Learning Models of Sequential Decision-Making with Partial Specification of Agent Behavior

Vaibhav V. Unhelkar, Julie A. Shah
Massachusetts Institute of Technology
77 Massachusetts Avenue, Cambridge, MA 02139
{unhelkar, julie_a_shah}@csail.mit.edu

Abstract

Artificial agents that interact with other (human or artificial) agents require models in order to reason about those other agents' behavior. In addition to the predictive utility of these models, maintaining a model that is aligned with an agent's true generative model of behavior is critical for effective human-agent interaction. In applications wherein observations and partial specification of the agent's behavior are available, achieving model alignment is challenging for a variety of reasons. For one, the agent's decision factors are often not completely known; further, prior approaches that rely upon observations of agents' behavior alone can fail to recover the true model, since multiple models can explain observed behavior equally well. To achieve better model alignment, we provide a novel approach capable of learning aligned models that conform to partial knowledge of the agent's behavior. Central to our approach are a factored model of behavior (AMM), along with Bayesian nonparametric priors, and an inference approach capable of incorporating partial specifications as constraints for model learning. We evaluate our approach in experiments and demonstrate improvements in metrics of model alignment.

Introduction

Artificial agents interact with other agents, including humans, in a variety of scenarios: for example, autonomous driving, human-robot collaborative manufacturing, intelligent tutoring, and serving as digital assistants. In order for such interactions to be successful, these agents require accurate behavioral models of the other agents' sequential decision-making process. Not only are these models useful for predicting agents' behavior, but maintaining shared mental models (i.e., the alignment of the learned model with the true behavior) is also critical for effective human-agent interaction (Jonker, Van Riemsdijk, and Vermeulen 2011).

Relying solely upon manual specification to develop models is expensive and typically leads to incomplete models, thus motivating the development of algorithmic approaches for learning models that use behavioral data (observations of another agent's behavior) (Albrecht and Stone 2018). While multiple data-driven approaches utilizing different model representations are capable of learning predictive models, they may fail to recover the true model of another agent. This occurs as multiple values of model parameters can explain the observed behavior equally well - e.g., the reward in inverse reinforcement learning (IRL) (Abbeel and Ng 2004).

Further, an agent's behavior is often influenced by factors that are not directly observable to another agent, or are difficult to manually specify a priori. For instance, in human-robot teamwork and assisted driving, the human's behavior depends on both observable features of the task/environment and latent states (e.g., trust, attention, and workload (Thomaz et al. 2016)); hand-coded categories are typically used to quantify such latent states (Fridman et al. 2018). More recently, principled approaches to learning models in the absence of missing decision factors have been developed (for example,(Panella and Gmytrasiewicz 2017)). However, due to their emphasis on prediction of agent's behavior and not model alignment, these approaches may also fail to recover the true model.

While developing these models, domain knowledge pertaining to the true model of behavior is often available or can be acquired by querying human experts (Chernova and Thomaz 2014). This domain knowledge corresponds to the partial specification of that agent's decision factors (states), state dynamics, change points of states, and policy (mapping from states to actions). For applications in human-agent interaction, ensuring that the learned model conforms to these partial specifications is necessary for maintaining model alignment. Moreover, we posit that these auxiliary inputs (partial specification of the agent's behavior) have the potential to alleviate the ambiguity between the true and learned models that exists when learning via behavioral data alone. However, algorithms capable of complying with and utilizing auxiliary inputs – especially given incomplete knowledge of an agent's states – are currently lacking.

In this paper, we introduce an algorithm capable of learning models in the absence of complete state specification by utilizing both behavioral data and auxiliary inputs. Both the model representation and learning algorithm have been designed to facilitate utilization of domain information (when available). Our model incorporates a factored state representation to encode knowledge of known decision factors, their dynamics, and their impact on an agent's policy. Further, we adopt a Bayesian approach and introduce a novel learning approach – termed as constrained variational inference –

that allows us to incorporate auxiliary inputs as constraints during the learning process. We evaluate our inference algorithm and its ability to incorporate auxiliary information in experiments, and demonstrate improvements in alignment between the true and the learned model.

Model of Sequential Decision-Making

In order to pose the model-learning problem, we begin with a definition of the decision-making model and discuss its relation to relevant models in the literature. Several representations that originated in varied modeling and application requirements have been proposed and analyzed in prior research (Albrecht and Stone 2018). Due to our focus on sequential decision-making, we adopt a representation inspired by controlled Markov chains, i.e., Markov chains with control inputs (Kumar and Varaiya 2016). To minimize ambiguity, we refer to the agent who seeks to learn the decision-making model as the observer.

Controlled Markov Chains (CMC) Briefly, a CMC models the impact of an input (denoted by action $a \in A$) on the sequential evolution of a random variable (denoted by state $f \in F$). Due to the Markov property, the distribution of the next state given the entire history depends only on the current state and action – i.e., $T_f \equiv \Pr(f_{t+1}|f_t, a_t) = \Pr(f_{t+1}|f_{0:t}, a_{0:t})$. For a stationary CMC, both the state transition probabilities T_f and the initial state probability $b_f \equiv \Pr(f_0)$ remain constant over time. The model parameters (F, A, b_f, T_f) and the state-action pair (f_t, a_t), completely specify the distribution of the next state f_{t+1}.

Agent Markov Model (AMM) In order to model sequential decision-making behavior, we build upon the CMC. The decision factors of an agent are modeled as the state of a factored CMC, $f \equiv [x, s]$. The decision-maker can observe both x and s. However, only some decision factors are known to the observer (denoted as known states $s \in S$), while others are unspecified (representing latent/mental states $x \in X$). This implies that neither the variable x nor the set X is known to the observer. (Note that $F = X \times S$ and $T_f = T_x \cdot T_s$.) Since the mental states x can impact the known states s only via the agent's actions, the transition probabilities have the following factored structure:

$$T_f(f_{t+1}|f_t, a_t) = T_s(s_{t+1}|s_t, a_t)T_x(x_{t+1}|x_t, s_t, a_t) \quad (1)$$

In order to model an agent's decision-making, we additionally require a mapping from decision factors to actions (i.e., policy). We model the decision maker to follow a stationary Markov policy: $\pi \equiv \Pr(a_t|f_t) = \Pr(a_t|f_{0:t})$. Further, the initial probability distribution of the unknown decision factor is denoted as b_x, and the observable component of the initial state as s_0. We term this generative model of sequential decision-making behavior, parametrized by the tuple $(X, S, A, b_x, T_x, T_s, \pi)$ and depicted in Fig. 1, as agent Markov model (AMM) (Unhelkar and Shah 2018).

