current states s_t and next states s_{t+1} have computational intractability because there are so many state candidates in a realistic problem;

(b) the computation for constructing the belief state $P(s_t|H_t)$ over possible current states is intractable due to the large state space and high dimensionality; and

(c) the prediction of possible next states is difficult because the environmental model $P(s_{t+1}|s_t, a_t)$ is unknown for the learning agent and may change in a multi-agent setting.

Some effective approximations, therefore, are required for avoiding the above difficulties. Note that these problems above are the same as the problems in our previous study (Ishii et al., 2005, described in Chapter 3).

To avoid the computational intractability problem (a), we use sampling-based approximation; the learning agent obtains independent and identically distributed (i.i.d.) random samples, $\hat{s}_t$ and $\hat{s}_{t+1}$, whose probabilities are proportional to the belief state $P(s_t|H_t)$ and the environmental model $P(s_{t+1}|s_t, a_t)$, respectively. Note here that each sampled $\hat{s}_{t+1}$ depends on a certain sampled $\hat{s}_t$. The utility function in equation

(4.2) can then be approximated as

$$U(H_t, a_t) \approx \sum_{j=1}^{N} P(\hat{s}_t^{(j)}|H_t) \sum_{k=1}^{K} P(\hat{s}_{t+1}^{(k)}|\hat{s}_t^{(j)}, a_t) \left[R(\hat{s}_t^{(j)}, a_t, \hat{s}_{t+1}^{(k)}) + \gamma V(\hat{s}_{t+1}^{(k)}) \right]. \quad (4.3)$$

Samples of the current state $\hat{s}_t$ can be obtained by sequential Monte Carlo methods such as particle filtering (Gilks, Richardson, & Spiegelhalter, 1996); samples of the previous state $\hat{s}_{t-1}$, whose probability is proportional to the previous belief state $P(s_{t-1}|H_{t-1})$, are diffused into the next time step according to equation (2.3). This process is repeated N times, and the agent obtains estimated current states $\{\hat{s}_t^{(j)}|j = 1, \cdots, N\}$. Samples of the next state $\hat{s}_{t+1}$ are obtained by a simple sampling technique; K samples are drawn from the environmental model $P(s_{t+1}|s_t, a_t)$ given a sampled current state $\hat{s}_t$ and an action a_t. This sampling is repeated K times, and the agent obtains predicted next states $\{\hat{s}_{t+1}^{(k)}|k = 1, \cdots, K\}$ for each of N possible current states. The two summations in equation (4.2) are thus simultaneously approximated by using KN samples in equation (4.3). In large-scale and multi-agent problems, however, this naive sampling approach is insufficient due to difficulties (b) and (c); we then need further devices, as described below.

To avoid difficulty (b), we do not deal with the whole history H_t but use a one-step history $h_t = \{(o_t, -), (o_{t-1}, a_{t-1})\}$, which leads us to make a simplification (A): a belief state represents a simple one-step prior knowledge about states, but does not carry the complete likelihood information. The history H_t contains two kinds of information. The first is about impossible states at the t-th turn; for example, in the game Hearts, if an agent played ♡9 after a leading card ♣3 in a past trick, the agent no longer has any club cards at the t-th turn and any state in which this agent holds club cards is impossible (cf. Appendix B). The second is about likelihood, considering the characteristics of the opponent agents; for example, in the same situation as above, it is unlikely for the agent to have any heart card higher than ♡9. Although the belief state $P(s_t|H_t)$, which is a sufficient statistic for the history H_t, should involve these two kinds of information, we partly ignore the latter kind by replacing the whole history H_t with a one-step history h_t; namely, the belief state $P(s_t|H_t)$ is approximated by the partial belief state $P(s_t|h_t)$. Although this simplification enables us to estimate unobservable states easily, it may lead to bias in the estimation of the complete belief state; in this approximation,

$$P(s_t|H_t) \approx P(s_t|h_t) \propto \sum_{s_{t-1}} P(s_{t-1}) P(o_t|s_t) P(s_t|s_{t-1}, a_{t-1}), \quad (4.4)$$

we assume uniform distribution for the previous unobservable states $P(s_{t-1})$, whereas the complete belief state $P(s_t|H_t)$ contains the previous information $P(s_{t-1}|H_{t-1})$ by

performing an exact belief propagation. Note, however, that this partial belief state attains a one-step belief propagation, unlike the previous study (Ishii et al., 2005, described in Chapter 3). No impossible state, on the other hand, is considered in light of the former type of information, but each possible state has a one-step likelihood between the $(t-1)$-th and t-th time steps. Although the maintenance of likelihood over all possible states requires heavy computation and a large amount of memory in many realistic problems, even with the sampling approximation, this simplification enables us to estimate internal states easily at each time step.

To solve problem (c), the learning agent uses action predictors, which is the same as our previous approach described in Chapter 3. Since there are M opponents' actions within the transition from a state s_t to the next state s_{t+1} in multi-agent problems, the transition probability $P(s_{t+1}|s_t, a_t)$ in equation (4.2) is represented by using the real action selection probability $P(a_t^i|o_t^i, \phi^i)$ of agent i:

$$P(s_{t+1}|s_t, a_t)$$
$$= \sum_{\{s_t^1,\cdots,s_t^M\}\in\mathcal{S}^M} \sum_{\{a_t^1,\cdots,a_t^M\}\in\mathcal{A}^M} \prod_{j=0}^{M} P(s_t^{j+1}|s_t^j, a_t^j) \prod_{i=1}^{M} \sum_{o_t^i\in\mathcal{O}} P(a_t^i|o_t^i, \phi^i) P(o_t^i|s_t^i). \quad (4.5)$$

In many multi-agent games with partial observability, the environmental dynamics are deterministic and have two properties:

(i) the state s_t^{i+1} can be uniquely determined given a state s_t^i and an action a_t^i, that is, $P(s_t^{i+1}|s_t^i, a_t^i)$ is 1 for a certain state s_t^{i+1}, otherwise 0; and

(ii) the observation o_t^i can also be determined without any ambiguity given a state s_t^i, that is, $P(o_t^i|s_t^i)$ is 1 for a certain observation o_t^i, otherwise 0.

Note that these properties above are the same as those of our previous study. Under these properties, equation (4.5) is simplified to

$$P(s_{t+1}|s_t, a_t) = \sum_{\{a_t^1,\cdots,a_t^M\}\in\mathcal{A}^M(s_t,a_t,s_{t+1})} \prod_{i=1}^{M} P(a_t^i|o_t^i, \phi^i), \quad (4.6)$$

where $\mathcal{A}^M(s_t, a_t, s_{t+1})$ denotes the set of possible sequences of opponents' actions $\{a_t^1, \cdots, a_t^M\}$ in which the state s_t reaches the next state s_{t+1} after the action a_t. According to property (i), given a current state s_t and an action a_t, determining the next state s_{t+1} is the same as determining opponent agents' actions $\{a_t^1, \cdots, a_t^M\}$. For obtaining samples of the next state $\hat{s}_{t+1}$, therefore, the agent predicts all opponent

agents' actions $\hat{a}_t^1, \cdots, \hat{a}_t^M$. A sample path for equation (4.6) is now given by

$$P(\hat{s}_{t+1}|\hat{s}_t, a_t) = \prod_{i=1}^{M} P(\hat{a}_t^i|\hat{o}_t^i, \phi^i), \tag{4.7}$$

where $\hat{s}_{t+1}$ is the sampled next state so that it is consistent with the current state $\hat{s}_t$, the action a_t and the opponent agents' actions $\{\hat{a}_t^1, \cdots, \hat{a}_t^M\} \in \mathcal{A}^M(\hat{s}, a_t, \hat{s}_{t+1})$, and the observation $\hat{o}_t^i$ of agent i is uniquely determined (that is, constant) by the current state $\hat{s}_t$, the action a_t and the previous opponents' actions $\hat{a}_t^1, \cdots, \hat{a}_t^{i-1}$ according to property (ii).

Since the action selection probability $P(a_t^i|o_t^i, \phi^i)$ of opponent agent i is unknown for the learning agent, as described in problem (b), the agent uses action predictors; each action predictor learns the action selection model of the corresponding opponent agent. The real environmental model in equation (4.7) is then approximated by M action predictors:

$$P(\hat{s}_{t+1}|\hat{s}_t, a_t) \approx P(\hat{s}_{t+1}|\hat{s}_t, a_t, \hat{\Phi}) = \prod_{i=1}^{M} P(\hat{a}_t^i|\hat{o}_t^i, \hat{\phi}^i), \tag{4.8}$$

where $\hat{\Phi} = \{\hat{\phi}^1, \cdots, \hat{\phi}^M\}$. The action selection probability of the i-th opponent agent, $P(a_t^i|o_t^i, \phi^i)$, is approximated by the i-th action predictor ($i = 1, \cdots, M$); $\hat{\phi}^i$ in equation (4.8) is not the real strategy ϕ^i by agent i but a strategy approximated by the i-th action predictor. Since each action predictor is realized as a function approximator, $\hat{\phi}^i$ denotes, in effect, its parameters (see section 4.2). The agent predicts that the i-th opponent agent selects an action a_t^i according to the soft-max policy:

$$P(a_t^i|o_t^i, \hat{\phi}^i) = \frac{\exp(F^i(o_t^i, a_t^i; \hat{\phi}^i)/T_m^i)}{\sum_{a_t^j \in \mathcal{A}} \exp(F^i(o_t^i, a_t^j; \hat{\phi}^i)/T_m^i)}, \tag{4.9}$$

where $F^i(o_t^i, a_t^i)$ denotes an assumed utility of taking action a_t^i for an observation o_t^i, and T_m^i is a constant which denotes the assumed randomness of agent i's policy.

Using the above two devices to deal with problems (b) and (c), equation (4.3) is now further approximated as

$$U(H_t, a_t) \approx \sum_{j=1}^{N} P(\hat{s}_t^{(j)}|h_t) \sum_{k=1}^{K} \prod_{i=1}^{M} P(\hat{a}_t^{i,(k)}|\hat{o}_t^{i,(j)}, \hat{\phi}^i) \left[R(\hat{s}_t^{(j)}, a_t, \hat{s}_{t+1}^{(k)}) + \gamma V(\hat{s}_{t+1}^{(k)}) \right], \tag{4.10}$$

by replacing the belief state $P(\hat{s}_t^{(j)}|H_t)$ and the environmental model $P(\hat{s}_{t+1}^{(k)}|\hat{s}_t^{(j)}, a_t)$ with the partial belief state $P(\hat{s}_t^{(j)}|h_t)$ and the action predictors $P(\hat{s}_{t+1}^{(k)}|\hat{s}_t^{(j)}, a_t, \hat{\Phi}) =$

$\prod_{i=1}^{M} P(\hat{a}_t^{i,(k)}|\hat{o}_t^{i,(j)}, \hat{\phi}^i)$, respectively, where $\hat{a}^{i,(k)}$ is a constituent of the action sequence $\{\hat{a}^{1,(k)}, \cdots, \hat{a}^{M,(k)}\} \in \mathcal{A}^M(\hat{s}_t^{(j)}, a_t, \hat{s}_{t+1}^{(k)})$. According to equation (4.4), samples of the current state $\hat{s}_t$ are obtained by the following four steps: the first step is to sample the previous state $\hat{s}_{t-1}^*$ according to uniform distribution so as not to violate the whole history H_t (no impossible state being sampled); the second step is to calculate the action selection probability $P(a_{t-1}^i|o_{t-1}^i, \hat{\phi}^i)$ for the action a_{t-1}^i actually taken by agent i according to equation (4.9) and to iterate this step for all opponent agents' actions $a_{t-1}^1, \cdots, a_{t-1}^M$; the third step is to calculate the one-step likelihood $P(\hat{s}_t^*|\hat{s}_{t-1}^*, a_{t-1}, \hat{\Phi})$ according to equation (4.8) and obtain a sample of the current state $\hat{s}_t^*$ given the previous state $\hat{s}_{t-1}^*$, the previous action a_{t-1} and the opponents' actual actions $a_{t-1}^1, \cdots, a_{t-1}^M$; and the last step is to accept $\hat{s}_t^*$ as $\hat{s}_t^{(j+1)}$ with probability $p = \min\left\{1, \frac{P(\hat{s}_t^*|\hat{s}_{t-1}^*, a_{t-1}, \hat{\Phi})}{P(\hat{s}^{(j)}|\hat{s}_{t-1}^{(j)}, a_{t-1}, \hat{\Phi})}\right\}$, otherwise $\hat{s}_t^*$ is rejected. These four steps are repeated N times, and the agent obtains estimated current states $\{\hat{s}_t^{(j)}|j = 1, \cdots, N\}$. The samples of the current state $\hat{s}_t$ include only a one-step likelihood information (simplification (A)); in the first step above, samples of the previous state $\hat{s}_{t-1}^*$ are obtained by uniform sampling, ignoring the previous history. For approximating a large-scale posterior by using a limited number of samples, this uniform prior sampling is a plausible approach, because the complete posterior belief is likely to be very sparse and many possible states tend to have similar probability. In addition, in problems whose state space is discrete with a deterministic observation process, samples $\hat{s}_t$ inconsistent with an actual observation o_t necessarily disappear; for example, if an agent played ♡9, all state candidate in which another agent has ♡9 are no longer useful. In large-scale and high-dimensional problems such as the game Hearts, especially, very few samples remain after each observation; this causes difficulty in performing the incremental maintenance of complete belief states by using a limited number of samples according to equation (2.3). The learning agent then discards the previous information $P(s_{t-1}|H_{t-1})$ and obtains new samples $\hat{s}_{t-1}^*$ with uniform probability $P(s_{t-1})$, before calculating a one-step likelihood. Although this sampling technique may lead to bias in the estimation of the belief state due to ignoring the previous history H_{t-1}, simplification (A) nonetheless provides us with an effective approximation to make intractable problems easier.

Samples of the next state $\hat{s}_{t+1}$ are obtained by a simple sampling technique. K samples are drawn by the following three steps: the first step is to calculate the action selection probability for the opponent agent i according to equation (4.9); the second step is to select a possible action $\hat{a}_t^i$ according to the action selection probability, and the first and second steps are iterated alternately for all opponent agents; and the last

step is to compute the next state $\hat{s}_{t+1}$ given the estimated current state $\hat{s}_t$, the action a_t and the opponent agents' actions $\hat{a}_t^1, \cdots, \hat{a}_t^M$. These three steps are repeated K times, and the agent obtains predicted next states $\{\hat{s}_{t+1}^{(k)} | k = 1, \cdots, K\}$, with the learned model, for each of the N possible current states.

The three approximations described above (the sampling technique, partial belief state and action predictor) enable us to solve large-scale and partially observable problems. Our method, for example, provides an effective solution to multi-agent problems whose underlying state space is discrete, including various multi-agent games.

4.2 Function approximators with feature extraction

In large-scale problems, the state space often has high dimensionality. In card games, for example, the state s_t is a 52-dimensional vector each of whose dimensions represents the status of the corresponding card. This high dimensionality causes the performance of the RL agent to deteriorate due to the large and redundant representation. To achieve effective learning in a realistic problem, therefore, it is beneficial to use feature extraction techniques, by considering the properties of the target problem, for the input and output of the value function and action predictors. Here, just same as our previous study (Ishii et al., 2005, described in Section 3.2 in the previous chapter), we explain the feature extraction techniques used to apply our approach to a specific domain: the card game Hearts.

To reduce the dimensionality, the 52-dimensional state s_t is converted to a 36-dimensional input p_t according to the following representation:

- $p_t[8 \times i + 1]$: the number of club cards in the agent i's ($i = 0, 1, 2, 3$) hand,
- $p_t[8 \times i + 2]$: same as $p_t[8 \times i + 1]$, but the suit is diamonds,
- $p_t[8 \times i + 3]$: same as $p_t[8 \times i + 1]$, but the suit is spades excepting ♠Q, ♠K and ♠A,
- $p_t[8 \times i + 4]$: a binary value for ♠Q,
- $p_t[8 \times i + 5]$: a binary value for ♠K,
- $p_t[8 \times i + 6]$: a binary value for ♠A,
- $p_t[8 \times i + 7]$: the number of heart cards in the agent i's hand which are weaker than the strongest card in the current trick (only if the leading card is a hart, otherwise zero),
- $p_t[8 \times i + 8]$: the number of heart cards in the agent i's hand which are stronger than the strongest card in the current trick (only if the leading card is a hart, otherwise the number of all heart cards in the agent i's hand), and
- $p_t[33]$ to $p_t[36]$: a bit sequence.

