Nonparametric Bayesian Learning of Other Agents' Policies in Interactive POMDPs

Alessandro Panella and Piotr Gmytrasiewicz
Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607

Abstract

We consider an autonomous agent facing a partially observable, stochastic, multiagent environment where the unknown policies of other agents are represented as finite state controllers (FSCs). We show how an agent can (i) learn the FSCs of the other agents, and (ii) exploit these models during interactions. To separate the issues of off-line versus on-line learning we consider here an off-line two-phase approach. During the first phase the agent observes as the other player(s) are interacting with the environment (the observations may be imperfect and the learning agent is not taking part in the interaction.) The collected data is used to learn an ensemble of FSCs that explain the behavior of the other agent(s) using a Bayesian non-parametric (BNP) approach. We verify the quality of the learned models during the second phase by allowing the agent to compute its own optimal policy and interact with the observed agent. The optimal policy for the learning agent is obtained by solving an interactive POMDP in which the states are augmented by the other agent(s)' possible FSCs. The advantage of using the Bayesian nonparametric approach in the first phase is that the complexity (number of nodes) of the learned controllers is not bounded a priori. Our two-phase approach is preliminary and separates the learning using BNP from the complexities of learning on-line while the other agent may be modifying its policy (on-line approach is subject of our future work.) We describe our implementation and results in a multiagent Tiger domain. Our results show that learning improves the agent's performance, which increases with the amount of data collected during the learning phase.

Introduction

An autonomous, rational agent operating in a stochastic, partially observable environment maximizes its expected utility, usually the discounted sum of future rewards, as in the case of partially observable Markov decision processes (POMDP)(Russell and Norvig 2009; Kaelbling, Littman, and Cassandra 1998). An additional layer of uncertainty in multiagent environments is due to the actions of other agents which affect the state of the world, and possibly our agent's payoff. To increase performance it is useful to predict the actions of other agents. In this work, we consider an agent that maintains explicit models of other agents.

Intentional models are specifications of the other agent's beliefs and preferences, which can be used to simulate their decision making process. For instance, a POMDP-based agent may consider the other agent(s) to be POMDP-based themselves, maintain a probability distribution over their POMDPs, and compute a solution to those models recursively, as in the case of interactive POMDPs (I-POMDPs) (Gmytrasiewicz and Doshi 2005). In general, this approach involves maintaining a probability distribution over all possible agents' specifications and beliefs. The resulting joint probability space is very complex.

An alternative we pursue in this paper is to consider *subintentional models*, which do not contain other agents' payoffs and beliefs. A subintentional model can be as simple as a static probability distribution over actions, or a mapping from observations to actions. The representation we use here is a variant of finite state controllers (FSCs), which provides a good trade-off between complexity and expressive power (Meuleau et al. 1999). We consider FSCs with deterministic transitions between nodes triggered by observations from the environment. Each node represents an internal state of the modeled agent that summarizes its past history, and contains a probabilistic mapping to the action space.

We consider a likely case in which the "protagonist" agent i is not given a finite set of possible models of the other agent (j) but considers the set of all possible controllers. To handle the fact that the size of the controllers is not bounded a priori, we use Bayesian nonparametric methods (Hjort et al. 2010). The crucial advantage of BNP is that it allows agents to represent probability distributions over objects of unbounded size, which is needed given the lack of bounds on the size of other agent's policy. In other words, BNP methods are flexible in that they allow for the learned representation to grow with the observed complexity of the data. Intuitively, we would like to learn a small FSC if it provides a sufficient account for the observed behavior. If the behavior is complex, however, we want to be able to learn a more complex FSC. The problems we tackle are too complex to be amenable to conjugate analysis (Kadane 2011) and we use the computational implementations of Bayesian inference based on Dirichlet processes and Gibbs sampling techniques (Hjort et al. 2010).

We view an on-line model-based Bayesian approach to learning the most realistic and practical. On-line approach assumes that the planning and learning are interleaved, with the agent updating the learned models as it observes and interacts with the other agent. The major complexity of an on-line approach is that it may be unrealistic to assume that the other agent's policy remains constant during interaction. To separate this issue from BNP learning itself in this paper we take an off-line two-phase approach of batch learning. During the *learning* phase the agent (i) accumulates its own observations which probabilistically depend on the state of the world and the actions of the other agent(s) (j). The agent then uses Bayesian nonparametric (BNP) techniques described below to compute an ensemble of plausible FSCs which explain the observations and what is known behavior of the other agent(s). As we mentioned the advantage of BNP is that is does not impose an a priori bound on the complexity of the behavior of the other agent. During the second testing phase the agents interact and agent i *exploits* the learned models, while also adjusting the probabilities of each of the FSCs in its ensemble based on incoming observations using the Bayes rule. We show how i can improve its performance during interactions by exploiting the information it obtained during observation, depending on the length of the observation phase. The online approach to interleaving of the learning and exploitation phases, that provides the capability to interact with adaptive agents (who themselves may also learn,) is the subject of future work.