As an example of a behavior modeled via the AMM, consider an agent navigating a grid world. To an observer, the agent's position is observable and thus is the known state, s, while its goal is the unspecified latent state, x. Consequently,

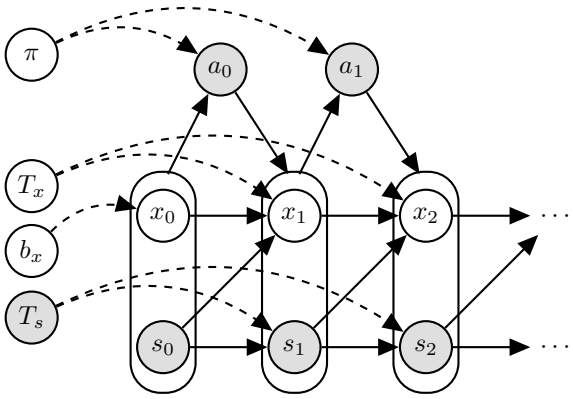

Figure 1: AMM, a model of sequential decision-making behavior. Each node represents a random variable, and the observed variables are shown in grey. Decision factors s and x (included in the oval supernode) impact the decision choices a, based on the policy $\pi(a|s, x)$. The parameters T_x and T_s specify the probability for the next state given the previous state and the action, and b_x models the initial probability.

both the number of goals, $|X|$, and their dynamics (the way in which the next goal is chosen), T_x, are unknown. While both (x, s) impact the agent's choice of action (the direction in which the agent moves), the next position depends only on the current position and the action. Thus, the transition probabilities exhibit the factored structure of Eq. 1.

Related Models The AMM shares properties with existing models for sequential decision-making. However, it also includes specific features for incorporating domain knowledge and facilitating model alignment. Recently, Panella and Gmytrasiewicz used probabilistic deterministic finite state controllers (PDFCs) to model the behavior of another agent. A PDFC models the state transition as deterministic, and thus can be derived as a special case of the AMM with deterministic T_f. Further, in contrast with the PDFC, the AMM uses a factored representation for the state variable, which allows the model to incorporate known dynamics of the state (via T_s), and prior knowledge regarding the impact of the known state on the agent's policy (via priors for π).

Inclusion of a reward function R within the AMM describes an agent executing policy π in a factored Markov decision process (MDP) given by the tuple (F, A, T_f, R) (Puterman 2014). Since agents may not behave rationally (Kahneman 2003), by directly representing π to describe decision-making, the AMM does not assume rationality or even goal-oriented behavior on the part of the other agent.

The AMM and partially observable MDPs (POMDPs) are also related but serve different purposes. POMDPs are used by a decision-maker agent to arrive at her policy (π). In contrast, AMM is designed for an observer agent seeking to infer and explain the decision-maker's policy (π) by observing her behavior. Further, the agent *solving* a POMDP to arrive at π may not observe the full state but has complete knowledge of the state space (S, X). In AMM, however, a subset of state space (X) is both unknown and unobservable.

Problem Definition

Consider an agent whose true behavior model is given by the AMM tuple $(X, S, A, b_x, T_x, T_s, \pi)$. An observer seeks to recover this true model, but has partial specification of the agent's behavior, described as follows:

- partial knowledge of the decision factors, $s \in S$,
- complete knowledge of the agent's action space, $a \in A$,
- dynamics of the known decision factors, T_s,
- I execution traces of the agent's behavior, where the i-th trace refers to the sequences $(s^i_{0:N_i}, a^i_{0:N_i})$,
- noisy change point information regarding the latent state – i.e., the indicator variable $c_t = \mathbb{I}(x_{t+1} = x_t)$ which is accurate with probability p_c, and
- estimates of some elements of the transition function of latent states (i.e., partial knowledge of T_x).

Thus, formally, the problem of learning the decision-making model corresponds to learning the full AMM tuple given the partial tuple $(\cdot, S, A, \cdot, \cdot, T_s, \cdot)$, execution traces, local auxiliary input (i.e., the change point of latent states), and global auxiliary input (i.e., partial knowledge of T_x). In this section, we elaborate upon various inputs for the learning problem.

Known AMM Parameters and Execution Traces In the grid-based example introduced above, we have already discussed the specification of the known states $s \in S$ and actions $a \in A$. The specification of T_s corresponds to the known dynamics of the agent motion and the map of the grid world. The factored representation of AMM allows us to directly incorporate this partial specification (S, A, T_s) of the agent's model. Execution traces, then, correspond to the sequences of the agent's position and action, both of which are readily observable within this domain. The problem of learning the model via execution traces and the partial AMM tuple bears similarity to the classical IRL setting; however, in the current problem, the execution trace includes only the partial state. While approaches exist for IRL in partially observable environments, these methods assume knowledge about the set of latent variables and their dynamics (Bogert et al. 2016; Choi and Kim 2011). However, this knowledge, namely X and T_x, is absent while learning the AMM.

Auxiliary Input While decision factors may not be completely known, partial specifications are often available. This information can be provided by human experts, other algorithms or directly from the decision-maker, e.g., (Chernova and Thomaz 2014; Johnson and Willsky 2013). For human-agent interaction, ensuring that the learned model conforms to this information is critical for maintaining model alignment. In the grid-world example, the number of goals, $|X|$, and their dynamics, T_x, are unknown to the observer; however, domain knowledge in the form of regions where no goal exists may be available. Such information constraints the dynamics of the latent state and partially specifies the transition function T_x. Furthermore, the goal is but one example of the latent state; in other applications, the latent state may correspond to other difficult-to-quantify variables, such as workload and attention. For such variables, constraints regarding their dynamics can be obtained based on cognitive theories and models (Proctor and Van Zandt 2018).

We refer to the partial specification of T_x as the global auxiliary input, since it applies to all execution traces of the agent. In addition, local information may be queried from a domain expert or the decision maker regarding a specific execution trace. For instance, in the grid-world example, for a given execution trace, an expert can provide estimates of when the agent changed its goal. This corresponds to the change point information regarding the latent state c_t, and is referred to as the local auxiliary input. To account for the fact that the expert can make errors, we model the input as accurate with a probability p_c. We emphasize that while both auxiliary inputs are related to the latent state, they focus on the change in the latent state and do not assume knowledge of the specific label (value) of the latent state. In our problem, x corresponds to an unknown variable, thus, obtaining labels of this variable from human experts is not possible.