The binary values of ♠Q, ♠K and ♠A take either 1 or 0, which represent whether agent i has the card or not, respectively. Since these three cards are the most important in the game Hearts, we allocate one dimension to each card. Because heart cards are also important, we allocate twice as many dimensions to heart cards as to other suits. The bit sequence represents the playing order in the current trick. For example, when the learning agent is the second player in the current trick (the t-th playing turn of the learning agent), $\{p_t[33], p_t[34], p_t[35], p_t[36]\} = \{0, 1, 0, 0\}$. When the next state s_{t+1} is obtained by the sampling process, equation (4.10) can be calculated by replacing $V(s_{t+1})$ with $V(p_{t+1})$ according to the above feature extraction.

The observation o_t^i for agent i is also a 52-dimensional vector; each dimension of the observation vector represents an observable status of the corresponding card such that each status represents whether the card has already played or not. To reduce the dimensionality, the 52-dimensional observation o_t^i is converted to a 26-dimensional input q_t^i according to the following representation:

- $q_t^i[1]$: the number of club cards in the agent i's hand which are weaker than the strongest card already played in the current trick (only if the leading card is a club, otherwise zero),
- $q_t^i[2]$: the number of club cards in the agent i's hand which are stronger than the strongest card already played in the current trick (only if the leading card is a club, otherwise the number of all club cards in the agent i's hand),
- $q_t^i[3]$: same as $q_t^i[1]$, but the suit is diamonds,
- $q_t^i[4]$: same as $q_t^i[2]$, but the suit is diamonds,
- $q_t^i[5]$: same as $q_t^i[1]$, but the suit is spades excepting ♠Q, ♠K and ♠A,
- $q_t^i[6]$: same as $q_t^i[2]$, but the suit is spades excepting ♠Q, ♠K and ♠A,
- $q_t^i[7]$: a binary value for ♠Q,
- $q_t^i[8]$: a binary value for ♠K,
- $q_t^i[9]$: a binary value for ♠A,
- $q_t^i[10]$ to $q_t^i[22]$: a binary value for each heart card, and
- $q_t^i[23]$ to $q_t^i[26]$: a bit sequence.

The binary values of ♠Q, ♠K, ♠A and heart cards are defined similar to p_t. Since the statuses of these cards are important for predicting the next action in the game Hearts, we allocate one dimension to each card. The bit sequence is the same as p_t.

Given an extracted input q_t^i, the utility function F^i of agent i returns a 26-dimensional output vector r_t^i each of whose dimensions represents:

- $r_t^i[1]$: the merit value where agent i plays an arbitrary club card which is weaker than the strongest card in the current trick,

- $r_t^i[2]$: the merit value where agent i plays the weakest club card (among the remaining club cards) which is stronger than the strongest card in the current trick,
- $r_t^i[3]$: the merit value where agent i plays a club card (neither the weakest nor the strongest among the remaining club cards) which is stronger than the strongest card in the current trick,
- $r_t^i[4]$: the merit value where agent i plays the strongest card (in remaining club cards) which is stronger than the strongest card in the current trick,
- $r_t^i[5]$: same as $r_t^i[1]$, but the suit is diamonds,
- $r_t^i[6]$: same as $r_t^i[2]$, but the suit is diamonds,
- $r_t^i[7]$: same as $r_t^i[3]$, but the suit is diamonds,
- $r_t^i[8]$: same as $r_t^i[4]$, but the suit is diamonds,
- $r_t^i[9]$: the merit value where agent i plays an arbitrary spade card (except ♠Q, ♠K and ♠A) which is weaker than the strongest card in the current trick,
- $r_t^i[10]$: the merit value where agent i plays an arbitrary spade card (except ♠Q, ♠K and ♠A) which is stronger than the strongest card in the current trick,
- $r_t^i[11]$: the merit value where agent i plays ♠Q,
- $r_t^i[12]$: the merit value where agent i plays ♠K,
- $r_t^i[13]$: the merit value where agent i plays ♠A, and
- $r_t^i[14]$ to $r_t^i[26]$: the merit value where agent i plays each heart card.

The merit value represents a tendency where agent i plays the corresponding card; the larger the merit value, the more likely the agent is to play the card (see equations (4.9) and (4.11)). The output representation r_t^i is the same as that of our previous study described in Chapter 3, while the input representation q_t^i is different due to the difference of the estimation process.

According to the above feature extraction technique, the action selection probability of agent i is calculated by the following four steps: first, the observation o_t^i for agent i can be obtained when the state s_t is given by the sampling process; second, the 52-dimensional observation vector o_t^i is converted to the extracted and hence compressed 26-dimensional input q_t^i according to the above representation; third, the utility function F^i returns the compressed 26-dimensional output r_t^i given the input q_t^i; and fourth, the i-th action predictor calculates the action selection probability:

$$P(r_t^i|q_t^i,\hat{\phi}^i) = \frac{\exp(F^i(q_t^i, r_t^i; \hat{\phi}^i)/T_m^i)}{\sum_{a_t^j \in \mathcal{A}^j} \exp(F^i(q_t^i, r_t^j; \hat{\phi}^i)/T_m^i)}, \tag{4.11}$$

where $\mathcal{A}^j$ denotes the set of possible actions for agent j. Note that the utility $F^i(o_t^i, a_t^i)$ in equation (4.9) is replaced with $F^i(q_t^i, r_t^i)$ by the feature extraction. The action

selection probability $P(r_t^i|q_t^i, \hat{\phi}^i)$ in equation (4.11) consists of a 26-dimensional vector each of whose dimensions represents an occurrence probability r_t^i, which is different from the primitive action a_t^i. The action selection probability for a compressed action r_t^i is, therefore, converted to that of a primitive action a_t^i.

The feature extraction conducted by considering the properties of the target problem allows us to reduce the dimensionality and improve the learning efficiency. Large-scale realistic problems, however, still have huge state spaces and high dimensionality; for example, there are about 10^{12} possible inputs q_t^i for the action predictor. This causes difficulty in learning the action selection model directly from a limited number of learning samples, by using a simple table lookup approach such as a multinomial model, due to the large number of effective parameters in the lookup table. To overcome this difficulty, we use function approximators for the value function and action predictors. A function approximator can approximate an input-output relation as a non-linear regression model with a reasonable number of parameters, and it enables the agent to use its generalization ability for unknown situations.

The value function is approximated by a normalized Gaussian network (NGnet) (Sato & Ishii, 2000) with the feature extraction technique for its input. The NGnet $V(p_t)$ is trained so as to learn the relationship between the compressed input p_t and the cumulative reward via the following three steps: first, the real state $\bar{s}_t$ and the discounted cumulative reward (return) $\bar{R}_t = \sum_{i=t}^{13} \gamma^i R(s_i, a_i, s_{i+1})$ for each time t are available after one game has finished, where the immediate reward is defined as $R(s_t, a_t, s_{t+1}) = n$ when the agent gets n penalty points (n may be 0) between the t-th and $(t+1)$-th play; second, the 52-dimensional state vector $\bar{s}_t$ is converted to the compressed 36-dimensional input $\bar{p}_t$ according to the above feature extraction; and third, these two values, the input vector $\bar{p}_t$ and the corresponding scalar output $\bar{R}_t$, are given to the NGnet for supervised learning, that is, the NGnet is updated so as to approximate the return according to the Monte Carlo RL method (Sutton & Barto, 1998). The discount factor γ is 1.0 in our application. The property that the sequence of the real state can be available for learning is specific for many card games. Nevertheless, several POMDP algorithms would be applicable to learn the state value function in various partially observable environments (Hauskrecht, 2000), for example by calculating the expectation of the value function with respect to the belief state. The NGnet showed the smallest TD error and achieved the fastest learning in our problem compared with other function approximators*2; this is partly attributable to

*2 We compared three function approximators in their approximation ability for the problem: normalized Gaussian network (NGnet), multi-layered perceptron (MLP) and radial basis function (RBF).

the fact that it is a piece-wise linear model with multiple Gaussian connections, and its learning is based on the on-line EM algorithm whose coordinate-ascent optimization is often faster than simple gradient methods.

The utility function F^i in equations (4.9) and (4.11) is represented by a multi-layered perceptron (MLP) with the feature extraction for its input and output. The MLPs are trained according to the four steps similar to those for the NGnet: first, the actual observation $\bar{o}_t^i$ for agent i is available after one game has finished; second, the 52-dimensional observation vector $\bar{o}_t^i$ is converted to the compressed 26-dimensional input $\bar{q}_t^i$; third, the 26-dimensional output $\bar{r}_t^i$ is calculated from the actual action $\bar{a}_t^i$; and fourth, these two vectors, the input $\bar{q}_t^i$ and output $\bar{r}_t^i$, are given to the MLP, that is, it is trained by supervised learning based on the error back-propagation method. The output vector $\bar{r}_t^i$ consists of elements of 1 or 0; the target value of the dimension corresponding to the actually taken action $\bar{a}_t^i$ is 1, otherwise 0. Although the parameters of the MLP are tuned so that the output represents non-negative values (from 0 to 1) in each dimension, the utility function does not directly represent the probability. In other words, the summation over possible actions of agent i is not always 1, because the set of possible actions $\mathcal{A}^j$ may depend on the state s_t. We then use the soft-max normalization in equation (4.11), after removing impossible actions. An MLP showed the best prediction accuracy in our problem compared with other function approximators; this is partly because it is a global nonlinear model and has an adequate representation ability in this problem whose input and output have high dimensionality.

To achieve effective learning, using feature extraction suited for target problems and using function approximators are crucial. Sturtevant and White (2006), for example, examined features of the game Hearts to avoid ♠Q and heart cards, and constructed a feature representation suited for playing the game. We used a similar idea; it enables the agent to understand the important information by taking advantage of the game's properties, and to improve the learning speed and approximation ability of function approximators.

4.3 Computer simulations

We applied our RL method to the card game Hearts, which is a well-defined example of large-scale and multi-agent problems with partial observability. To evaluate our method, we carried out computer simulations where an agent trained by our RL method played against rule-based agents which have 66 general rules for playing cards from their hands. The performance of an agent can be evaluated by the acquired

penalty ratio, which is the ratio of the penalty points acquired by the agent to the total penalty points of the four agents. If the four agents have equal strength, their penalty ratio averages 0.25. The rule-based agent used in this study is stronger than the previous one (Ishii et al., 2005, described in Section 3.3 of the previous chapter), due to the improvement in the rules; comparisons between the previous rule-based agent and the current rule-based agent are summarized in Table 4.1. Although the previous rule-based agent was an "experienced"-level player, the current rule-based agent has almost the same strength as a human Hearts player; when this rule-based agent challenged a human player, the acquired penalty ratio was 0.256 (Table 4.1). The learning agent based on our previous RL method then remained weaker than this rule-based agent even after 100,000 training games (data not shown).

Since the outcome of this game tends to depend on the initial card distribution (for example, an expert player with a bad initial hand may be defeated by an unskilled player), we prepared a fixed data set for the evaluation; the data set is a collection of initial card distributions for 100 games, each of which was generated randomly in advance. In the evaluation games, the initial cards were distributed according to this data set. Since performance is influenced by seat position (that is, an agent may have an advantage/disadvantage based on its seat position if the agents have different strengths), we rotated the agents' positions for each initial hand to eliminate this bias; each of the 100 evaluation games was repeated four times with the four types of seating position. The performance of each agent, therefore, is evaluated by the 400 fixed and unbiased games. Note that learning of the agent was suspended during the evaluation games. Each learning run comprised several sets of 500 games, in which initial cards were distributed to the four agents at random and seat positions of the agents were determined randomly. In each learning run, accordingly, 400 evaluation games and 500 learning games were alternated.

4.3.1 Single agent learning in stationary environment

Figure 1 shows the result when the agent trained by our method challenged the three rule-based agents. Each point and error bar represent the average and standard deviation of the penalty ratio, respectively, for the 400 evaluation games over 17 learning runs. The penalty ratio of the RL agent decreased as learning progressed, and after 5,000 training games the agent became significantly stronger than the rule-based agents. Since the agent showed a better performance than the rule-based agents after only several thousand training games, the new RL method based on a sampling method showed a dramatic improvement over the previous one, both in learning speed and in strength; our previous RL agent required about twenty times as many training

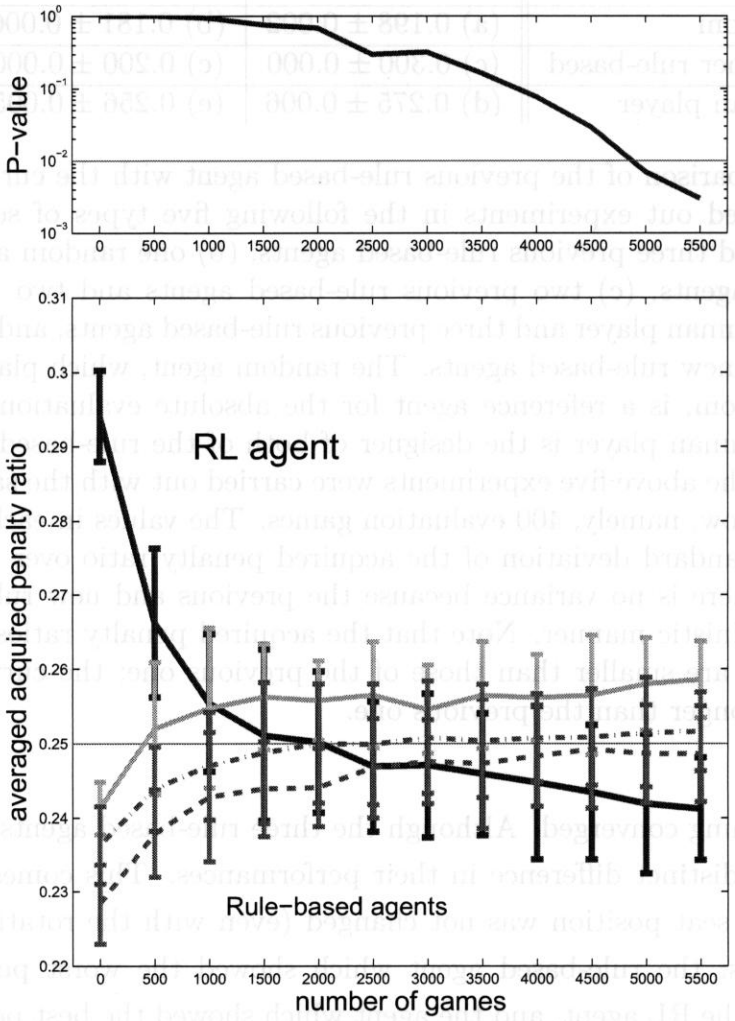

Fig. 4.1: Computer simulation result in an environment consisting of one learning agent trained by our RL method and three rule-based agents.

upper panel: P-values of the t-test where the null hypothesis is "the RL agent has the same strength as the rule-based agents" and the alternative hypothesis is "the RL agent is stronger than the rule-based agents". The test was done independently at each point on the abscissa. The horizontal line denotes the significance level of 1%. After 5,000 training games, the RL agent was significantly stronger ($p < 0.01$) than the rule-based agents.

lower panel: The abscissa denotes the number of training games and the ordinate denotes the penalty ratio acquired by each agent. We executed 17 learning runs, each consisting of 5,500 training games. Each point and error bar represent the average and standard deviation, respectively, for the 400 evaluation games over the 17 runs. The discount factor γ in equation (4.2) was 1.0, the constant T_m^i in equation (4.11) was 1.0, and the numbers of samples in equation (4.10) were $N = 80$ and $K = 20$. These parameters were heuristically determined.

vs.	Previous	Current
Random	(a) 0.198 ± 0.002	(b) 0.181 ± 0.006
Another rule-based	(c) 0.300 ± 0.000	(c) 0.200 ± 0.000
Human player	(d) 0.275 ± 0.006	(e) 0.256 ± 0.005

Table. 4.1: Comparison of the previous rule-based agent with the current rule-based agent. We carried out experiments in the following five types of setting: (a) one random agent and three previous rule-based agents, (b) one random agent and three new rule-based agents, (c) two previous rule-based agents and two new rule-based agents, (d) one human player and three previous rule-based agents, and (e) one human player and three new rule-based agents. The random agent, which played cards from its hand at random, is a reference agent for the absolute evaluation of the agents' strength. The human player is the designer of both of the rule-based agents. Three runs for each of the above five experiments were carried out with the same data set as in the figures below, namely, 400 evaluation games. The values in each cell represent the mean and standard deviation of the acquired penalty ratio over the three runs. In setting (c), there is no variance because the previous and new rule-based agents play in a deterministic manner. Note that the acquired penalty ratios of the current rule-based agent are smaller than those of the previous one; the current rule-based agent is thus stronger than the previous one.

games until learning converged. Although the three rule-based agents have the same rules, there is a distinct difference in their performances. This comes from the fact that the relative seat position was not changed (even with the rotation) during the evaluation games; the rule-based agent which showed the worst performance was always opposite the RL agent, and the agent which showed the best performance was always at the left side. Although the evaluation is unbiased, the agents' strength is biased (the RL agent is weaker than the rule-based agents before learning, but stronger after 5,000 training games), so that the trajectories of the learning curves diverged from each other. Figure 4.2 shows frequency distributions of penalty points for the RL agent and the rule-based agent in the same experiment as Figure 4.1. After 5,500 training games, the frequency of penalty points obtained by the rule-based agent is mainly distributed over higher penalty points than that of the RL agent. The frequencies of the RL agent for incurring 4 and 5 penalty points, on the contrary, markedly increased after the training games. This result suggests that the agent learned the policy so as to avoid many penalty points by instead receiving relatively few penalty points. This figure then supports the previous observation; the RL agent could acquire a good policy by interacting with the environment. We obtained a similar result to that shown in Figures 4.1 and 4.2 by using another evaluation data set (data not shown).

