Scaling Expectation-Maximization for Inverse Reinforcement Learning to Multiple Robots under Occlusion

Kenneth Bogert
Department of Computer Science
University of North Carolina
Asheville, NC 28804
kbogert@unca.edu

Prashant Doshi
THINC Lab, Dept. of Computer Science
University of Georgia
Athens, GA 30602
pdoshi@cs.uga.edu

ABSTRACT

We consider inverse reinforcement learning (IRL) when portions of the expert's trajectory are occluded from the learner. For example, two experts performing tasks in close proximity may block each other from the learner's view or the learner is a robot observing mobile robots from a fixed position with limited sensor range. Previous methods mitigate this challenge by either focusing on the observed data only or by forming an expectation over the missing portion of the expert's trajectories given observed data. However, not only is the resulting optimization nonlinear and nonconvex, the space of occluded trajectories may be very large especially when multiple agents are observed over an extended time, which makes it intractable to compute the expectation. We present methods for speeding up the computation of conditional expectations by employing blocked Gibbs sampling. Challenged by a time-limited, multi-robot domain we explore various blocking schemes and demonstrate that our methods offer significantly improved performance over existing IRL techniques under occlusion.

1. INTRODUCTION

Inverse reinforcement learning (IRL) [1, 2] refers to both the problem and associated methods by which an agent learns the preferences of another agent engaged in performing a task simply by passively observing it. The expert is modeled as a Markov decision process (MDP) [3] whose parameters except for the reward function are known to the learner. The problem is usually made feasible by assuming that the reward function is a linear combination of binary feature functions, and the problem reduces to that of finding the weights associated with the feature functions.

IRL in the real world is often challenged by *occlusion*. For example, two experts performing tasks in close proximity may block each other from the learner's view [4, 5] or the learner is a robot observing mobile robots in motion from a fixed position with limited sensor range. This makes the optimization further degenerate. A straightforward method in this context is to simply remove the occluded states and actions from consideration [6]. While this omission is not egregious in some applications, in others it may be that some features predominantly activate in the occluded portions only. For example, the learner observing a cyclically patrolling robot from a vantage point with a limited field of view may never see the robot turning around. Consequently, no data is available to guide the weighting of these features.

A more effective approach is to form an expectation over the missing data conditioned on the observed portions. Bogert et al. [7] adapt maximum-entropy optimization involving latent variables situated within the expectation-maximization schema [8] for application in IRL. Specifically, a distribution over the set of all trajectories with the maximum entropy is sought, which is constrained to match the observed feature expectations. When portions of the trajectory are hidden, a distribution over possible hidden portions conditioned on the observed part is formed. However, a significant limitation is that computing this distribution becomes intractable when the occluded portion is large because the size of the underlying set of possible trajectories grows exponentially with length. This growth exacerbates to being doubly exponential in domains with two experts that interact due to which joint trajectories must be considered.

Clearly, IRL involving multiple observed agents and exhibiting moderate to high occlusion of their trajectories require computing the conditional expectation over a large latent space. This quickly precludes solving multi-expert problems exactly. The primary contribution of this paper is to reclaim tractability by introducing a method for computing the conditional expectations more efficiently while flexibly trading off exactness. Our method intuitively models the stochastic process generating the joint trajectories of length T as a dynamic Bayesian network (DBN) of T slices, thereby reducing the problem to one of Bayesian inference. However, some slices of this DBN, corresponding to the occluded sequences, are fully hidden while perfect evidence is available for others. As hidden slices could alternate with observed ones, an exact inference technique such as the forward-backward algorithm is appropriate. While linear in T, the complexity of forward-backward remains exponential in the number of agents. Consequently, we show how we may utilize Gibbs sampling [9] for performing fast inference. The typically slow convergence rate of Gibbs sampling is speeded up by using a *coordinate (variable) blocking* scheme.

We comprehensively evaluate our method for multi-expert IRL under occlusion on a previously introduced robotic application [6]. It involves a robot L observing from a fixed vantage point two robotic patrollers executing cyclic trajectories. L's view is drastically limited and it is tasked with penetrating the patrol to reach a goal state. We compare our approximate solution to a previous method in both simulation and on physical robots, and show that the blocked Gibbs sampling is effective in quickly computing the conditional expectations. It offers improved inverse learning accuracy under time constraints.

Rest of the paper is structured as follows. In Section 2, we briefly review IRL and a well-known method that maximizes entropy. We

Appears in: *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

also review how this method is extended to settings involving occlusion using EM. Section 3 details the methodological contribution of this paper by introducing a technique for speeding up the computation of the E-step, and gives the main algorithm. The domain and evaluation metrics are presented in Section 4 followed by the experiments in Section 5. We contrast our approach with related work in Section 6, and conclude in Section 7.

2. BACKGROUND

Informally, IRL refers to both the problem and method by which an agent learns preferences of another agent that explain the latter's observed behavior [1]. Usually considered an "expert" in the task that it is performing, the observed agent, say I, is modeled as executing the optimal policy of a standard MDP defined as $\langle S_I, A_I, T_I, R_I \rangle$. The learning agent is assumed to perfectly know the parameters of the MDP except the reward function. Consequently, the learner's task may be viewed as finding a reward function under which the expert's observed behavior is optimal.

This problem in general is ill-posed because for any given behavior there are infinitely-many reward functions which align with the behavior. Abbeel and Ng [10] present an algorithm that allows the expert I to provide task demonstrations instead of its policy. The reward function is modeled as a linear combination of K binary features, ϕ: $S_I \times A_I \rightarrow \{0, 1\}$, each of which maps a state from the set of states S_I and an action from the set of I's actions A_I to either a 0 or 1. Note that non-binary feature functions can always be converted into binary feature functions although there will be more of them. Throughout this article, we assume that these features are known to or selected by the learner. The reward function for expert I is then defined as $R_I(s,a) = \sum_{k=1}^{K} \theta_k \cdot \phi_k(s,a)$, where θ_k are the *weights*. The learner's task is reduced to finding a vector of weights that complete the reward function, and subsequently the MDP such that the demonstrated behavior is optimal.