Our work is related to research on *plan recognition* (Charniak and Goldman 1993); and on goal-based POMDPs (Ramirez and Geffner 2011). In game theory, deterministic finite automata (DFA) have been employed to represent strategies in the presence of *bounded rationality* (Rubinstein 1998) when the opponent's actions are assumed observable. (Carmel and Markovitch 1996) provides a heuristic for the on-line inference of a consistent DFA, which does not guarantee any bound on the complexity of the learned model. However, we do not use the classical game-theoretic solution concept of a Nash equilibrium, building on the growing body of work that uses the decision-theoretic solution concept (Oliehoek and Amato 2014) and behavioral game theory (Wright and Leyton-Brown 2012).

BNP learning techniques have been employed in the context of partially observable policy-based (Liu, Liao, and Carin 2011) and model-based (Doshi-Velez et al. 2010; 2013) reinforcement learning (RL). The difference between that work and ours is that we focus on learning explicit model of the other agent and assume that the transition and reward models (for agent i) are known. In particular, the reward signal is not perceived by agent i at each time step during execution. While it is possible to fold the information about the other agent's actions and how they change the physical state into the transition model, doing so would treat the other agent as a constant noise factor. This would preclude more nuanced coordination based on tracking its policy.

The remainder of this paper is organized as follows. Section provides background on POMDP and FSC concepts. Section describes the BNP learning model and the sampling-based inference algorithm. Section explains how the learned models are used for planning optimally, and experimental results are provided in Section . Section concludes the paper and charts directions for future work.

Background: POMDPs and Finite State Controllers

A Partially observable Markov decision process (POMDP) (Kaelbling, Littman, and Cassandra 1998) is a general model for planning and acting in partially observable, stochastic domains. It is a tuple $P = (S, A, \Omega, T, O, R)$, where: S is the set of possible states of the world; A is the set of agent's actions; Ω is the set of observations the agent can receive; $T : S \times A \times S \to [0, 1]$ is the state transition function; $O : A \times S \times \Omega \to [0, 1]$ is the observation function; $R : S \times A \to \mathbb{R}$ is the reward function. Frequently, an optimal POMDP policy can be represented as a deterministic finite state controller. Some solution methods compute POMDP policies by performing a search directly in the space of FSCs (Hansen 1998), or search in the more general space of stochastic FSCs (Meuleau et al. 1999; Poupart and Boutilier 2003).

We use a version of FSCs called probabilistic deterministic finite controllers (PDFC) to model other agent(s) policies. In PDFCs the transitions between the memory states (nodes) are deterministic but the actions are chosen stochastically in each state of the controller. The actions are stochastic in order to enable more efficient search through the space of all models. Formally, a PDFC is a tuple $c = (\Omega, A, Q, \tau, \theta, q_0)$, where: Ω is the set of observations of the agent; A is the set of actions the agent can execute; Q is the set of nodes, the internal states of the agent; $\tau : Q \times \Omega \to Q$ is the deterministic node transition function; $\theta : Q \to \Delta(A)$ is the probabilistic emission function; q_0 is the initial node.

Bayesian Nonparametric Learning of Finite State Controllers

Description of The Model

In this paper, we propose a Bayesian methodology to learn over a class C of finite state controllers which implement agent j's policy, given a sequence of observations of agent i $\omega^i_{1:T}$ of length T. We wish to infer the posterior distribution over all possible models $c \in C$. Formally, we have:

$$p(c|\omega^i_{1:T}) \propto p(\omega^i_{1:T}|c)p(c), \tag{1}$$

where $p(\omega^i_{1:T}|c)$ is the likelihood of the observed data given the model, and $p(c)$ is the prior distribution over the space C of PDFCs.