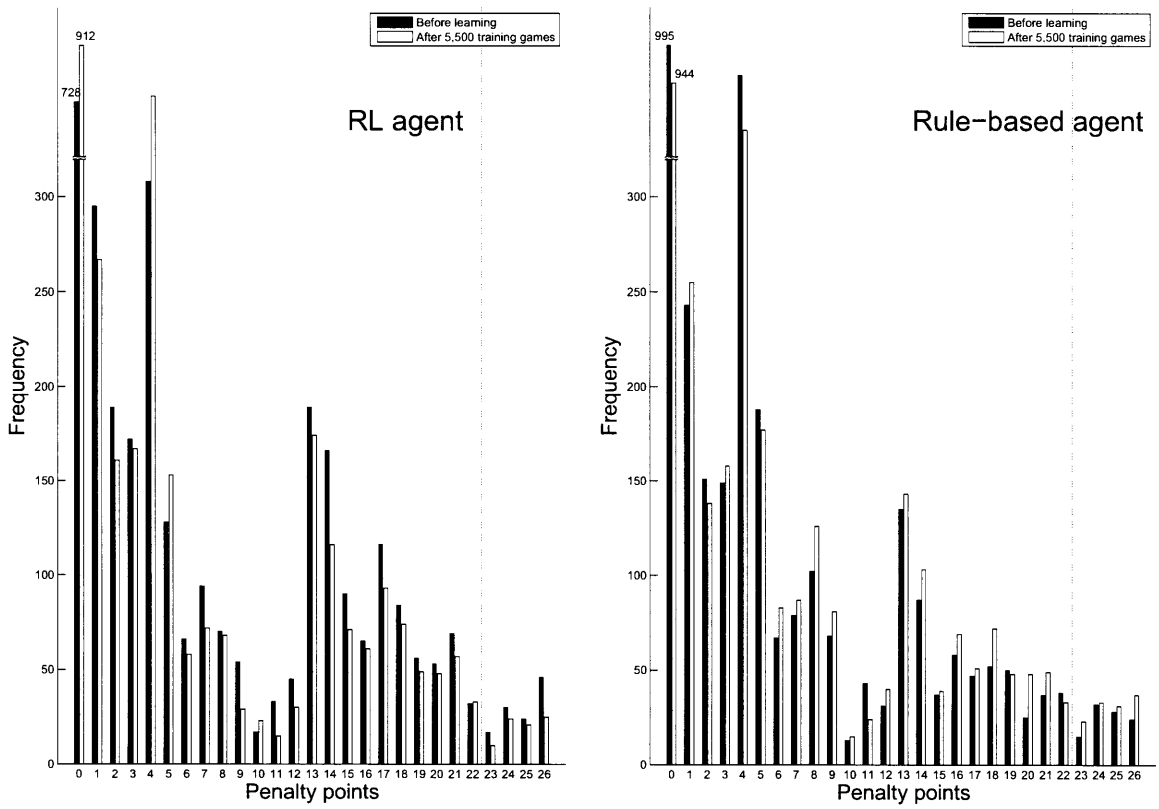

Fig. 4.2: Frequency distributions of penalty points obtained by the RL agent and the rule-based agent, before learning and after 5,500 training games.

left: The abscissa denotes the penalty point number and the ordinate denotes the frequencies of penalty points acquired by the RL agent. The vertical dashed line represents the boundary between incurring '23 or less' and '23 or more' penalty points. The total frequencies at the right side of the boundary are 117 and 80 before learning and after 5,500 training games, respectively.

right: The representation is the same as in the left panel except that the ordinate denotes the frequencies of penalty points acquired by the rule-based agent. The total frequencies at the right side of the boundary are 99 and 124 before learning and after 5,500 training games, respectively.

When the RL agent challenged the three rule-based agents used in our previous work (Ishii et al., 2005, described in Chapter 3), it showed an improved performance from the beginning of learning and finally became better than the rule-based agents; this is why we developed the new rule-based agent, which is stronger than the previous one. The improvement by our new RL method is attributed to the following two facts. First, the ability to approximate the utility function in equation (4.2) was improved by replacing the analog approximation method with a discrete sampling-based one.

In our previous work, to calculate the utility function, we applied the mean-field-like analog approximation to the problem, whose state space is discrete, by changing the order of summations; we calculated the summation over the current state with the approximation before calculating the summation over the next state. In this study, on the contrary, the summations are calculated in a straightforward way with the sampling-based approximation; each sampled state represents a discrete state, and therefore our new approximation is more suitable than the previous one. This enables us to calculate the expectation with a higher accuracy. Second, the expected future reward could be evaluated more accurately by the state value function. In our previous work, we made the value function learn over the observation space. Although this is an effective method for large-scale POMDP problems, it is difficult to obtain an accurate value due to the perceptual aliasing property in partially observable environments; the same observation may come from different states whose state values should be different, but the value function defined on the observation space cannot detect the difference between the values. In this study, in contrast, the learning agent predicts possible next states and evaluates a value from the value function defined on the state space. This enables the agent to perform accurate value prediction. The ideas used in our previous work were adequate for an environment with moderate complexity, constituted by the previous rule-based agents, but the limitation of that method precluded a more remarkable result. In this study, we have improved the old model so that the method works well within only several thousand training games, even in the harder environment constituted by the stronger rule-based agents.

4.3.2 Multi-agent learning in dynamic environments

In the experiment described above, our RL method was applied to the problem under the POMDP assumption that there is only one learning agent in the stationary environment. In the following, we apply our method directly to multi-agent environments where there are multiple learning agents, making the environment dynamic.

Figure 4.3 (left) shows the result when one agent trained by our RL method, one agent trained by the REINFORCE algorithm (Williams, 1992), a policy gradient-based RL method, and two rule-based agents played against each other. Note that our RL agent sat at the left side of the REINFORCE agent. The REINFORCE agent learned an action selection probability $P(a_t|o_t)$ with a feature extraction technique applied to its 52-dimensional input and output; an observation o_t was converted to a 25-dimensional input whose representation of each dimension was the same as that in our previous work, and its output was also converted to a 26-dimensional output whose representation was the same as that of the action predictor in our current

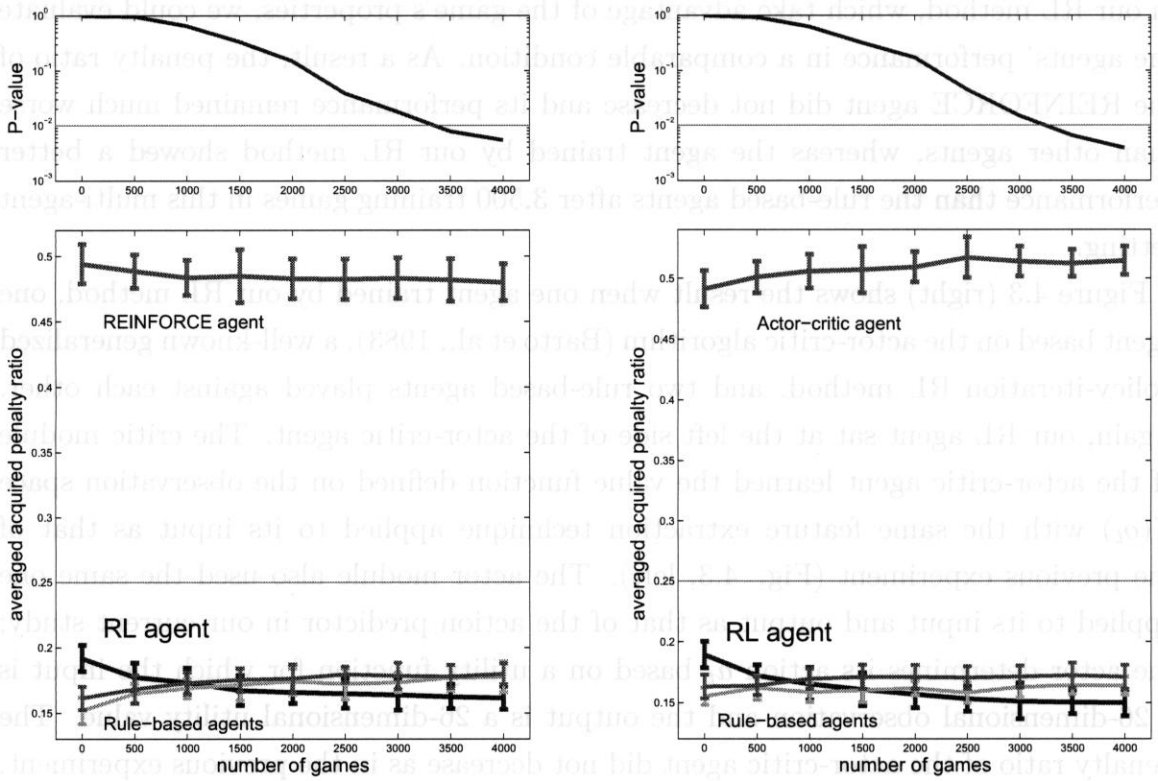

Fig. 4.3: Computer simulation result in an environment consisting of one learning agent trained by our RL method, one learning agent trained by another algorithm, and two rule-based agents.

left panels: Computer simulation result in an environment consisting of one learning agent trained by our RL method, one learning agent trained by the REINFORCE algorithm, and two rule-based agents.

right panels: Computer simulation result in an environment consisting of one learning agent trained by our RL method, one learning agent trained by the actor-critic algorithm, and two rule-based agents.

upper panels: P-values of the t-test where the null and alternative hypotheses are the same as in the previous experiment (Fig. 4.1). After 3,500 training games, the RL agent became significantly stronger than the rule-based agents, but the agent trained by another algorithm did not, in both experiments (P-values are not shown because they remained around 1).

lower panels: The abscissa and the ordinate denote the same as in Fig. 4.1, but the scale of the ordinate is larger here. We executed 15 learning runs, each consisting of 4,000 training games in both experiments. The parameter values and other experimental setups were also the same.

study (see section 4.2). Since these feature extraction processes are similar to those in our RL method, which take advantage of the game's properties, we could evaluate the agents' performance in a comparable condition. As a result, the penalty ratio of the REINFORCE agent did not decrease and its performance remained much worse than other agents, whereas the agent trained by our RL method showed a better performance than the rule-based agents after 3,500 training games in this multi-agent setting.

Figure 4.3 (right) shows the result when one agent trained by our RL method, one agent based on the actor-critic algorithm (Barto et al., 1983), a well-known generalized policy-iteration RL method, and two rule-based agents played against each other. Again, our RL agent sat at the left side of the actor-critic agent. The critic module of the actor-critic agent learned the value function defined on the observation space $V(o_t)$ with the same feature extraction technique applied to its input as that of the previous experiment (Fig. 4.3, left). The actor module also used the same one applied to its input and output as that of the action predictor in our current study; the actor determines its action a_t based on a utility function for which the input is a 26-dimensional observation and the output is a 26-dimensional utility value. The penalty ratio of the actor-critic agent did not decrease as in the previous experiment, whereas the agent trained by our RL method became significantly stronger than the rule-based agents after 3,500 training games.

These results are attributed to two points. The first is the disadvantage of learning over the observation space: as described above, it is difficult to obtain a good performance without solving ambiguity in partially observable problems, even with effective feature extraction. The second is the limitation of model-free approaches: those such as the REINFORCE and actor-critic methods are easy to apply to various problems, including POMDPs, but it is in fact hard to achieve a good result in complex multi-agent environments with highly restricted observations. Our experimental results show that our model-based RL method with sampling-based state estimation could overcome such difficulties, and then achieved a better performance than conventional RL methods.

It would be impractical to apply other existing RL methods to our problem. For example, using the LSTD algorithm (Bradtke & Barto, 1997) for Hearts would be infeasible, because it is very difficult to obtain a least-square solution in a large state space, due to the inevitable sparseness of learning samples. Using belief-state POMDP methods is also difficult, even with an appropriate approximation (Hauskrecht, 2000), because learning over the belief space is computationally heavy in large-scale problems, as discussed in section 2. On the contrary, our method can solve such problems.

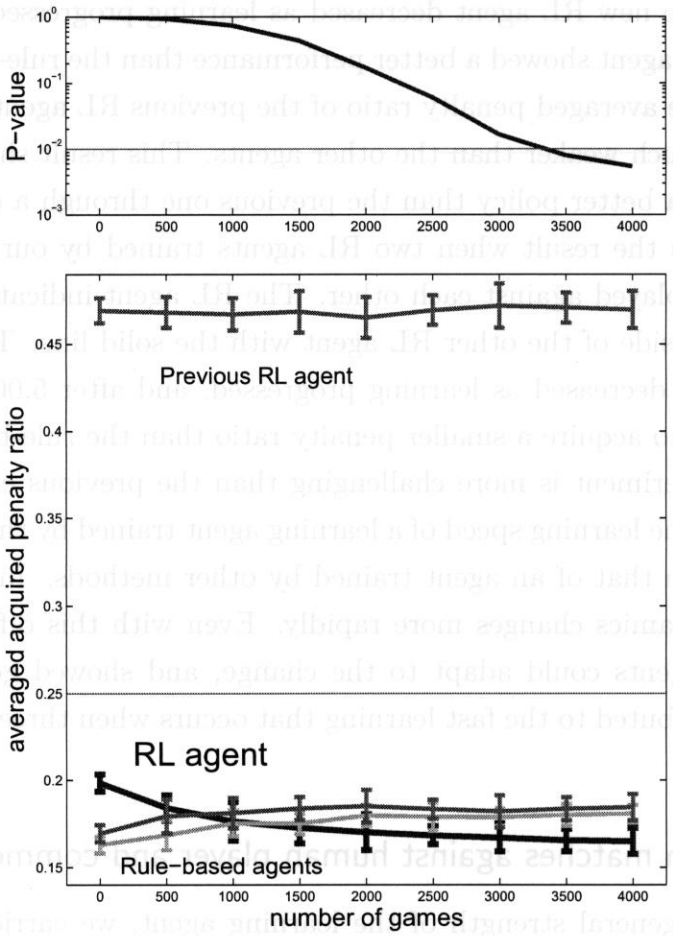

Fig. 4.4: Computer simulation result in an environment consisting of one learning agent trained by our RL method, one learning agent trained by our previous method, and two rule-based agents.

upper panel: P-values of the t-test where the null and alternative hypotheses are the same as in the previous experiment (Fig. 4.1). After 3,500 training games, the RL agent became significantly stronger than the rule-based agents, but the previous RL agent did not (P-values are not shown because they remained around 1).

lower panel: The abscissa and the ordinate denote the same as in Fig. 4.1, but the scale of the ordinate is larger here. We executed 16 learning runs, each consisting of 4,000 training games. The parameter values and other experimental setups were also the same.