To assist in finding the weights, feature expectations are calculated for the expert's demonstration and compared to those of possible trajectories [11]. A demonstration is provided as one or more *trajectories*, which are a sequence of length-T state-action pairs, $(\langle s,a \rangle_1, \langle s,a \rangle_2, \ldots \langle s,a \rangle_T)$, corresponding to an observation of the expert's behavior across T time steps. Feature expectations of the expert are averages over all observed trajectories, $\hat{\phi}_k = \frac{1}{|X|} \sum_{x \in X} \sum_{\langle s,a \rangle \in x} \phi_k(s,a)$, where x is a trajectory in the set of all observed trajectories, X.

Given a set of reward weights the expert's MDP is completed and solved optimally to produce π_I^*. The difference $\hat{\phi} - \phi^{\pi_I^*}$ provides a gradient with respect to the reward weights for a numerical solver. To resolve the degeneracy of this problem, Abbeel and Ng [10] maximize the margin between the value of the optimal policy and the next best policy. The resulting program may be solved with a quadratic program solver such as a support vector machine.

2.1 Maximum Entropy IRL

While expected to be valid in some contexts, the max-margin approach introduces a bias into the learned reward function in general. To address this, Ziebart et al. [11] find the distribution with *maximum entropy* over all trajectories that is constrained to match the observed feature expectations.

$$\begin{aligned} &\max_{\Delta} \left(- \textstyle\sum_{X \in \mathbb{X}} Pr(X)\ log\ Pr(X) \right) \\ &\textbf{subject to} \quad \textstyle\sum_{X \in \mathbb{X}} Pr(X) = 1 \\ &\textstyle\sum_{X \in \mathbb{X}} Pr(X) \sum_{\langle s,a \rangle \in X} \phi_k(s,a) = \hat{\phi}_k \qquad \forall k \end{aligned} \tag{1}$$

Here, Δ is the space of all distributions $Pr(X)$. The benefit is that this distribution makes no further assumptions beyond those which are needed to match its constraints and is maximally noncommittal to any one trajectory. As such, it is most generalizable by being the least wrong most often of all alternative distributions. *A disadvantage of this approach is that it becomes intractable for long trajectories because the set of trajectories grows exponentially with length.* In this regard, another formulation defines the maximum entropy distribution over policies [12], the size of which is also large but fixed.

2.2 IRL under Occlusion

Our motivating application involves a subject robot that must observe other mobile robots from a fixed vantage point. Its sensors allow it a limited observation area; within this area it can observe the other robots fully, outside this area it cannot observe at all. Previous methods [6, 13] denote this special case of partial observability where certain states are either fully observable or fully hidden as *occlusion*. Subsequently, the trajectories gathered by the learner exhibit missing data associated with time steps where the expert robot is in one of the occluded states. The empirical feature expectation of the expert $\hat{\phi}_k$ will therefore exclude the occluded states (and actions in those states).

To ensure that the feature expectation constraint in IRL accounts for the missing data, Bogert and Doshi [6] while maximizing entropy over policies [12] limit the calculation of feature expectations for policies to observable states only. However, an associated limitation of this method labeled as mIRL*, is that features active in the occluded states only exhibit a gradient of 0 due to which the gradient of the Lagrangian dual may be unavailable. Therefore, to find the solution a numerical function minimizer that does not use the gradient is required such as Nelder-Mead's simplex [14].

A recent approach [7] improved on the limitations of mIRL* by taking an expectation over the missing data conditioned on the observations. Completing the missing data in this way allows the use of all states in the constraint and with it the Lagrangian dual's gradient as well. The nonlinear program in (1) is modified to account for the hidden data and its expectation.

Let Y be the observed portion of a trajectory, Z is one way of completing the hidden portions of this trajectory, $\mathbb{Z}$ is the set of all possible Z, and $X = (Y \cup Z)$. Now we may treat Z as a latent variable and take the expectation to arrive at a new definition for the expert's feature expectations:

$$\hat{\phi}_k^{Z|_Y} \triangleq \frac{1}{|\mathcal{Y}|} \sum_{Y \in \mathcal{Y}} \sum_{Z \in \mathbb{Z}} Pr(Z|Y;\boldsymbol{\theta}) \sum_{\langle s,a \rangle \in Y \cup Z} \phi_k(s,a) \tag{2}$$

where $\mathcal{Y}$ is the set of all observed Y and $\mathcal{X}$ is the set of all complete trajectories. The program in (1) is modified by replacing $\hat{\phi}_k$ with $\hat{\phi}_k^{Z|_Y}$, as we show below. Notice that in the case of no occlusion $\mathbb{Z}$ is empty and $\mathcal{X} = \mathcal{Y}$. Therefore $\hat{\phi}_k^{Z|_Y} = \hat{\phi}_k$ and this method reduces to (1). Thus, this method generalizes the previous maximum entropy IRL method.

$$\begin{aligned} &\max_{\Delta} \left(- \textstyle\sum_{X \in \mathbb{X}} Pr(X)\ log\ Pr(X) \right) \\ &\textbf{subject to} \quad \textstyle\sum_{X \in \mathbb{X}} Pr(X) = 1 \\ &\textstyle\sum_{X \in \mathbb{X}} Pr(X) \sum_{\langle s,a \rangle \in X} \phi_k(s,a) = \hat{\phi}_k^{Z|_Y} \qquad \forall k \end{aligned} \tag{3}$$

However, the program in (3) becomes nonconvex due to the presence of $Pr(Z|Y)$. As such, finding its optima by Lagrangian relaxation is not trivial. Wang et al. [15] suggests a log linear approximation to obtain maximizing $Pr(X)$ and casts the problem of finding the reward weights as likelihood maximization that can

be solved within the schema of expectation-maximization [16]. An application of this approach to the problem of IRL under occlusion yields the following two steps with more details in [7]:

E-step This step involves calculating Eq. 2 to arrive at $\hat{\phi}_k^{Z|_Y,(t)}$, a conditional expectation of the K feature functions using the parameter $\boldsymbol{\theta}^{(t)}$ from the previous iteration. We may initialize the parameter vector randomly.