Bayesian AMM Learning

We adopt a Bayesian approach to recover the AMM parameters given the partial AMM tuple and data (execution traces and auxiliary input). We view the unknown model parameters and state sequences as latent random variables, and seek to infer their posterior using the data. In this paper, we limit our scope to those AMMs in which x is a scalar.

Prior Distributions The AMM tuple provides a generative model for an agent's behavior; however, since the tuple is only partially known, we include priors for the unknown AMM parameters in order to compute their posterior. As the set of unknown factors X is unknown a priori, the number of factors x is also unknown. This motivates the use of nonparametric priors for the number of unknown states $n_x = |X|$, initial distribution b_x and transition function T_x. We use hierarchical Dirichlet process (HDP) priors inspired by the infinite HMM model (Teh et al. 2006). Under this HDP prior, the popularity of the latent states is generated via a Dirichlet process (DP). The base distribution over possible latent states, given by β, can be obtained using a stick-breaking construction with hyper-parameter γ, i.e.,

$$\beta(\cdot) \sim \text{GEM}(\gamma) \tag{2}$$

$$T_x(\cdot|x, s, a) \sim \text{DP}(\alpha \cdot \beta), \quad \forall (s, a) \text{ and } x = 1, 2, \cdots \tag{3}$$

The rows of the transition function T_x are also generated using a Dirichlet process (DP) with base distribution β and scaling parameter α. This allows us to share parameters (in this case, the state space of latent states) between the rows of the transition function. The latent states are indexed as positive integers. A similar process is used for b_x.

In addition to priors for latent states and their dynamics, our model includes a prior for the policy $\pi(a|s, x)$. This prior is modeled as a probability distribution over the decision-maker's action space and is identical for each of the potentially countably infinite latent states x. This policy prior allows us to incorporate prior domain knowledge (if any) regarding the agent's policy, as it can vary based on the

known state s. In absence of additional knowledge, a Dirichlet distribution can serve as the policy prior,

$$\pi(a|s,x) \sim \textsc{Dir}(\rho_{1:|A|}), \quad \forall(s) \text{ and } x = 1, 2, \cdots \tag{4}$$

The graphical model of Fig. 1, along with the priors specified in Eq.2-4, provide a generative model for the AMM. In line with other nonparametric Markov models, we term this model as the infinite AMM (iAMM). The iAMM incorporates a factorization structure common to several statistical models (cf. Eq. 1 Hoffman et al.), which includes global variables (T_x, b_x, π, β), local hidden variables $(x_{0:N})$, observations ($s_{0:N}$ and $a_{0:N}$) and hyper-parameters (α, γ, ρ).

Variational Inference with Execution Traces

In general, obtaining the exact posterior distribution of the iAMM is intractable. Hence, we explore approaches that approximate this distribution. Guided by the recent success of stochastic variational inference (SVI) for sequential models with factorization structures similar to the iAMM (Johnson and Willsky 2014), we provide a mean-field variational inference algorithm in order to approximate the posterior distribution the iAMM. We first derive the algorithm for the case of execution traces as data, then augment it to incorporate auxiliary input.

In this setting, the data consists of (s, a) traces from I sequences: $\mathbf{x}=\{x^i_{0:N_i}\}^I$, $\mathbf{s}=\{s^i_{0:N_i}\}^I$, $\mathbf{a}=\{a^i_{0:N_i}\}^I$. The posterior distribution of the hidden variables of the iAMM is then denoted as $p(\mathbf{x}, T_x, b_x, \pi, \beta|\mathbf{s}, \mathbf{a})$. Mean-field inference approximates this posterior by the product of variational factors $q(\mathbf{x})q(T_x)q(b_x)q(\pi)q(\beta)$ – i.e., by assuming independence between the hidden variables. Each variational factor is a distribution with separate parameters. The posterior is obtained as the $\arg\max$ of the evidence lower bound $\mathcal{L}$;

$$\mathcal{L}(q) \equiv \mathbb{E}_q\left[\frac{p(\mathbf{x}, T_x, b_x, \pi, \beta|\mathbf{s}, \mathbf{a})}{q(\mathbf{x})q(T_x)q(b_x)q(\pi)q(\beta)}\right] \tag{5}$$

Due to the mean-field assumption, this optimization problem can be solved by iteratively optimizing and updating the parameters of the local $q(\mathbf{x})$ and global $q(T_x), q(b_x), q(\pi), q(\beta)$ variational factors. Following Hoffman et al., we use natural gradient ascent for the global variational updates. While the structure of the algorithm is similar to that of the SVI algorithm for the iHMM (Johnson and Willsky 2014), the variational factors and update equations differ. Here, we include the key terms for inference of the iAMM; for an excellent introduction to variational inference, we refer the reader to (Hoffman et al. 2013).

Local Variational Factor In order to efficiently estimate the latent state sequences in the presence of a nonparametric prior, we utilize the direct assignment truncation (Johnson and Willsky 2014). This requires a truncation parameter K as input and models the assumption that K latent states at most are present in the execution traces, i.e., $q(\mathbf{x})=0$ if any $x > K$. The mean-field update for the local variational factors is given as follows:

$$q(\mathbf{x}) \propto \exp(\mathbb{E}_q[\ln p(\mathbf{x}, \text{data}|T_x, b_x, \pi, \beta, T_s)]) = \textstyle\prod_i q(x^i)$$

$$\begin{aligned} q(x^i) &\propto \exp\left(\mathbb{E}_q[\ln p(x^i_{0:N}, s^i_{0:N}, a^i_{0:N}|T_x, b_x, \pi, \beta, T_s)]\right) \\ &\propto \exp\left(\mathbb{E}_q[\ln p(x^i_{0:N}, s^i_{0:N}, a^i_{0:N}|T_x, b_x, \pi, T_s)]\right) \\ &= p(x^i_{0:N}, s^i_{0:N}, a^i_{0:N}|\tilde{T}_x, \tilde{b}_x, \tilde{\pi}, T_s)/Z^i \qquad (6) \\ &\propto \exp\Bigg(\ln \tilde{b}_x(x^i_0) + \sum_0^{N_i-1}\big(\ln \tilde{T}_x(x^i_{t+1}|x^i_t, s^i_t, a^i_t) + \\ &\qquad \ln T_s(s^i_{t+1}|s^i_t, a^i_t) + \ln \tilde{\pi}(a^i_t|x^i_t, s^i_t)\big)\Bigg) \end{aligned}$$