Figure 4.4 shows the result when one agent trained by our RL method, one agent trained by our previous RL method, and two rule-based agents played against each other. As before, the new RL agent sat at the left side of the previous RL agent. The penalty ratio of the new RL agent decreased as learning progressed, and after 3,500 training games the agent showed a better performance than the rule-based agents. On the other hand, the averaged penalty ratio of the previous RL agent did not decrease and it remained much weaker than the other agents. This result shows that our new agent can acquire a better policy than the previous one through a direct match.

Figure 4.5 shows the result when two RL agents trained by our method and two rule-based agents played against each other. The RL agent indicated by the dashed line sat at the left side of the other RL agent with the solid line. The penalty ratios of both RL agents decreased as learning progressed, and after 5,000 training games both agents came to acquire a smaller penalty ratio than the rule-based agents. The setting of this experiment is more challenging than the previous experiments (Figs. 4.3, 4.4), because the learning speed of a learning agent trained by the new RL method is much faster than that of an agent trained by other methods. In other words, the environmental dynamics changes more rapidly. Even with this difficult multi-agent setting, the RL agents could adapt to the change, and showed good performance. This ability is attributed to the fast learning that occurs when three action predictors are used.

4.3.3 Validation matches against human player and commercial software

To validate the general strength of the learning agent, we carried out evaluation games with the human player, who is the same person evaluated in Table 4.1. Figure 4.6 shows the result when one RL agent trained by our method, two rule-based agents and the human player played together, with the RL agent sitting next to the human player. We used another evaluation data set for 25 games with seat rotation (that is, 100 games), and the learning run was executed twice. Each point of Figure 4.6, therefore, represents a mean over 200 games for the evaluation games.[*3] One hundred evaluation games in both learning runs were done before learning and after 1,000, 2,000, 3,000, 4,000 and 5,000 training games, and these training games were carried out with three rule-based agents; one rule-based agent was then replaced by the human player in each evaluation phase. This may cause the action predictor to deviate from the human player's action selection model, due to the difference between the strategy

[*3] To prevent the human player from remembering the fixed card distributions, the order of the evaluation games was shuffled in each evaluation on the abscissa.

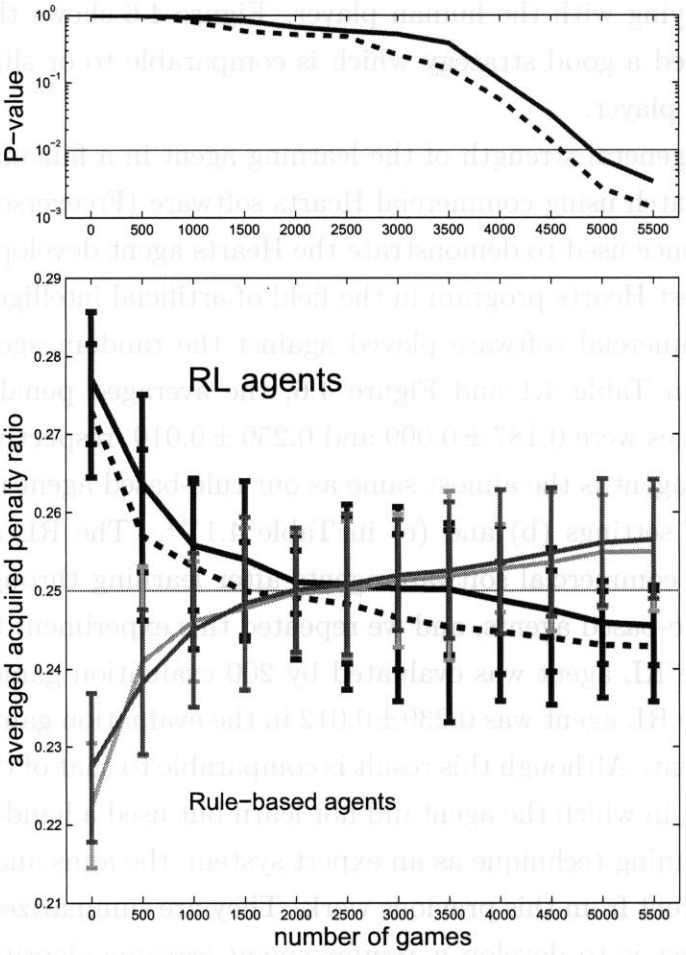

Fig. 4.5: Computer simulation result in an environment consisting of two learning agents trained by our RL method and two rule-based agents.

upper panel: P-values of the t-test where the null and alternative hypotheses are the same as in the previous experiment (Fig. 4.1). After 5,000 training games, the two RL agents were significantly stronger than the rule-based agents.

lower panel: The abscissa and the ordinate denote the same as in Fig. 4.1. We executed 18 learning runs, each consisting of 5,500 training games. The parameter values and other experimental setups were also same.

estimated by the action predictor in the training phase and the strategy of the human player in the evaluation phase. The action predictor, nonetheless, could have come to predict a standard card playing by its generalization ability from playing against the rule-based agents with general strategies, and such a standard card prediction worked well in playing with the human player. Figure 4.6 shows that the RL agent successfully acquired a good strategy which is comparable to or slightly better than that of the human player.

To examine the general strength of the learning agent in a fair manner, we carried out a validation match using commercial Hearts software (Freeverse-Software, 2004); this software was once used to demonstrate the Hearts agent developed by Sturtevant (2003), the strongest Hearts program in the field of artificial intelligence. When three agents of this commercial software played against the random agent and the same human player as in Table 4.1 and Figure 4.6, the averaged penalty ratios of each agent over 200 games were 0.187 ± 0.009 and 0.259 ± 0.010, respectively; the strength of the commercial agent is the almost same as our rule-based agent used in this study (see the results of settings (b) and (e) in Table 4.1)[*4]. The RL agent played 100 games against the commercial software agents after learning through 5,000 training games with our rule-based agents, and we repeated this experiment twice; that is, the performance of the RL agent was evaluated by 200 evaluation games. The averaged penalty ratio of the RL agent was 0.239 ± 0.012 in the evaluation games, stronger than the commercial agent. Although this result is comparable to that of the previous study (Sturtevant, 2003), in which the agent did not learn but used a hand-tuned evaluation function with a pruning technique as an expert system, the aims and contributions of our study are different from this previous work. They are summarized as the following two points: the first is to develop a *reinforcement learning* algorithm applicable to large-scale multi-agent environments with partial observability; and the second is to apply our method to the card game Hearts, which exemplifies such an environment, and to demonstrate that the agent trained by our RL algorithm attains a comparable performance to the human player.

[*4] Since the interface of our simulation program is different from that of the commercial software, the evaluation games were carried out by manually inputting cards played by the commercial agents into our program; the strengths were evaluated under the initiative of the commercial software. Since, for this reason, the initial card distribution was not fixed but distributed randomly, the exact comparison of the penalty ratios with those in Table 4.1 may be difficult. These results, however, show that there is only a small difference in the strengths between the current rule-based agent and the commercial agent.

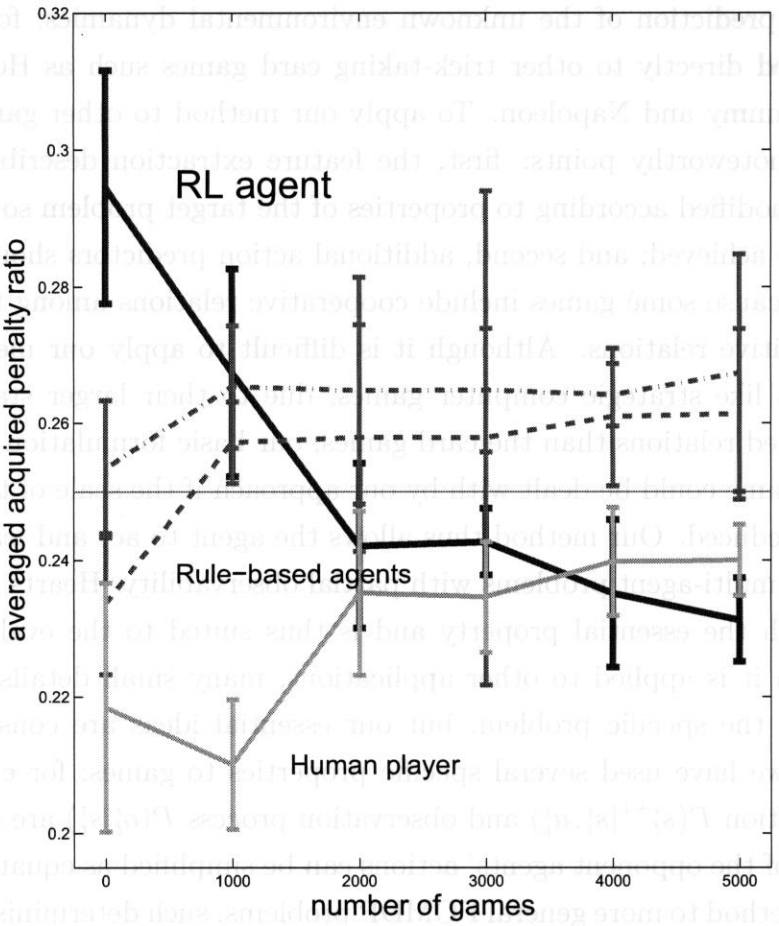

Fig. 4.6: Computer simulation result when one learning agent trained by our RL method, one human player and two rule-based agents played together. One hundred evaluation games were done before learning and after 1,000, 2,000, 3,000, 4,000 and 5,000 training games (with three rule-based agents). We repeated the training and evaluation runs twice. The abscissa and the ordinate denote the same as in Fig. 4.1. The parameter values were also the same. Each point denotes the average of 200 (2 $\times$ 100) evaluation games.

4.4 Discussion

Our RL formulation provides a general solution for partially observable games which can be solved by sequential decision-making based on the estimation of unobservable states and the prediction of the unknown environmental dynamics; for example, it could be applied directly to other trick-taking card games such as Hoist, Contract Bridge, Gin Rummy and Napoleon. To apply our method to other games, however, there are two noteworthy points: first, the feature extraction described in Section 4.2 should be modified according to properties of the target problem so that effective learning can be achieved; and second, additional action predictors should sometimes be prepared because some games include cooperative relations among the players as well as competitive relations. Although it is difficult to apply our method directly to other games like strategic computer games, due to their larger state space and more complicated relations than the card games, our basic formulation and ideas are available; the game could be dealt with by our approach if the scale of the problem is appropriately reduced. Our method thus allows the agent to act and learn in various large-scale and multi-agent problems with partial observability. Hearts is an example application with the essential property and is thus suited to the evaluation of our method. When it is applied to other applications, many small details may have to be changed for the specific problem, but our essential ideas are consistent. Note, however, that we have used several specific properties to games; for example, since the state transition $P(s_t^{i+1}|s_t^i, a_t^i)$ and observation process $P(o_t^i|s_t^i)$ are deterministic, the prediction of the opponent agents' actions can be simplified as equation (4.7). For applying our method to more general POMDP problems, such deterministic properties should be relaxed. Even in probabilistic situations, our sampling-based method with the simplified belief calculation would be effective.

Thrun (2000) proposed the Monte Carlo POMDP; it estimated unobservable states using the Monte Carlo method, which is similar to our method because all summations were approximated by a plausible number of samples in a partially observable problem. In the problem setting, however, the environmental model was given to the agent, and the state space was reasonably sized so that the simple Monte Carlo integration worked well. If an adequate number of samples could be obtained from the given distribution, then the summations were approximated with high accuracy, and it was possible to deal with the propagation of belief states. On the contrary, in the problem of the game Hearts, the environmental model is unknown and it is necessary to approximate the integration process with some types of devices. This

article shows that a sampling technique with model identification works well for a large-scale problem whose state space is discrete. We have not used complete belief maintenance throughout the game process, but have calculated a one-step belief, because a good incremental approximation of the belief is not easy with a restricted computer resource for problems whose observation process is deterministic.

Ginsberg (2001) designed an automatic player of the card game "Bridge," which called GIB, the most powerful computer program for the game. It used the same QMDP approximation as that of our RL scheme, and the unobservable card distribution was also estimated by a sampling technique. Since, however, possible card allocation was sampled from the uniform distribution under the assumption that the unobservable cards are distributed with uniform probability, this estimation process does not include any belief information. Therefore, though the agent could take an optimal action so that the expected return was maximized by calculating the utility value of each possible action using the samples, a large number of samples is required for selecting every single action, and this may cause a long computational time in large-scale problems. On the other hand, in our technique, the effective sampling process can be attained by calculating the partial likelihood information according to the acquired environmental model. This enables us to solve larger problems.

The dynamics of the game Hearts can be represented by products of opponent agents' action probabilities, as in equation (4.8). In our method, the policies of opponent agents are estimated by corresponding action predictors, and the utility function in equation (4.2), which is necessary for action selection by the agent, is calculated based on these predictors. Since such a model-based approach is effective in unknown and partially observable environments, many effective methods have been proposed (Chrisman, 1992; Whitehead & Lin, 1995; Nikovski & Nourbakhsh, 2000; Yoshimoto et al., 2003). This may, however, make the problem more difficult and complicated than model-free approaches, for two reasons: first, the computational cost for learning of the model with the estimation process is expensive in general problems; and second, if learning of the model fails, it may work against the estimation and policy acquisition processes, and vice versa, because they rely on each other. On the other hand, in our method, learning of the model is independent of the estimation or learning of the value function; each predictor is trained by available information, which is given at the end of each game, according to the supervised learning method. Learning of the model then always goes well, and policy learning accelerates with the improvement of prediction. Predictors are prepared for each opponent agent, and they are learned independently. When one opponent agent changes its policy, it is enough to make the corresponding predictor adapt to the change. Since this enables the agent

to adapt quickly, our RL method becomes applicable to complex multi-agent settings where the Markov property fails*5.

Our RL method is based on the Monte Carlo RL method (Sutton & Barto, 1998) in learning of the value function. When we carried out the same experiment as shown in Figure 4.1 without learning of the value function, the policy of the agent was not improved (data not shown). The value function, therefore, should be learned properly for a large state space. In general, however, it is difficult to learn the value function effectively for a large-scale and high-dimensional problem like Hearts, even with function approximators, because the number of parameters increases exponentially; this is known as the curse of dimensionality. If a large problem can be reduced by being divided into multiple subproblems with a hierarchical structure (Barto & Mahadevan, 2003), the agent will be able to learn more effectively.

*5 To evaluate the adaptability of the action predictor in a dynamic environment, we carried out an additional experiment; this is shown in Appendix C.

Chapter 5

Human Decision-Planning with Exploratory and Exploitative Strategies in a Partially Observable Environment

This chapter demonstrates the performance of a probabilistic model for the human decision-making and estimation process in a partially observable environment. Since theoretical POMDP studies have indicated as presented in the previous chapters that optimal decision-planning in partially observable environments has computational difficulties, the human brain copes with such difficulty based on some ingenious mechanism, because we are familiar with solving complex partially observable problems. To reveal how this mechanism operates in the human brain, this study presents probabilistic modeling approaches for higher-order cognitive process implemented in the brain and decision-making process based on the cognition, in partially observable environments. Model-based analyses show that our models can reproduce the subjects' behaviors with high accuracy, and indicate that humans estimate unobservable states based on the framework of the incremental Bayes estimation.