Our learning task is equivalent to performing inference in a dynamic Bayesian network such as the one in Figure 1, where the nodes $q_{1:T}$ represent the sequence of internal nodes of the finite controller, $a_{1:T}$ are the actions generated in such nodes, and $\omega^j_{1:T}$ are the observations received by the modeled agent j. As we mentioned, the transitions between nodes are deterministic and represented by a transition function τ, depending on the previous node and the current observation, while the actions are stochastically dependent on

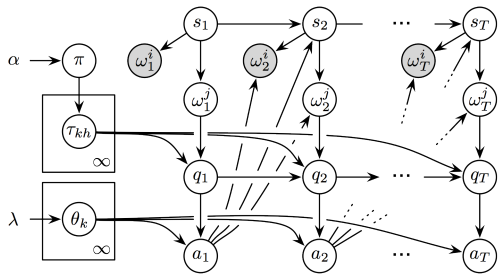

Figure 1: Bayesian model for PDFC learning.

the current node of the controller, i.e.

$$\begin{aligned} q_t &= \tau_{q_{t-1}\omega^j_t} \\ a_t &\sim \theta_{q_t}. \end{aligned} \tag{2}$$

Following the hierarchical Bayesian approach to learning graphical models (Koller and Friedman 2009), we treat the transition and emission functions as stochastic variables. Since we do not know a priori the number of nodes $|Q|$ in the controller, we place a stick-breaking prior distribution (GEM) parameterized by α (Sethuraman 1994) on the values of τ for each node-observation pair (for brevity we write τ_{kh} instead of τ_{q_{t-1},ω_t} below):

$$\begin{aligned} \pi &\sim \mathrm{GEM}(\alpha) \\ \tau_{kh} &\sim \pi \quad \begin{array}{l}\forall k = 1, 2, \dots \\ \forall h = 1, \dots, |\Omega|\end{array}. \end{aligned} \tag{3}$$

The stick-breaking distribution GEM serves to provide a prior over the size of PDFCs. It does so by dividing the unit range $[0, 1]$ into an infinite number of intervals (as in repeatedly breaking a stick) defining variable $\pi = \{\pi_k\}_{k=1}^{\infty}$ as an infinite discrete probability vector over the potentially infinite number of nodes of a PDFC. The "concentration parameter" α affects the values of consecutive π_k which decay exponentially with k: smaller values of α favor a steeper decay, resulting in fewer nodes of the PDFC each with more incoming transitions τ_{kh}. Conversely, larger values of α place a bias towards a more uniform π, hence favoring PDFCs with more nodes.

The actions generated in each node are sampled from a probability vector θ_k and therefore follow a multinomial distribution. We place a Dirichlet distribution as hierarchical prior over these parameters to exploit its conjugacy in the model. We use symmetric Dirichlet distributions over the simplex defined by the action set, i.e. $\theta_k \sim \mathrm{Dir}_{|A|}(\lambda) = \mathrm{Dir}\left(\frac{\lambda}{|A|}, \dots, \frac{\lambda}{|A|}\right)$.

Inference Via Gibbs Sampling

By exploiting the stick-breaking construction to implement Equation 1, we build a collapsed Gibbs sampler to estimate the posterior distribution over PDFCs, whose pseudocode is shown in Algorithm 1, that repeatedly samples each variable given the current values of all the other variables in the model. We note that variables π and θ_k's are not being sampled directly, since they are integrated out analytically. In the remainder of this section, we define the conditional distributions that are used during each step of the Gibbs sampler. Also the nodes $q_{1:T}$ are not sampled since they are uniquely determined by the values of the other variables.

Sampling the transition function (Algorithm 1, line 10). Given the value of the other variables, the probability of τ for each node and observation pair is:

$$\begin{aligned} &p(\tau_{kh}|\alpha, \tau_{-(kh)}, \omega^j_{1:T}, a_{1:T}) \\ &\quad \propto p(\tau_{kh}|\alpha, \tau_{-(kh)})\; p(a_{1:T}|\tau, \omega^j_{1:T}), \end{aligned} \tag{4}$$

where $\tau_{-(kh)}$ denotes all current values of τ except the one being sampled (so that $\tau = \tau_{kh} \cup \tau_{-(kh)}$). The first term on the right-hand side of this equation can be derived from the properties of the so-called Chinese restaurant process (Hjort et al. 2010), that allows to sample τ from the stick-breaking distribution directly, without explicitly representing the vector π. Let us denote as v_i the count of how many times an existing node i happens to be the destination node of any transition in $\tau_{-(kh)}$; we have:

$$\begin{aligned} p(\tau_{kh} = i|\alpha, v) &\propto v_i \quad \text{for existing node } i \\ p(\tau_{kh} = \bar{i}|\alpha, v) &\propto \alpha \quad \text{for new node } \bar{i}. \end{aligned} \tag{5}$$