M-step In this step, the modified version of the constrained maximum entropy program of (1) is optimized by utilizing $\hat{\phi}_k^{Z|_Y,(t)}$ from the E-step above as the expert's feature expectations to obtain $\boldsymbol{\theta}^{(t+1)}$. An adaptive exponentiated gradient descent [17] solves the program.

As EM may converge to local minima, this process is repeated with random initial $\boldsymbol{\theta}$ and the solution with the maximum entropy is chosen as the final one.

3. MULTI-EXPERT IRL UNDER OCCLUSION

We focus on domains involving multiple experts as exemplified by the two-robot patrolling scenario. IRL in such contexts may model the experts as a multiagent MDP [18], where the joint state and actions of all observed agents influence the transitions and rewards. Formally, let $S = S_I \times S_J$ be the state of the observed system consisting of two experts, I and J, whose set of states is S_I and S_J, respectively. Let A_I and A_J be the sets of I's and J's actions, respectively. Dynamics of the multiagent MDP are generally modeled using a two-time slice DBN as shown in Fig. 1. As the next states of the two experts in our domain of interest are not explicitly correlated, the joint transition probability $\mathcal{T}(\langle S_I^{t+1}, S_J^{t+1}\rangle|\langle A_I^t, A_J^t\rangle, \langle S_I^t, S_J^t\rangle)$ is factored by applying conditional independence, and it becomes $\mathcal{T}_I(S_I^{t+1}|\langle A_I^t, A_J^t\rangle, \langle S_I^t, S_J^t\rangle) \times \mathcal{T}_J(S_J^{t+1}|\langle A_I^t, A_J^t\rangle, \langle S_I^t, S_J^t\rangle)$.

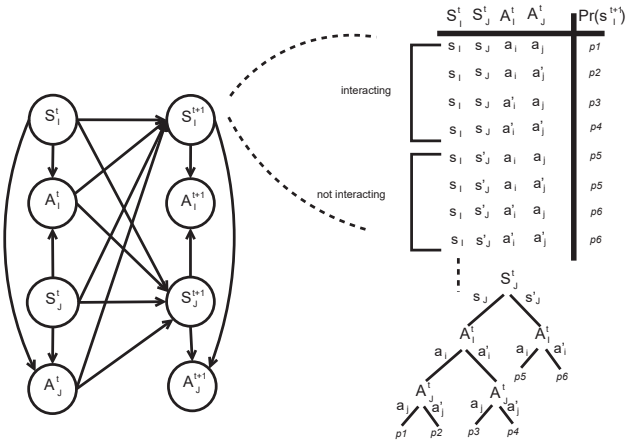


Figure 1: DBN modeling the dynamics and policy of a two-agent MDP. Assuming that each node is binary valued for illustration, the state and actions of both agents I and J influence the transition functions in the interacting state $\langle s_I, s_J\rangle$. However, in the non-interacting state $\langle s_I, s'_J\rangle$, the distribution over S_I^{t+1} is independent of J's state and actions. This context-specific independence facilitates a more efficient representation of the CPT as a tree.

While supporting full generality, this approach makes IRL significantly intractable because implementations of IRL must solve this large MDP repeatedly during the search for optimal reward weights. Occlusion further complicates matters because one of the robots may be observed at a time step while the other is not, resulting in the state of the joint MDP being partially observed. The resulting problem may not be solved by traditional IRL methods that assume perfect observability of the trajectories.

However, in domains where the experts interact *sparsely* such as only when the two patrolling robots approach each other in the narrow corridor, we may continue to model each robot using a separate, individual MDP. The behavior of the robots during an interaction is modeled separately and overrides the behavior prescribed by these MDPs for the duration of the interaction. To avoid a collision, one of the patrollers stops while the other slows down to sidestep it.

Let $S_{int} \in S$ be the set of those joint states where I and J interact. In our patrolling example, these could be the same or adjacent grid cells and the two patrollers are facing each other. Then, $\bar{S}_{int}$ are those states of the multi-expert system devoid of any interaction. For any state $s = \langle s_I, s_J\rangle \in \bar{S}_{int}$, transition probability for an expert say I, $\mathcal{T}_I(S_I^{t+1}|\langle A_I^t, A_J^t\rangle, \langle s_I^t, s_J^t\rangle) = T_I(S_I^{t+1}|A_I^t, s_I^t)$, and analogously for J. Recall that T_I is the transition function of I's individual MDP. For all other states, $\mathcal{T}_I(S_I^{t+1}|\langle A_I^t, A_J^t\rangle, \langle s_I^t, s_J^t\rangle)$ may not be simplified due to the interaction. The above simplification is a classic example of *context-specific independence* [19], which allows us to represent the conditional probability tables of the chance nodes S_I^{t+1} and S_J^{t+1} as efficient trees. We illustrate this in Fig. 1 and point out the unbalanced tree due to the context-specific independence.

Our aim is to generalize the EM based maximum-entropy IRL reviewed in the previous section to multiple agents by using this approach. The outcome will be a method that operates well under occlusion and scales to multiple experts that interact sparsely. The computational challenge is that the set of all possible trajectories grows exponentially in the number of observed agents and doubly exponentially with length. Specifically, $X = ((S_I \times A_I)^N)^T$, where N is the number of observed agents each of whose state and action sets are assumed to be same as I's for simplicity and T is the length of the trajectory. We present an approach to quickly infer the relevant joint distributions next.