The tilde denotes the operator $\tilde{A}=\exp(\mathbb{E}_{q(A)}[\ln A])$ and is used to provide expectation with respect to the global variational factors. The variable Z^i denotes the normalization constant. To compute the normalization constant and given their utility in updating global factors, we compute the forward F and backward messages B for each sequence.

$$\begin{aligned} F_{(t,j)} &\equiv \Pr(x_t{=}j, s_{0:t}, a_{0:t}) \qquad (7) \\ &= \sum_k \Big(F_{(t-1,k)}\tilde{T}_x(j|k, s_{t-1}, a_{t-1}) \\ &\qquad T_s(s_t|s_{t-1}, a_{t-1})\tilde{\pi}(a_t|j, s_t)\Big) \\ B_{(t,j)} &\equiv \Pr(s_{t+1:N}, a_{t+1:N}|x_t{=}j, s_{0:t}, a_{0:t}) \qquad (8) \\ &= \sum_k \Big(B_{(t+1,k)}\tilde{T}_x(k|j, s_t, a_t) \\ &\qquad T_s(s_{t+1}|s_t, a_t)\tilde{\pi}(a_{t+1}|k, s_{t+1})\Big) \end{aligned}$$

$$F_{(0,j)} = \tilde{b}_x(j)\tilde{\pi}(a_0|j, s_0), \quad B_{(N,j)} = 1, \quad Z = \textstyle\sum_j F_{(N,j)}$$

The message computation takes $\mathcal{O}(NK^2)$ time.

Global Variational Factors The direct assignment truncation allows us to represent the base distribution $\beta = (\beta_{1:K}, \beta_{rest})$, where $\beta_{1:K}$ represents the probability of the first K states and $\beta_{rest} \equiv 1 - \sum_{k=1}^K \beta_k$ represents the probability of truncated states. For the posterior inference, this results in a Dirichlet prior for the rows of T_x and b_x given as $\textsc{Dir}\big(\alpha{\cdot}(\beta_{1:K}, \beta_{rest})\big)$. Due to the conjugacy of the Dirichlet and multinomial distributions, we set the variational factors (approximate posterior) for each row of T_x as follows:

$$q\big(T_x(\cdot|x=j, s=s, a=a)\big) = \textsc{Dir}(\lambda_{jsa}) \tag{9}$$

In order to update the global parameters, expected statistics are necessary with respect to the local factor $q(\mathbf{x})$. For the transition function this corresponds to the expected transition counts which can be efficiently computed using the forward and backward messages as follows:

$$\begin{aligned} \hat{u}^{T_x}_{kjsa} &\equiv \mathbb{E}_{q(\mathbf{x})}\textstyle\sum_t \mathbb{I}[x_{t+1}{=}k, x_t{=}j, s_t{=}s, a_t{=}a] \qquad (10) \\ &= \textstyle\sum_t F_{(t,j)}T_s(s'|s, a)\tilde{T}_x(k|j, s, a)\tilde{\pi}(a'|k, s')B_{(t+1,k)}/Z \end{aligned}$$

where $a'{=}a_{t+1}$ and $s' = s_{t+1}$. Given the expected transition counts and conjugacy, the global variational parameters are updated by maximizing the evidence lower bound,

$$\begin{aligned} \lambda_{jsa} &= \arg\max_\lambda \mathcal{L}\big(\lambda; \mathbb{E}_q[\eta_g]{=}(\alpha \cdot \beta + \hat{u}^{T_x}_{\cdot jsa})\big) \qquad (11) \\ \mathcal{L}(\lambda) &\equiv (\mathbb{E}_q[\eta_g]{-}\lambda) \cdot \nabla_\lambda \ln B(\lambda) + \ln B(\lambda) + \text{const.} \end{aligned}$$

$B(\lambda)$ represents the multivariate Beta function, and $\ln B(\lambda)$ is the log normalizer of the Dirichlet distribution (cf. Eq. 13 Hoffman et al.). This optimization problem is solved by equating the natural gradient (Amari 1998) of the objective function to zero, resulting in the update:

$$\lambda_{jsa} \leftarrow (\alpha \cdot \beta + \hat{u}^{T_x}_{\cdot jsa}) \tag{12}$$

This update in the variational parameter λ_{jsa} corresponds to a natural gradient ascent of step size 1. For large datasets, stochastic natural gradient ascent can be used by considering only a subset of a dataset when computing the expected statistics.

Due to conjugate priors, analysis similar to $q(T_x)$ follows for updating $q(b_x)$ and $q(\pi)$. However, different expected statistics are required, which are computed as follows:

$$\hat{u}^{b_x}_j \equiv \mathbb{E}_{q(\mathbf{x})}\mathbb{I}[x_0{=}j] = B_{0,j}/Z \tag{13}$$

$$\begin{aligned}\hat{u}^{\pi}_{ajs} &\equiv \mathbb{E}_{q(\mathbf{x})}\textstyle\sum_t \mathbb{I}[a_t{=}a, x_t{=}j, s_t{=}s] \\ &= \textstyle\sum_t F_{t,j}B_{t,j}\mathbb{I}[a_t{=}a, s_t{=}s]/Z\end{aligned} \tag{14}$$

To obtain a variational estimate for β, following (Johnson and Willsky 2014), we compute a point estimate β^* by minimizing the evidence lower bound with the constraint that all elements are non-negative.

Variational Inference with Auxiliary Inputs

Priors and hyper-parameters provide one avenue for incorporating domain knowledge in our Bayesian approach. For instance, knowledge about the policy can be incorporated via the policy priors, ρ. However, the auxiliary inputs cannot be incorporated within the priors, motivating the need for modifications to the model or the inference approach.

