Bayesian Methods for Constraint Inference in Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

Learning constraints from demonstrations provides a natural and efficient way to improve the safety of AI systems; however, prior work only considers learning a single, point-estimate of the constraints. By contrast, we consider the problem of inferring constraints from demonstrations using a Bayesian perspective. We propose *Bayesian Inverse Constraint Reinforcement Learning* (BICRL), a novel approach that infers a posterior probability distribution over constraints from demonstrated trajectories. The main advantages of BICRL, compared to prior constraint inference algorithms, are *(1)* the freedom to infer constraints from partial trajectories and even from disjoint state-action pairs, *(2)* the ability to infer constraints from suboptimal demonstrations and in stochastic environments, and *(3)* the opportunity to use the posterior distribution over constraints in order to implement active learning and robust policy optimization techniques. We show that BICRL outperforms pre-existing constraint learning approaches, leading to more accurate constraint inference and consequently safer policies. We further propose Hierarchical BICRL that infers constraints locally in sub-spaces of the entire domain and then composes global constraint estimates leading to accurate and computationally efficient estimation.

1 Introduction

Reinforcement Learning (RL) algorithms have proven effective in providing policies for a wide range of dynamic decision making tasks (Mnih et al., 2013; Polydoros & Nalpantidis, 2017; Charpentier et al., 2021). However, manually specifying a reward function in an environment to encourage an agent to perform a specific task is a nontrivial process. To alleviate this issue, *Inverse Reinforcement Learning* (IRL) aims at inferring a reward function by observing the behavior of an expert agent performing a specified task (Russell, 1998). While many different IRL approaches have been proposed to infer non-degenerate reward functions from a finite set of expert trajectories (Arora & Doshi, 2021), in many cases it may not be necessary to infer the entire reward function, since partial information regarding the reward function might be available. For example, in safety critical applications such as robotic surgery (Lanfranco et al., 2004) and autonomous driving (Shafaei et al., 2018), the basic goal of a task may be known (e.g. move the robot end effector to a particular position or minimize energy usage), but there may be user specific constraints that are unknown (e.g. proximity to people or other objects). In these cases, we desire algorithms that can infer the unknown constraints by observing demonstrations from an expert in the environment.

Prior work has considered constraint learning from demonstrations in a *maximum likelihood* setting, without considering or utilizing a representation of uncertainty of the constraints. Chou et al. (2018) reason that trajectories that result in lower cumulative costs than the demonstrated ones must be associated with constraints. Based on the same notion Scobee & Sastry (2019) propose a greedy method to add constraints in a MDP so that the expert demonstrated trajectories are more likely under that choice of constraint allocation. Finally, Anwar et al. (2020) extend the aforementioned method to continuous state spaces with unknown transition models. Park et al. (2020) use a Bayesian non-parametric approach to estimate a sequence of subgoals and corresponding constraints from demonstrations; however, they only obtain the MAP solution and assume the demonstrator never violates constraints. By contrast, we infer a full Bayesian posterior distribution over constraints while considering demonstrators that are imperfect and may sometimes accidentally violate

constraints. Maintaining a belief distribution over the location and likelihood of constraints is important for many downstream tasks such as active query synthesis (Settles, 2009), Bayesian robust optimization (Brown et al., 2020a; Javed et al., 2021) and safe exploration (Garcıa & Fernández, 2015).

In this work we formulate the constraint inference problem using a Bayesian perspective and argue that this approach has a number of advantages over approaches that are based on maximum likelihood estimation. For example, as opposed to the maximum likelihood counterpart (Scobee & Sastry, 2019), our method does not require full expert trajectory demonstrations and can work with partial trajectories as well. This fact can also be utilized in active learning settings in which the agent can query the expert for specific state actions or partial demonstrations, without requiring full trajectories. Such a feature could be crucial in applications where demonstrations are not readily available. This flexibility allows our method to be independently implemented in distinct sub-parts of the state space in a Hierarchical manner, learning from local demonstrations and accelerating inference via a divide and conquer approach. Another contribution of our work is that constraint estimates are associated with confidence levels obtained from the posterior distribution. These confidence levels of the constraint configuration can be used to design policies that satisfy certain safety criteria, as agents can utilize this information to keep, for example, certain distance from areas where the existence of constraints is highly uncertain.

2 Related Work

Constraint Inference: Constraint inference has generally been studied (Chou et al., 2018; 2020) with focus on inferring unknown constraints of specific types e.g. geometric (D'Arpino & Shah, 2017; Subramani et al., 2018), sequential (Pardowitz et al., 2005) or convex (Miryoosefi et al., 2019). As an important innovation, Scobee & Sastry (2019) formulated the problem of inferring general constraints in the framework of inverse reinforcement learning and provided a greedy algorithm to infer constraints in deterministic tabular environments. Their proposed approach has since been extended to work with stochastic demonstrations (McPherson et al., 2021) and continuous state spaces (Stocking et al., 2022). Anwar et al. (2020) developed an alternative approach with focus on scaling to high dimensional continuous state space environments. Chou et al. (2022) in parallel have developed an approach to learn chance-constraints using Gaussian processes for motion planning.

Preference Learning and Inverse RL: Constraint inference can also be viewed as a special case of preference learning (Christiano et al., 2017). Preference learning focuses on learning the preference order over outcomes through means of ratings (Daniel et al., 2014), comparisions (Christiano et al., 2017; Sadigh et al., 2017), human interventions (MacGlashan et al., 2017; Hadfield-Menell et al., 2017a) or other forms of human feedback (Jeon et al., 2020). Demonstrations (Ziebart et al., 2008; Finn et al., 2016; Brown et al., 2020c) are a particularly popular form of feedback. In imitation learning (Ross et al., 2011), provided demonstrations are considered optimal and the AI agent is trained to reproduce the demonstrated behavior. However, imitation learning is known to suffer from issues such as lack of robustness to noisy observations (Reddy et al., 2020), distribution shift (Ross et al., 2011) and fragile learning (Zolna et al., 2019). Inverse reinforcement learning (Russell, 1998; Ng et al., 2000; Abbeel & Ng, 2004) avoids some of the issues of imitation learning by learning an explicit reward function from demonstrations first and using regularizers (Finn et al., 2016; Fu et al., 2018), priors (Ramachandran & Amir, 2007b; Michini & How, 2012; Jeon et al., 2018) or robust optimization (Javed et al., 2021) to generalize faithfully. Bayesian IRL (Ramachandran & Amir, 2007b; Brown et al., 2020a;b; Chan & van der Schaar, 2020), in particular, attempts to learn a distribution over possible reward functions.

Reward Function Design: Our work also connects with the wider literature on safe reward function design which focuses on minimizing or avoiding *side effects* (Turner et al., 2020) due to reward misspecification. The approaches in this area either manually design a reward function regularizer that inhibits the tendency of RL agent to act destructively in the environment or learn such a regularizer from information leaked or provided by humans. Turner et al. (2020) use attainable utility preservation as a regularizer which is closely linked to the idea of reachability (Krakovna et al., 2018). Shah et al. (2019) learn the regularizer term by ensuring that the world state in the final state of a demonstration trajectory is maintained. Hadfield-Menell et al. (2017b) consider the given reward function as true reward function only on the given environment and then use Bayesian inference to infer the correct reward function when the environment undergoes any change.

Safe Reinforcement Learning: In safe reinforcement learning (Garcıa & Fernández, 2015), the objective is to learn a policy that maximizes the given reward while ensuring minimal constraint violations. In most of the prior works (Garcıa & Fernández, 2015; Achiam et al., 2017; Tessler et al., 2018; Calian et al., 2020; Srinivasan et al., 2020), it has been assumed that the constraint set is known and the focus has been on improving algorithms to learn improved constraint abiding policies efficiently. Yang et al. (2021) is a notable exception to this trend which assumes that constraints are specified through natural language.