We note here that new nodes can be generated while sampling τ, and the probability of sampling a new node is proportional to α, the concentration parameter of the Dirichlet process underlying this sampling schema. As discussed in the previous section, larger α values promote the generation of new nodes. Moreover, the probability of generating a new node decays as the set of nodes grows, so to define a proper prior distribution over the number of nodes. The second term on the right-hand side of Equation 4 can be computed by considering that τ and $\omega^j_{1:T}$ jointly determine the values of the node sequence. Let us introduce a count matrix d, where each element d_{ky} represents how many times action y is generated in node k in such sequence. If the current number of node in τ is K, and given that each action is conditionally independent, we can use the properties of the Dirichlet-multinomial model (Gelman et al. 2003), and marginalize over the parameters θ_k's, to obtain:

$$p(a_{1:T}|\tau, \omega^j_{1:T}) = \prod_{k=1}^{K} \left[\frac{\Gamma(\lambda)}{\Gamma(d_{k\cdot} + \lambda)} \prod_{y=1}^{|A|} \frac{\Gamma(d_{ky} + \lambda/|A|)}{\Gamma(\lambda/|A|)} \right]. \tag{6}$$

The quantity $d_{k\cdot}$ is the number of times node k is visited, i.e. $d_{k\cdot} = \sum_{y=1}^{|A|} d_{ky}$.

Sampling the observation sequence (Algorithm 1, line 13). To estimate a posterior distribution over PDFCs, we also need to sample the observation sequence $\omega^j_{1:T}$ of the modeled agent j. Given all the other variables, we can sample $\omega^j_{1:T}$ by using a variation of the forward-backward algorithm for hidden Markov models (Rabiner 1989). From the structure of the graphical model, we can infer that the probability of each observation is:

$$\begin{aligned} &p(\omega^j_t|s_t, a_{t-1:T}, q_{t-1}, \tau) \\ &\quad \propto p(\omega^j_t|a_{t-1}, s_t)p(a_{t:T}|q_{t-1}, \omega^j_t, \tau). \end{aligned} \tag{7}$$

Algorithm 1 Gibbs sampler for PDFC learning.

```
1: $K \leftarrow 1$                                              ▷ Initialize number of nodes to 1
2: $\tau_{1\omega^j} \leftarrow 1 \ \forall \omega^j \in \Omega^j$          ▷ Initialize all transitions from the only node to itself
3: $s_{1:T} \sim p(s_{1:T}|\omega^i_{1:T})$                        ▷ Initialize sequence of states of the world,
4: $a_{1:T} \sim p(a_{1:T}|\omega^i_{1:T}, s_{1:T})$                                         j's actions,
5: $\omega^j_{1:T} \sim p(a_{1:T}|\omega^i_{1:T}, s_{1:T}, a_{1:T})$                         and j's observations

6: for $n = 1..N_{iter}$ do                           ▷ $N_{iter}$ is the number of iterations of the Gibbs sampler
7:     $p_Q \leftarrow$ rand-perm$(1..K)$                        ▷ Permute nodes and observations to
8:     $p_\Omega \leftarrow$ rand-perm$(1..|\Omega|)$            sample transitions in random order to avoid bias
9:     for $k$ in $p_Q$, $h$ in $p_\Omega$ do
10:        Sample $\tau_{kh} \sim p(\tau_{kh}|\alpha, \tau_{-(kh)}, \omega^j_{1:T}, a_{1:T})$
11:    end for
12:    Sample $\omega^j_{1:T} \sim p(\omega^j_{1:T}|s_{1:T}, a_{1:T}, q_{1:T}, \tau)$     ▷ Sample the sequences of j's observations,
13:    Sample $s_{1:T} \sim p(s_{1:T}|s_{1:T}, a_{1:T}, \omega^i_{1:T}, \omega^j_{1:T})$                    states of the world,
14:    Sample $a_{1:T} \sim p(a_{1:T}|q_{1:T}, \omega^i_{1:T}, \omega^j_{1:T}, s_{1:T})$                      and j's actions
15: end for
```

The first term is the observation function of the modeled agent, while the second term can be computed efficiently through a backward probability message (we omit the details here for brevity.)

Note that the assumption that the observation function of the modeled agent is known can be relaxed. If this observation function is not known the modeling agent has to marginalize over every possible value of this function. To do this, we could use hierarchical modeling approach and introduce a variables ν_{sa} which, for each world state s and action a, denote the probabilities of j's observations. We could then place a Dirichlet prior over these quantities.