Local Input Here, we first discuss how local information regarding change points of the latent states can be incorporated by augmenting the model. We consider the change point information, when available, as an additional observation with two levels, 0 and 1, and the likelihood function ψ,

$$\psi(c_t{=}1|x_{t+1}{=}x_t){=}\ p_c \ \text{ and } \ \psi(c_t{=}0|x_{t+1}{\neq}x_t){=}\ p_c \tag{15}$$

p_c represents the known accuracy of the domain expert. This observation depends on both the current and next states, requiring modifications to the local variational factors. The modified message computations and sufficient statistics u are derived as follows (due to space constraints, we list the function arguments only if they are modified):

$$q(x^i) = p(x^i_{0:N}, s^i_{0:N}, a^i_{0:N}, c^i_{0:N}|\tilde{T}_x, \tilde{b}_x, \tilde{\pi}, T_s, \psi) \tag{16}$$

$$\begin{aligned}F_{(t,j)} &\equiv \Pr(x_t{=}j, s_{0:t}, a_{0:t}, c_{0:t-1}) \\ &= \textstyle\sum_k F_{(t-1,k)}\tilde{T}_x(\cdot)T_s(\cdot)\tilde{\pi}(\cdot)\psi(c_{t-1}|k,j)\end{aligned} \tag{17}$$

$$\begin{aligned}B_{(t,j)} &\equiv \Pr(s_{t+1:N}, a_{t+1:N}, c_{t:N-1}|x_t{=}j, s_{0:t}, a_{0:t}) \\ &= \textstyle\sum_k B_{(t+1,k)}\tilde{T}_x(\cdot)T_s(\cdot)\tilde{\pi}(\cdot)\psi(c_t|j,k)\end{aligned} \tag{18}$$

$$\hat{u}^{T_x}_{kjsa} = \textstyle\sum_t F_{(t,j)}T_s(\cdot)\tilde{T}_x(\cdot)\tilde{\pi}(\cdot)B_{(t+1,k)}\psi(c_t|j,k)/Z$$

By utilizing the mean-field approximation, given the modified sufficient statistics, no other changes to the global update equations are necessary.

Global Input Another available auxiliary input is partial knowledge of the transition function, T_x. For analyzing this global input, without loss of generality, we focus on one row of the transition function $\theta \equiv T_x(\cdot|j,s,a)$ with a noisy estimate of its k-th element. The auxiliary input corresponds to θ_k, the estimate for the k-th parameter of the multinomial distribution θ. While the local change point information can be included by augmenting the model, incorporating global auxiliary input as additional observations is not straightforward. If θ_k is modeled as a novel observation of T_x, the benefits of conjugacy are lost, necessitating computationally expensive inference schemes. This occurs since the posterior of T_x given this novel observation, despite the mean-field assumption, is no longer a Dirichlet distribution.

In order to utilize the global auxiliary input and still preserve the benefits of conjugacy, we utilize a novel approach that incorporates this input as constraints for the variational inference. In this method (briefly), the information about θ is converted into a set of constraints for the variational parameters – corresponding to a constraint for λ_{jsa}, the variational parameters of $q(T_x(\cdot|x,s,a))$. To derive this constraint, we use properties (in the current example, the expected value) of the Dirichlet distribution as follows:

$$\lambda_{jsa}(k) = \theta_k \textstyle\sum_i \lambda_{jsa}(i) \tag{19}$$

To compute the posterior, we solve a constrained optimization problem in which the objective is identical to $\mathcal{L}$ and the constraints are derived from the auxiliary input (e.g., Eq. 19). Due to the mean-field assumption, these constraints apply only to a subset of the global update equations – specifically, to the update of global parameter λ_{jsa} given in Eq. 11 – and do not impact the local variational updates. Thus, the computationally faster (unconstrained) variational inference can be used for the remaining parameters for which no auxiliary input is available.

We term this inference approach constrained variational inference (CVI). By incorporating auxiliary information as constraints over the variational parameters, the support of the posterior distribution is limited to those regions of variational parameters λ that satisfy the constraints. Thus, our approach functions, in essence, as a weighted prior. We note that CVI provides an approximate posterior by limiting the posterior to be in the same family as the prior, and also that a computationally expensive approach that sacrifices conjugacy can lead to better estimates. Instead, by utilizing constrained optimization, our approach emphasizes the benefits of conjugacy – allowing us, in this case, to approximate the posterior of T_x with the Dirichlet distribution.

Related Approaches for Learning Models of Other Agent's Sequential Decision-Making

Recent research into algorithmic human-robot interaction provides several examples of employing (variants of) inverse reinforcement learning (IRL) (Ziebart et al. 2008; Ramachandran and Amir 2007) to estimate humans' policies during sequential tasks (Sadigh et al. 2016; Majumdar et al. 2017) – albeit with the complete specification of state features available. In contrast, we consider the learning problem when complete state specification is unavailable.

Teh et al. provided an inference algorithm for the iHMM, a Bayesian nonparametric (BNP) hidden Markov model for scenarios in which the latent states are unknown (2006). Both sampling and variational inference algorithms for several extensions of iHMM have since been developed (Fox et al. 2011; Johnson and Willsky 2013; Saeedi et al. 2016). These approaches have been applied to segmentation and clustering of sequential data; however, they do not model agent policy or the decision-making process. Our approach to modeling the unknown state space, X, is inspired by these models and includes the explicit dependence of actions on states in order to model agent decision-making.

BNP extensions of decision-making models, both for planning (Doshi-Velez et al. 2015; Liu, Liao, and Carin 2011) and IRL (Michini and How 2012; Ranchod, Rosman, and Konidaris 2015; Krishnan et al. 2016), have also been developed. Nonparametric IRL approaches incorporate execution traces as the input data and aim to recover the decision-maker's latent state dependent reward/policy. They result in better performance than parametric IRL approaches when complete state specification is unavailable, but aim to maximize accrued reward and do not seek to recover the true behavioral model. In contrast, our approach explicitly models the dependence of latent state transitions upon action and known states, and includes mechanisms to ensure alignment of the learned model with the auxiliary information. In order to relax the assumption of rationality, in AMM we directly model policy and do not explicitly represent reward. However, our approach for incorporating partial specifications of agent behavior is general and complementary to prior IRL approaches. In domains that include goal-directed or near-rational behavior, future extensions that combine constrained variational inference and IRL hold the potential to improve sample complexity.

In their work, Panella and Gmytrasiewicz provided a Bayesian nonparametric approach to learning the PDFC from execution traces. They further used the learned PDFC to define subintentional interactive POMDPs and generate autonomous agent behavior during multi-agent tasks. Our model representation and the associated generative model (iAMM) generalize the PDFC model representation by considering stochastic transition of latent states. Further, in order to facilitate model alignment, our approach (a) includes a factored state representation to incorporate available information regarding the known states (S, T_s, ρ^s), and (b) provides a mechanism to utilize auxiliary inputs regarding latent state sequences and their dynamics.

