Bayesian Inverse Constrained Reinforcement Learning

Dimitris Papadimitriou
UC Berkeley
dimitri@berkeley.edu

Usman Anwar
Information Technology
University, Lahore
usmananwar391@gmail.com

Daniel S. Brown
UC Berkeley
dsbrown@berkeley.edu

Abstract

We consider the problem of inferring constraints from demonstrations from a Bayesian perspective. We propose *Bayesian Inverse Constraint Reinforcement Learning* (BICRL), a novel approach that infers a 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 learn constraints from suboptimal demonstrations and to learn constraints in stochastic environments, and *(3)* the opportunity to estimate a posterior distribution over constraints that enables active learning and robust policy optimization.

1 Introduction

Manually specifying the 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 actions of an expert agent performing a specified task [14]. A number of different IRL approaches infer non-degenerate reward functions from a finite set of expert trajectories [1, 10, 13, 19]. However, in many cases it may not be necessary to infer the entire reward function. For example, in safety critical applications such as robotic surgery [9] and autonomous driving [17], 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 unknown constraints by observing demonstrations in an environment with a known nominal reward function.

Most prior work has considered constraint learning from demonstrations in a maximum likelihood setting, without considering or utilizing a representation of uncertainty with respect to constraint location. In [6] the authors reason that trajectories that result in higher reward than the demonstrated ones must be associated with constraints. Based on the same notion [15] propose a greedy method to add constraints in a MDP so that the resulting trajectories are more likely under that choice of constraint allocation. Finally, [2] extend the aforementioned method to continuous state-action spaces with unknown transition models. By contrast, we infer a fulll Bayesian posterior distribution over constraints. Maintaining a belief distribution over the location and likelihood of constraints is important for many downstream tasks such as active query synthesis [16], Bayesian robust optimization [3, 8], and safe exploration [7]. Park et al. [11] use Bayesian non-parametrics to estimate a sequence of

Workshop on Safe and Robust Control of Uncertain Systems at the 35th Conference on Neural Information Processing Systems (NeurIPS 2021), Online.

subgoals and corresponding constraints from demonstrations; however, they only obtain the MAP solution and assume the demonstrator never violates constraints. By contrast, we seek to leverage the posterior distribution for active learning and consider demonstrators that are imperfect and may sometimes accidentally violate constraints.

In this work we argue that a Bayesian perspective on the constraint inference problem has a number of advantages over approaches that are solely based on maximum likelihood estimation. Bayesian methods usually do not require full expert trajectory demonstrations, as opposed to the maximum likelihood counterparts [15], which can reduce the data complexity of the problem. This fact can also be utilized in active learning settings in which the agent can query the expert for specific expert actions or partial demonstrations, without requiring full trajectories. Furthermore, the posterior distribution of the constraint configuration can be used to design policies that satisfy certain safety criteria, as agents can now utilize this information to keep for instance a distance from areas where the existence of constraints is highly uncertain. Finally, our Bayesian method is not restricted to work only with optimal expert demonstrations as it allows for suboptimal demonstrations and occasional constraint violations by the expert.

2 Bayesian Constraint Inference

2.1 Preliminaries

Constrained Reinforcement Learning (CRL) 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 discrete state space of size n and A the action space. We will denote the transition probability $P : S \times A \rightarrow S$ from a state s following action a with $P(s'|s, a)$. We denote with $R : S \times A \rightarrow \mathbb{R}$ the reward function, with C the set of constraints and with $\gamma \in (0, 1)$ the discount factor. The constraint set $\mathbf{C} \in \{0, 1\}^n$ is modeled as an n-ary binary product where $\mathbf{C}[i] = 1$ means that the state $i, i \in \{1, \ldots, n\}$ is a constraint state, with n being the number of states. Further, we use the indicator function $\mathbb{I}_c$ to denote membership over the constraint set.

The CRL objective can be written as follows, where we assume that the environment makes transitions between states according to P.

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

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, r_p denotes the penalty an agent has to incur in terms of reward in order to violate a constraint. Prior work [12] shows that the problem has zero duality gap and hence the solutions of the aforementioned two problems are equivalent.