3 Bayesian Constraint Inference

3.1 Preliminaries

Constrained Reinforcement Learning (CRL) (Garcıa & Fernández, 2015) is generally studied in the context of Constrained Markov Decision Processes (CMDP). A CMDP, M_C, is a tuple (S, A, P, R, C, γ), where S denotes the state space and A the action space. We denote the transition probability $P : S \times A \to S$ from a state s following action a with $P(s'|s, a)$. We denote with $R : S \times A \to \mathbb{R}$ the reward function, with C the set of constraints and with $\gamma \in (0, 1)$ the discount factor. If the state space is discrete with $|S| = n$, then the constraint set $C \in \{0, 1\}^n$ can be modeled as an n-ary binary product where $C[j] = 1$ means that the state $j \in \{1, \ldots, n\}$ is a constraint state. Further, we use the indicator function $\mathbb{I}_C$ to denote membership over the constraint set. Denoting the action policy with π, the CRL objective can be written as follows

$$\max_{\pi} \quad \mathbb{E}_{a\sim\pi}\left[\sum_{t=1}^{T} \gamma^t R(s, a)\right] \quad \text{s.t.} \quad \mathbb{E}_{a\sim\pi}\left[\sum_{t=1}^{T} \mathbb{I}_C(s, a)\right] = 0. \tag{1}$$

One way to solve Eq. (1) is by formulating the Lagrangian of the optimization problem

$$\mathcal{L}(\pi, r_p) = \mathbb{E}_{a\sim\pi}\left[\sum_{t=1}^{T} \gamma^t R(s, a)\right] + r_p\left(\mathbb{E}_{a\sim\pi}\left[\sum_{t=1}^{T} \mathbb{I}_C(s, a)\right]\right), \tag{2}$$

where $r_p \in (-\infty, 0]$ denotes the Lagrange multiplier. Intuitively, the Lagrange multiplier r_p can be interpreted as the penalty an agent will incur in terms of reward for violating a constraint. Prior work by Paternain et al. (2019) shows that the problem has zero duality gap and hence the solutions of the aforementioned two problems are equivalent. We leverage this fact to pose the constraint inference problem as the inverse problem of Eq. (2). This novel perspective helps us utilize off-the-shelf efficient RL solvers as opposed to prior works (Scobee & Sastry, 2019; Anwar et al., 2020) which formulate the constraint inference problem as the inverse problem of Eq. (1) and require the use of constrained RL solvers which are generally much less stable and efficient.

3.2 Problem Statement

Bayesian constraint inference is the problem of inferring a distribution over possible constraint sets given $M_C \setminus C$, which denotes the *nominal* MDP with unknown constraints, and a set of demonstrations. Denote the set of expert demonstrations with $\mathcal{D}$, where $\mathcal{D} = \{\xi_1, \xi_2, \ldots\}$ is composed of a number of expert trajectories, with each individual trajectory being denoted with $\xi_i = \{(s_1^i, a_1^i), (s_2^i, a_2^i), \ldots\}$. Furthermore, as opposed to prior works (Scobee & Sastry, 2019; Anwar et al., 2020), we leverage the fact that problems (1) and (2) are equivalent and hence we use the formulation in (2) for constraint inference. This novel perspective has multiple benefits: *(1)* learning constraints does not require making any modifications to the model of the environment (for example there is no need in modifying the action space in particular states as done in Scobee & Sastry (2019)); *(2)* it allows for learning the expert's "risk tolerance" level through learning of r_p; and *(3)* this in turn allows the use of standard, and more stable, RL algorithms to solve for the CMDP in the place of CRL algorithms, which are often difficult to tune (Anwar et al., 2020).

In our formulation an agent can take an action that leads to a constraint state in which case the agent incurs a penalty reward of $r_p < 0$, which is incorporated in the reward function. More specifically, when the agent takes action a and transitions to state $s \in C$ the observed reward is r_p. For a transition in a state $s \notin C$

the accrued nominal reward is $r = R(s, a)$. We will be referring to the nominal MDP that is augmented with a constraint set C and a penalty reward r_p as M_{C,r_p} which is defined by $(S, A, P, R, C, r_p, \gamma)$. With this modified reward function we can use a MDP solver and obtain an optimal policy for any choice of constraint configuration and penalty reward. Inspired by the classic Bayesian IRL (BIRL) framework (Ramachandran & Amir, 2007a) we propose a modification of the Grid Walk algorithm (Vempala, 2005) to jointly perform Markov Chain Monte Carlo (MCMC) sampling over the constraint set C and the penalty reward r_p, as detailed in the next section.

3.3 Bayesian Constraint Inference Algorithm

In what follows we detail our Bayesian constraint inference method called Bayesian Inverse Constraint Reinforcement Learning (BICRL). The basic concept behind our Bayesian method follows the approach proposed in Ramachandran & Amir (2007a) and can be summarized in the following steps: *(1)* we sample a candidate solution, in this case a candidate constraint allocation and a penalty reward, from the neighborhood of the current solution; *(2)* we compare the likelihood functions of the expert demonstrations under this proposal and the current solution and probabilistically accept the candidate solution based on that comparison. We further allow for randomly accepting proposals even if they are not associated with a higher likelihood to enhance exploration in the Markov Chain. In our implementation at each iteration we choose whether to sample a candidate constraint or penalty reward. This choice can either be random or follow a deterministic schedule. When sampling constraints we select a random index j from $\{1, \dots, n\}$ and only change the constraint status of state j: $C'[j] = \neg C[j]$, $C'[i] = C[i], \forall i \neq j$. For the penalty reward we sample r_p from a Gaussian proposal distribution. The main computational burden of the algorithm lies in the value iteration method that is called in each of the K iterations to evaluate the likelihood given the proposal.

We compute the likelihood of a sample by assuming a Boltzmann-type choice model (Ziebart et al., 2008). Under this model the likelihood of a trajectory ξ of length m is given by

$$\mathcal{L}(C, r_p) \coloneqq P(\xi|C, r_p) = \prod_{i=1}^{m} \frac{e^{\beta Q^*(s_i, a_i)}}{Z_i}, \tag{3}$$

where Z_i is the partition function and $\beta \in [0, \infty)$ is the inverse of the temperature parameter. Assuming a prior distribution over constraints and penalties $P(C, r_p)$ the posterior distribution is given by

$$P(C, r_p|\xi) = \frac{P(\xi|C, r_p)P(C, r_p)}{P(\xi)}. \tag{4}$$

We choose an uninformative prior in our experiments, but given some domain knowledge informative priors can also be incorporated. The detailed process of sampling from the posterior distribution using BICRL can be seen in Algorithm 1. The Maximum a Posteriori (MAP) estimates for the constraint allocation and the penalty reward can be obtained as

$$C_{\text{MAP}}, r_{p\text{MAP}} = \arg\max_{C, r_p} P(C, r_p|\xi), \tag{5}$$

and the Expected a Posteriori (EAP) estimates as

$$C_{\text{EAP}}, r_{p\text{EAP}} = \mathbb{E}_{C, r_p \sim P(C, r_p|\xi)}[C, r_p|\xi]. \tag{6}$$

The MAP and EAP estimates will be used to evaluate the performance of BICRL.

3.4 Active Constraint Learning

One of the benefits of using a Bayesian approach to infer constraints is the quantification of the uncertainty, or confidence, in the estimation. Safety critical applications require safety guarantees during the deployment of an agent. In that direction we are interested in mechanisms that can improve the agent's confidence regarding constraint inference. We examine the utility of a simple active learning acquisition function which is based on the variance of the constraint estimates. One of the benefits of BICRL lies in that it does not

Algorithm 1 BICRL

1: **Parameters:** Number of iterations K, standard deviation σ
2: Randomly sample $C \in \{0,1\}^n$
3: Randomly sample $r_p \in \mathbb{R}$
4: $chain_C[0] = C$
5: $chain_{r_p}[0] = r_p$
6: Compute Q^* on M_{C,r_p}
7: **for** $i = 1, \ldots, K$ **do**
8: **if** sample constraints **then**
9: Randomly sample state j from $\{1, \ldots, n\}$
10: Set $C'[j] = \neg C[j]$, $r'_p = r_p$
11: **else**
12: Set $r'_p = r_p + \mathcal{N}(0, \sigma)$, $C' = C$
13: Compute Q^* on M_{C',r'_p}
14: **if** $\log \mathcal{L}(C', r'_p) \geq \log \mathcal{L}(C, r_p)$ **then**
15: Set $C = C'$, $r_p = r'_p$
16: **else**
17: Set $C = C'$, $r_p = r'_p$ w.p. $\mathcal{L}(C', r'_p)/\mathcal{L}(C, r_p)$
18: $chain_C[i] = C$
19: $chain_{r_p}[i] = r_p$
20: **Return** $chain_C$, $chain_{r_p}$

require entire trajectory demonstrations but specific state action demonstrations suffice. We propose an active learning method in which the agent can query the expert K_Q times for K_D specific state demonstrations associated with high uncertainty in the current estimation. Between queries BICRL is run for K_A iterations. The outline of this process is summarized in Algorithm 2. At every iteration of the active learning algorithm