3.1 Blocked Gibbs Sampling

Equation 2 in the E-step requires us to calculate $\sum_{Z\in\mathbb{Z}} Pr(Z|Y; \boldsymbol{\theta}^{(t)})$ for each incomplete trajectory Y.

$$Pr(Z|Y;\boldsymbol{\theta}^{(t)}) = \frac{Pr(X;\boldsymbol{\theta}^{(t)})}{\Pr(Y;\boldsymbol{\theta}^{(t)})} \propto Pr(X;\boldsymbol{\theta}^{(t)}) \qquad (4)$$

Computing $Pr(Z|Y;\boldsymbol{\theta}^{(t)})$ is feasible for small-sized $\mathbb{Z}$ by enumerating all possible ways in which a given Y may be completed and obtaining the probability of each complete trajectory $Pr(X)$. In the event that an expert is occluded from view the corresponding state and action nodes at that time step will be hidden. Consequently, if T' is the length of the trajectory that is occluded from the learner, $|\mathbb{Z}| = \mathcal{O}((((|S_{occ}||A_I|)^N)^{T'})$ where S_{occ} is the subset of states occluded from the learner. These would be the coordinate locations (mapped to grid cells) that are not visible to the learner in our patrolling domain.

As Fig. 2 illustrates, hidden nodes in several contiguous time slices may be interspersed with a few observed time slices when the agent comes into view. The two agents may not be in the view of L simultaneously, as in our patroller domain, due to which all nodes in the same time slice may not be hidden. The extent of hidden nodes presents a significant challenge to inferring the joint $Pr(X;\boldsymbol{\theta}^{(t)})$ tractably as hidden nodes are summed out in normalizing it.

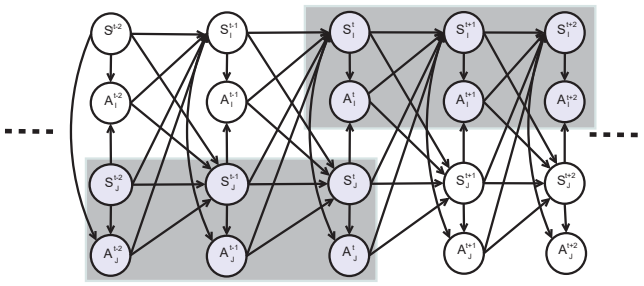


Figure 2: A DBN model of the two-expert trajectory. There are as many time slices in the DBN as the number of steps in the trajectory. State-action pairs of each agent where the state belongs to S_{occ} are hidden, and these are shown shaded. Here, J is two time steps behind I and the learner does not observe any robot currently. J will come into L's view in the next time step.

An appealing method for computing the joint that is well suited to extensive occlusion is the forward-backward message passing algorithm [20]. This filtering and smoothing algorithm makes use of dynamic programming to calculate the posterior marginal probabilities of the hidden states of a Markov chain and has a time complexity of $O(T'(|S_{occ}||A_I|)^N)$. This is no longer exponential in time as compared to a naive inferencing scheme.

3.1.1 Gibbs Sampling

In robotic applications, the state is often multi-dimensional because just representing a robot's pose requires tracking its two or three-dimensional coordinates along with its orientation. A principled method for efficiently computing the posterior of a DBN that models a multi-dimensional evolving state, and which is exact in the limit is Gibbs sampling [9]. A Markov chain Monte Carlo method, Gibbs sampling is well suited for approximating the distribution of a BN with hidden variables. It is a special case of the Metropolis–Hastings algorithm in which the probabilities of each individual node are known and can be sampled from, but the distribution over the entire network is intractable. Sampling proceeds by first randomly assigning all hidden nodes and then repeatedly sampling each node conditioned on the current or just sampled values of all other nodes. Gibbs sampling distinguishes itself by utilizing newly sampled values of variables as soon as we obtain them. This procedure generates a Markov chain of samples where each complete network generated from samples depends on the previous one only, and over time the sequence of networks approaches the true joint distribution.

As each node in a BN is conditionally independent of all other nodes given its Markov blanket, the nodes in multi-agent Z may be sampled from these distributions:

$$Pr(s_I^t|MB(s_I^t)) = \eta\, T_I(s_I^{t-1}, s_J^{t-1}, a_I^{t-1}, a_J^{t-1}, s_I^t)\times$$
$$T_I(s_I^t, s_J^t, a_I^t, a_J^t, s_I^{t+1})\, T_J(s_J^t, s_I^t, a_J^t, a_I^t, s_J^{t+1})\times$$
$$Pr(a_I^t|s_I^t, s_J^t)\; Pr(a_J^t|s_J^t, s_I^t); \text{and}$$
$$Pr(a_I^t|MB(a_I^t)) = \eta'\, T_I(s_I^t, s_J^t, a_I^t, a_J^t, s_I^{t+1})\; Pr(a_I^t|s_I^t, s_J^t)$$

Sampled nodes are states or actions; $MB(\cdot)$ is the Markov blanket of the argument node; η and η' are normalizers.

3.1.2 Blocking

To improve the convergence rate of Gibbs sampling – a well-known impediment to using it – we may employ blocking [21]. Similarly to blocked coordinate descent [22], variables are grouped into acyclic blocks and the joint distribution of all nodes within the block are computed and sampled as one unit. The size of the block and the order in which the blocks should be visited are often determined empirically.

We may easily calculate the joint distribution when a block size of only one time step of the trajectory is used. However, if block sizes incorporate multiple time steps the joint distribution may become intensive to compute due to its size. We could utilize a forward-backward algorithm to obtain the distribution in this case. As multi-agent trajectories greatly increase the size of the sample space we develop and evaluate variants of the blocking and visitation schemes in an attempt to improve convergence rates of Gibbs sampler:

- **Blocked Gibbs** - Samples the state and action of one agent as a block, alternating between agents and proceeding in time step order.
- **Multiagent Blocked Gibbs** - Samples the joint state and action of all agents at a given time step of the joint trajectory as a single block; reduces to the above *Blocked Gibbs* if only one agent is occluded at a time.
- **X Time-steps Blocked Gibbs** - Samples X time steps of a single agent as a block, alternating between agents. Uses a forward-backward algorithm to obtain the joint distribution over the block, which is sampled.
- **X Time-steps Multiagent Blocked Gibbs** - Samples X time steps of all agents jointly as a block. Also uses forward-backward message passing for the joint distribution of the block.

Markov blankets of all these blocking schemes are the states and actions in the time steps surrounding the block - these nodes are treated as perfectly observed in the forward-backward. If a value for a node in the blanket is not available to the learner, it uses an uninformative dummy observation that weights all occluded states and all actions equally.