2.2 Problem Statement

Bayesian Constraint Inference (BCI) denotes the problem of inferring a distribution over possible constraint sets given $M_c \setminus C$ and a set of demonstrations. We will denote the expert demonstrations with $\mathcal{D}$, where $\mathcal{D} = \{(s_1, a_1), \ldots\}$ and each individual trajectory with ξ. Further, as opposed to prior works [2, 15], we leverage the fact that problems (1) and (2) are equivalent and hence we solve for the inverse problem using (2). This novel perspective has multiple benefits: *(1)* it does not require making any modifications to the model of the environment (as done in [15]), *(2)* it allows for learning the expert's risk tolerance level through learning of r_p and *(3)* this in turn allows 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 [2].

In our formulation an agent can take an action that leads to a constraint state in which case the agent incurs a reward penalty of $r_p < 0$, which is incorporated in the reward function. With this modified reward function we can use a MDP solver and obtain an optimal policy for any choice of constraint configuration and reward penalty. Borrowing from the classic Bayesian IRL (BIRL) framework [13] we propose a modification of the Grid Walk algorithm [18] to jointly perform MCMC over the constraint set $\mathbf{C}$ and the constraint penalty r_p, as detailed in the next section.

2.3 Bayesian Constraint Inference Algorithm

The basic concept behind our Bayesian approach follows the approach proposed in [13] and can be summarized in the following steps: *(1)* sample a candidate solution, in this case a candidate constraint allocation and a constraint penalty, from the neighborhood of the current solution; *(2)* compare the likelihood functions of the expert demonstrations under this proposal and the original solution; and *(3)* probabilistically accept the candidate solution based on the likelihood of the proposal compared to the likelihood of the current solution. In our paper we iteratively sample different constraint configurations and penalty rewards at each step, allowing us to sample from the posterior distribution over likely constraints given demonstrations. At each iteration we choose whether to sample a candidate constraint or reward penalty. When sampling constraints we select a random index i from $\{1, \dots, n\}$ and only change the constraint status of state i: $C'[i] = \neg C[i],\ C'[j] = C[j], \forall j \neq i$. For the reward penalty, r_p we sample using a Gaussian proposal distribution. The detailed process can be seen in Algorithm 1.

We compute the likelihood of a sample by assuming a Boltzmann-type choice model [19]. Under this model the likelihood of a trajectory ξ of length m is given by

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

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 penalty rewards $P(\mathbf{C}, r_p)$ the posterior distribution is given by

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

The normalizing constant can be neglected as it does not affect the comparison of the likelihoods between alternative constraint allocations and reward choices. We choose an uninformative prior for our experiments, but plan to incorporate informative priors into future work.

We denote a MDP with constraint set C and penalty reward r_p with $M_{\mathbf{C}, r_p}$ for conciseness. Essentially the algorithm alternates between sampling constraint sets and penalty rewards. When sampling constraints the algorithm randomly selects a single state and switches that state from “no constraint" to “constraint" or vice versa (if the state was already a constraint). We found that sampling each time new constraints for all the states was too noisy and led to poor performance. On the other hand the sampling of penalty rewards takes place by injecting random normal noise of variance σ^2 to the proposals. For each of these samples the log-likelihood of the new configuration is computed and compared with the log-likelihood of the existing configuration. The Q-values used in the likelihood functions are computed by running value iteration on the MDP. If the new choice of variables is more likely then it is accepted by the algorithm. We further allow for randomly accepting proposals even if they are not associated with higher log-likelihood to enhance exploration. The initialization of both the constraint set and the penalty reward is done randomly.

2.4 Active Constraint Learning

One of the benefits of using a Bayesian approach to infer constraints is the quantification of the uncertainty in that estimation. Safety critical applications require safety guarantees during the deployment of an agent. We examine the utility of a simple active learning acquisition function which is based on the variance of the constraint estimates. We propose an active learning method in which the agent can query the expert M_Q times for M_D specific state demonstrations associated with high uncertainty in the current estimation. The outline of this process is summarized in Algorithm 2.