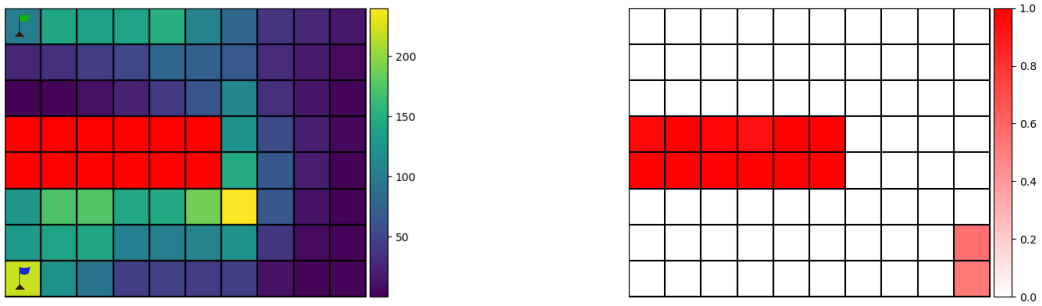
Algorithm 2 Active Constraint Learning

1: **Parameters:** Number of iterations K_Q, K_D, K_A
2: **for** $i = 1, \ldots, K_Q$ **do**
3: Run BICRL for K_A iterations
4: Compute $var(C[i])$, $\forall\, i$
5: Select state $i^* = \text{argmax}_i\ var(C[i])$
6: Query expert for K_D state i^* demonstrations
7: Add demonstrations to $\mathcal{D}$
8: **Return** $chain_C$, $chain_{r_p}$

the BICRL Algorithm is called to provide a new estimate of the constraint allocation. To expedite the process initialization in BICRL can be set using a warm start. More specifically, after the first iteration, each of the subsequent calls to BICRL uses the MAP solution of the previous iteration as the constraint allocation and penalty reward initialization (lines 2 and 3 in Algorithm 1).

4 Deterministic Environments

Most recent work in constraint inference in IRL is based on a Maximum Likelihood framework (Scobee & Sastry, 2019; Stocking et al., 2022). Before introducing the main body of our results, we carry out simulations in deterministic state space grid world environments to compare our method to the Greedy Iterative Constraint Inference (GICI) method proposed by Scobee & Sastry (2019). At each iteration GICI computes the most likely state, among the ones not appearing in the expert demonstrations, to appear in optimal trajectories obtained for the current constraint allocation and classifies it as a constraint state. GICI, like our method, assumes knowledge of the nominal reward function which refers to the rewards of the constraint free states.



(a) (b)

Figure 1: (1a) True constraints (red) along with expert state visitation frequencies. (1b) EAP constraint estimates obtained from BICRL.

One of the restrictions of GICI is that it assumes deterministic transition dynamics and hence in this section we restrict BICRL in deterministic MDPs for a fair comparison.

4.1 Classification Performance and Convergence Rate

We first demonstrate the performance of our method in the discrete grid world environment shown in Figure 1a. The goal state, which is marked with a green flag at the top left, is associated with a known reward of 2 while each other constraint-free state is associated with a known reward of -1. The environment includes 12 constraint states that can be seen in Figure 1a colored in red. Each constraint state is associated with an unknown penalty reward of $r_p = -10$. We synthesize expert trajectories using a Boltzmann policy

$$\pi(a|s) \propto e^{\beta Q^*(s,a)}, \tag{7}$$

with $\beta = 1$. We provide to our learning algorithm 100 trajectories from the demonstrator, each having as a starting state the bottom left corner, designated with a blue flag. Figure 1a shows the state visitation frequencies of those trajectories. We run BICRL for $K = 2000$ iterations and we sample constraints (line 8 in Algorithm 1) 50 times more frequently than constraint penalty rewards r_p. Our Bayesian method correctly identifies the majority of the actual constraints. Figure 1b shows the mean predictions for the constraints with values approaching 1.0 corresponding to high belief of that state being constrained. Mean values close to 0.5 designate states with high uncertainty regarding their constraint status. Given the modeling in BICRL these values are essentially the parameters of Bernoulli random variables that model the likelihood of constraint existence. The algorithm further manages to infer the penalty reward r_p returning an estimated mean value of -9.96. As expected, the agent demonstrates high uncertainty in areas that are far away from the expert demonstrations, like the bottom right section of the grid.

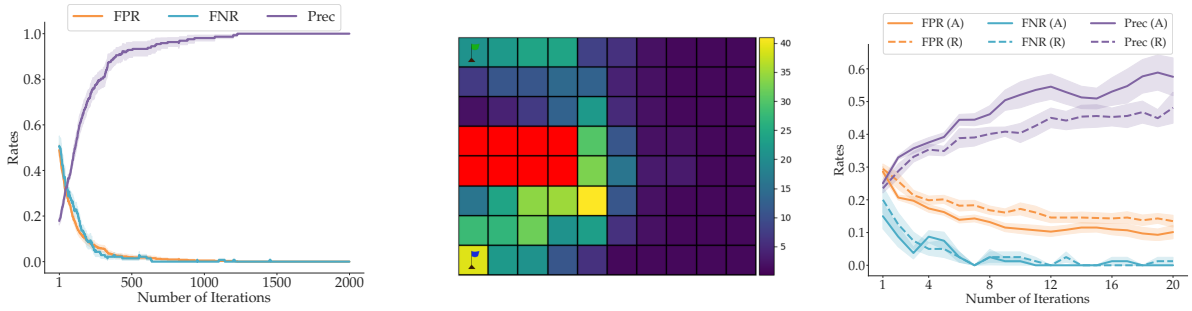
To quantify the convergence rate and the performance of BICRL we further provide a plot in Figure 2a of the False Positive Rate (FPR), the False Negative Rate (FNR) and the Precision (Prec) of the MAP estimates at each BICRL iteration. We average the rates over 10 independent runs of BICRL each using 100 new expert demonstrations. The true constraints are the ones specified in Figure 1a. The rates are not necessarily monotonic as at each iteration of BICRL we allow sub-optimal propositions to be accepted to enhance exploration in the Markov Chain. As the number of iterations increases the MAP estimates tend towards the true constraint allocation.

4.2 Bayesian vs Maximum Likelihood Estimation

To further quantify the benefits of our method we also compare it with GICI. For the same environment as in Figure 1a we compare the classification performance of both methods in Table 1.

GICI			BICRL		
FPR	FNR	Precision	FPR	FNR	Precision
0.0	0.42	**1.0**	0.08	**0.06**	**1.0**

Table 1: False Positive, False Negative and Precision classification rates for GICI and BICRL. Results averaged over 10 runs.



(a) (b) (c)

Figure 2: (2a) Classification rates of MAP constraint estimates from Section 4.1. Results averaged over 10 independent experiments. (2b) True constraints (red) along with expert state visitation frequencies. (2c) Classification rates for Active (A) and Random (R) learning methods. Results averaged over 10 independent experiments.

For these simulations we utilized 100 expert trajectories. The detailed parameters of the simulation can be found in the Appendix. GICI utilizes a KL-divergence based stopping criterion with which it terminates when the distribution of the trajectories under the inferred constraints is within a threshold of the distribution of the expert trajectories. Following the authors we use 0.1 as the KL divergence threshold value. Given the Bayesian nature of BICRL we average the classification results over 10 simulations. BICRL manages to infer significantly more accurately the constraints while infrequently reporting false positives. A low False Negative Rate in the estimation can lead to the acquisition of significantly safer policies.

4.3 Active Learning

Another advantage of BICRL over GICI is having access to a posterior distribution of the constraint allocation which inherently allows for active learning. Querying the expert for specific state-action pair demonstrations can allow for faster and more accurate inference. To obtain intuition about the significance of active learning, we compare the active learning method outlined in Algorithm 2 with a random approach that each time randomly selects a state to query the expert for demonstrations. The MDP environment used is shown in Figure 2b. In each experiment we have a set $\mathcal{D}$ of 20 initial expert demonstrations and we use the active and the random method to query the expert $K_Q = 20$ times for $K_D = 20$ state-action pair demonstrations at a particular state each time with $K_A = 100$. After each iteration $i = 1, \ldots, K_Q$ of the active and random learning methods we compute the FPR, FNR and Precision.