3.2 Algorithm

Our sampling in the E-step proceeds by initializing the large number of nodes in Z in time step order: first sample from the distribution over an expert's state at time t using $T(s^{t-1}, a^{t-1}, s^t)$ and then over its actions using $Pr(a^t|s^t)$. If blocking is utilized, the joint distribution over the block is computed first as mentioned in the previous subsection.

As a full trajectory X is produced by Gibbs sampling, we may obtain the feature expectations due to each trajectory and update the mean of feature expectations so far at regular intervals; this approximates computing Eq. 2. We may stop this sampling pass when the change in the mean feature expectations has remained consistently below a small threshold for some number of iterations (we use last 20 in our experiments). This mean is then added to a mean-of-means and Gibbs is repeated until the mean-of-means has converged. [1]

For the M-step, we minimize the dual of (3) using the adaptive, unconstrained and exponentiated gradient descent algorithm [17] with variance bounds. The gradient involves a summation over all possible trajectories, which may be a very large set. With the

[1]Convergence of Gibbs sampling to the expected distribution is not guaranteed except in specific cases involving nodes that exhibit high levels of mixing. This is not necessarily the case with transition functions used in robotic applications (a robot may not arrive at any other location given its current). Convergence of mean-of-means is a simple method to incorporate restarts into Gibbs sampling.

goal of speeding up the computation, we avoid this summation by expressing the feature expectations in terms of state-visitation frequencies at each time step:

$$\begin{aligned}\nabla\Theta &= \sum_{X\in\mathbb{X}} Pr(X) \sum_{(s,a)\in X} \phi_k(s,a) - \hat{\phi}^{Z|_Y} \\ &= \sum_{s\in S_I} \mu(s) \sum_{a\in A_I} Pr(a|s)\ \phi_k(s,a) - \hat{\phi}^{Z|_Y} \\ &= \sum_s \sum_t \mu_t(s) \sum_a Pr(a|s)\ \phi_k(s,a) - \hat{\phi}^{Z|_Y}\end{aligned} \quad (5)$$

Here, $\nabla\Theta$ denotes the gradient; $\mu(s)$ is the state-visitation frequency, $\mu_t(s)$ is the state-visitation frequency at time t, which is obtained as, $\mu_t(s) = \sum_{s'}\sum_{a} Pr(s|s',a)\ Pr(a|s')\ \mu_{t-1}(s')$. The overall state-visitation frequency, $\mu(s) = \sum_{t\in T} \mu_t(s)$, and $Pr(a|s)$ is calculated using softmax value iteration [11]. Next, we decompose this gradient along time steps with the goal of using an online stochastic gradient descent approach. We begin by using Eq. 2,

$$\begin{aligned}\nabla\Theta =& \sum_{s\in S_I} \sum_{t\in T} \mu_t(s) \sum_{a\in A_I} Pr(a|s)\ \phi_k(s,a) \\ & - \frac{1}{|\mathcal{Y}|} \sum_{Y\in\mathcal{Y}} \sum_{Z\in\mathbb{Z}} Pr(Z|Y;\boldsymbol{\theta}) \sum_{\langle s,a\rangle\in Y\cup Z} \phi_k(s,a) \\ \approx& \sum_{s\in S_I} \sum_{t\in T} \mu_t(s) \sum_{a\in A_I} Pr(a|s)\ \phi_k(s,a) \\ & - \tilde{Pr}(s,a)^t\ \phi_k(s,a) \\ =& \sum_{t\in T} \sum_{s\in S_I} \sum_{a\in A_I} \phi_k(s,a)\ (\mu_t(s) Pr(a|s) \\ & - \tilde{Pr}(s,a)^t)\end{aligned}$$

The gradient at a particular time step t is,

$$\nabla\Theta_t = \sum_{s\in S_I} \sum_{a\in A_I} \phi_k(s,a)\ (\mu_t(s) Pr(a|s) - \tilde{Pr}(s,a)^t)$$

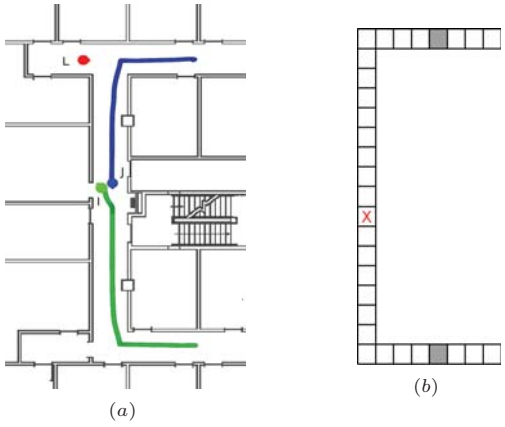
where, $\tilde{Pr}(s,a)^t$ is the empirical distribution of state-action pairs at time t across all trajectories obtained from the E-step. Notice that as t grows, computing μ_t begins to dominate the run time due to its recursive nature. Unfortunately, this additional run time challenges the online use of this algorithm in our experiments, thereby suggesting an additional approximation.

Because the learner receives a long trajectory from each expert in our problem domain thereby providing several state-action pairs, we explore simply replacing the exact computation of the state-visitation frequency by an empirical estimate. In order words, we allow, $\mu_t(s) = \sum_{a\in A_I} \tilde{Pr}(s,a)^t$, in the equation above. The stochastic gradient descent is performed for 10,000 iterations, and we update $Pr(a|s)$ after blocks of 10 iterations.

4. ROBOTIC DOMAIN, METRICS AND MODEL

We utilize the multi-robot patrolling domain for our empirical evaluations. While a previous method [6] let the patrollers' interaction behavior be unknown, here we assume that the learner knows how the overlaid interaction proceeds. This permits a focus on occlusion. Patrollers I and J operate in narrow hallways of a building as shown in Fig. 3. L observes from a corner as the patrollers move in a cyclic trajectory using way points unknown to L. Each robot is a Turtlebot equipped with a video camera and laser scanner (Kinect 360). A time step is two seconds.