Sampling the state and action sequences (Algorithm 1, lines 13, 14). The modeling agent i does not perceive directly the sequence of world states and actions of agent j, and only perceives the environment through its own observation model $p(\omega^i_t|a_{t-1}, s_t)$, where the observations belong to a set Ω^i, which of course does not have to be identical to the modeled agent j.

Therefore, in the Gibbs sampler, we sample $s_{1:T}$ and $a_{1:T}$ given the values of the other variables. The world state at each time step can be sampled from

$$\begin{aligned} &p(s_t|s_{t-1}, a_{t-1:T}, \omega^i_{t:T}, \omega^j_{t:T}) \\ &\quad \propto p(s_t|s_{t-1}, a_{t-1})p(\omega^i_{t:T}, \omega^j_{t:T}|s_t, a_{t-1}), \end{aligned} \tag{8}$$

where the first term on the right-hand side is the environment's transition model, that we assume both agents know, and the second term can be efficiently computed using a backward probability message. The sequence of actions can be sampled from

$$\begin{aligned} &p(a_t|q_t, \omega^i_{t+1}, \omega^j_{t+1}, s_t, s_{t+1}) \\ &\quad \propto p(a_t|q_t)p(s_{t+1}|a_t, s_t)p(\omega^i_{t+1}, \omega^j_{t+1}|a_t, s_{t+1}). \end{aligned} \tag{9}$$

Interactive POMDPs

As in the interactive POMDP framework (Gmytrasiewicz and Doshi 2005), we extend the definition of POMDP to multiagent settings by defining a tuple

$$\bar{P}_i = (\bar{S}_i, A, \Omega_i, T, O_i, R_i),$$

where A is the set of *joint* actions $a = (a_i, a_{-i})$ and the transition function T describes how the world evolves as an effect of joint actions; similarly, O_i and R_i specify how agent i receives observation and rewards, depending on the state and the joint action performed. The "interactive state space" $\bar{S}_i = S \times M_{-i}$ is the cross product of the physical state and the set of possible models of other agent(s), each of which is a tuple $m_j = (h_j, f_j, O_j)$, where $f_j : H_j \rightarrow \Delta(A_j)$ is an agent function mapping observation histories to distributions over actions, and h_j is a particular observation history. Here, we consider models where the agent function is implemented by a PDFC $c_j \in C_j$ and the history of observations is replaced by the internal state q_j, i.e. $m_j = (q_j, c_j, O_j)$. Note that, with respect to the more general I-POMDP case, we only consider a class of *subintentional* models for other agents (i.e. their reward function is not explicitly modeled.) However, this formalization maintains the perspective of an autonomous agent during planning and execution, assuming that the features of other agents are not known and observable, unlike the case partially observable stochastic games (POSG) (Hansen, Bernstein, and Zilberstein 2004). Our model is closer to the *best-response model* (BRM) defined in (Oliehoek and Amato 2014).

In the following, we focus on the formalization which includes one other agent j, but the N-agents generalization is straightforward. An element of $\bar{S}_i$ is therefore a tuple $\bar{s} = (s, q_j, c_j, O_j)$. We assume that the PDFCs of the other agent does not change during execution, and that the observation model of the other agent O_j is known. Moreover, we consider a finite set of PDFCs C_j (it will be the finite ensemble of models obtained during the learning phase.) The belief update function returns the probability of an interactive state when action a_i is executed and observation ω_i is received, given the previous belief over $\bar{S}_i$. The formula can be derived as in I-POMDPs, by considering agent i's predic-

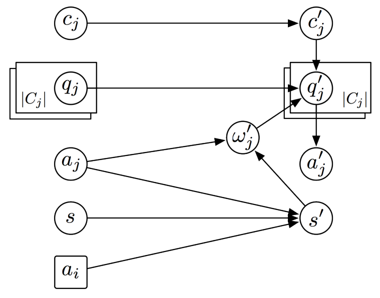

Figure 2: Two-slice dynamic Bayesian network representing the interactive state transition in an I-POMDP with subintentional models.

tion over j's action, as follows:

$$p(\bar{s}'|\bar{b}, a_i, \omega_i) = \\ \beta \sum_{\bar{s}\,:\,c_j = c'_j} \bar{b}(\bar{s}) \sum_{a_j} p(a_j|q_j, c_j)\; O_i(a_i, a_j, s', \omega_i) \\ \times T(s, a_i, a_j, s') \sum_{\omega_j} O_j(a_i, a_j, s', \omega_j)\; p(q'_j|q_j, c_j, \omega_j), \tag{10}$$

where β is a normalization constant, and $p(a_j|q_j, c_j)$ and $p(q'_j|q_j, c_j, \omega_j)$ are derived from the components θ and τ of PDFC c_j. The value function is defined similarly to POMDPs, and its property of piece wise linearity and convexity carries over from the single-agent POMDP case (Gmytrasiewicz and Doshi 2005).