Recently, approaches that model latent states in sequential behavior using generative adversarial networks (GANs) and conditional variational autoencoders (CVAEs) have been developed (Li, Song, and Ermon 2017; Schmerling et al. 2017). By modeling latent states, these approaches can predict and generate multimodal behavior. However, they require specification of the number of latent modes and do not model their dynamics. In contrast, by utilizing BNP priors, our approach can jointly learning the number of latent states and their impact upon an agent's behavior. Another interesting direction for future work would be to explore the integration of these techniques with Bayesian nonparametrics.

Experiments

We conducted numerical experiments in order to confirm that the proposed approach can successfully incorporate partial specification of an agent's behavior. Specifically, through the experiments, we measured the ability of the approach to learn models that are aligned with an agent's true model. Access to the true model is necessary for measuring model alignment and validating our approach; hence, in the experiments, we utilized two simulated scenarios for which the true model was accurately known.

For each scenario, we first specified the true model as a complete AMM tuple. Using this tuple, we generated the problem inputs (namely, the partial AMM tuple, execution traces, and local and global auxiliary inputs). We also generated a test dataset (a set of execution traces) to compare the predictive performance of the learned model. The problem inputs were then used to learn the missing elements of the AMM tuple and infer the latent state sequences.

Metrics In order to measure the state estimation performance, we computed the normalized Hamming distance between the inferred and true state sequences using the Munkres algorithm (Saeedi et al. 2016). The Munkres algorithm provides a correspondence between the inferred and true state labels, such that the normalized Hamming distance is minimized. We used this correspondence to measure the predictive performance on the test set and the metrics of model alignment. For measuring model alignment, we utilized a metric inspired by the weighted KL divergence (Panella and Gmytrasiewicz 2017). Specifically, to quantify error in learning the transition probabilities T_x, we first used the Munkres correspondence to match the rows and columns of the learned $\hat{T}_x$ and true T_x, and computed the KL divergence between each matched row. Finally, the metric was obtained as an average score weighted by the relative counts of the input data η described as follows:

$$w\text{KL}(T_x, \hat{T}_x) = \sum_{x,s,a} \eta_{xsa} \text{KL}(T_x(\cdot|x,s,a) || \hat{T}_x(\cdot|x,s,a))$$

Similar procedures were used for measuring error for the parameters π and b_x. We also quantified model alignment by computing weighted L2 norm $(wL2)$ in a similar fashion.

Baselines Due to different model structures and input information it is difficult to compare model alignment performance of our approach with related algorithms. Prior approaches cannot utilize the auxiliary inputs. Further, for methods that do not model unknown states several metrics of interest are undefined. Hence, to validate the ability of our approach to incorporate partial specifications, we compared the performance of four versions of our algorithm – namely, variational inference with execution traces (VI), variational inference with execution traces and local input (VI-L), constrained variational inference with execution traces and global input (CVI-G), and constrained variational inference with execution traces and both auxiliary inputs (CVI-LG). The first version (VI) served as a proxy for approaches that exclusively rely on execution traces, while the last version demonstrated our complete approach.

Domain	Line World					Highway				
	MaxEnt	VI	VI-L	CVI-G	CVI-LG	MaxEnt	VI	VI-L	CVI-G	CVI-LG
Hamming dist. (train)	—	0.51	0.39	0.30	0.13	—	0.56	0.36	0.48	0.35
Hamming dist. (test)	—	0.53	0.40	0.30	0.14	—	0.63	0.44	0.55	0.42
$w\text{KL}(T_x, \hat{T}_x)$	—	2.01	1.38	0.98	0.43	—	2.11	1.12	1.03	0.64
$w\text{KL}(\pi, \hat{\pi})$	1.08	0.77	0.56	0.41	0.19	1.04	0.48	0.38	0.38	0.33
$w\text{KL}(b_x, \hat{b}_x)$	—	0.85	0.87	0.53	0.51	—	1.67	1.75	1.50	1.58

Table 1: State inference and model alignment errors. The results are averaged over twenty-five trials for each domain. For both MaxEnt and VI, the input was agent's execution traces. Auxiliary inputs – either local (L), global (G) or both (LG) – were provided as additional inputs to the other approaches. All algorithms but MaxEnt model unknown states by utilizing AMM as the underlying model. Metrics for state estimation (Hamming dist.) and alignment of T_x and b_x are undefined for MaxEnt.

The remaining versions demonstrated the utility of our approach for different input settings. Identical priors, hyperparameters, initialization and termination conditions were used for all the algorithms.

The four versions of our algorithm all perform inference using the AMM model and thus learn the impact of unknown states. In order to evaluate the effect of modeling unknown states (x), we further compared the performance of our algorithms with a representative approach that does not model unknown states, namely, Maximum Entropy IRL (MaxEnt) (Ziebart et al. 2008). Inference of unknown states is not possible using MaxEnt and the metrics Hamming distance, $w\text{KL}(T_x, \hat{T}_x)$ and $w\text{KL}(b_x, \hat{b}_x)$ are undefined. However, the policy alignment can be quantified using the weighted KL divergence between the true and learned policies.

For each domain, we compared the performance of the algorithms across twenty-five simulation trials. We implemented our algorithms as an extension of `pyhsmm`, a Python library for approximate unsupervised inference. For CVI, a constrained optimizer is required to solve the resulting constrained optimization problem; towards this end, we utilized SLSQP, a sequential least squares programming optimization algorithm as implemented in `scipy` (Kraft 1994; Jones et al. 2001).

Line World

The grid world example described earlier was used as the first evaluation scenario. An agent navigating a one-dimensional grid of length five was considered, with its goal as the latent state x and position as the known state s. The action space of the agent included three actions: left, right and wait. The known transition dynamics T_s were modeled as deterministic. The set of agent goals X included either ends of the grid. The agent exhibited goal-directed motion, and switched its goal after reaching the current goal.

The inputs to the learning algorithm included the partial AMM tuple (S, A, T_s), five execution traces (each of length 20), and the auxiliary inputs. Noisy change point information for two execution traces with accuracy p_c=0.9 served as the local input. The region of the grid where the agent did not switch its goal was randomly selected and specified as the global input. Additionally, five execution traces were generated as the test set. The objective of the learning algorithm was to recover the set of goals X, their switching dynamics T_x and the agent's policy π. Despite the small domain size, the learning problem was challenging since different model parameters (T_x, π) could generate identical behavior. Further, depending on its goal the agent chose different actions for the same location, and execution traces included cycles.