At every iteration of the active learning algorithm the BIRLC Algorithm is called to provide a new estimate of the constraint allocation. To expedite the process we initialize the constraint allocation in BIRLC using a warm start. More specifically, after the first iteration, each of the subsequent calls to BIRLC uses the MAP solution of the previous iteration as the constraint allocation and penalty reward initialization.

Algorithm 1 BICRL

```
1: Parameters: Number of iterations $M$, noise standard deviation $\sigma$
2: Randomly sample constraint vector $\mathbf{C} \in \{0,1\}^N$
3: Randomly sample reward penalty $r_p \in \mathbb{R}$
4: Compute $Q^\pi(s, a, \mathbf{R}, \mathbf{C}, r_p)$ on $M_{\mathbf{C}, r_p}$
5: for $i = 1, \dots, M$ do
6:     if sample constraints then
7:         Randomly sample state $i$ from $\{1, \dots, N\}$
8:         Set $\mathbf{C}'[i] = \neg\mathbf{C}[i]$, $r'_p = r_p$
9:     else
10:        Set $r'_p = r_p + \mathcal{N}(0, \sigma)$, $\mathbf{C}' = \mathbf{C}$
11:    Compute $Q^\pi(s, a, \mathbf{R}, \mathbf{C}', r'_p)$ on $M_{\mathbf{C}', r'_p}$
12:    if $\log \mathcal{L}(\mathbf{C}', r'_p) \geq \log \mathcal{L}(\mathbf{C}, r_p)$ then
13:        Set $\mathbf{C} = \mathbf{C}'$, $r_p = r'_p$
14:    else
15:        Set $\mathbf{C} = \mathbf{C}'$, $r_p = r'_p$ w.p. $\mathcal{L}(\mathbf{C}', r'_p)/\mathcal{L}(\mathbf{C}, r_p)$
16: Return $\mathbf{C}$, $r_p$
```

Algorithm 2 Active Constraint Learning

```
1: Parameters: Number of iterations $M_Q$ and of expert demonstrations $M_D$
2: for $i = 1, \dots, M_Q$ do
3:     Run BICRL and compute $var(\mathbf{C}[i]), \ \forall\ i$
4:     Select state $i^* = \text{argmax}_i \ var(\mathbf{C}[i])$
5:     Query expert for $M_D$ state $i^*$ demonstrations
6:     Add demonstrations to $\mathcal{D}$
```

3 Experiments

We carry out experiments mainly in a discrete state space environment and we present the bulk of our results for that case. In subsection 3.2 we provide some preliminary results regarding a possible extension of our method to continuous state space environments.

3.1 Discrete State Space

We first demonstrate the performance of our method in the discrete navigation environment shown in Figure 1. The goal state s_g (marked with a green X), 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 6 constraint states that can be seen in Figure 1 colored with red. We further assume that the dynamics are stochastic in that when the agent tries to move towards a direction there is a 10% chance that the agent will move to a neighboring state instead. To model a noisy optimal demonstrator that seeks to avoid the constraints, we associate each constraint with a reward of -10 and synthesize expert trajectories using a Boltzmann policy with $\beta = 1$. The Q-values needed for the policy are obtained by running value iteration on the MDP. We provide to our learning algorithm 40 such trajectories from the demonstrator, each having as a starting state the bottom right corner. Figure 1 shows the state visitation frequencies of those trajectories along with the original constraints in the environment.

We run BICRL for $M = 4000$ iterations with a proposal mean and standard deviation of $\mu = 0$ and $\sigma = 1$ respectively, for sampling r_p. During BICRL we choose to sample constraints 20 times more frequently than rewards. Our Bayesian method correctly identifies the majority of the actual constraints. The right figure in Figure 1 shows the mean predictions for the constraints with values approaching 1.0 corresponding to high chances of that state being constrained. The algorithm further manages to infer the reward penalty r_p returning an estimated mean value of -9.96. Expectedly, the agent demonstrates high uncertainty in areas that are far away from the expert demonstrations, like the bottom left section of the grid.