We deliberately restricted to few expert demonstrations and imposed a smaller constraint set in this MDP in order to increase uncertainty especially on the states on the right side of the grid, as very few trajectories now pass through that region. Figure 2c contains the classification rates averaged over 10 independent experiments. The active learning method outperforms the random one especially in the case of false positives and precision. In these simulations we assumed there was a budget of 20 iterations for the active learning algorithm. As the number of active learning iterations increases classification accuracy naturally further improves.

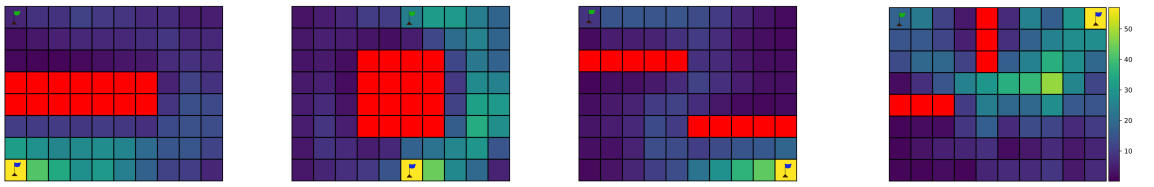


Figure 3: The four grid world environments (from left to right $\text{MDP}_i, i = 1, \ldots, 4$) used to evaluate BICRL. Red squares denote constraint states while the colormap quantifies the state visitation frequencies of the expert trajectories. Start states are denoted with a blue flag and goal states with a green.

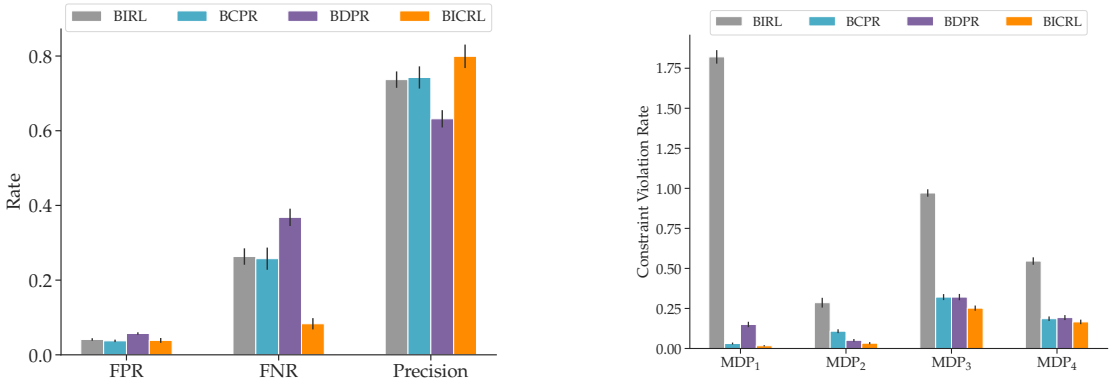
5 BICRL Motivation

In this section we motivate BICRL by showing that the decoupling of constraint allocation and the corresponding penalty rewards is indeed needed for accurate constraint inference. In that direction we compare BICRL to three alternative methods. The simplest approach in inferring constraint states associated with low rewards is via Bayesian Inverse Reinforcement Learning (BIRL) (Ramachandran & Amir, 2007a). In this case however, we do not assume knowledge of a nominal reward function but infer the entire reward vector instead. In the other two methods we assume knowledge of the nominal reward function and on top of that we infer the penalty rewards $\mathbf{r}_p \in \mathbb{R}^n$, that are now state dependent, without including in the MCMC sampling the indicator variable set C to designate constraint states. These approaches can essentially be considered as an implementation of BIRL with knowledge of the nominal rewards. In the first variation, called Bayesian Continuous Penalty Reward (BCPR), we assume that $\mathbf{r}_p$ takes on continuous values while in Bayesian Discrete Penalty Reward (BDPR) we assume they are discrete. The detailed algorithms for these two methods, as well the details of the experiments, can be seen in the Appendix.

To compare the above alternatives with BICRL we use four different MDP environments as seen in Figure 3. Each $\text{MDP}_i, i = 1, 2, 3, 4$ is associated with k_i constraints, $k_1 = 14$, $k_2 = 16$, $k_3 = 10$, $k_4 = 6$, depicted in red. In each case an expert provides noisy demonstrations from a start to a goal state which are designated with blue and green flags, respectively. The dynamics are considered to be stochastic in the MDPs. More specifically, when the agent tries to move in a particular direction there is an ϵ probability that the agent will move to a neighboring state instead. Stochastic dynamics, except for providing a more general framework to the deterministic ones, can be seen as a more accurate depiction of real world robot transitions that are not always perfect due to sensor errors, actuator errors or interference from the surrounding environment. For the remainder of this paper we will assume that the transition dynamics are stochastic.

We first compare the False Positive, False Negative and Precision classification rates that we obtain from the MAP estimates of the four methods as shown in Figure 4a. For BIRL, BDPR and BCPR we classify the k states with the lowest r_ps (from the vector of $\mathbf{r}_p$s) as constraint states, where k is the number of constraints in the original MDP. It should be noted that assuming knowledge of the actual number of constraint states gives a significant advantage to the three comparison methods. Furthermore, in Figure 4b we report the average number of constraint violations per trajectory for each method. These trajectories are acquired from an optimal policy that is obtained for the MDPs with the inferred constraints and penalty rewards and evaluated on the true MDPs. For each MDP we carry out 10 independent constraint inference estimations using the four algorithms and for each of those we obtain 100 optimal trajectories to evaluate the average constraint violation.

Both the classification results as well as the average constraint violation metric showcase that a decomposition of inferring constraint indicator functions and the associated penalty reward leads to more accurate inference and safer policies. Constraint inference via BIRL leads to significantly higher constraint violation. BDPR



(a) (b)

Figure 4: (a) Classification results of BIRL, BCPR, BDPR and BICRL. Results averaged over the four MDPs and 10 independent simulations. (b) Average constraint violation for the four MDPs. Results averaged over 10 independent simulations.

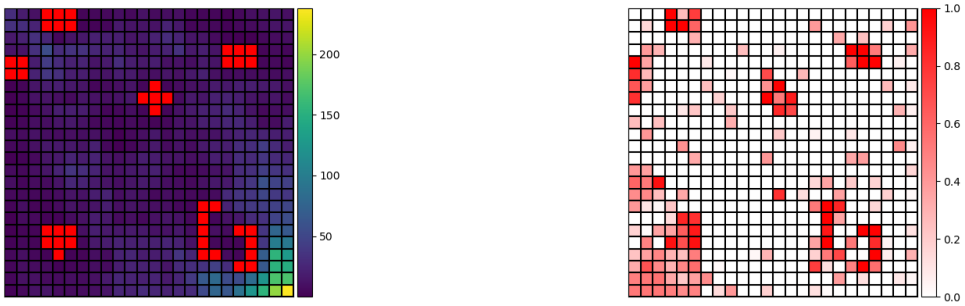
and BCPR also underperform and some times lead to inferred penalty rewards that can alter the reward in a way that changes the actual goal state.

6 Hierarchical BICRL

For a large number of environments and tasks safety constraints are compositional. In this section we take advantage of that fact and we propose learning local constraint allocations by observing agents perform sub-tasks. These local estimates can then be synthesized to obtain an estimate of the constraints in the entire space. Concretely, if the state space S is made of distinct sub-domains $S_1, S_2, \ldots, S_n$, each with constraint sets $C_1, C_2, \ldots, C_n$, then the constraint set associated with the full state space is $C_1 \cup C_2 \cup ... \cup C_n$. By observing experts interact in those distinct domains we can infer constraints that, when composed together, provide more accurate constraint estimation as compared to a global inference approach, in which one agent tries to complete one task on S. Learning constraints from sub-tasks has also the benefit of mitigating the high dimensionality of the original problem. BICRL can be run in parallel for each sub-domain increasing the computational efficiency of our approach.