Standard POMDP algorithms can be adapted to solve subintentional I-POMDPs as the one described since there is no nested intentional beliefs. Moreover, the interactive state space $\bar{S}_i$ can be factorized into individual random variables, so that solving methods can be devised that do not work on the full joint state space. In this paper, we adopt Symbolic Perseus (Poupart 2005), which is a point-based approximation algorithm that exploits state factorization by representing the POMDP as a dynamic Bayesian network, and context-specific independence by using algebraic decision diagrams to represent the conditional probabilities as depicted in Figure 2.

Experiments

We report the experimental results involving agent i observing an agent, j, acting in three different instances of the Tiger Problem (Kaelbling, Littman, and Cassandra 1998), with three different hearing accuracies of 0.96, 0.85 and 0.8. In each case we assume that the modeled agent j acts according to the exact optimal solution of its POMDP. These solutions can be represented as deterministic finite state controllers containing 3, 5, and 7 nodes for each of the hearing accuracies above. They are depicted in Figure 4. We assume that the modeling agent i hears a growl (GL and GR) from the correct door (one with the tiger) with 0.85 accuracy. Additionally, if agent j opens a door, i hears a creak (CL or CR) coming from the door being opened with probability 0.9, a creak from the other direction with probability 0.05, and silence (S) with probability 0.05. If j does not open any doors i hears a silence with probability 0.9 and CL and CR with probabilities 0.05 each.

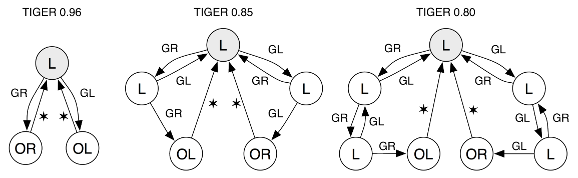

Figure 4: Diagrams of the POMDP policies generating the data.

In order to fully specify a prior distribution over the space of PDFCs, we have to pick some values for the parameters of the models. We use $\lambda = 0.1|A|$ as the parameter of the Dirichlet prior over the probabilities, θ_k, which govern the selection of actions from set $A = \{Listen, OpenLeft, OpenRight\}$ (henceforth designated as L, OL and OR.) The prior distribution over the number of nodes of the learned PDFCs is parameterized by quantity α via the stick breaking construction (Antoniak 1974). In order to make the learning process more flexible we place a hierarchical prior $\text{Gamma}(1, 0.1)$ over α itself (this is not depicted in Figure 1 above.)

During the interaction phase both agents can open doors or listen. Any opening of the doors resets the location of the agent, so agent i has to use the information it has from the learning phase, as well as from creaks during interactions, to align its own behavior with the expected behavior of agent j. For each value of T_{learn} we ran 40 leaning trials each resulting in an ensemble of PDFCs for agent j, by running the Gibbs sampler for 100 iterations. From this ensemble we picked the set of four most likely (a posteriori) PDFCs and as the set $|C_j|$ of models of agent j in agent i's I-POMDP. The solution of this I-POMDP was then ran 100 times (for 50-step interaction sequences each.) The standard deviation bars in Figure 3 are fairly wide due to occasional high magnitude payoffs of -100 when the agent i encounters the tiger. The full horizontal lines in the figures mark the performance of the optimal policy of agent i when full knowledge of the FSC of the agent j, and the dashed lines represent the standard deviation of the rewards obtained in such ideal case.

For each value of the hearing accuracy of agent j (and the corresponding deterministic finite state controllers containing 3, 5 and 7 nodes, respectively,) we show how the performance of agent i during a 50-step long interaction phase depends of the length of the observation sequence T_{learn}. They are depicted in Figure 3. We see that the increased length of the observation phase allows agent i to learn better quality models of agent j and, in general, increase its reward during the subsequent interactions. Note also that the increase of the accumulated reward due to learning better models of j shows up later (after a greater number of observations) for more complex models. This makes sense and shows that it takes more observations to learn a useful model if the behavior is generated by a more complex policy.