5.1 Experiments

In this study, we carried out behavioral experiments using a virtual maze, as shown in Figure 5.1(A). The task objective of our experiments is to reach a goal from a starting position in as few trials as possible. Thirteen normal subjects (11 males and 2 females, ages 23-28) participated in the experiments, and brain images of the subjects were simultaneously taken using functional Magnetic Resonance Imaging (fMRI). Our experimental setting was approved by the ethical committee of Advanced

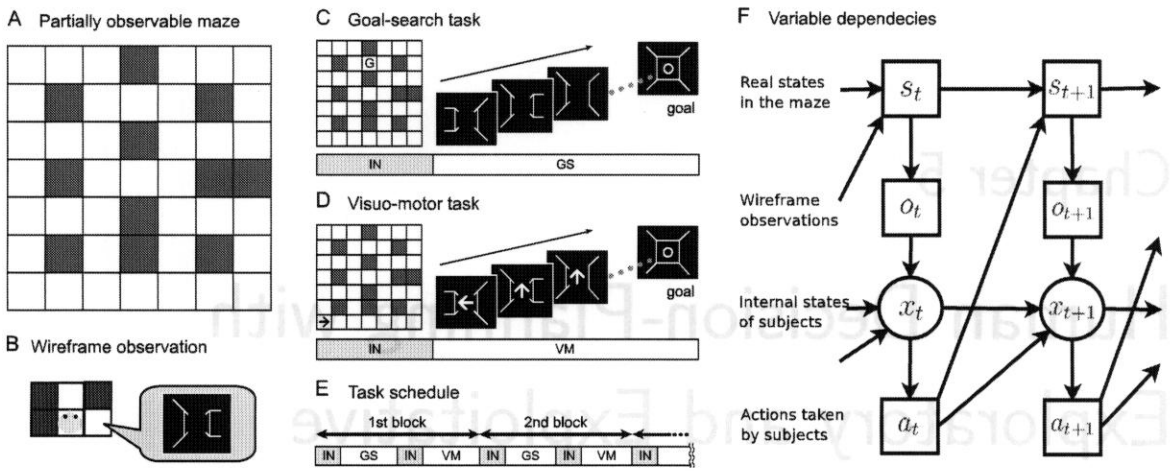

Fig. 5.1: Experimental design and variable dependencies of our model. (A) The partially observable maze used in the experiment; it is enclosed in walls, and there is no crossroad and dead end; (B) an example of the wireframe observation given to the subjects; since an immediate observation does not necessarily determine the subject's position and body orientation due to the perceptual aliasing property, it is essential to estimate them and to resolve uncertainty based on the observation sequence; (C) an example observation sequence in a goal-search (GS) task; (D) an example stimulus sequence in a visuo-motor (VM) task; (E) block design for fMRI imaging; instruction periods (IN) were inserted between GS and VM tasks, and (F) variable dependencies of our model in time series sequence, where the circles denote unobservable variables and the squares denote observable ones for the experimenters.

Telecommunications Research Institute International, Japan and all subjects fully consented to the participation prior to our experiment.

The subjects carried out two sessions in the same maze as that shown in Figure 5.1(A), and each session comprised several sets of a goal-search task and a visuo-motor task. The subjects were required to cope with the task as many times as possible in each session, where maximum number of trials was 300.[*1] In the goal-search tasks, the subjects tried to reach the instructed goal from an unknown starting position, shown in Figure 5.1(C); the goal position was shown in a 2-dimensional maze map at the beginning of each goal-search task, but the starting position was not. The subjects could not observe their real position in the maze, but obtain partial observation; they observed only a 3-dimensional wireframe observation in each trial, which represented the presence or absence of a wall in each 6-grid in front of the subject, as shown in Figure 5.1(B). To solve the problem effectively, therefore, it is important to estimate unobservable self positions and body orientations based on the observation sequence.

[*1] The tasks were aborted even in the middle of an active session; only the data for completed blocks were used for analyses.

The visuo-motor tasks were control tasks; the same observations and actions as those in the previous goal-search task were displayed, as shown in Figure 5.1(D), and the subjects were required to reproduce all sensori motor events in the previous task, according to the observations and actions presented by three kinds of arrows. These two tasks, goal-search and visuo-motor, were alternated with instruction periods, shown in Figure 5.1(E).

At the beginning of each goal-search task, a goal position was provided for 4 seconds, and an initial observation at an unknown starting position was presented. To reach the goal, the subjects selected one of three actions, forward move, left turn and right turn, by a button press action within 1.8 second. Note that the forward move changed subjects' positions to a place in front of them, but the left and right turns did not; the subjects turned to the correspond direction while staying in the same position. The intertrial interval was fixed at 2 seconds with the observation displayed for that interval time before the next trial began. The goal mark was displayed upon arrival at the goal, and an instruction period was inserted for 4 seconds before a visuo-motor task was started. Although the start and goal positions were different between each block and session, the minimum number of steps and branches was almost the same over all blocks (12±2 steps and 4±1 branches). Every subject was familiar with the maze structure due to their receiving sufficient training prior to the experiment, where they learned the correspondence between an arbitrary state and a correct observation so that their performance reached a satisfactorily high level. We then assumed that the subjects did not fail to estimate their current unobservable position and orientation when getting enough information and time to determine them.

5.2 Model

In this study, we assume that the subjects take a stochastic action $a_t \in \mathcal{A}$ based on their internal states $x_t = \{y_t, c_t\} \in \mathcal{X}$, where $y_t \in \mathcal{S}$ and $c_t \in \mathcal{C}$ denote a state (combination of a position and a body orientation) estimated by the subjects and a confidence for the state estimation, respectively. Note that the estimated state y_t includes 196 candidates by considering four orientations on each position in the 7×7 grid and the confidence c_t takes two values for representing *confidence* ($c_t = 1$) or *no confidence* ($c_t = 0$). Figure 5.1(D) shows the variable dependencies of our model. Sequence of subject's internal states, actions taken by the subjects, observations obtained from the maze are denoted by $\boldsymbol{x} = \{x_1, \cdots, x_{T+1}\}$, $\boldsymbol{a} = \{a_1, \cdots, a_T\}$ and $\boldsymbol{o} = \{o_1, \cdots, o_T\}$, respectively, where T is the terminal (goal arrival) time. The

likelihood function is then given as

$$l(\boldsymbol{\theta}, \boldsymbol{\sigma}) = \sum_{\boldsymbol{x}} P(\boldsymbol{a}, \boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\sigma}) = P(x_1) \sum_{\boldsymbol{x}} \prod_{t=1}^{T} P(a_t|x_t) P(x_{t+1}|x_t, a_t). \tag{5.1}$$

Our aim is to obtain the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\sigma}$, related to the internal state transition and observation processes, respectively, based on the maximum likelihood inference, so that the likelihood function in equation (5.1) is maximized. More concretely, $\boldsymbol{\theta} \equiv \{\theta_{ij} = P(x_{t+1} = j|x_t = i, a_t)|i = 1, \ldots, |\mathcal{X}|, j = 1, \ldots, |\mathcal{X}|\}, \boldsymbol{\sigma} \equiv \{\sigma_{ik} = P(a_t = k|x_t = i)|i = 1, \ldots, |\mathcal{X}|, k = 1, \ldots, |\mathcal{A}|\}$. Note, however, that hyperparameters $\boldsymbol{\delta} = \{\epsilon_{\mathrm{c}}, \epsilon_{\mathrm{exp}}, \epsilon_{\mathrm{opt}}\}$ were actually calculated, which define the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\sigma}$. First, under a factorization assumption $P(x_{t+1}|x_t, a_t) = P(y_{t+1}|y_t, a_t) P(c_{t+1}|y_t, c_t, a_t)$, the transition probability $\boldsymbol{\theta}$ is defined as

1. if $o^*_{t+1} = \hat{o}_{t+1}$, then $P(y_{t+1}|y_t, a_t) = 1$, where y_{t+1} is the next state reached from y_t by an action a_t, or 0 for the other next states, and
2. if $o^*_{t+1} \neq \hat{o}_{t+1}$, then $P(y_{t+1}|y_t, a_t) = 1/N_{\mathrm{s}}$, where y_{t+1} is an arbitrary next state whose observation o_{t+1} is consistent with an actual observation o^*_{t+1}, or 0 for the other next states.

Here, o^*_{t+1} and $\hat{o}_{t+1}$ denote an actual observation at time $(t+1)$ and an observation at $\hat{y}_{t+1}$ which is a predicted next state determined from the estimated current state $\hat{y}_t$ and action a_t, respectively, and N_{s} denotes the number of next states which do not violate the actual observation o^*_{t+1}. The initial distribution $P(y_1)$ is uniform for all states y_1 whose observation corresponds to the actual observation o^*_1. The model of the state transition can be explained as follows: if an actual observation o^*_{t+1} is the same as a predicted observation $\hat{o}_{t+1}$, the subject becomes convinced of the current estimation $\hat{y}_t$ and moves it to the next state $\hat{y}_{t+1}$ with probability 1 (case 1), otherwise the subject makes uniform estimation of the unobservable state to the states whose observation is consistent with the actual one (case 2). Next, $P(c_{t+1}|y_t, c_t, a_t)$ is defined as

3. if $o^*_{t+1} = \hat{o}_{t+1}$, then
   (a) if $c_t = 0$, then $P(c_{t+1}|y_t, c_t, a_t) = 1 - \epsilon_{\mathrm{c}}$ for $c_{t+1} = 0$, or ϵ_{c} for $c_{t+1} = 1$, and
   (b) if $c_t = 1$, then $P(c_{t+1}|y_t, c_t, a_t) = 0$ for $c_{t+1} = 0$, or 1 for $c_{t+1} = 1$,
4. if $o^*_{t+1} \neq \hat{o}_{t+1}$, then
   (a) if $c_t = 0$, then $P(c_{t+1}|y_t, c_t, a_t) = 1$ for $c_{t+1} = 0$, or 0 for $c_{t+1} = 1$, and
   (b) if $c_t = 1$, then $P(c_{t+1}|y_t, c_t, a_t) = 1$ for $c_{t+1} = 0$, or 0 for $c_{t+1} = 1$.

The initial distribution $P(c_1)$ is 1 for $c_1 = 0$, or 0 for $c_1 = 1$. Note that the initial distribution of the internal state $P(x_1)$ is given by $P(x_1) = P(s_1)P(c_1)$. The model of the confidence transition can be explained as follows: if the actual observation o^*_{t+1} is the same as the predicted observation $\hat{o}_{t+1}$, the subject changes his/her confidence from *no confidence* to *confidence* with probability ϵ_c (case 3a), or keeps it with probability 1 (case 3b), otherwise the subject does not get the confidence (case 4a) but rather loses it immediately with probability 1 (case 4b).

Second, the observation process $\boldsymbol{\sigma}$ is defined as

5. if $c_t = 0$, then $P(a_t|x_t) = (1 - \epsilon_{\text{exp}})/N_{\text{exp}}$, where a_t is an exploratory action, or $P(a_t|x_t) = \epsilon_{\text{exp}}/(N_{\text{a}} - N_{\text{exp}})$, where a_t is a non-exploratory action, and
6. if $c_t = 1$, then $P(a_t|x_t) = (1 - \epsilon_{\text{opt}})/N_{\text{opt}}$, where a_t is an optimal action, or $P(a_t|x_t) = \epsilon_{\text{opt}}/(N_a - N_{\text{opt}})$, where a_t is a non-optimal action.

Here, N_{exp} and N_{opt} denote the number of exploratory actions and that of optimal actions, respectively. The exploratory action is defined as a forward movement or a turn to any non-wall direction. At a turning point toward the right-hand side, for example, a forward move and a left turn are non-exploratory actions, but a right turn is a unique exploratory action; $N_{\text{exp}} = 1$ in this example. At a T-junction, $N_{\text{exp}} = 2$, but at a state where there is no wall in three directions, a forward move that brings 3-grid information is assumed to be a unique exploratory action. The optimal action is defined as an action to follow the shortest path to the goal. At many states, $N_{\text{opt}} = 1$, but at states which have multiple shortest paths, $N_{\text{opt}} = 2$.

In the above model, the hyperparameters $\boldsymbol{\delta} = \{\epsilon_{\text{c}}, \epsilon_{\text{exp}}, \epsilon_{\text{opt}}\}$ which maximize the likelihood function in equation (5.1) are simply given by

$$\epsilon = \frac{\langle N_2 \rangle}{\langle N_1 \rangle + \langle N_2 \rangle}, \tag{5.2}$$

where ϵ corresponds to any of ϵ_{c}, ϵ_{exp}, ϵ_{opt}. N_1 and N_2 denote the numbers of event occurrences related to each parameter, and $\langle \cdot \rangle$ denotes expectation with respect to the posterior distribution of the unobservable state x_t. We obtain the maximum likelihood estimates $\epsilon_{\text{c}} = 0.9451$, $\epsilon_{\text{exp}} = 0.0079$, $\epsilon_{\text{opt}} = 0.0100$ by using the forward-backward algorithm in the HMM framework; these parameter values are quite reasonable.

5.3 Results

Given the maximum likelihood estimates $\boldsymbol{\delta}^*$, the model can predict the actions

$$\begin{aligned}\hat{a}_{t+1} &= \underset{a_{t+1}}{\text{argmax}}\ P(a_{t+1}|a_{1:t}) \\ &= \sum_{x_{t+1}} P(a_{t+1}|x_{t+1}) \sum_{x_t} P(x_{t+1}|x_t, a_t) P(x_t|a_{1:t}),\end{aligned} \tag{5.3}$$

where $\hat{a}_{t+1}$ denotes a predicted action. The posterior distribution $P(x_t|a_{1:t})$ in equation (5.3) can be calculated as $P(x_t|a_{1:t}) = P(x_t, a_{1:t}) / \sum_{x_t} P(x_t, a_{1:t})$, where the joint distribution $P(x_t, a_{1:t})$ is obtained by the forward algorithm in the HMM framework. We evaluated the model by calculating the concordance rate:

$$\rho = \frac{\#(\hat{a}_{t+1}; t = 1 : T-1)}{T-1}, \tag{5.4}$$

where the numerator is the number of trials in which the predicted action $\hat{a}_{t+1}$ is the same as the actual action a^*_{t+1} from time 1 to $(T-1)$; equation (5.4) denotes the rate of concordance: $\hat{a}_{t+1} = a^*_{t+1}$, over $t = 1, \cdots, T-1$. This concordance rate was calculated with the leave-one-out validation method via the following three steps: the first step is to remove an action sequence $\boldsymbol{a}^k = \{a^k_1, \cdots, a^k_T\}$ of the k-th block from the complete action data $\boldsymbol{a} = \{\boldsymbol{a}^1, \cdots, \boldsymbol{a}^N\}$, where N denotes the number of action sequences; the second step is to obtain the maximum likelihood estimates $\boldsymbol{\delta}^*$ according to equation (5.2) by using the data of $(N-1)$ blocks; and the last step is to calculate the concordance rate ρ^k for the action sequence $\boldsymbol{a}^k$ according to equation (5.4). These three steps were iterated N times for all action sequences, and the averaged concordance rate $\langle\rho\rangle = \frac{1}{N}\sum_{i=1}^{N} \rho^k$ was calculated. The averaged concordance rate calculated by the above procedure was $\langle\rho\rangle = 0.8621$, that is, the model can reproduce about 86.2% of the behaviors taken by the subjects in the partially observable maze task.

About 13.8% of actions deviated from the model's predictions. This deviation may arise from two factors: the time requirement for the button-press action within only 1.8 seconds and irrational tendencies of human action selection. First, we assume in this study that the subjects do not fail to estimate their current unobservable state, if enough information and time is available, because all subjects are familiar with the maze structure due to their receiving sufficient training prior to the experiment. If, therefore, the subject can identify his/her state from the observation sequence, the following actions must be optimal according to the model.*2 Because of the time

*2 Nonetheless, the error rate ϵ_{opt} exists in the action selection process, as shown in case 6.

restriction for action selection, however, the subject may fail to select an optimal action or to estimate the unobservable state even with the adequate ability for correct estimation. Second, for example, if the subjects detect that the instructed goal is behind their back, most subjects do not turn around at the same position, but rather select an indirect route with forward moves. These facts imply that the assumption is slightly strong in that it does not explain such irregular behaviors, and this leads to deterioration of the concordance rate. A higher accuracy, about 95%, would be attained if such irregular behaviors were removed from the data.

Figure 5.2 shows routes and transition processes of the internal state for two different subjects, in the setting whose start and goal positions were the same. Figures 5.2(a) and (c) for each subject represent an actual route followed by the subjects and the changing process of the estimated state, respectively. Figures 5.2(b) and (d) represent the time-series of $P(c_t = 0|a_{1:T})$ and $P(y_t|a_{1:T})$, respectively.*3 The dotted lines in Figure 5.2(b) denote times when an estimated confidence $\hat{c}_t$ at time t is different from an estimated one $\hat{c}_{t+1}$ at time $(t+1)$; $\hat{c}_{t+1} \neq \hat{c}_t$, where $\hat{c}_t = \text{argmax}_{c_t} P(c_t|a_{1:T})$, and the dotted lines in Figure 5.2(d) denote times when an estimated state $\hat{y}_t$ at time t is different from an estimated one $\hat{y}_{t+1}$ at time $(t+1)$; $\hat{y}_{t+1} \neq \hat{y}_t$, where $\hat{y}_t = \text{argmax}_{y_t} P(y_t|a_{1:T})$. Note that the arrows' colors in (a) and (c) correspond to the lines' colors in (b) and (d), respectively; for example, the gray lines in Figure 5.2(a) represent periods when the model estimated that the subject had no confidence, $\hat{c}_t = 0$.