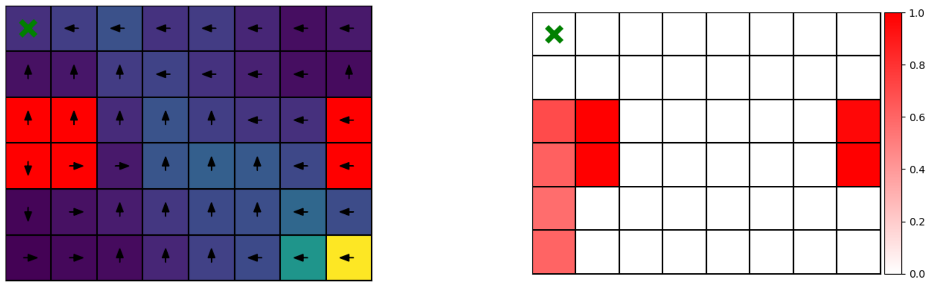

Figure 1: *(left)* True constraints (red) along with optimal policy and expert state visitation frequencies. *(right)* Mean constraint estimation using BICRL Algorithm 1.

3.1.1 Classification Performance

To quantify the performance of the constraint classification task we provide a plot in Figure 2 of the True Positive Rate (TPR), the False Positive Rate (FPR) and the False Negative Rate (FNR) of the MAP estimate at each BICRL iteration. We average the rates over 10 independent runs of BICRL each using 40 expert demonstrations. The true constraints are the ones specified in Figure 1. The rates are not necessarily monotonic as a number of constraint configurations can lead to high likelihood function evaluations due to the noisy trajectories and the fact that they only partially cover the state space. As the number of iterations increases the MAP estimates tend towards the true constraint allocation.

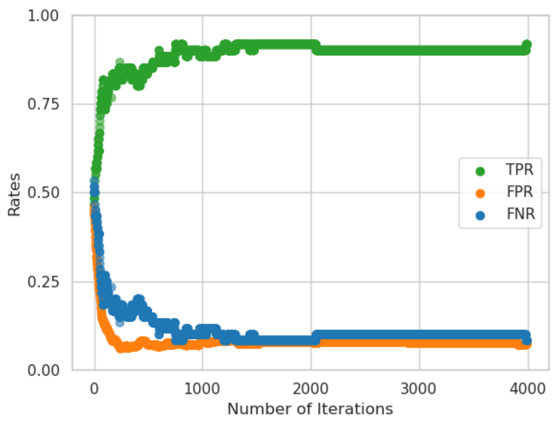

Figure 2: True Positive, False Positive and False Negative classification rates of MAP constraint estimates. Results averaged over 10 independent experiments.

3.1.2 Active Learning

Having access to the posterior distribution of the constraint allocation allows for an active learning approach in which the agent queries the expert for specific state-action pair demonstrations that can allow for faster and more accurate inference. To showcase this approach we reduce the number of expert demonstrations to 4 and rerun BICRL. We report the mean prediction for the constraints learned from 4 demonstrations in Figure 3. As expected, uncertainty is now significantly higher. Applying Algorithm 2 from Section 2.4 with $M_Q = 20$ and $M_D = 4$ we obtain mean estimates that are significantly more accurate as seen in Figure 3.

To obtain intuition about the significance of an active learning approach we further compare the active learning method outlined in Algorithm 2 with a naive approach that each time randomly selects a state

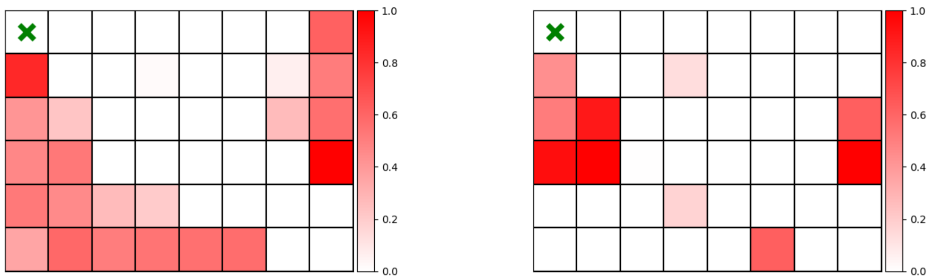

Figure 3: *(left)* Mean constraint estimation using only 4 expert trajectories. *(right)* Mean constraint estimation improved by active learning.