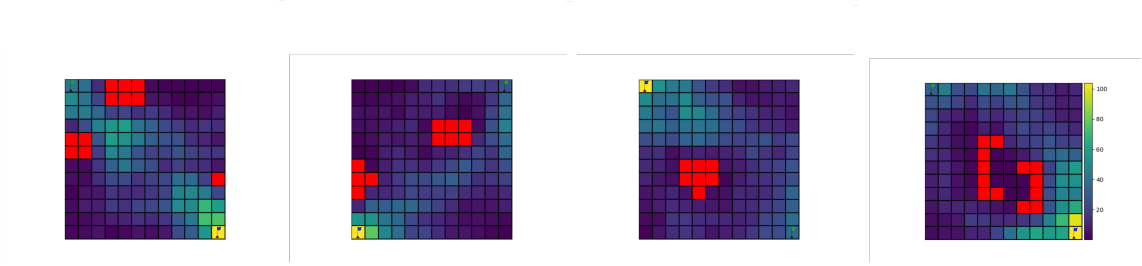
To showcase the benefits of learning from sub-tasks we design an experiment on a 24×24 state grid world environment with constraints as seen in Figure 5a. We compare the case of *Global* inference in which the agent tries to traverse from the bottom right to the top left cell with the *Hierarchical* case in which the domain is split into four non-overlapping 12×12 sub-domains in which agents complete specific sub-tasks. Figure 5a shows the original grid world along with the constraints and state visitation frequencies of the demonstrations while Figure 6 shows the sub-domains extracted from the main environment and the associated sub-tasks. In each of the four sub-domains the experts have to complete a different task as shown by the blue (start state) and green (target state) flags respectively. The posterior mean estimates of the constraints are shown in Figure 7. In comparison, the posterior mean estimates of the *Global* inference case shown in Figure 5b are considerably more uncertain.

To further motivate *Hierarchical* BICRL we compare the classification performance of the MAP estimates of the two methods. Given that trajectories, and hence the total number of state-action demonstration pairs, can be of varying length due to the grid size differences, for fairness we compare the two methods by varying the total number of state-action transition pairs used during inference. Table 2 clearly showcases that *Hierarchical* BICRL achieves significantly higher classification accuracy while providing a more computationally efficient alternative.



(a) (b)

Figure 5: Even given 10000 samples, performing global constraint inference results in much more uncertainty than when using Hierarchical BICRL (Figures 6 and 7). (a) Original constraints along with state occupancies from 10000 samples from expert trajectories. (b) Global EAP constraint estimates.



(a) (b) (c) (d)

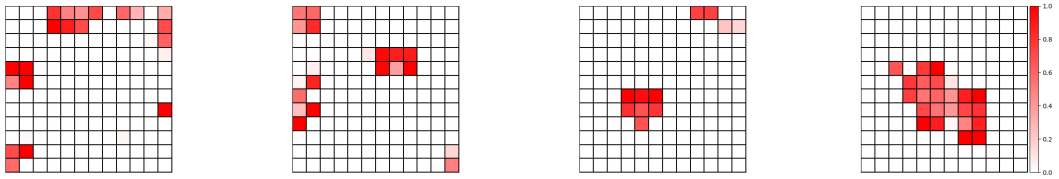
Figure 6: Original sub-task constraints and state occupancies for 2500 samples (for each sub-domain) from expert trajectories. The sub-tasks were obtained by splitting the original 24×24 grid into four equally sized quadrants.

6.1 Home Navigation Task

In this section we present a more realistic constraint inference scenario for Hierarchical BICRL in which an agent infers obstacles in a home environment. An example of such an agent could be a home appliance robot trying to map the rooms of a house. Figure 8 shows the floor plan of a single bedroom apartment obtained from the iGibson dataset (Li et al., 2021). A natural way to divide the entire apartment domain into sub-domains is to consider each room individually. The four room spaces are discretized and for each room we provide a number of expert demonstrations for a particular navigation sub-task. In Figure 8 we further show the mean constraint estimates of the posterior distribution for each sub-domain and ultimately for the entire space. BICRL manages to identify with high certainty most of the constraints in the rooms. This example showcases that given the decomposability of constraints, they can be efficiently inferred from independent sub-tasks. Having access to that information an agent can then design policies and safely complete any task, possibly involving a subset of the sub-domains.

7 Feature-Based BICRL

Finally, we introduce a feature-based version of BICRL. Estimating feature-based constraints allows for better generalization to different environments and tasks. In the feature-based case the walk takes place on the feature weight vector and given that this is usually of smaller dimension than for instance the number of states in Section 5, BICRL may require fewer iterations for the posterior distribution to converge. Feature-based BICRL, detailed in Algorithm 5 in the Appendix, follows the same logic as the original version of the algorithm.



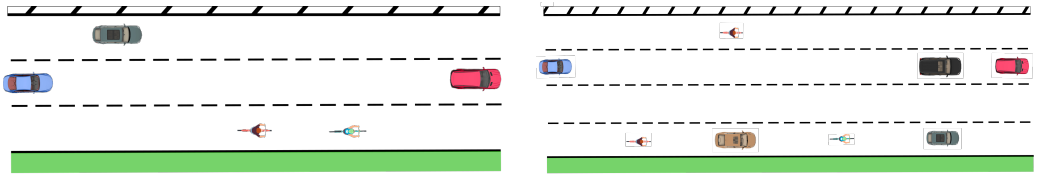
(a) (b) (c) (d)

Figure 7: Hierarchical BICRL learns more accurate and less uncertain constraints than using global inference. Shown are the EAP constraint estimates for the four sub-domains of Figure 6.

	BICRL (*Global*)			BICRL (*Hierarchical*)		
Samples	FPR	FNR	Precision	FPR	FNR	Precision
1000	0.34	0.38	0.12	**0.26**	**0.35**	**0.15**
5000	0.25	**0.13**	0.23	**0.08**	0.2	**0.46**
10000	0.12	**0.08**	0.43	**0.03**	0.11	**0.67**

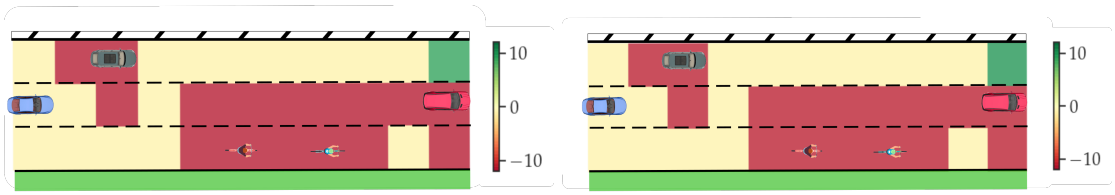
Table 2: False Positive, False Negative and Precision classification rates for *Global* and *Hierarchical* BICRL over a varying number of state-action demonstration sample sizes. Results averaged over 10 experiments.

The main difference is that this time the features and not the states are considered to be constrained or not. At each iteration the algorithm either switches the label of a feature between constraint and free or samples a value for the feature weight associated with constraint features.



(a) Train Environment (b) Test Environment

Figure 9: BICRL learns constraints from demonstrations in (a) that better transfer to new settings (b).



(a) Ground Truth (b) BICRL

Figure 10: BICRL is able to learn the correct constraint function from driving demonstrations.

We study a highway environment in which the reward in each state s is given as a linear function over features $R(s) = \mathbf{w}^T \boldsymbol{\phi}(s)$. The nine binary features $\boldsymbol{\phi} \in \mathbb{R}^9$ and the corresponding weights $\mathbf{w} \in \mathbb{R}^9$ are shown in Table 7 in the Appendix. The dimensions of the original highway environment shown in Figure 9a are 3×11. The goal of the driver of the blue car is to overtake the rest of the vehicles and the cyclists. For tailgating a car, overtaking from the right side of another car, passing close to the cyclists or any collision the driver incurs a penalty of -10. The driver obtains a reward of 10 for completing the task while being penalized for

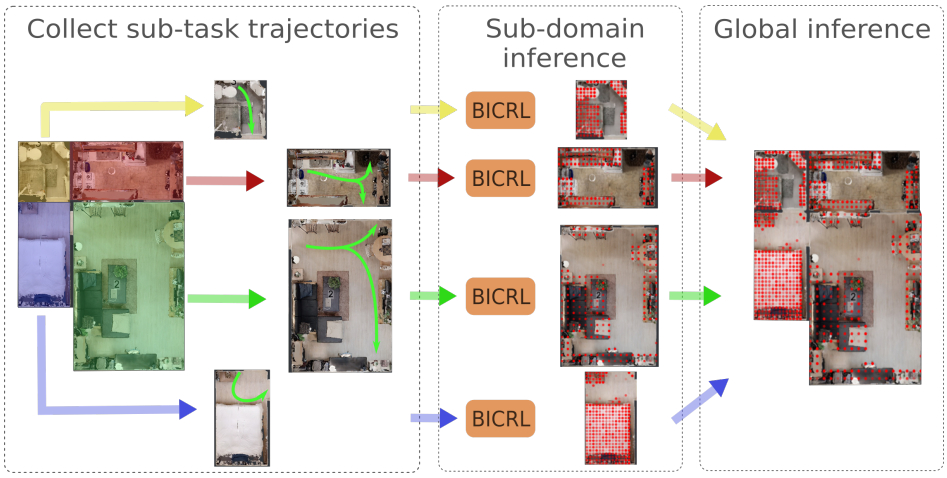


Figure 8: *Hierarchical* BICRL in a home navigation environment with posterior mean results.