According to the analysis of subject A's behaviors shown in the upper panel of Figure 5.2, this subject was supposed to have made a correct estimation by chance, from the beginning of the task. There were, however, many consistent states with actual observations, and the subject was not convinced of the state estimation till $t = 4$. By obtaining the observation where the subject selected forward actions from the starting position and a right turn at the end of the forward path (at the fourth trial), he/she was supposed to acquire confidence. According to the analysis of subject B's behaviors shown in the lower panel of Figure 5.2, on the contrary, the subject was supposed to have retried an estimation for every inconsistent observation against his/her predicted one. When the subject went around the lower right-hand corner of the maze before coming back to the starting position, he/she was supposed to acquire confidence and then take the shortest path to the goal.

*3 Since the posterior distribution $P(x_t|a_{1:T})$ can be calculated by the forward-backward algorithm in the HMM framework, the distribution of a confidence c_t and a state y_t can be calculated as $P(c_t = 0|a_{1:T}) = \sum_{y_t, c_t=1} P(x_t|a_{1:T})$ and $P(y_t|a_{1:T}) = \sum_{c_t} P(x_t|a_{1:T})$, respectively.

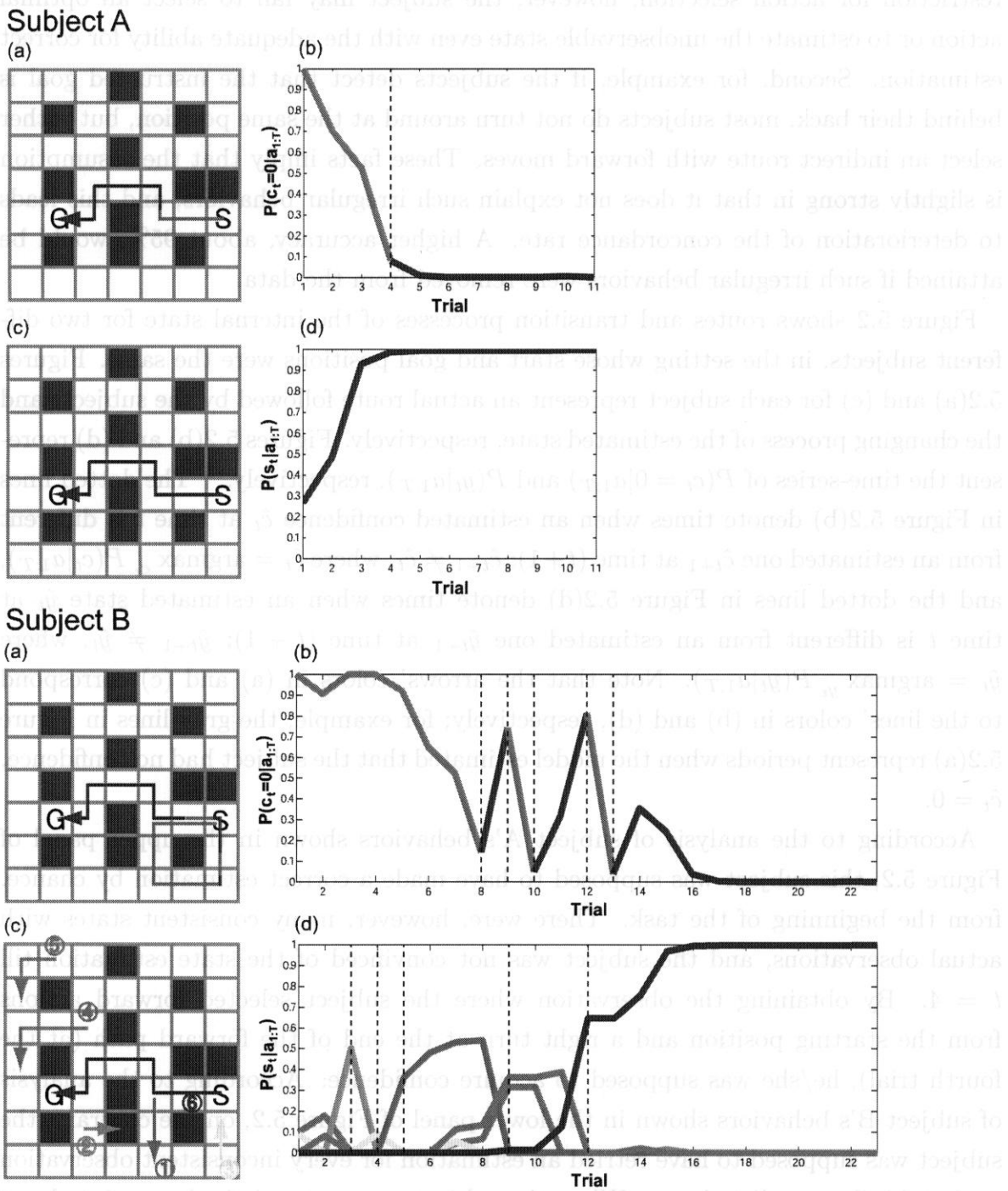

Fig. 5.2: The routes and transition processes of the internal state for two different subjects, in the setting whose start and goal positions were the same. S and G in the maze denote the positions of the start and goal, respectively. Subject A took the shortest path and arrived at the goal in 11 trials. Subject B, on the other hand, went around the lower right-hand corner of the maze and acquired confidence after coming back to the start position. The circled numbers in (c) denote the changing numbers of the state estimation. This subject arrived at the goal in 23 trials.

We carried out the imaging regression analysis in which brain images obtained by using fMRI were analyzed by a regression function defined on the model. The analysis implicates the neural correlates involved in exploratory and exploitative behavioral models in partially observable domains; this result is shown in Appendix D.

5.4 Discussion

We demonstrated the performance of an HMM-based probabilistic model of human decision-making and estimation process during a partially observable maze task, and showed the associated brain activity by evaluating the behavioral performance of subjects in the maze, with taking brain images simultaneously using functional Magnetic Resonance Imaging (fMRI). The model enabled us to analyze the behaviors of thirteen subjects for the maze task, based on the maximum likelihood inference, under the assumption that the subjects take a stochastic action according to their internal states, and to estimate a cognitive load related to uncertainty resolution and decision-making processes by means of a regression analysis. The results showed that our model based on the estimation of the internal states according to the probabilistic model-based approach can reproduce the subjects' behaviors with high accuracy and the imaging regression analysis based on the model implicated the neural correlates involved in exploratory and exploitative behavioral modes.

Since the model reproduces the estimation process based on the incremental Bayes formulation, the results, which are consistent with findings of previous researches, imply the possibility that humans estimate hidden information by performing the incremental belief update based on the Bayesian probabilistic inference. Although this implication is feasible, three issues still remain. First, since we focus in this model on the human estimation process of unobservable states, the action selection model is then defined as a simple sub-optimal reactive policy depending on the confidence variable. It is, however, impractical to assume that the subjects take such reactive responses to immediate stimuli. We improve, therefore, the decision-making model under the assumption that they determine their actions based on reward prediction, and in our future work, present a value-based decision-making model to describe subjects' behaviors more precisely. Second, the decision-making process presented in this dissertation is defined so that the action selection depends on the internal state, and the parameters for this action selection is calculated along with the estimation of the internal state, based on the action sequences of the subjects. This approach is not direct modeling of a human behavior, but just a search of the most-likely point over a restricted parameter space. In a statistical modeling, the human decision-making

process should be strictly determined based on a feasible mechanism, that is, the value-based action selection. Last, although our current model includes the confidence variable and it discriminates between exploitative and exploratory behavioral modes, the model does not deal sufficiently with the exploitation-exploration problem. Our future work will include using a mechanism for solving the problem.

Chapter 6

Conclusion

This dissertation explores the possibility that learning and decision-making in the real world require the following three factors: the first is to estimate hidden information with effective approximation techniques; the second is to predict environmental behavior by learning its dynamics; and the last is to evaluate action values based on the estimation and prediction. To demonstrate the importance of these factors, we have carried out several studies using two-sided approach: theoretical approaches with computer simulations and modeling approaches with a behavioral experiment. This chapter gives a summary of contributions in this dissertation and an outline of future work.

6.1 Contributions

In theoretical studies for controlling autonomous agents, we developed new model-based RL schemes for large-scale multi-agent environments with partial observability. It is necessary for learning and decision-making in such environments to estimate unobservable states and predict opponents' actions. The computational cost for such processes, however, increases exponentially with the enlargement in the scale of the environment, so it is infeasible to estimate all possible current states and predict all possible next states in realistic problems. We therefore developed two types of devices to avoid the exhaustive state enumeration for the estimation and prediction. First, we proposed the mean-field-like analog approximation, described in Chapter 3. This method approximated a belief state as an expected observation which contains the estimation process of unobservable states within its representation. Opponents agents' actions subsequent from the expected observation can be predicted according to the corresponding behavior model represented by the action predictor. This approximation idea has enabled us to alleviate the heavy integrations effectively and showed a good performance in the application to a real card game, Hearts, which is a

typical example of such difficult environments. Our approach, however, may cause the deviation to approximate the real distribution of the unobservable states due to the following two facts; first, this technique represents the distribution of possible discrete states as an analog averaged point; and second, it does not propagate belief information over time due to the uniform assumption. The approximation error incurred by these two properties may prevent the learning agent from evaluating the action value accurately, and lead to deterioration of its performance. Second, therefore, we used a sampling-based approximation technique with one-step belief maintenance, in which the heavy integration required for the estimation and prediction can be approximated by using a plausible number of discrete samples, presented in Chapter 4. This alternative approach succeeded to improve the performance due to removing the evaluation error of the action value. This result implies that the sampling-based state estimation is better than the mean-field one for controlling the autonomous agent in realistic partially observable environments, and may be more practical for considering the human estimation process of hidden information.

According to the theoretical findings, in studies for modeling human decision-making process of the partially observable maze task, we presented a probabilistic model, described in Chapter 5. To reveal the human estimation process, we presented the HMM-based probabilistic model, which used incremental Bayes filter for the estimation of unobservable states, and showed the possibility that humans estimate hidden information based on the belief update according to the Bayesian probabilistic inference, under the assumption that the subjects take reactive actions to immediate stimuli. This result indicates that humans estimate unobservable states based on the framework of the incremental Bayes estimation.

6.2 Future works

When considering a probabilistic model available to learning and decision-planning in realistic environments and practical for describing the human behavior, several ideas of previous theoretical POMDP research introduced in Chapter 2 are beneficial.

First, it is a promising approach to exploit certain types of problem structure to make the value function and/or the belief state representation more compactly by using a dynamic Bayesian network, described in Section 2.2.2; for example, to estimate the current state in the partially observable maze, it is feasible to focus attention on a subregion based on hierarchical features such as "facing a northerly or southerly direction" or "being at an outer or inner region of the maze", before estimating the detailed current state. The estimation accuracy and required computational cost may

be improved by using distinct features with hierarchical structure; we plan to deal with this hierarchical formulation for the belief update.

Second, the idea to use memory states of the FSM model introduced in Section 2.3.1 is beneficial. Since the memory state can be an essential feature necessary to learning and decision-making, it is important for solving realistic problems to convert the primitive belief space to an effective feature space represented by the set of memory states, according to a projection scheme such as the principal component analysis described in Section 2.2.2. Although we used a similar scheme as the feature extraction technique described in Sections 3.2 and 4.2, this was completely based on a heuristics process; we also plan to explore the automatic feature projection technique with dimension estimation using non-parametric Bayesian inference in our future work. Since the belief update is performed over the feature space, the approach based on this idea can be interpreted as a hybrid solution between the belief-state MDP and the FSM model.

Last, the policy gradient RL approach with learning basis functions is important for learning in partially observable environments. Although our current studies focus relatively on decision-planning processes, our future work will include more effective learning schemes, based on the policy gradient RL method with learning basis functions, which are conducted over the feature space with hierarchical structure.

Appendix A

Publications

A.1 Journal papers

1. Hajime Fujita and Shin Ishii
   Model-based reinforcement learning for partially observable games with sampling-based state estimation
   Neural Computation (to appear).

2. Hajime Fujita and Shin Ishii
   Model-based reinforcement learning for a multi-player card game with partial observability *(in Japanese)*
   IEICE Transactions on the Institute of Electronics, Information and Communication Engineers, Vol.J88–D–II, No.11, pp.2277–2287, 2005.

3. Shin Ishii, Hajime Fujita, Masaoki Mitsutake, Tatsuya Yamazaki, Jun Matsuda, and Yoichiro Matsuno
   A reinforcement learning scheme for a partially-observable multi-agent game
   Machine Learning, vol.59, pp.31–54, 2005.

A.2 Conference presentation

1. Hajime Fujita, Yutaka Nakamura and Shin Ishii
   Feature extraction for decision-theoretic planning in partially observable environments
   International Conference on Artificial Neural Networks (ICANN '06), LNCS 4131, pp.820–829, 2006.

2. Hajime Fujita and Shin Ishii
   Model-based reinforcement learning for large-scale multi-agent games with sampling-based state estimation

Artificial Life and Robotics (AROB '06), GS3–1, 2006.

3. Hajime Fujita and Shin Ishii
   Model-based reinforcement learning for partially observable games with sampling-based state estimation
   Advances in Neural Information Processing Systems, Workshop on Game Theory, Machine Learning and Reasoning under Uncertainty (NIPS Workshop GMR '05), 2005.

4. Hajime Fujita and Shin Ishii
   Model-based reinforcement learning for a multi-player card game with partial observability
   IEEE/WIC/ACM International Joint Conference on Intelligent Agent Technology (IEEE IAT '05), pp.467–470, 2005.

5. Hajime Fujita and Shin Ishii
   A reinforcement learning scheme for a multi-agent card game with Monte Carlo state estimation
   International Conference on Computational Intelligence for Modeling Control and Automation (CIMCA '04), pp.799–806, 2004.

6. Hajime Fujita, Yoichiro Matsuno, and Shin Ishii
   A reinforcement learning scheme for a multi-agent card game
   IEEE International Conference System, Man. and Cybernetics (IEEE SMC '03), pp.4071–4078, 2003.

A.3 Technical reports

1. 藤田 肇, 石井 信
   部分観測ゲームのためのモデル同定型強化学習
   日本神経回路学会 第 16 回全国大会, pp.10–11, 2006.

2. 藤田 肇, 中村 泰, 石井 信
   部分観測環境での意思決定に必要な特徴空間の抽出
   電子情報通信学会技術報告, Vol.106, No.102, pp.13–18, 2006.

3. Hajime Fujita and Shin Ishii
   Model-based reinforcement learning for partially observable games with sampling-based state estimation
   NAIST Technical Report, No. 2005009, 2005.

4. 藤田 肇, 石井 信

部分観測カードゲームのためのモデル同定型強化学習法
日本神経回路学会 第 14 回全国大会, pp.46–47, 2004.

5. 藤田 肇, 石井 信
マルチエージェントカードゲームのための信念状態強化学習法
電子情報通信学会技術報告, Vol.103, No.734, pp.67–72, 2004.

6. 藤田 肇, 松野 陽一郎, 石井 信
マルチエージェントカードゲームのための強化学習
日本神経回路学会 第 13 回全国大会, pp.28–29, 2003.

7. 藤田 肇, 松野 陽一郎, 石井 信
マルチエージェントカードゲームのための強化学習法の改良
電子情報通信学会技術報告, Vol.102, No.731, pp.167–172, 2003.

Appendix B

Hearts

The game of Hearts is played by four players and uses an ordinary 52-card deck. There are four suits - spades(♠), hearts(♡), diamonds(♢), and clubs(♣) - and there is an order of strength within each suit (decreasing from A, K, Q, $\cdots$, 2). There is no strength order among the suits. Cards are distributed to the four players so that each has 13 cards at the beginning of a game. After that, according to the rules below, each player plays a card in clockwise order. When each of the four players has played a card, it is called a *trick*; each player plays a card once in a trick. The first card played in a trick is called the *leading card* and the player who plays this card is called the *leading player*. A single game ends when 13 tricks have been carried out.