to query the expert for demonstrations. We run 10 independent experiments on the same environment as shown in Figure 1. In each experiment we have 10 initial expert demonstrations and we use the active and the naive method to query the expert $M_Q = 10$ times for $M_D = 10$ state-action pair demonstrations at a particular state each time. After each iteration $i = 1, \ldots, M_Q$ of the active and naive learning methods we compute the TPR, FPR and FNR. Figure 4 contains those rates averaged over the 10 experiments. The active learning method outperform the naive case after 3 queries.

Another motivation for our Bayesian method is outlined in the next subsection in which we compare it to a recent RL constraint inference method [15] that is based on Maximum Likelihood Estimation. In a number of environments with varying number of constraints and randomized constraint allocations our method manages to infer the underlying constrained states more accurately.

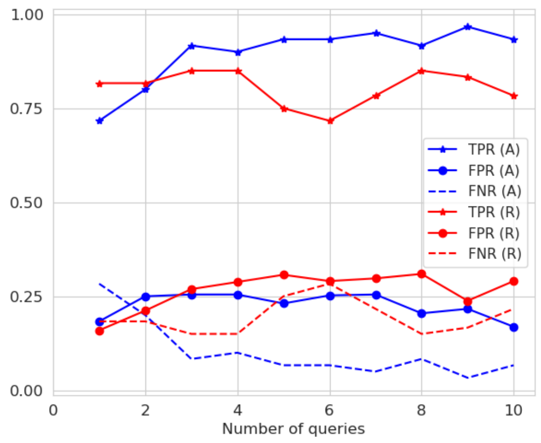

Figure 4: True Positive, False Positive and False Negative classification rates of MAP constraint estimates for active (A) and naive (random) (R) learning methods. Results averaged over 10 independent experiments.

3.1.3 Bayesian vs Maximum Likelihood Estimation

To further quantify the benefits of our method we also compare it with Greedy Iterative Constraint Inference (GICI) [15] which is a Maximum Likelihood approach for constraint inference. We simulate three types of environments with $m = 6, 8$ and 10 total constraints in each. For each of those environments we randomly allocate the m constraints in the state space and we report the classification metrics for both algorithms. For each value of m we perform 20 simulations each time sampling a random constraint allocation and we report the averaged results in Table 1.

	BICRL			GICI		
m	TPR	FPR	FNR	TPR	FPR	FNR
6	**0.96**	**0.01**	**0.03**	0.24	0.04	0.75
8	**0.96**	**0.01**	**0.04**	0.24	0.07	0.75
10	**0.92**	**0.02**	**0.08**	0.26	0.06	0.74

Table 1: True Positive, False Positive and False Negative classification rates for BICRL and GICI over a varying number of randomly chosen constraint allocations. Results averaged over 20 runs.

For these simulations we utilize 40 expert trajectories each time sampled using a Boltzmann policy with $\beta = 1$. The number of iterations for BICRL is again $M = 4000$ and the results do not include the active learning improvement. For GICI we infer for each number of true constraints m the m most likely constraints after which we terminate the algorithm, while for BICRL we return the m most likely constraints, as measured by the mean of the posterior of the constraints. BICRL manages to infer significantly more accurately the constraints in most cases.