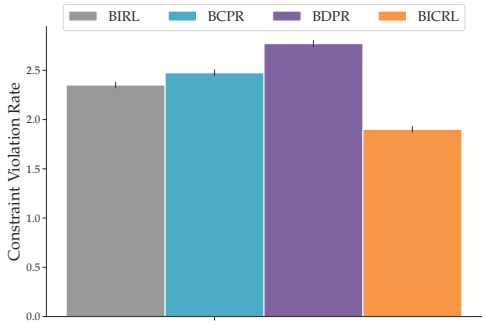


Figure 11: Average constraint violation in the test highway environment of Figure 9b. Results averaged over 10 independent simulations.

driving slowly. For 20 expert demonstrations and $K = 2000$ iterations, Figures 10a and 10b show the original reward and the one recovered from BICRL. The latter results are average over 10 independent simulations.

Finally, we compare how the inferred feature weights from BIRL, BCPR, BDPR and BICRL generalize to a new unseen environment shown in Figure 9b. The new environment models a longer highway stretch with more lanes and vehicles. For each method we run 10 independent inference simulations using each time new expert demonstrations and for each of those we obtain 100 optimal trajectories using the inferred features. Figure 11 shows the average constraint violation in the test environment, providing evidence that BICRL leads to safer policies.

8 Conclusion

In this work we proposed BICRL, a Bayesian approach to infer the unknown constraints in discrete MDPs. BICRL can be utilized in both deterministic and stochastic environments as well as under complete or partial demonstrations. We showed that BICRL outperforms in constraint classification established approaches in both types of environments. The posterior distribution obtained allows for active learning tools to be utilized to further improve classification, especially in infrequently visited by the expert sections of the state space. We further proposed an Hierarchical version of BICRL that allows for independent constraint inference in distinct sub-domains of the entire state space. Finally, we extended BICRL to feature-based environments and we showed that the estimated feature vectors can be used to obtain safe policies in new environments.

References

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1, 2004.

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pp. 22–31. PMLR, 2017.

Usman Anwar, Shehryar Malik, Alireza Aghasi, and Ali Ahmed. Inverse constrained reinforcement learning. *arXiv preprint arXiv:2011.09999*, 2020.

Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, 2021.

D Brown, S Niekum, and M Petrik. Bayesian robust optimization for imitation learning. In *Neural Information Processing Systems*, 2020a.

Daniel Brown, Russell Coleman, Ravi Srinivasan, and Scott Niekum. Safe imitation learning via fast Bayesian reward inference from preferences. In *International Conference on Machine Learning*, pp. 1165–1177. PMLR, 2020b.

Daniel S Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on robot learning*, pp. 330–359. PMLR, 2020c.

Dan A Calian, Daniel J Mankowitz, Tom Zahavy, Zhongwen Xu, Junhyuk Oh, Nir Levine, and Timothy Mann. Balancing constraints and rewards with meta-gradient d4pg. *arXiv preprint arXiv:2010.06324*, 2020.

Alex James Chan and Mihaela van der Schaar. Scalable Bayesian inverse reinforcement learning. In *International Conference on Learning Representations*, 2020.

Arthur Charpentier, Romuald Elie, and Carl Remlinger. Reinforcement learning in economics and finance. *Computational Economics*, pp. 1–38, 2021.

Glen Chou, Dmitry Berenson, and Necmiye Ozay. Learning constraints from demonstrations. *arXiv preprint arXiv:1812.07084*, 2018.

Glen Chou, Necmiye Ozay, and Dmitry Berenson. Learning constraints from locally-optimal demonstrations under cost function uncertainty. *IEEE Robotics Autom. Lett.*, 5(2):3682–3690, 2020.

Glen Chou, Hao Wang, and Dmitry Berenson. Gaussian process constraint learning for scalable chance-constrained motion planning from demonstrations. *IEEE Robotics and Automation Letters*, 7(2):3827–3834, 2022.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, 2017.

Christian Daniel, Malte Viering, Jan Metz, Oliver Kroemer, and Jan Peters. Active reward learning. In *Proceedings of Robotics: Science and Systems*, 2014.

Claudia Pérez D'Arpino and Julie A. Shah. C-learn: Learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, 2016.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018.

Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

Dylan Hadfield-Menell, Anca Dragan, Pieter Abbeel, and Stuart Russell. The off-switch game. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017a.

Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. Inverse reward design. *Advances in neural information processing systems*, 30, 2017b.

Zaynah Javed, Daniel S Brown, Satvik Sharma, Jerry Zhu, Ashwin Balakrishna, Marek Petrik, Anca D Dragan, and Ken Goldberg. Policy gradient bayesian robust optimization for imitation learning. In *International Conference on Machine Learning*, 2021.

Hong Jun Jeon, Smitha Milli, and Anca Dragan. Reward-rational (implicit) choice: A unifying formalism for reward learning. *Advances in Neural Information Processing Systems*, 33:4415–4426, 2020.

Wonseok Jeon, Seokin Seo, and Kee-Eung Kim. A bayesian approach to generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, 2018.

Victoria Krakovna, Laurent Orseau, Ramana Kumar, Miljan Martic, and Shane Legg. Penalizing side effects using stepwise relative reachability. *arXiv preprint arXiv:1806.01186*, 2018.

Anthony R Lanfranco, Andres E Castellanos, Jaydev P Desai, and William C Meyers. Robotic surgery: a current perspective. *Annals of surgery*, 239(1):14, 2004.

Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Elliott Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, Andrey Kurenkov, Karen Liu, Hyowon Gweon, Jiajun Wu, Li Fei-Fei, and Silvio Savarese. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. In *5th Annual Conference on Robot Learning*, 2021. URL `https://openreview.net/forum?id=2uGN5jNJROR`.

James MacGlashan, Mark K. Ho, Robert Loftin, Bei Peng, Guan Wang, David L. Roberts, Matthew E. Taylor, and Michael L. Littman. Interactive learning from policy-dependent human feedback. In *International Conference on Machine Learning*, 2017.

David L McPherson, Kaylene C Stocking, and S Shankar Sastry. Maximum likelihood constraint inference from stochastic demonstrations. In *2021 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 1208–1213. IEEE, 2021.

Bernard Michini and Jonathan P. How. Bayesian nonparametric inverse reinforcement learning. In *Machine Learning and Knowledge Discovery in Databases*, 2012.

Sobhan Miryoosefi, Kianté Brantley, Hal Daume III, Miro Dudik, and Robert E Schapire. Reinforcement learning with convex constraints. In *Advances in Neural Information Processing Systems*, 2019.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pp. 2, 2000.

Michael Pardowitz, Raoul Zöllner, and Rüdiger Dillmann. Learning sequential constraints of tasks from user demonstrations. In *5th IEEE-RAS International Conference on Humanoid Robots*, 2005.

Daehyung Park, Michael Noseworthy, Rohan Paul, Subhro Roy, and Nicholas Roy. Inferring task goals and constraints using Bayesian nonparametric inverse reinforcement learning. In *Conference on Robot Learning*, pp. 1005–1014. PMLR, 2020.

Santiago Paternain, Luiz Chamon, Miguel Calvo-Fullana, and Alejandro Ribeiro. Constrained reinforcement learning has zero duality gap. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/file/c1aeb6517a1c7f33514f7ff69047e74e-Paper.pdf.

Athanasios S Polydoros and Lazaros Nalpantidis. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):153–173, 2017.

Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *IJCAI*, volume 7, pp. 2586–2591, 2007a.

Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *20th International Joint Conference on Artifical Intelligence*, 2007b.

Siddharth Reddy, Anca D. Dragan, and Sergey Levine. SQIL: imitation learning via reinforcement learning with sparse rewards. In *International Conference on Learning Representations*, 2020.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011.

Stuart Russell. Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 101–103, 1998.

Dorsa Sadigh, Anca D. Dragan, Shankar Sastry, and Sanjit A. Seshia. Active preference-based learning of reward functions. In *Robotics: Science and Systems XIII*, 2017.