- In the first trick, ♣2 is the leading card, so that the player holding this card is the leading player.
- Except for the first trick, the winner of the current trick becomes the leading player of the next trick.
- Each player must play a card of the same suit as the leading card.
- If a player does not have a card of the same suit as the leading card, he or she can play any card. When a heart is played in this way for the first time in a game, the play is called *breaking hearts*.
- Until breaking hearts occurs, the leading player may not play a heart. If the leading player has only hearts, it is an exceptional case and the player may lead with a heart.
- After a trick, the player who has played the strongest card of the same suit as the leading card is the winner of that trick.
- Each heart equals a one-point penalty and the ♠Q equals a 13-point penalty, so the total number of penalty points is 26. The winner of a trick receives all of the penalty points of the cards played in the trick.

According to the rules above, a single game is played, and at the end of a game, the penalty points of each player are determined as the sum of the received points. The lower the points, the better. This game, therefore, represents a competitive situation because each player has to avoid penalty points by pushing them to the opponents. For simplicity, *shooting the moon* is removed from our setting.

Appendix C

Prediction accuracy

To examine the adaptability of the action predictors in a dynamic environment, we calculated the Kullback-Leibler (KL) divergence between the real empirical action probability, $P(a_t^i|o_t^i, \phi^i)$, and the action probability approximated by the agent, $P(a_t^i|o_t^i, \hat{\phi}^i)$, by changing the environment every 100 training games. We carried out the experiment using four types of rule-based agents; their acquired penalty ratios were (A) 0.206, (B) 0.193, (C) 0.186 and (D) 0.180 when they played against the random agent. This experiment used four types of evaluation data sets, which are collections of actions taken by the corresponding rule-based agent, each of which was generated according to the agent's own rules within 1,000 games in advance. We switched the rule-based agents in alphabetical order, (A), (B), (C) and (D), after every 1,000 training games; the opponent agents gradually became stronger, and the performance of the predictors was evaluated based on the data set from the corresponding agent. In this experiment, accordingly, 1,000 evaluation games and 100 training games were alternated, and three rule-based agents were replaced every 1,000 training games. Each point of Figure C.1 represents an average for 1,000 such evaluation games over 10 learning runs. Although the performance deteriorated slightly just after every switching of the rule-based agents, the action predictor could adapt to the changes quickly, and its performance improved steadily as learning progressed even in this dynamic environment. This result implies that the adaptability of our action predictor is so fast that the POMDP assumption is approximately valid even in dynamic environments constituted by multiple learning agents.

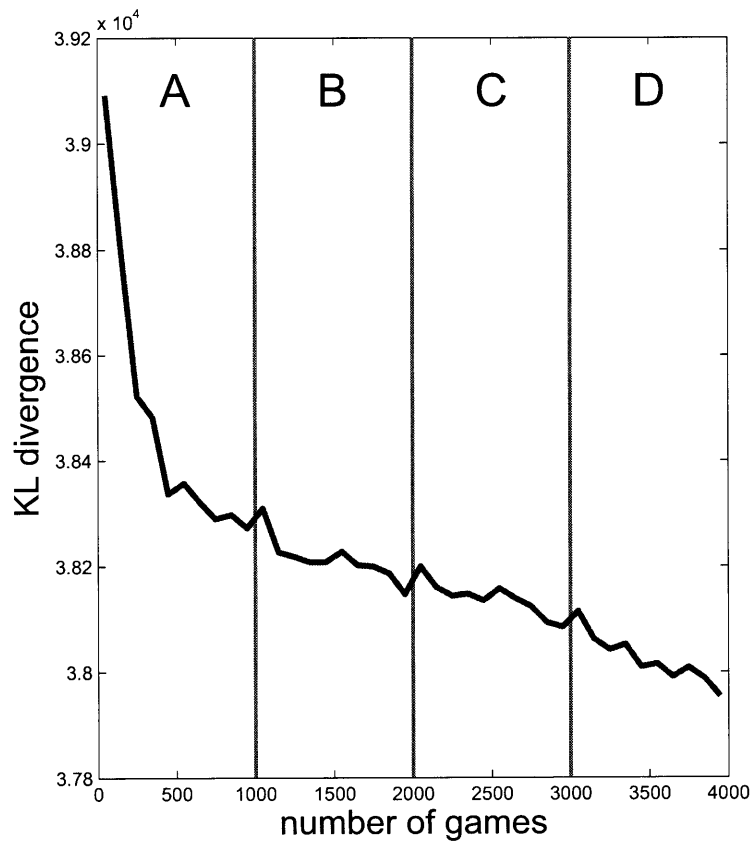

Fig. C.1: The Kullback-Leibler (KL) divergence between the real empirical action probability, $P(a_t^i|o_t^i, \phi^i)$, and the action probability approximated by the agent, $P(a_t^i|o_t^i, \hat{\phi}^i)$. The abscissa denotes the number of training games and the ordinate denotes the KL divergence. We executed 10 learning runs, each consisting of 4,000 training games. The constant T^i in equation (4.11) was set at 1.0.

Appendix D

Brain activities

Functional images were obtained with EPIs using BOLD contrast on a 1.5-tesla scanner (Magnetic Eclipse; Shimadzu Marconi, Kyoto, Japan). Volumes, acquired in synchronization with stimulus presentation (TR: 2 sec), contained 20 slices each measuring 5 mm in thickness. The first six (12 sec) EPIs in each session were not evaluated as part of the scanning data to avoid T1 equilibrium effects. Each scanning run began with a T1-weighted anatomical image acquisition (voxel size: 1 mm^3).

According to our model, we assumed the subjects switch between exploratory (most informative) and exploitative (optimal) strategies depending on their confidence in the state estimation. These behavioral modes could be evaluated as $P(c_t = 0|a_{1:T})$ (Fig. 5.2(b)) and $P(c_t = 1|a_{1:T})$, which are marginalized over the unobservable state variable s_t by averaging with respect to posterior probabilities for removing dependencies on the sequence of the subjects' internal cognitive conditions. Using statistical parametric mapping (SPM), we conducted a multiple regression analysis with these probabilistic measures, called exploration and exploitation regression functions respectively, to identify neural activities correlated with these behavioral modes. After convolution with a hemodynamic response function, these functions were input to a general linear convolution model of evoked hemodynamic responses in the usual way to form statistical parametric maps. We calculated a group random effects statistic from the combination of the individual correlation maps with statistical thresholds at the voxel level of $p < 0.001$ (uncorrected) and at the cluster level of $p < 0.05$ (corrected).

For the exploration mode, we observed correlated activity in the fronto-polar prefrontal cortex (BA10; $x, y, z = 0, 54, 14$; $Z = 4.24$), bilateral inferior parietal cortex (BA39/40; right: $x, y, z = 62, -56, 16$; $Z = 4.56$, left: $x, y, z = -44, -56, 24$; $Z = 4.53$), and the posterior cingulate cortex (BA31; $x, y, z = 2, -50, 30$; $Z = 4.32$) (Fig. D.1A), which were more apparent than for the exploitation mode. For the opposite contrast, we found significant activations in the regions of the cerebral cor-

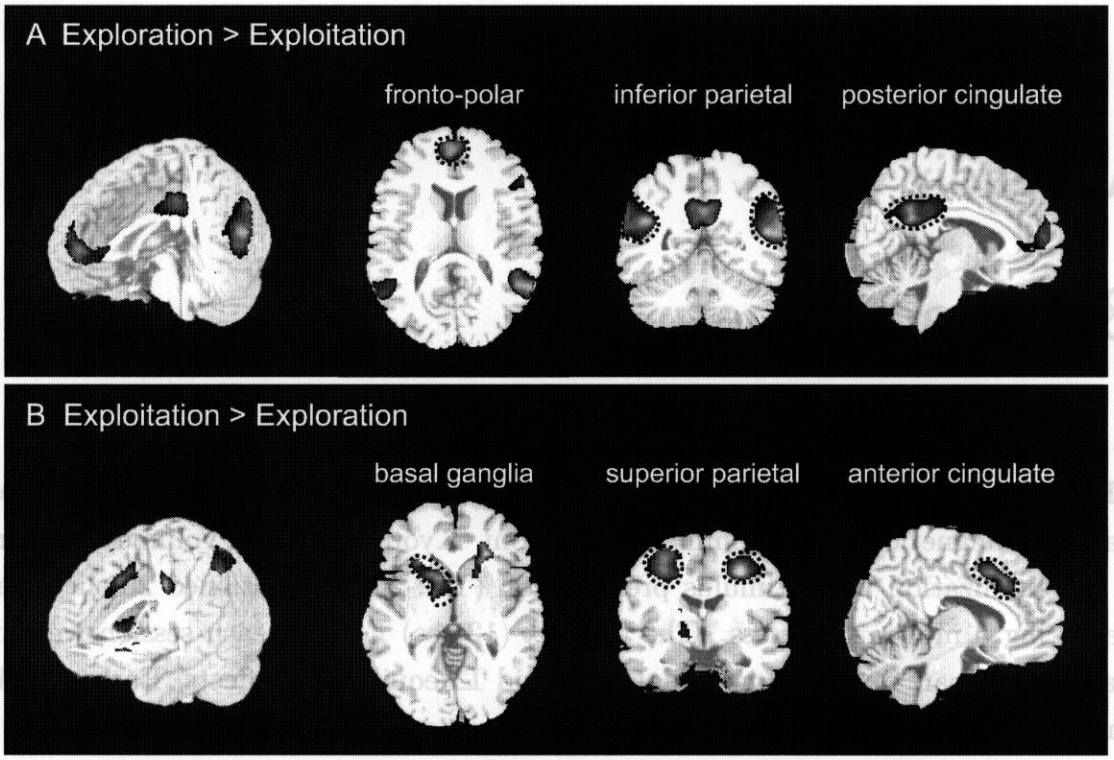

Fig. D.1: Brain areas that are more active in exploration than exploitation modes (A) and in exploration than exploitation modes (B).

tex, including the bilateral superior parietal cortex (BA7; right: $x, y, z = 32, -46, 48$; $Z = 4.94$, left: $x, y, z = -20, -56, 62$; $Z = 4.78$), bilateral premotor cortex (BA6; right: $x, y, z = 26, 4, 48$; $Z = 5.55$, left: $x, y, z = -22, 4, 46$; $Z = 5.09$), and anterior cingulate cortex (BA32; $x, y, z = 4, 18, 44$; $Z = 5.13$), and basal ganglia areas (bilateral putamen, caudate, and globus pallidus) (Fig. D.1B).

The fronto-polar cortex, the most rostral part of the prefrontal cortex, is known to be activated during complex cognitive tasks, in particular those which involve multiple competitive rules (Koechlin, Corrado, Pietrini, & Grafman, 2000; Strange, Henson, Friston, & Dolan, 2001), and a recent imaging study using a model-based analysis (Yoshida & Ishii, 2006) suggested that this area is involved in the internal conflict among multiple candidates of the hidden state. The fronto-polar activity in our task could reflect the uncertainty of state estimation which induces exploratory behaviors. Behavioral and neuropsychological studies in humans indicate that multiple spatial reference frames are used to guide behavior and that parietal cortex is central to the construction of these egocentric spatial representations (Colby & Goldberg, 1999). The cingulate cortex, which is the major cortical component of a distributed network of spatial attention, was activated both in the exploration and exploitation. The

posterior part is known to be activated while the risky choice in a gambling task (McCoy & Platt, 2005), and it is consistent with that exploratory behaviors could be directed away from the goal. The activation of the anterior part, however, which is associated with the detection of behavioral errors (Braver, Barch, Gray, Molfese, & Snyder, 2001) and response conflict (Botvinick, Nystrom, Fissell, Carter, & Cohen, 2001), was not observed with the exploration, and this might be attributable to the subjects just selecting intuitive actions without conflict and not predicting the next scene attentively. We also observed activation of the basal ganglia nuclei during exploitation; these areas are known to constitute parallel loops with the cortex, which are important for controlling involuntary motion (Alexander, Delong, & Strick, 1986), and may be involved in motor information processing directed to the goal (reward).

Bibliography

Aberdeen, D., & Baxter, J. (2002). Scaling internal-state policy-gradient methods for POMDPs. In *Proceedings of the 19th International Conference on Machine Learning*, pp. 1–12.

Alexander, G. E., Delong, M. R., & Strick, P. L. (1986). Parallel organization of functionally segregated circuits linking basal ganglia and cortex. *Annual Review Neuroscience*, *9*, 357–381.

Astrom, K. J. (1965). Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, *10*, 174–205.

Baird, L., & Moore, A. (1998). Gradient descent for general reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS '98)*, Vol. 11, pp. 968–974.

Barraclough, D. J., Conroy, M. L., & Lee, D. (2004). Prefrontal cortex and decision making in a mixed-strategy game. *Nature Neuroscience*, *7*, 404–410.

Barto, A. G., & Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, *13*, 341–379.

Barto, A. G., Sutton, R. S., & Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man and Cybernetics*, *13*, 834–846.

Bellman, R. E. (1957a). *Dynamic Programming*. Princeton University Press, Princeton.

Bellman, R. E. (1957b). A Markov decision process. *Journal of Mathematical Mechanics*, *6*, 679–684.

Bengio, Y., & Frasconi, P. (1996). Input-output HMM's for sequence processing. *IEEE Transactions on Neural Networks*, *7*, 1231–1249.

Botvinick, M. M., Nystrom, L., Fissell, K., Carter, C. S., & Cohen, J. D. (2001). Conflict monitoring vs. selection-for-action in anterior cingulate cortex. *Nature*, *402*, 179–181.

Boutilier, C., & Poole, D. (1996). Computing optimal policies for partially observable decision processes using compact representations. In *Proceedings of the 13th*

National Conference on Artificial Intelligence (AAAI '96), pp. 9–16.

Bowling, M., & Veloso, M. (2000). An analysis of stochastic game theory for multiagent reinforcement learning. Tech. rep., CS-00-165, Carnegie Mellon.

Boyen, X., & Koller, D. (1998). Tractable inference for complex stochastic processes. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI '98)*, pp. 33–42.

Bradtke, S., & Barto, A. (1997). Linear least-squares algorithms for temporal difference learning. *Machine Learning*, *22*, 33–57.

Brafman, R. I. (1997). A heuristic variable grid solution method for POMDPs. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI '97)*, pp. 727–733.

Braver, T. S., Barch, D. M., Gray, J. R., Molfese, D. L., & Snyder (2001). Anterior cingulate and response conflict: Effects of frequency, inhibition, and errors. *Cerebral Cortex*, *11*, 825–836.

Cassandra, A. (1998). *Exact and approximate algorithms for partially observable Markov decision processes.* Ph.D. thesis, Brown University.

Cassandra, A. R., Littman, M. L., & Zhang, N. L. (1997). Incremental pruning: a simple, fast, exact algorithm for partially observable Markov decision processes. In *Proceedings of the 13th Annual Conference on Uncertainty in Artificial Intelligence (UAI '97)*, pp. 54–61.

Chang, Y.-H., Ho, T., & Kaelbling, L. P. (2003). All learning is local: Multi-agent learning in global reward games. In *Advances in Neural Information Processing Systems (NIPS '03)*, Vol. 16, pp. 807–814.

Cheng, H.-T. (1988). *Algorithms for partially observable Markov decision processes.* Ph.D. thesis, University of British Columbia.

Chrisman, L. (1992). Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI '92)*, pp. 183–188.

Chrisman, L., & Littman, M. L. (1993). Hidden state and short-term memory. Presentation at Reinforcement learning workshop, the 10th International Conference on Machine Learning (ICML '93).

Colby, C. L., & Goldberg, M. E. (1999). Space and attention in parietal cortex. *Annual Review Neuroscience*, *22*, 319–349.

Collins, S., Ruina, A., Tedrake, R., & Wisse, M. (2005). Efficient bipedal robots based on passive dynamic walkers. *Science*, *307*, 1082–1085.

Crites, R. H. (1996). *Large-scale dynamic optimization using teams of reinforcement learning agents.* Ph.D. thesis, University of Massachusetts, Amherst.

Crites, R. H., & Barto, A. G. (1996a). Elevator group control using multiple reinforcement learning agents. *Machine Learning, 33*, 235–262.

Crites, R. H., & Barto, A. G. (1996b). Improving elevator performance using reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS '96)*, Vol. 8, pp. 1017–1023.