The results of the experiments, averaged over the twenty-five trials, are summarized in Table 1. We observed that our complete approach CVI-LG resulted in models with better alignment (lower wKL scores) as compared to the remaining baselines. Similar trends were observed for the model alignment metrics computed using the weighted L2 norm. Along with improved model alignment, CVI-LG resulted in lower Hamming distance between the true and inferred sequences for both the training and test datasets. Utilization of only one auxiliary input also improved upon the baseline VI that relies only on execution traces. This validates that our approach is capable of utilizing auxiliary information when available. Interestingly, while the auxiliary inputs did not include information regarding the agent's policy, the alignment between the true and learned policy improved by incorporating the auxiliary input. This is possible since our approach performs joint inference of the agent's model parameters (e.g., T_x and π).

All versions of our algorithm resulted in lower policy alignment error as compared to MaxEnt. Despite identical input data, VI (variational inference applied to AMM) demonstrated better model alignment and could perform state inference. This result indicates the utility of explicitly modeling the unknowns as part of the AMM and learning their impact on the agent's behavior.

Highway Domain

As the second domain, we utilized a modified version of the highway domain (Ranchod, Rosman, and Konidaris 2015; Abbeel and Ng 2004). Briefly, for the experiments, this domain modeled a highway with two lanes and two shoulders (one on each side), multiple civilian cars and a police car. The police car could use the entire road, while the other cars were limited to the two lanes. The objective of the learning algorithm was to model the behavior of the police car (henceforth, referred to as the agent). The agent drove at a constant speed, which was faster than that of the other cars

on the highway, and could choose to switch lanes at each step. The agent's actions depend on known, observable factors (distance to cars and current lane) as well as latent states (whether it is driving nominally or pursuing a civilian car). The agent switched from nominal driving to pursuing a civilian car if two civilian cars seemed to be racing (which the agent determined based on the relative distances of the cars), and returned to nominal mode after catching a car. The learning algorithm had access to the known states of the agent, its action space and dynamics of the highway domain T_s; however, the number of latent states x and their dynamics T_x were unknown.

We conducted experiments for the highway domain in a similar fashion to that for the line world scenario; however, longer execution traces (each of length 40) were used for both training and testing datasets. For this domain, the learning algorithm had to reason over a much larger problem with an observable state space of size 200. The results of the experiments, averaged over the twenty-five trials, are summarized in Table 1. Similar to the first scenario, the effect of modeling unknowns was evident, as all versions of our approach (including VI) resulted in lower policy alignment error as compared to MaxEnt. Further, we observed that our approach could successfully incorporate auxiliary inputs and improve model alignment when one or both auxiliary inputs are available. Lastly, we observed reductions in the state decoding and prediction errors (normalized Hamming distance on training and test data, respectively) when the local and global auxiliary inputs were utilized.

Discussions

Through the numerical experiments, we demonstrate that the proposed approach to model learning is capable of learning AMM tuples that are aligned with an agent's true model and that conform to the partial knowledge of an agent's behavior. Furthermore, as more information of varied types (e.g., execution traces, change point sequences, constraints on model parameters) is provided to the inference algorithm, the model alignment improves. This is possible despite incomplete specification of the agent's decision factors.

The ability to incorporate these novel input types enables the use of high-level information of the agent's behavior during the model learning and specification process. This approach is made possible through the use of a factored model representation and a novel inference algorithm. While prior approaches to learning an agent's model largely rely on either execution traces or label/feature queries, we provide an approach that can efficiently utilize different input types (including input about model parameters) and does not assume complete knowledge of the agent's feature set (state space).

In our ongoing work, we are exploring interactive extensions of the proposed approach, wherein the observer can actively choose to query a human decision-maker (or a domain expert) regarding additional and potentially informative specifications. The algorithms presented in this work will enable the observer to utilize the queried specifications efficiently. Another application of our approach includes learning models from behavioral data that conform to preferences provided, as logical or procedural constraints, by a human user; the global inputs provide one avenue to incorporate preferences in our approach.

As mentioned in the introduction, a key motivation behind learning agent models is to enable effective human-machine interaction. The algorithms presented in this work can be used both by a collaborative robot (for learning models of human behavior to improve robot decision-making) and by a human (to learn transparent models of robot behavior to better calibrate trust) (Javdani et al. 2018; Yang et al. 2017). Both of these use cases offer novel research avenues, such as the efficient specification of decision-theoretic models for interaction (such as, POMDPs and decentralized-POMDPs) and evaluation of the utility of aligned models as compared to purely predictive models for human-machine interaction (Oliehoek, Amato, and others 2016).

We identify three areas of further improvement of our approach. Firstly, our approach is limited in that it can learn the latent states and the AMM tuple, but not the semantic labels or interpretation of these latent states. We posit that interaction between a human (domain expert) and the algorithm will be necessary for incorporating interpretability in the latent states learned via Bayesian nonparametrics. Secondly, our approach is limited to a scalar latent state and thus utilizes a flat state representation for the latent state. Lastly, we utilize tabular representations for the unknown AMM parameters (namely, π, T_x). We aim in future work to address these limitations through the use of function approximation to represent model variables and factored latent state representations.

Conclusion

We consider the problem of learning an agent's true decision-making model with partial specification of its behavior. To formalize and address this problem, we utilize a factored representation of sequential decision-making behavior, the agent Markov model (AMM). We pose the learning problem as one of Bayesian inference, and provide novel variational inference algorithms for the AMM that can utilize different types of information – including, sequences of observed behavior, prior knowledge of agent's policy, and partial specification of agent's state and dynamics. Through numerical experiments, we validate that our approach is capable of utilizing varied information types and, consequently, learning models that align with the true behavioral model.

Acknowledgments

We thank Ardavan Saeedi for fruitful discussions on Bayesian inference and learning.

References

Abbeel, P., and Ng, A. Y. 2004. Apprenticeship Learning via Inverse Reinforcement Learning. In *Intl. Conf. on Machine Learning (ICML)*. ACM.

Albrecht, S. V., and Stone, P. 2018. Autonomous Agents Modelling Other Agents: A Comprehensive Survey and Open Problems. *Artificial Intelligence* 258:66 – 95.

Amari, S.-I. 1998. Natural Gradient Works Efficiently in Learning. *Neural Computation* 10(2):251–276.

Bogert, K.; Lin, J. F.-S.; Doshi, P.; and Kulic, D. 2016. Expectation-Maximization for Inverse Reinforcement Learning with Hidden Data. In *Intl. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 1034–1042.