Let us look at some special cases. The optimal policy of

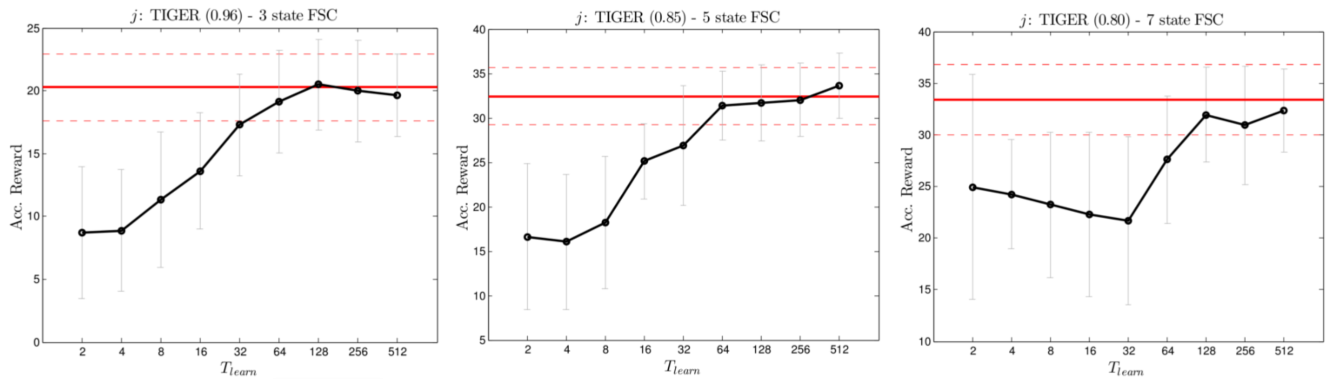

Figure 3: Mean reward over 50 steps for three different policies of agent j. The vertical bars represent the standard deviation.

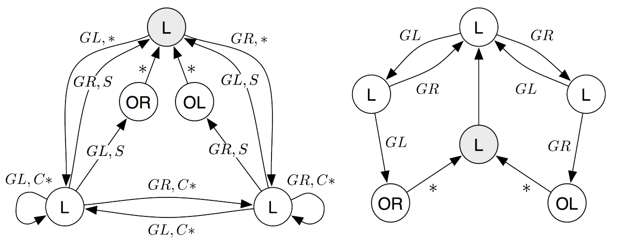

Figure 5: I-POMDP optimal policies, computed after learning from very few observations (left), and given a sequence of 256 observations (right.) The '$*$' symbol represents "don't care's".

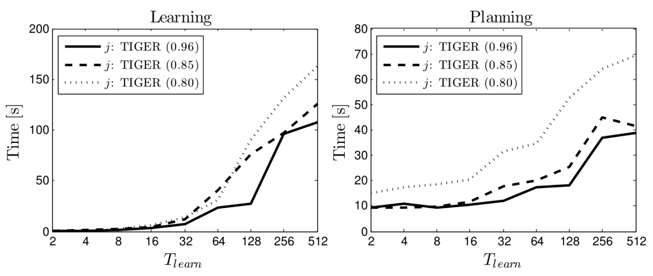

Figure 6: Timing results for learning and planning.

agent i for small values of T_{learn} is depicted in Figure 5-left. This policy does not rely on the learned model of agent j, which is non-informative anyway, and instead relies on the information contained in the creaks. It uses the absence of creaks to wait for a sequence of two interactions during which i can listen and hope for two consistent growls which give it enough confidence to open the opposite door. Note that i gets an informative growl immediately after j's opening of the door; the direction of the growl informs i about the tiger's position after the reset. The performance of this optimal policy is modest, since it does not include much useful information from the learning phase.

Let us contrast the above with i's optimal policy after a long observation sequence (T_{learn} was close to 256) of agent j equipped with quite accurate hearing and using the 3-nodes FSC depicted in Figure 4. Agent i's optimal policy is depicted in Figure 5-right. Here i was able to closely approximate the real policy of j and align its own policy to maximize its own reward. Note that the creaks are irrelevant now because i knows that j alternates listening and opening of the doors. Its own policy aligns its listening actions to hope to get two consistent growls from the same physical state when j also listens and does not reset the tiger.

Figure 6 reports timing results. The plots depict the average time measured across the simulations described above. As expected, the time required by learning grows with the length of the training sequence. Moreover, the charts show that it takes slightly longer to learn more complex models; this is explained by the fact that the size K of the controller being learned has an impact on the running time, due to the inner loop in Algorithm 1. The time spent for planning also grows with the size of the observation sequence, albeit the growth is less pronounced. This increase in planning time is due to the fact that the PDFCs tend to have more nodes when learned from longer observation sequences and for more complex models, and therefore the planning algorithm has to deal with a larger probability space.