Dexter RR Scobee and S Shankar Sastry. Maximum likelihood constraint inference for inverse reinforcement learning. *arXiv preprint arXiv:1909.05477*, 2019.

Burr Settles. Active learning literature survey. 2009.

Sina Shafaei, Stefan Kugele, Mohd Hafeez Osman, and Alois Knoll. Uncertainty in machine learning: A safety perspective on autonomous driving. In *International Conference on Computer Safety, Reliability, and Security*, pp. 458–464. Springer, 2018.

Rohin Shah, Dmitrii Krasheninnikov, Jordan Alexander, Pieter Abbeel, and Anca Dragan. Preferences implicit in the state of the world. *arXiv preprint arXiv:1902.04198*, 2019.

Krishnan Srinivasan, Benjamin Eysenbach, Sehoon Ha, Jie Tan, and Chelsea Finn. Learning to be safe: Deep rl with a safety critic. *arXiv preprint arXiv:2010.14603*, 2020.

Kaylene C Stocking, David L McPherson, Robert P Matthew, and Claire J Tomlin. Maximum likelihood constraint inference on continuous state spaces. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

Guru Subramani, Michael Zinn, and Michael Gleicher. Inferring geometric constraints in human demonstrations. In *2nd Conference on Robot Learning (CoRL)*, 2018.

Chen Tessler, Daniel J Mankowitz, and Shie Mannor. Reward constrained policy optimization. *arXiv preprint arXiv:1805.11074*, 2018.

Alex Turner, Neale Ratzlaff, and Prasad Tadepalli. Avoiding side effects in complex environments. *Advances in Neural Information Processing Systems*, 33:21406–21415, 2020.

Santosh Vempala. Geometric random walks: a survey. *Combinatorial and computational geometry*, 52 (573-612):2, 2005.

Tsung-Yen Yang, Michael Y Hu, Yinlam Chow, Peter J Ramadge, and Karthik Narasimhan. Safe reinforcement learning with natural language constraints. *Advances in Neural Information Processing Systems*, 34:13794–13808, 2021.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

Konrad Zolna, Scott Reed, Alexander Novikov, Sergio Gomez Colmenarejo, David Budden, Serkan Cabi, Misha Denil, Nando de Freitas, and Ziyu Wang. Task-relevant adversarial imitation learning. *arXiv preprint arXiv:1910.01077*, 2019.

A Appendix

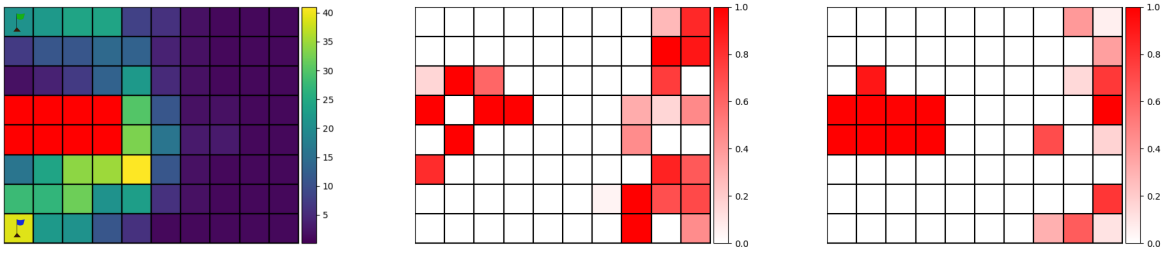
A.1 Details for Simulations in Section 4

This section contains the details for the deterministic transition dynamics environment simulations. The MDP has 8×10 number of states. The possible actions are *up*, *down*, *left* and *right*. The discount factor is $\gamma = 0.95$. Starting and goal states are depicted with a blue and green flag respectively. The expert trajectories are obtained with a Boltzmann policy model with $\beta = 1$. For this example, and all the rest of the simulations in this paper, we gather demonstrated trajectories by letting the expert agent follow a Boltzmann policy (7) from the start state until the goal state is reached or a predetermined number of steps has been made.

Hyperparameters	Sec. 4.1-4.2	Sec. 4.3
# Expert trajectories	100	20
n	80	80
γ	0.95	0.95
β	1	1
K	2000	200
σ	1	1

Table 3: Hyperparameters of Section 4 simulations.

Here, we also report the mean prediction for the constraints learned in Section 4.3. As expected, uncertainty is now significantly higher before running active learning. Applying Algorithm 2 from Section 3.4 with $K_Q = 20$, $K_A = 100$ and $K_D = 20$ we obtain mean estimates that are significantly more accurate as seen in Figure 12c. This feature of BICRL makes it appropriate for tasks where demonstrations are scarce and there is a limitation on the number of queries. In the active learning case we first obtained initial constraint estimates by running BICRL on the initial 20 trajectories for 200 iterations and then used the active learning algorithm to further refine the classification.



(a) (b) (c)

Figure 12: (12a) Original 20 trajectories and true constraint allocation. (12b) EAP constraint estimation without active learning querying. (12c) EAP constraint estimation after 10 active learning queries.

A.2 Details for Simulations in Section 5

This section contains the details on the simulations we run to compare our method with three other Bayesian approaches. The four MDP environments have 8×10 number of states. The possible actions are *up*, *down*, *left* and *right*. There is uncertainty in the transition model as with certain probability the agent moves to a neighboring state instead of the target state. For instance for noise level ϵ when the agent takes the action up then the agent will end up taking the action *up* with probability $1 - 2 * \epsilon$ and the actions *left* and *right* each with probability ϵ. If the agent tries to transition outside the grid boundaries then the agent remains in the same state as shown in Figure 13.

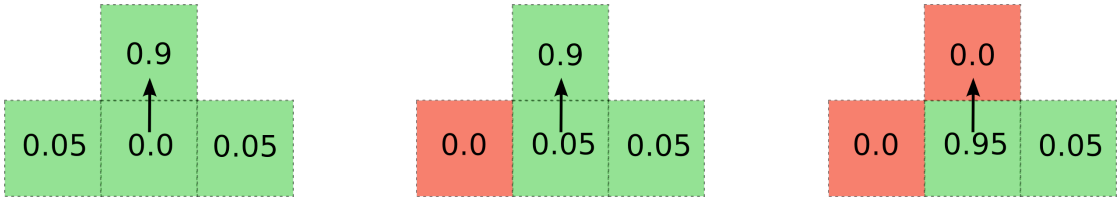


Figure 13: Stochastic dynamics with $\epsilon = 0.05$ for taking action *up*. Red squares represent grid walls. The rest of the actions are analogous.

The discount factor is $\gamma = 0.95$. Starting and goal states are depicted with a blue and green flag respectively. For each of the MDPs we used 20 expert trajectories which we assume are noisy. We model this noise by using a Boltzmann policy (7) for the expert with $\beta = 1$.

Hyperparameters	Values
# Expert trajectories	20
n	80
γ	0.95
ϵ	0.1
β	1
K	4000
σ	1

Table 4: Hyperparameters of Section 5 simulations.

The algorithms for BDPR and BCPR are given below. We denote the reward penalty $\mathbf{r}_p \in \mathbb{R}^n$ with bold letter as now it is a vector since we infer a state-dependent penalty. Both BCPR and BDPR assume knowledge of the nominal reward and sample a state-dependent penalty reward that is added on top of the nominal reward. The MDP notation $M_{\mathbf{r}_p}$ refers to the MDP that has the penalty reward term added to its nominal reward function.