Dahl, F. A. (2002). The lagging anchor algorithm: Reinforcement learning in two-player zero-sum games with imperfect information. *Machine Learning, 49*, 5–37.

Daw, N. D., & Doya, K. (2006). The computational neurobiology of learning and reward. *Current Opinion in Neurobiology, 16*, 199–204.

Dean, T., & Kanazawa, K. (1989). A model for reasoning about presistence and causation. *Computational Intelligence, 5(3)*, 142–150.

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 1–38.

Doucet, A., Freitas, N., & Gordon, N. (2001). *Sequential Monte Carlo Methods in Practice*. Springer.

Doya, K., Samejima, K., Katagiri, K., & Kawato, M. (2002). Multiple model-based reinforcement learning. *Neural Computation, 14*, 1347–1369.

Emery-Montemerlo, R., Gordon, G., Schneider, J., & Thrun, S. (2004). Approximate solutions for partially observable stochastic games with common payoffs. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS '04)*, pp. 136–143.

Estelle, J. (2003). Reinforcement learning in POMDPs: Instance-based state identification vs. fixed memory representations. Tech. rep., Cornell University.

Freeverse-Software (2004). 3D Hearts Deluxe..

Fudenberg, D., & Tirole, J. (1991). *Game Theory*. MIT Press.

Fujita, H., & Ishii, S. (2007). Model-based reinforcement learning for partially observable games with sampling-based state estimation. *Neural Computation, (to appear)*.

Fujita, H., Nakamura, Y., & Ishii, S. (2006). Feature extraction for decision-theoretic planning in partially observable environments. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN '06)*, Vol. LNCS 4131, pp. 820–829.

Fujita, H., & Ishii, S. (2005). Model-based reinforcement learning for partially observable games with sampling-based state estimation. Tech. rep., Nara Institute of Science and Technology, No. 2005009.

Fujita, H., & Ishii, S. (2006). Model-based reinforcement learning for large-scale multi-

agent games with sampling-based state estimation. In *Proceedings of Artificial Life and Robotics (AROB '06)*, pp. GS3–1.

Fujita, H., Matsuno, Y., & Ishii, S. (2003). A reinforcement learning scheme for a multi-agent card game. In *Proceedings of IEEE International Conference System, Man. and Cybernetics (IEEE SMC '03)*, pp. 4071–4078.

Geffner, H., & Bonet, B. (1998). Solving large POMDPs by real time dynamic programming. Working Notes Fall AAAI Symposium on POMDPs.

Gilks, W. R., Richardson, S., & Spiegelhalter, D. J. (1996). *Markov Chain Monte Carlo in Practice*. Chapman and Hall.

Ginsberg, M. (2001). Gib: imperfect information in a computationally challenging fame. *Journal of Artificial Intelligence Research*, *14*, 303–358.

Hansen, E. A. (1998). Solving POMDPs by searching in policy space. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI '98)*, pp. 211–219.

Hansen, E. A., Bernstein, D. S., & Zilberstein, S. (2004). Dynamic programming for partially observable stochastic games. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI '04)*, pp. 709–715.

Hartley, H. (1958). Maximum likelihood estimation from imcomplete data. *Biometrics*, *14*, 174–194.

Hauskrecht, M. (1997). Incremental methods for computing bounds in partially observable Makorv decision processes. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI '97)*, pp. 734–739.

Hauskrecht, M. (2000). Value-function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, *13*, 33–94.

Hu, J., & Wellman, M. P. (1998). Multiagent reinforcement learning: theoretical framework and an algorithm. In *Proceedings of the 15th International Conference on Machine Learning*, pp. 242–250.

Ishii, S., Fujita, H., Mitsutake, M., Yamazaki, T., Matsuda, J., & Matsuno, Y. (2005). A reinforcement learning scheme for a partially-observable multi-agent game. *Machine Learning*, *59*, 31–54.

Ishii, S., Yoshida, W., & Yoshimoto, J. (2002). Control of exploitation-exploration meta-parameter in reinforcement learning. *Neural Networks*, *15(4–6)*, 665–687.

Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, *101*, 99–134.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, *4*, 237–285.

Koechlin, E., Corrado, G., Pietrini, P., & Grafman, J. (2000). Dissociating the role of

the medial and lateral anterior prefrontal cortex in human planning. In *National Academic Science*, Vol. 97, pp. 7651–7656.

Lin, L.-J., & Mitchell, T. M. (1992). Memory approaches to reinforcement learning in non-Markovian domains. Tech. rep., CS-92-138, Carnegie Mellon.

Littman, M. L. (1994a). Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning*, pp. 157–163.

Littman, M. L. (1994b). Memoryless policies: theoretical limitations and practical results. In *From Animals to Animats 3: Proceedings of the 3rd International Conference on Simulation of Adaptive Behavior (SAB '94)*, pp. 297–305.

Littman, M. L., Cassandra, A. R., & Kaelbling, L. P. (1995). Learning policies for partially observable environments: Scaling up. In *Proceedings of the 12th International Conference on Machine Learning (ICML '95)*, pp. 363–370.

Loch, J., & Singh, S. P. (1998). Using eligibility traces to find the best memoryless policy in partially observable Markov decision processes. In *Proceedings of the 15th International Conference on Machine Learning (ICML '98)*, pp. 323–331.

Lovejoy, W. S. (1991). A survey of algorithmic methods for partially observed Markov decision processes. *Management Science, 28(1)*, 1–16.

Lovejoy, W. S. (1993). Suboptimal policies with bounds for parameter adaptive decision processes. *Operations Research, 41*, 583–599.

Madani, O., Hanks, S., & Condon, A. (1999). On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision processes. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI '99)*, pp. 541–548.

Matsuno, Y., Yamazaki, T., Matsuda, J., & Ishii, S. (2001). A multi-agent reinforcement learning method for a partially-observable competitive game. In *Proceedings of the 5th international conference on autonomous agents*, pp. 39–40.

McCallum, A. K. (1996). *Reinforcement learning with selective perception and hidden state*. Ph.D. thesis, University of Rochester.

McCallum, A. (1993). Overcoming incomplete perception with util distinction memory. In *Proceedings of the 10th International Conference on Machine Learning (ICML '93)*, pp. 190–196.

McCoy, A. N., & Platt, M. L. (2005). Risk-sensitive neurons in macaque posterior cingulate cortex. *Nature Neuroscience, 8*, 1129–1130.

Meuleau, N., Kim, K.-E., Kaelbling, L. P., & Cassandra, A. R. (1999a). Solving POMDPs by searching the space of finite policies. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI '99)*, pp. 417–

426.

Meuleau, N., Peshkin, L., Kim, K.-E., & Kaelbling, L. P. (1999b). Learning finite-state controllers for partially observable environments. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI '99)*, pp. 427–436.

Monahan, G. E. (1982). A survey of partially observable Markov decision processes: Theory, models, and algorithms. *Management Science, 28(1)*, 1–16.

Moore, A. W., & Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with les data and less real time. *Machine Learning, 13*, 103–130.

Mori, T., Nakamura, Y., & Ishii, S. (2004). Reinforcement learning for CPG-driven biped robot. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI '04)*, pp. 623–630.

Morimoto, J., & Doya, K. (2001). Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning. *Robotics and Autonomous Systems, 36*, 37–51.

Murphy, K. P. (2000). A survey of POMDP solution techniques. Tech. rep., Dept. of Computer Science, U. C. Berkeley.

Nagayuki, Y., Ishii, S., & Doya, K. (2000). Multi-agent reinforcement learning: an approach based on the other agent's internal model. In *Proceedings of the 4th International Conference on MultiAgent Systems*, pp. 215–221.

Nair, R., Marsella, S., Tambe, M., Pynadath, D., & Yokoo, M. (2003). Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI '03)*, pp. 705–711.

Nikovski, D., & Nourbakhsh, I. (2000). Learning probabilistic models for decision-theoretic navigation of mobile robots. In *Proceedings of the 17th International Conference on Machine Learning (ICML '00)*, pp. 671–678.

Nishimura, M., Yoshimoto, J., & Ishii, S. (2004). Acrobot control by learning the switching of multiple controllers. In *Proceedings of the Ninth International Symposium on Artificial Life and Robotics (AROB)*, Vol. 2, pp. 633–636.

Nourbakhsh, I., Powers, R., & Birchfield, S. (1995). DERVISH: an office-navigating robot. *AI Magazine, 16(2)*, 53–60.

Papadimitrious, C. H., & Tsitsiklis, J. N. (1987). The complexity of Markov decision processes. *Mathematics of Operations Research, 12(3)*, 441–450.

Parr, R., & Russell, S. (1995). Approximating optimal policies for partially observable stochastic domains. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence (IJCAI '95)*, pp. 1088–1094.

Perkins, T. (1998). Two search techniques for imperfect information games and application to hearts. Tech. rep., Computer Science at University of Massachusetts.

Peshkin, L., Meuleau, N., & Kaelbling, L. P. (1999). Learning policies with external memory. In *Proceedings of the 16th International Conference on Machine Learning (ICML '99)*, pp. 307–314.

Pfahringer, B., Kaindl, H., Kramer, S., & Furnkranz, J. (1999). Learning to make good use of operational advice. Presentation at International Conference on Machine Learning (ICML '99), Workshop on Machine Learning in Game Playing.

Pineau, J., Gordon, G., & Thrun, S. (2003). Point-based value iteration: An anytime algorithm for POMDPs. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI '03)*, pp. 1025–1032.

Platzman, L. K. (1980). A feasible computational approach to infinite-horizon partially observed Makorv decision problems. Tech. rep., Georgia Institute of Technology.

Poupart, P., & Boutilier, C. (2000). Value-directed belief state approximation for POMDPs. In *Proceedings of the 16th Annual Conference on Uncertainty in Artificial Intelligence (UAI '00)*, pp. 409–416.

Poupart, P., & Boutilier, C. (2004). Bounded finite state controllers. In *Advances in Neural Information Processing Systems (NIPS '04)*, Vol. 16, pp. 1–8.

Poupart, P., Ortiz, L. E., & Boutilier, C. (2001). Value-directed sampling methods for monitoring POMDPs. In *Proceedings of the 17th Annual Conference on Uncertainty in Artificial Intelligence (UAI '01)*, pp. 453–461.

Rescorla, R. A., & Wagner, A. R. (1972). A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. *Classical conditioning: Current research and theory, II*, 64–99.

Rodriguez, A., Parr, R., & Koller, D. (1999). Reinforcement learning using approximate belief states. In *Advances in Neural Information Processing Systems (NIPS '99)*, Vol. 12, pp. 1036–1042.

Ron, D., Singer, Y., & Tishby, N. (1994). The power of amnesia: Learning probabilistic automata with variable memory length. In *Advances in Neural Information Processing Systems (NIPS '94)*, Vol. 6, pp. 176–183.

Roy, N., & Gordon, G. (2003). Exponential family PCA for belief compression in POMDPs. In *Advances in Neural Information Processing Systems (NIPS '03)*, Vol. 15, pp. 1043–1049.

Salustowicz, R. P., Wiering, M. A., & Schmidhuber, J. (1998). Learning team strategies: soccer case studies. *Machine Learning, 33*, 263–282.

Sandholm, T. W., & Crites, R. H. (1995). Multiagent reinforcement learning in the

iterated prisoner's dilemma,. *Biosystems*, *37*, 147–166.

Sato, M., & Ishii, S. (2000). On-line EM algorithm for the normalized Gaussian network. *Neural Computation*, *12*, 407–432.

Schultz, W., Dayan, P., & Montague, P. R. (1997). A neural substrate of prediction and reward. *Science*, *275*, 1593–1599.

Sen, S., Sekaran, M., & Hale, J. (1994). Learning to coordinate without sharing information. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI '94)*, pp. 426–431.

Shani, G. (2004). A survey of model-based and model-free methods for resolving perceptual aliasing. Tech. rep., Department of Computer Science at Ben-Gurion University.

Shelton, C. (2001). Policy improvement for POMDPs using normalized importance sampling. In *Proceedings of the 17th Annual Conference on Uncertainty in Artificial Intelligence (UAI-01)*, pp. 496–50.

Singh, S. P., & Bertsekas, D. (1996). Reinforcement learning for dynamic channel allocation in cellular telephone systems. In *Advances in Neural Information Processing Systems (NIPS '96)*, Vol. 9, pp. 974–980.

Singh, S. P., & Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Machine Learning*, *22(1–3)*, 123–158.

Singh, S. P., Jaakkola, T., & Jordan, M. I. (1994). Learning without state-estimation in partially observable Markovian decision processes. In *Proceedings of the 11th International Conference on Machine Learning (ICML '94)*, pp. 284–292.

Smallwood, R. D., & Sondik, E. J. (1973). The optimal control of partially observable processes over a finite horizon. *Operations Research*, *21*, 1071–1088.

Sondik, E. J. (1971). *The optimal control of partially observable Markov processes.* Ph.D. thesis, Stanford University.

Sondik, E. J. (1978). The optimal control of partially observable Markov decision processes over the infinite horizon: Discounted costs. *Operations Research*, *26(2)*, 282–304.

Stone, P., & Veloso, M. M. (2000). Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, *8*, 345–383.

Strange, B. A., Henson, R. N. A., Friston, K. J., & Dolan, R. J. (2001). Anterior prefrontal cortex mediates rule learning in humans. *Cerebral Cortex*, *11*, 1040–1046.

Sturtevant, N. R. (2003). *Multi-player games: Algorithms and Approaches.* Ph.D. thesis, Computer Science Department, UCLA.

Sturtevant, N. R., & White, A. M. (2006). Feature construction for reinforcement

learning in Hearts. In *Computers and Games.*

Sutton, R. S. (1988). Learning to predict by the method of temporal differences. *Machine Learning, 3*, 9–44.

Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the 7th International Conference on Machine Learning (ICML '90)*, pp. 216–224.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction.* MIT Press.

Tan, M. (1993). Multi-agent reinforcement learning: independent vs. cooperative agents. In *Proceedings of the 10th International Conference on Machine Learning*, pp. 330–337.

Tesauro, G. (1994). TD-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation, 6*, 215–219.

Theocharous, G., & Mahadevan, S. (2002). Approximate planning with hierarchical partially observable Markov decision process models for robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '02)*, pp. 1374–1352.

Thorndike, E. L. (1911). *Animal Intelligence.* Hafner, Darien, CT.

Thrun, S. (2000). Monte Carlo POMDPs. In *Advances in Neural Information Processing Systems (NIPS '00)*, Vol. 12, pp. 1064–1070.

Ueno, T., Nakamura, Y., Takuma, T., Shibata, T., Hosoda, K., & Ishii, S. (2006). Fast and stable learning of quasi-passive dynamic walking by an unstable biped robot based on off-policy natural actor-critic. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '06)*, pp. 5226–5231.

Watkins, C., & Dayan, P. (1992). Q-learning. *Machine Learning, 8*, 279–292.

White, C. C. (1991). A survey of solution techniques for the partially observed Markov decision process. *Annals of Operations Research, 32*, 215–230.

Whitehead, S. D., & Lin, L.-J. (1995). Reinforcement learning of non-Markov decision processes. *Artificial Intelligence, 73*, 271–306.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning, 8*, 229–256.

Yoshida, W., & Ishii, S. (2006). Resolution of uncertainty in prefrontal cortex. *Neuron, 50(5)*, 781–789.

Yoshimoto, J., Ishii, S., & Sato, M. (2003). System identification based on on-line variational Bayes method and its application to reinforcement learning. In *Proceedings of the International Conference on Artificial Neural Networks and Neural*

Information Processing, Vol. 2714, pp. 123–131.

Zhang, N. L. (1995). Efficient planning in stochastic domains through exploiting problem characteristics. Tech. rep., HKUST-CS95-40, Hong Kong University of Science and Technology.

Zhang, N. L., & Lee, S. S. (1998). Planning with partially observable Markov decision processes: Advances in exact solution method. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI '98)*, pp. 523–530.

Zhang, N. L., & Liu, W. (1997). A model approximation scheme for planning in partially observable stochastic domains. *Journal of Artificial Intelligence Research*, *7*, 472–480.

Zhang, N. L., & Zhang, W. (2001). Speeding up the convergence of value iteration in partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, *14*, 29–51.

Zhou, R., & Hansen, E. A. (2001). An improved grid-based approximation algorithm for POMDPs. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI '01)*, pp. 707–714.