Learner L is tasked *online* with (i) passively observing the patrollers to gather data for IRL; (ii) using IRL to recover the patrollers' reward functions and subsequently the policy they are using; (iii) utilizing the found policies to predict the future path of both patrollers; (iv) planning a route that avoids the patrollers' predicted path while reaching the goal state.



(a) (b)

Figure 3: (a) A diagram of the map for our patrolling experiment. (b) Corresponding MDP state space for each patroller. Shaded squares are the turn-around states, which is not known to the learner, and the red X is L's goal state.

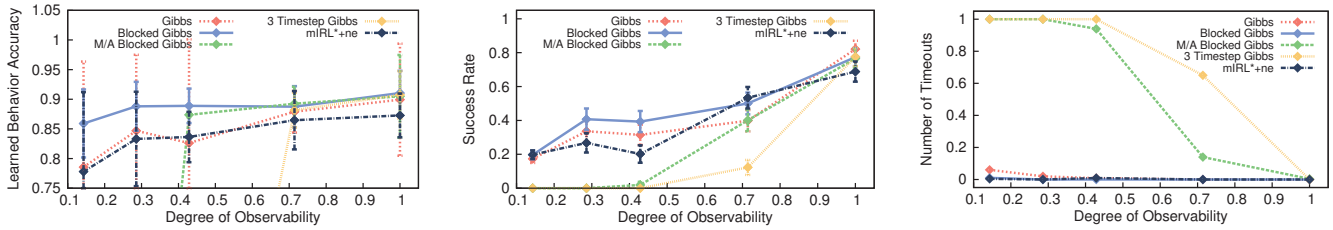
Two measures allow an evaluation of the ability of IRL to learn the behavior of the patrollers. The first is *learned behavior accuracy*, which is the proportion of states in which the optimal policy for the learned reward function gives the observed action. This gives a measure of the prediction accuracy of a given technique. The second measure is the *success rate* of L as a proportion of penetration attempts that result in L reaching its goal without being detected. This measures all aspects of the experiment but suffers from the possibility that L could fail to learn the patrollers' policies accurately but still reach its goal undetected, by chance.

Additionally, we report the number of *time outs* for each method, a measure of L taking excessively long to perform its task. A run times out after *25 minutes* and if L has not reached the goal until then, the run is counted as a failure. Time outs may occur because the inverse learning takes too long or the learned policies produce no useful predictions causing L to believe that there is no safe path to the goal.

4.1 Model

States of each patroller's MDP are cell decompositions of the patrolled area (x, y) and an additional discretized orientation ψ. Each patroller may take one of 4 actions: Move forward, Stop, Turn left, and Turn right. The transition function models the probability of any action succeeding at 92.5% with the remaining probability mass distributed uniformly among the intended states of other actions. As L is expected to move through the same space as the patrollers, its MDP is similar to theirs with the important addition of a discretized time variable. This 4-dimensional MDP is needed otherwise L's reward function would be dynamic as patrollers move constantly [6]. All MDPs are solved for infinite horizon until convergence.

On learning the policy of each patroller, L jointly projects these forward in time starting from the last position each patroller was observed, to arrive at a prediction for the future positions of each patroller. These positions are noted in L's MDP and any states which are visible from a patroller's position at a given time step receive a negative reward. Goal locations at all time steps are given a positive reward and the MDP is solved optimally. L requires a positive value at its starting position at the current time or some future time step; this implies that L expects there to be a path to the goal that avoids detection starting from that time. L waits until that time step has arrived and begins following its policy. In the event that no positive value is found the positions of the patrollers



(a) (b) (c)

Figure 4: (a) Effect of degree of observability on average learned behavior accuracy (error bars are standard deviation) (b) Success rates achieved by each method as observability changes (error bars are 95% confidence intervals) (c) Proportion of runs that timed out, we stop a run after 25 mins. We used standard computers for running the simulations – Intel Core 2 Duo @ 3.00GhZ and 4GB RAM on Ubuntu.

are updated due to any new observations and the process is repeated until one is found or the run times out.

Interactions between the robots occur as they attempt to pass each other without collision, resulting in one robot stopping while the other moves slowly around it. It takes 3 time steps to model the slower movement of the robots during interaction.

The reward functions of I and J are a linear combination of three feature functions:

1. *Has moved*, which returns 1 if the action causes the patroller to leave its current location s, otherwise 0;
2. *Turn around at state s*, which returns 1 if the robot turns at the location given by s, otherwise 0; and
3. *Catch All*, which returns 1 for all other state-action pairs not matched by the above two features.

The true reward function penalizes turning around in the middle of the hallway and rewards turning at the shaded states of Fig. 3(b). As a result, the optimal policies of the patrollers generate trajectories that move through the hallways, turn around at the ends, and move back. This scenario is illustrated in Fig. 3 (a), which shows a trace of the patrollers. The state space of a robot's MDP is shown in Fig. 3 (b) where the shaded squares are the states in which the patrollers turn around. L must move from its position in the top left corner to reach the red X undetected by either I or J.

4.2 Baselines

Several methods described previously are evaluated in this scenario. As the size of $\mathbb{Z}$ is very large due to long trajectories, two experts, and the large number of states per agent the original E-step that seeks to compute $Pr(Z|Y;\boldsymbol{\theta}^{(t)})$ exactly is not employed here. Instead, we test the schemes *Gibbs Sampling*, *Blocked Gibbs*, *Multiagent Blocked Gibbs*, and *3 Time-steps Blocked Gibbs*, which were previously described. Computing the joint of the large block in the *3 Time-steps Multiagent Blocked Gibbs* quickly proved to be infeasible within the time limit and was not utilized further. EM iterations were limited to a maximum of 7 and the Gibbs sampling over the entire network was iterated up to 50 times. Additionally, we compare the methods to a baseline mIRL*+ne, a variant of mIRL* described in Section 2 in which the interaction behavior is known. This approach does not compute the expectation over trajectories and is expected to take least time with respect to $|S_{occ}|$.