Algorithm 3 BDPR

Parameters: Number of iterations K
Randomly sample reward vector $\mathbf{r}_p \in \mathbb{Z}^n$
$chain_{\mathbf{r}_p}[0] = \mathbf{r}_p$
Compute Q^* on $M_{\mathbf{r}_p}$
for $i = 1, \ldots, K$ **do**
    Randomly sample state j from $\{1, \ldots, K\}$
    Set $\mathbf{r}'_p[j] = \mathbf{r}_p[j] + 1$ or $\mathbf{r}_p[j] - 1$ with equal probability
    Compute Q^* on $M_{\mathbf{r}'_p}$
    if $\log \mathcal{L}(\mathbf{r}'_p) \geq \log \mathcal{L}(\mathbf{r}_p)$ **then**
        Set $\mathbf{r}_p = \mathbf{r}'_p$
    else
        Set $\mathbf{r}_p = \mathbf{r}'_p$ w.p. $\mathcal{L}(\mathbf{r}'_p)/\mathcal{L}(\mathbf{r}_p)$
    $chain_{\mathbf{r}_p}[i] = \mathbf{r}_p$
Return $chain_{\mathbf{r}_p}$

Algorithm 4 BCPR

Parameters: Number of iterations K, standard deviation σ
Randomly sample penalty reward vector $\mathbf{r}_p \in \mathbb{R}^n$
$chain_{\mathbf{r}_p}[0] = \mathbf{r}_p$
Compute Q^* on $M_{\mathbf{r}_p}$
for $i = 1, \ldots, K$ **do**
    Randomly sample state j from $\{1, \ldots, n\}$
    Set $\mathbf{r}'_p[j] \sim \mathcal{N}(\mathbf{r}_p[j], \sigma)$
    Compute Q^* on $M_{\mathbf{r}'_p}$
    if $\log \mathcal{L}(\mathbf{r}'_p) \geq \log \mathcal{L}(\mathbf{r}_p)$ **then**
        Set $\mathbf{r}_p = \mathbf{r}'_p$
    else
        Set $\mathbf{r}_p = \mathbf{r}'_p$ w.p. $\mathcal{L}(\mathbf{r}'_p)/\mathcal{L}(\mathbf{r}_p)$
    $chain_{\mathbf{r}_p}[i] = \mathbf{r}_p$
Return $chain_{\mathbf{r}_p}$

B Hierarchical BICRL

For the Hierarchical BICRL in Section 6 we compare the methods using the varying number of state-action tuples obtained from the expert and not the number of trajectories. The reason for that is that the heterogeneity in the constraints and grid sizes can lead to imbalanced numbers of total state-action demonstrations for the same number of expert trajectories. In practice we keep gathering trajectories until the certain limit of state-action tuples is reached.

Hyperparameters	Values
# Expert samples	$1000, 5000, 10000$
n (Gloval, Hierachical)	$576, 244$
γ	0.95
ϵ	0.1
β	1
σ	1
K	4000

Table 5: Hyperparameters of grid world Hierarchical BICRL.

B.1 Home Navigation with Hierarchical BICRL

For the home navigation task we use one of the apartments in the iGibson dataset Li et al. (2021). Each room is discretized, counting from the top left clockwise, into 16×15, 12×24, 23×17 and 29×15 states respectively. In this set of experiments the reward associated with the goal state is now 10 while the penalty reward and the living reward are -10 and -1 respectively.

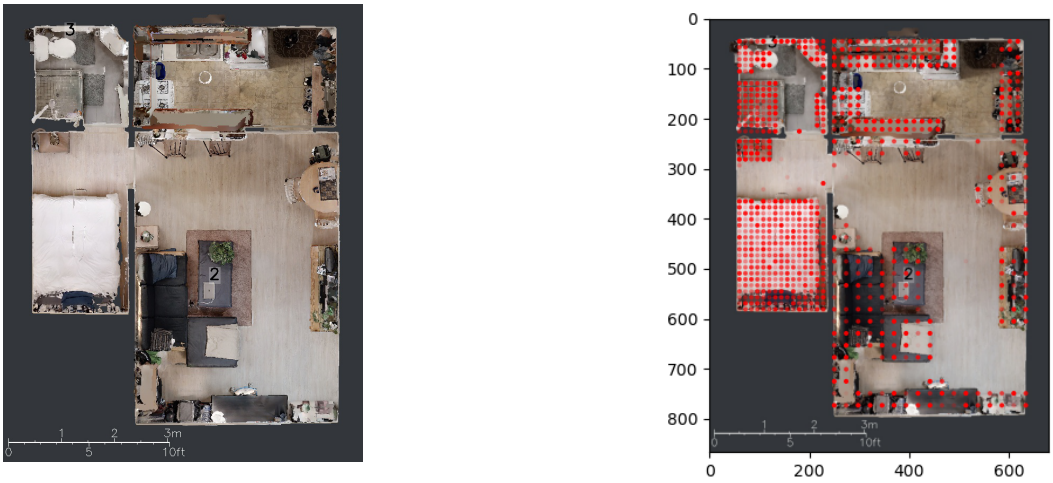


Figure 14: Original four room layout and mean constraint estimates on the four rooms.

Figure 14 contains the original apartment layout and the posterior mean constraint estimates. The brightness of each grid point in the Figure 14 on the right is proportional to the likelihood of that state being constrained. Colorless grid points designate states that are estimated to be free of obstacles. The parameters used for each room can be seen in Table 6.

Hyperparameters	Values
# Expert trajectories	100
n (for each room)	$240, 288, 391, 435$
γ	0.95
ϵ	0.1
β	1
σ	1
K	4000

Table 6: Hyperparameters of home navigation simulations.

C Feature-Based BICRL

Figure 15 shows the highway environment used for the feature-based simulations. The original environment is discretized in 3×11 states. The feature weights are shown in Table 7. The blue car pays a "living" reward penalty of -1, which penalizes driving slowly, while obtaining a small positive reward for overtaking a car from the left lane (car on the right). Overtaking a car from the right lane (car on the left), along with proximity to a cyclist, tailgating and collisions are considered constraints and are heavily penalized with a penalty reward of -10. The feature-based version of BICRL can be seen in Algorithm 5. BICRL, BCPR and BDPR assume knowledge of the "nominal" feature weights (features $4, 5, 6, 7$ and 9), and they sample feature weights on top of these. In lines 5 an 12 in Algorithm 5 the Q values are computed on the MDP that has its nominal feature weights along with the inferred constraint ones that are associated with a penalty reward (weight) of r_p. For instance, if the current weight sample is $\mathbf{w}' = [0, 1, 0, 0, 0, 0, 0, 0, 0]$ and $r'_p = -2.5$ then the weight vector of $M_{\mathbf{w}', r'_p}$ is $\mathbf{w}' = [0, -2.5, 0, 1, -1, -1, -1, 0, 10]$. In this particular MDP, the goal state is reaching the top right cell by overtaking all the vehicles and cyclists according to the rules.

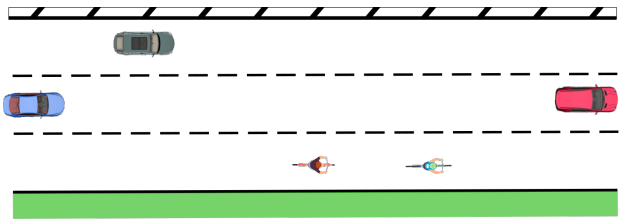


Figure 15: Highway environment.

The hyperparameters of the simulations can be seen in Table 8.

Feature ID	Feature	Weight
1	Tailgating	-10.0
2	Close to cyclist	-10.0
3	Car on the left	-10.0
4	Car on the right	1.0
5	Driving on left lane	-1.0
6	Driving on middle lane	-1.0
7	Driving on right lane	-1.0
8	Collision	-10.0
9	Goal State	10.0

Table 7: Features and weights of highway environment.

Hyperparameters	Values
# Expert trajectories	20
n	33
n_ϕ	9
γ	0.95
ϵ	0.1
β	1
σ	0.2
K	2000

Table 8: Hyperparameters of highway environment.

Algorithm 5 Feature-Based BICRL

Parameters: Number of iterations K, standard deviation σ
Randomly sample $\mathbf{w} \in \mathbb{R}^{n_\phi}$
$chain_{\mathbf{w}}[0] = \mathbf{w}$
$chain_{r_p}[0] = r_p$
Compute Q^* on $M_{\mathbf{w},r_p}$
for $i = 1, \ldots, K$ **do**
    if sample weights **then**
        Randomly sample feature j from $\{1, \ldots, n_\phi\}$
        Set $\mathbf{w}'[j] = \neg\mathbf{w}[j]$, $r'_p = r_p$
    else
        Set $r'_p = r_p + \mathcal{N}(0, \sigma)$, $\mathbf{w}' = \mathbf{w}$
    Compute Q^* on $M_{\mathbf{w}',r'_p}$
    if $\log \mathcal{L}(\mathbf{w}', r'_p) \geq \log \mathcal{L}(\mathbf{w}, r_p)$ **then**
        Set $\mathbf{w} = \mathbf{w}'$, $r_p = r'_p$
    else
        Set $\mathbf{w} = \mathbf{w}'$, $r_p = r'_p$ w.p. $\mathcal{L}(\mathbf{w}')/\mathcal{L}(\mathbf{w})$
    $chain_{\mathbf{w}}[i] = \mathbf{w}$
    $chain_{r_p}[i] = r_p$
Return $chain_{\mathbf{w}}$, $chain_{r_p}[i] = r_p$