Chernova, S., and Thomaz, A. L. 2014. Robot Learning from Human Teachers. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 8(3):1–121.

Choi, J., and Kim, K.-E. 2011. Inverse Reinforcement Learning in Partially Observable Environments. *Journal of Machine Learning Research* 12(Mar):691–730.

Doshi-Velez, F.; Pfau, D.; Wood, F.; and Roy, N. 2015. Bayesian Nonparametric Methods for Partially-Observable Reinforcement Learning. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 37(2):394–407.

Fox, E. B.; Sudderth, E. B.; Jordan, M. I.; and Willsky, A. S. 2011. A Sticky HDP-HMM with application to Speaker Diarization. *The Annals of Applied Statistics* 1020–1056.

Fridman, L.; Reimer, B.; Mehler, B.; and Freeman, W. T. 2018. Cognitive Load Estimation in the Wild. In *CHI Conf. on Human Factors in Computing Systems*, 652. ACM.

Hoffman, M. D.; Blei, D. M.; Wang, C.; and Paisley, J. 2013. Stochastic Variational Inference. *Journal of Machine Learning Research* 14(1):1303–1347.

Javdani, S.; Admoni, H.; Pellegrinelli, S.; Srinivasa, S. S.; and Bagnell, J. A. 2018. Shared Autonomy via Hindsight Optimization for Teleoperation and Teaming. *The International Journal of Robotics Research*.

Johnson, M. J., and Willsky, A. S. 2013. Bayesian Nonparametric Hidden Semi-Markov Models. *Journal of Machine Learning Research* 14(Feb):673–701.

Johnson, M., and Willsky, A. 2014. Stochastic Variational Inference for Bayesian Time Series Models. In *Intl. Conf. on Machine Learning*, 1854–1862.

Jones, E.; Oliphant, T.; Peterson, P.; et al. 2001. SciPy: Open Source Scientific Tools for Python.

Jonker, C. M.; Van Riemsdijk, M. B.; and Vermeulen, B. 2011. Shared Mental Models: A Conceptual Analysis. In *Coordination, Organizations, Institutions, and Norms in Agent Systems (COIN)*. Springer. 132–151.

Kahneman, D. 2003. Maps of Bounded Rationality. *The American Economic Review* 93(5):1449–1475.

Kraft, D. 1994. Algorithm 733: TOMP–Fortran modules for Optimal Control Calculations. *ACM Trans. on Mathematical Software (TOMS)* 20(3):262–281.

Krishnan, S.; Garg, A.; Goldberg, K.; et al. 2016. SWIRL: A Sequential Windowed Inverse Reinforcement Learning Algorithm for Robot Tasks with Delayed Rewards. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*.

Kumar, P., and Varaiya, P. 2016. *Stochastic Systems*. SIAM.

Li, Y.; Song, J.; and Ermon, S. 2017. InfoGAIL: Interpretable Imitation Learning from Visual Demonstrations. In *Advances in Neural Information Processing Systems (NIPS)*, 3812–3822.

Liu, M.; Liao, X.; and Carin, L. 2011. The Infinite Regionalized Policy Representation. In *Intl. Conf. on Machine Learning (ICML)*. ACM.

Majumdar, A.; Singh, S.; Mandlekar, A.; and Pavone, M. 2017. Risk-Sensitive Inverse Reinforcement Learning via Coherent Risk Models. In *Robotics: Science and Systems*.

Michini, B., and How, J. 2012. Bayesian Nonparametric Inverse Reinforcement Learning. In *Joint European Conf. on Machine Learning and Knowledge Discovery in Databases*. Springer.

Oliehoek, F. A.; Amato, C.; et al. 2016. *A Concise Introduction to Decentralized POMDPs*, volume 1. Springer.

Panella, A., and Gmytrasiewicz, P. 2017. Interactive POMDPs with Finite-State Models of Other Agents. *Autonomous Agents and Multi-Agent Systems* 31(4):861–904.

Proctor, R. W., and Van Zandt, T. 2018. *Human Factors in Simple and Complex Systems*. CRC Press.

Puterman, M. L. 2014. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley & Sons.

Ramachandran, D., and Amir, E. 2007. Bayesian Inverse Reinforcement Learning. In *Intl. Joint Conf. on Artificial Intelligence (IJCAI)*.

Ranchod, P.; Rosman, B.; and Konidaris, G. 2015. Nonparametric Bayesian Reward Segmentation for Skill Discovery using Inverse Reinforcement Learning. In *Intl. Conf. on Intelligent Robots and Systems (IROS)*, 471–477. IEEE.

Sadigh, D.; Sastry, S.; Seshia, S. A.; and Dragan, A. D. 2016. Planning for Autonomous Cars that Leverages Effects on Human Actions. In *Robotics: Science and Systems (R:SS)*.

Saeedi, A.; Hoffman, M.; Johnson, M.; and Adams, R. 2016. The Segmented iHMM: a Simple, Efficient Hierarchical Infinite HMM. In *Intl. Conf. on Machine Learning (ICML)*.

Schmerling, E.; Leung, K.; Vollprecht, W.; and Pavone, M. 2017. Multimodal Probabilistic Model-Based Planning for Human-Robot Interaction. *arXiv preprint*.

Teh, Y. W.; Jordan, M. I.; Beal, M. J.; and Blei, D. M. 2006. Hierarchical Dirichlet Processes. *Journal of the American Statistical Association* 101(476):1566–1581.

Thomaz, A.; Hoffman, G.; Cakmak, M.; et al. 2016. Computational Human-Robot Interaction. *Foundations and Trends in Robotics* 4(2-3):105–223.

Unhelkar, V. V., and Shah, J. A. 2018. Learning Models of Sequential Decision-Making without Complete State Specification using Bayesian Nonparametric Inference and Active Querying. Technical report, Massachusetts Institute of Technology (MIT).

Yang, X. J.; Unhelkar, V. V.; Li, K.; and Shah, J. A. 2017. Evaluating Effects of User Experience and System Transparency on Trust in Automation. In *Intl. Conf. on Human-Robot Interaction (HRI)*, 408–416. ACM.

Ziebart, B. D.; Maas, A. L.; Bagnell, J. A.; and Dey, A. K. 2008. Maximum Entropy Inverse Reinforcement Learning. In *AAAI Conf. on Artificial Intelligence*, volume 8, 1433–1438.