5. EXPERIMENTS AND RESULTS

We empirically evaluate the methods mentioned in Section 4.2 in simulation and on physical robots. In particular, we seek a Gibbs sampling scheme that results in most success. Each run lasts for a total of 25 minutes both in simulation and using the physical robots. Out of these, the learner observed the patrols for at most 6 minutes. About 8-10 minutes of the remaining time was spent in performing the IRL and solving the MDP, and then it waited until the right time step to launch a penetration attempt.

5.1 Simulations

We first examine the learned behavior accuracy of all methods in simulation as the proportion of the patrollers' trajectories visible to L is varied. Figure 4 (a) shows that the Blocked Gibbs scheme exhibited the highest accuracy for low degrees of observability until other methods reach its performance as observability improves. Observe the large standard deviation of mIRL*+ne and the non-blocking Gibbs as compared to Blocked Gibbs. When the entire trajectory is observable, all EM-based methods perform similarly as without missing data no E step is needed. Each data point is the result of 230 runs, resulting in very low standard error that ranges from 0.0024 (Gibbs, highest observability) to 0.010 (Gibbs, lowest observability). Consequently, the improved performance of Blocked Gibbs at low observability is statistically significant.

Next, we report the success rates achieved by all methods as the degree of observability is again varied. As we may expect, Fig. 4 (b) shows that Blocked Gibbs again provides the best overall success rate, matching or outperforming mIRL*+ne significantly and all other EM-based methods. As observability decreases these methods must sample from a greatly increased space of possible trajectories thereby increasing the time to find a solution. This leaves less time to find an opportunity to successfully penetrate the patrol and results in an increased number of time outs. As can be seen in Fig. 4 (c), mIRL*+ne experiences the fewest time outs of all methods followed closely by Blocked Gibbs. Notice the high proportion of time outs experienced by 3 Timestep Gibbs and Multiagent Blocked Gibbs; this explains their poor performances as these sampling algorithms take excessive time to sample $\mathbb{Z}$ under large amounts of occlusion.

These experiments strongly indicate that Blocked Gibbs where a block comprises a single agent's state-action nodes at a time step strikes a careful balance between tractability and learning performance. Larger-sized blocks in the multiagent and extended time-step schemes require increased time in computing the joint that leads to time outs in high occlusion scenarios. Indeed, our results indicate that it is better to not group variables based on the performance of Gibbs, which is second best.

5.2 Physical robots

To verify the applicability of the EM-based IRL to tasks in the physical world, we performed a set of experiments with physical

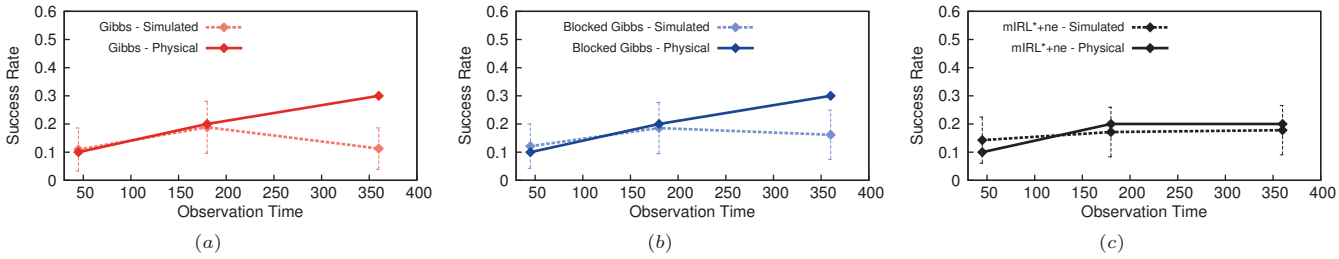


Figure 5: Success rates based on observation times in the physical runs for (a) Gibbs sampling (b) Blocked Gibbs and (c) mIRL*+ne. We compare with those obtained from the simulations for the same low degree of observability. The vertical bars are 95% confidence intervals. Laptops on the TurtleBots have Intel i3 processors with 4GB RAM on Ubuntu.

Turtlebots in conditions that matched the lowest degree of observability in our simulations. As altering the observability is not possible in the physical setup we instead varied the observation time and performed 10 runs per data point. Figure 5 demonstrates that all methods closely matched simulation results for 45 and 180 seconds of observation times. However, interestingly both EM methods outperformed their simulation results at 360 seconds. One hypothesis for this result is, surprisingly, computing power. While the program in physical experiments had dedicated access to the processing power, simulations shared compute time with the ROS environment and two patroller stacks. We think that this slowed down the solver to counteract the improved performance due to more data. Importantly, EM-based methods are overall robust to observations of trajectories that are often more noisy in the real world.

6. RELATED WORK

Ng and Russell [2] formalized IRL as a problem involving a single subject agent learning from a single expert modeled as a MDP. We may view IRL as a special case of inverse optimal control [23], which allows for other frameworks as well, such as inverse linear-quadratic regulators [24].

Ziebart et al. [11] developed maximum entropy IRL as a way of removing bias from the learned reward function. The technique utilized the principle of maximum entropy [25] to obtain a distribution over all possible trajectories, constrained to match feature expectations with those calculated from observed trajectories of the expert agent. A different formulation of maximum entropy IRL maintains a distribution over all candidate policies [12]. Bogert and Doshi [6] extended maximum entropy IRL to settings involving occlusion of portions of the trajectory, by limiting the constraints to the observed portion of the trajectories only. As a result the Lagrangian gradient becomes undefined, slowing the optimization step. In contrast, an EM based approach [7] forms an expectation over the missing data to allow the use of the Lagrangian gradient. However, this method suffers from a computationally intensive expectation step when the occluded portions of the trajectories are long; a limitation addressed by this paper.

Nguyen et al. [26] also combined EM with IRL to find multiple locally-consistent reward functions. The identity of the reward function and its parameters in use at each time step of the expert's demonstration is obtained by employing the EM scheme for clustering to find the maximum likelihood parameterization. In contrast, our method assumes a single reward function but due to missing data it may not be uniquely determined. We use EM to find the maximum likelihood reward weights.