3.2 Continuous State Space

In this section we extent our results to the continuous state space in a two-dimensional navigation task. The agent transitions using a simple point mass model with the control actions being the direction $a_o \in [0, 2\pi]$ and velocity $a_v = \{0.02, 0.04, 0.08\}$. The direction action is also discrete with $a_o = \{0, \pi/4, \ldots\}$ taking 8 equally distanced values in $[0, 2\pi]$ and the system transitions are deterministic. Optimal and Boltzmann-like policies are obtained by using a thin discretization of the state space into 30×30 grid points and solving the discrete MDP. The starting state s_s is $(0.5, 0)$ and the goal state s_g is $(0.5, 1)$. The agent receives a reward of $+2$ upon arriving at a minimum distance of 0.1 from the goal, goal set S_g, at which point the episode ends. The constraint set S_c is a disc or radius 0.15 centered at $(0.5, 0.5)$. All states inside that disc are associated with a penalty reward of -10 while the living reward is -1. The environment along with a subsample of Boltzmann trajectories are shown in Figure 5.

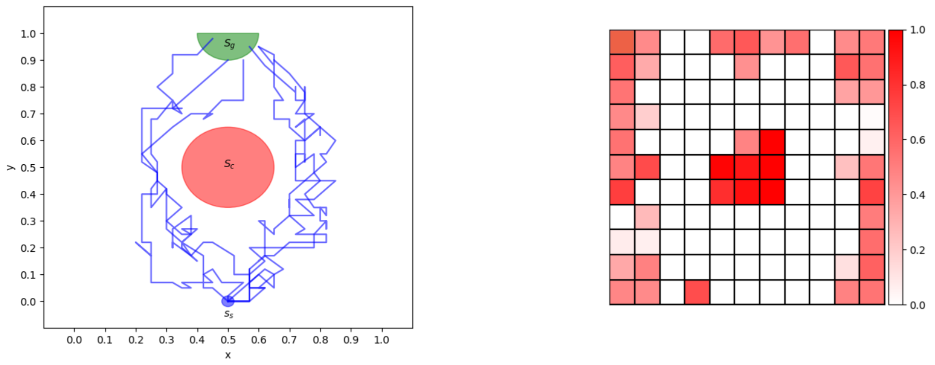

Figure 5: *(left)* True constraints set S_c (red), goal set S_g (green) and starting state s_s (blue) along with sampled expert trajectories. *(right)* Mean constraint estimation using BICRL Algorithm.

We run the BICRL Algorithm for $M = 4000$ iterations with the proposal mean being zero and standard deviation being $\sigma = 1$ for sampling r_p. We sample constraint allocations (line 6 in Algorithm 1) 50 times more frequently than we sample penalty rewards. The 100 expert trajectories were obtained using a Boltzmann policy with $\beta = 2$. The algorithm manages to recover most of the true constraints while it maintains some uncertainty for the states located the furthest away from the expert trajectories. Given that the expert trajectories terminate as soon as the mass moves inside the set S_g and given the discretization method, the Algorithm predicts that part of the goal set is a constraint set as in that part we essentially observe very few demonstrations. The predicted penalty reward in this case has a mean value between -3 and -2 which is significantly different from the actual value of -10.

4 Conclusion and Future Extensions

We proposed a Bayesian approach to infer the unknown constraints in a discrete and continuous state MDP. Our method manages to infer with high accuracy most of the constraints while providing confidence values on the estimation that can be valuable in safety critical applications. It is also suitable for environments with stochastic transitions as well as for applications in which individual, and potentially suboptimal, state-action demonstrations are provided. Finally, we proposed an active learning scheme that reduces the posterior variance and improves the mean and MAP constraint estimates. For ease of interpretability, our experiments used a simple 2-D environment. We are currently working to extend our results to more complex environments and exploring whether we can apply recently proposed preference learning [4] or variational inference [5] approaches to enable scalable Bayesian inference in high-dimensional continuous state MDPs.

References

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

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

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

[4] 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*, pages 1165–1177. PMLR, 2020.

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

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

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

[8] 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.

[9] 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.

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

[11] 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*, pages 1005–1014. PMLR, 2020.

[12] 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.

[13] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *IJCAI*, volume 7, pages 2586–2591, 2007.

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

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

[16] Burr Settles. Active learning literature survey. 2009.

[17] 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*, pages 458–464. Springer, 2018.

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

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