Conclusions and Future Work

We described an approach to learning subintentional models of an agent based on imperfect observations. The models we use are probabilistic deterministic finite controllers. We avoid a priori assumptions about the complexity of the learned model by using hierarchical Bayesian nonparametric approach to learning. Our implementation uses Gibbs sampling methods. The results are intuitive: The learned models are able to recover the complexity of the policies the other agent uses if the learning can use sufficient amount of observations. If the training data is sparse the learning is less informative and the quality of interaction is lower, but still optimal given the data.

Our future work will concentrate to enabling the modeling agent to interleave on-line learning and interactions. This leads to the challenging case of both agents possibly learning while interacting, discussed for example in (Shoham and Leyton-Brown 2008).

References

Antoniak, C. E. 1974. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics* 2(6):1152–1174.

Carmel, D., and Markovitch, S. 1996. Learning models of intelligent agents. In *Proceedings of the 13th National Conference on Artificial intelligence*, 62–67.

Charniak, E., and Goldman, R. P. 1993. A Bayesian model of plan recognition. *Artificial Intelligence* 64(1):53–79.

Doshi-Velez, F.; Wingate, D.; Roy, N.; and Tenenbaum, J. 2010. Nonparametric Bayesian policy priors for reinforcement learning. In *Advances in Neural Information Processing Systems 23*, 532–540.

Doshi-Velez, F.; Pfau, D.; Wood, F.; and Roy, N. 2013. Bayesian nonparametric methods for partially-observable reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 99(PrePrints):1.

Gelman, A.; Carlin, J. B.; Stern, H. S.; and Rubin, D. B. 2003. *Bayesian Data Analysis, Second Edition*. Chapman and Hall/CRC, 2 edition edition.

Gmytrasiewicz, P. J., and Doshi, P. 2005. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research* 24(1):49–79.

Hansen, E. A.; Bernstein, D. S.; and Zilberstein, S. 2004. Dynamic programming for partially observable stochastic games. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, 709–715.

Hansen, E. 1998. Solving POMDPs by searching in policy space. In *Proceedings of the 14th International Conference on Uncertainty In Artificial Intelligence*, 211–219.

Hjort, N. L.; Holmes, C.; Müller, P.; and Walker, S. G., eds. 2010. *Bayesian Nonparametrics*. Cambridge University Press.

Kadane, J. B. 2011. *Principles of Uncertainty*. Chapman and Hall/CRC, 1st edition.

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

Koller, D., and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

Liu, M.; Liao, X.; and Carin, L. 2011. The infinite regionalized policy representation. In Getoor, L., and Scheffer, T., eds., *Proceedings of the 28th International Conference on Machine Learning*, 769–776.

Meuleau, N.; Peshkin, L.; Kim, K.-e.; and Kaelbling, L. P. 1999. Learning finite-state controllers for partially observable environments. In *Proceedings of the 15th International Conference on Uncertainty In Artificial Intelligence*, 427–436.

Oliehoek, F. A., and Amato, C. 2014. Best response bayesian reinforcement learning for multiagent systems with state uncertainty. In *AAMAS Workshop on Multiagent Sequential Decision Making Under Uncertainty*.

Poupart, P., and Boutilier, C. 2003. Bounded finite state controllers. In *Advances in Neural Information Processing Systems 16*.

Poupart, P. 2005. *Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes*. Ph.D. Dissertation, University of Toronto, Toronto, Ont., Canada, Canada. AAINR02727.

Rabiner, L. R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, 257–286.

Ramirez, M., and Geffner, H. 2011. Goal recognition over POMDPs: inferring the intention of a POMDP agent. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, 2009–2014.

Rubinstein, A. 1998. *Modeling Bounded Rationality*. MIT Press.

Russell, S., and Norvig, P. 2009. *Artificial Intelligence: A Modern Approach (3rd Edition)*. Prentice Hall, 3rd edition edition.

Sethuraman, J. 1994. A constructive definition of Dirichlet priors. *Statistica Sinica* 4:639–650.

Shoham, Y., and Leyton-Brown, K. 2008. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. New York, NY, USA: Cambridge University Press.

Wright, J. R., and Leyton-Brown, K. 2012. Behavioral game theoretic models: a bayesian framework for parameter analysis. In *International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2012, Valencia, Spain, June 4-8, 2012 (3 Volumes)*, 921–930.