Other approaches to improving the performance of IRL seek to avoid calculating the forward problem by inversely learning the value function [27] and then performing a regression step to recover the reward function [28].

7. CONCLUDING REMARKS

Interest in IRL has increased tremendously in the past few years because of potential applications in learning from demonstrations and in imitation learning. Significant advances are readying IRL for real-world impact. A basic challenge in this regard is that the observed agents may become partially occluded from the learner as they go about performing their tasks; this exacerbates the degeneracy of the optimization. While previous methods seek to address this challenge, this paper shows that the EM-based method which previously demonstrated better learning is intractable for longer trajectories and multiple agents. It introduces Gibbs sampling as a way of speeding up the inference required in the E-step, and empirically explores various blocking schemes. While Gibbs sampling is known for estimating distributions in dynamic Bayesian networks, its embedding in a forward-backward algorithm for IRL under occlusion is novel. The net result is an IRL technique that scales to both long trajectories of as many as 180 time steps and multiple observed agents under occlusion.

Of course, the space of trajectories is smaller in a domain with less stochasticity, but generally, the space grows exponentially with length. Given the favorable performance in scaling from one to two patrollers, we believe that Gibbs sampling can scale reasonably well to more experts. However, a key challenge that we encountered is that with 3 and more patrollers, it becomes very difficult to penetrate patrols and reach the target, despite good learning accuracy; the success rate is low. Consequently, a different problem domain is needed to evaluate this method with more experts.

Further speed up of the E-step seems possible through the careful use of parallelization. Another significant avenue of future work is to explore methods for automatically discovering variable grouping schemes from data [29].

Acknowledgments

This research was funded, in part, by a grant from ONR N-00-0-141310870. We thank Brian Ziebart for useful discussions.

REFERENCES

[1] Stuart Russell. Learning agents for uncertain environments (extended abstract). In *Eleventh Annual Conference on Computational Learning Theory*, pages 101–103, 1998.

[2] Andrew Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *ICML*, pages 663–670, 2000.

[3] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[4] Y. Iwashita, R. Kurazume, T. Hasegawa, and K. Hara. Robust motion capture system against target occlusion using fast level set method. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 168–174, 2006.

[5] Christian Schmaltz, Bodo Rosenhahn, Thomas Brox, Joachim Weickert, Lennart Wietzke, and Gerald Sommer. Dealing with self-occlusion in region based motion capture by means of internal regions. In *5th International Conference on Articulated Motion and Deformable Objects (AMDO)*, pages 102–111, 2008.

[6] Kenneth Bogert and Prashant Doshi. Multi-robot inverse reinforcement learning under occlusion with interactions. In *International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 173–180, 2014.

[7] Kenneth Bogert, Jonathan Feng-Shun Lin, Prashant Doshi, and Dana Kulic. Expectation-Maximization for Inverse Reinforcement Learning with Hidden Data. In *International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1034–1042, 2016.

[8] Shaojun Wang Shaojun Wang, R. Rosenfeld, Yunxin Zhao Yunxin Zhao, and D. Schuurmans. The latent maximum entropy principle. *Proceedings IEEE International Symposium on Information Theory,*, pages 131–167, 2002.

[9] Robert Casella. *Monte Carlo Statistical Methods*. Springer-Verlag, 2004.

[10] Pieter Abbeel and AY Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML*, page 1, 2004.

[11] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, pages 1433–1438, 2008.

[12] Abdeslam Boularias, O Kromer, and J Peters. Structured apprenticeship learning. *Machine Learning and Knowledge Discovery in Databases*, 7524:227–242, 2012.

[13] Kenneth Bogert and Prashant Doshi. Toward estimating others' transition models under occlusion for multi-robot IRL. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1867–1873, 2015.

[14] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

[15] Shaojun Wang, Dale Schuurmans, and Yunxin Zhao. The latent maximum entropy principle. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(2):1–8, 2012.

[16] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistics Society. Series B Methodological*, 39(1):1–38, 1977.

[17] Jacob Steinhardt and Percy Liang. Adaptivity and optimism:an improved exponentiated gradient algorithm. In *International Conference on Machine Learning (ICML)*, pages 1593–1601, 2014.

[18] Craig Boutilier. Planning, learning and coordination in multiagent decision processes. In *6th Conference on Theoretical Aspects of Rationality and Knowledge (TARK)*, pages 195–210, 1996.

[19] Craig Boutilier, Nir Friedman, Moises Goldszmidt, and Daphne Koller. Context-specific independence in bayesian networks. In *Twelfth international conference on Uncertainty in artificial intelligence (UAI)*, pages 115–123, 1996.

[20] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (Third Edition)*. Prentice Hall, 2010.

[21] Claus Jensen, Uffe Kjaerulff, and Augustine Kong. Blocking gibbs sampling in very large probabilistic expert systems. *International Journal of Human Computer Studies*, 42(6), 1995.

[22] P. Tseng. Convergence of block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109:475–494, 2001.

[23] RW Obermayer and Frederick A Muckler. *On the inverse optimal control problem in manual control systems*, volume 208. NASA, 1965.

[24] BD Ziebart, JA Bagnell, and AK Dey. Modeling interaction via the principle of maximum causal entropy. In *ICML*, 2010.

[25] Edwin T. Jaynes. Where do we stand on maximum entropy. In Levin and Tribus, editors, *The Maximum Entropy Formalism*, pages 15–118. MIT Press, 1979.

[26] Quoc Phong Nguyen, Bryan Kian Hsiang Low, and Patrick Jaillet. Inverse reinforcement learning with locally consistent reward functions. In *Advances in Neural Information Processing Systems*, pages 1747–1755, 2015.

[27] Krishnamurthy Dvijotham and Emanuel Todorov. Inverse optimal control with linearly-solvable MDPs. In *ICML*, pages 335–342, 2010.

[28] Edouard Klein and M Geist. Inverse Reinforcement Learning through Structured Classification. *Advances in Neural Information Processing Systems*, pages 1–9, 2012.

[29] Charlie Frogner and Avi Pfeffer. Discovering weakly-interacting factors in a complex stochastic process. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.