Safe Inverse Reinforcement Learning via Control Barrier Function

Anonymous Author(s)
Affiliation
Address
`email`

Abstract: Learning from Demonstration (LfD) is a powerful method for enabling robots to perform novel tasks as it is often more tractable for a non-roboticist end-user to demonstrate the desired skill and for the robot to efficiently learn from the associated data than for a human to engineer a reward function for the robot to learn the skill via reinforcement learning (RL). Safety issues arise in modern LfD techniques, e.g., Inverse Reinforcement Learning (IRL), just as they do for RL; yet, safe learning in LfD has received little attention. In the context of agile robots, safety is especially vital due to the possibility of robot-environment collision, robot-human collision, and damage to the robot. In this paper, we propose a safe IRL framework, CBFIRL, that leverages the Control Barrier Function (CBF) to enhance the safety of the IRL policy. The core idea of CBFIRL is to combine a loss function inspired by CBF requirements with the objective in an IRL method, both of which are jointly optimized via gradient descent. In the experiments, we show our framework performs safer compared to IRL methods without CBF, that is $\sim 15\%$ and $\sim 20\%$ improvement for two levels of difficulty of a 2D racecar domain and $\sim 50\%$ improvement for a 3D drone domain.

Keywords: Agile Robot, Learning from Demonstration, Control Barrier Function

1 Introduction

Agility is an indispensable feature for robots applied in manufacturing or everyday life [1, 2] because the physical space can change very fast and the robots need to react quickly. Recent advances in robot learning have offered the potential to improve the agility of various robots, including high-speed cars [3, 4, 5, 6], drones [7, 8], legged robots [9, 10], and robots in sports [11, 12]. Reinforcement learning (RL) is a ubiquitous approach to robot learning for developing high-performance controllers for robots. Although various RL-based methods have shown promising results in both simulation and real robots, the design of reward functions that elicit desired behaviors could still be laborious and time-consuming [13]. Also, agents trained with RL can behave unnaturally [13]. Although it's hard to design controllers, humans can demonstrate robots for agile control (e.g., racecar driving and drone flying). As such, Learning from Demonstration (LfD), a field empowering end-users to program robots by demonstrations instead of a computing language [13, 14, 15, 16], can help address the issues by learning from experts and work in a more sample-efficient way.

Inverse Reinforcement Learning (IRL) [17] is a technique in LfD research that aims to infer a demonstrator's underlying objective function (i.e., reward) from demonstrations. However, the safety of IRL approaches is yet to be explored. Previous works in safe IRL [18, 19, 20, 21, 22] mainly focus on adding high-confidence bounds on the learned policy's performance, which is an indirect approach to promote safety and dangerous cases can still happen as the underlying objective function can guide the agent into danger. Therefore, a direct approach to avoid dangerous configurations in IRL, instead of hoping for safety in a performance-focused way, is needed.

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

To address safety in RL [23, 24, 25, 26], researchers have leveraged the control barrier function (CBF) [27, 28], which is a method to synthesize a control policy that maintains the system within a safe set of states. Although CBF can help directly avoid dangerous cases in RL, to the best of our knowledge, there's no work incorporating CBFs with IRL algorithms to mitigate possible safety issues. The most related works focus on synthesizing CBF from data (e.g., expert demonstrations) and combine the CBF with hand-designed control methods [29, 30, 31]. However, these methods fail to show how to make use of the synthesized CBF to enforce the learned policy in IRL safer.

In this paper, we propose a framework named CBFIRL where the CBF, approximated by a neural network, is learned and utilized to enforce the learned policy in IRL to take safe actions by optimizing a joint loss. We present empirical results over two simulated agile robot control tasks and find the proposed CBFIRL has $\sim 15\%$ and $\sim 20\%$ less collision for two levels of difficulty of one 2D racecar domain and $\sim 50\%$ less collision for one 3D drone domain than just IRL.

2 Preliminaries

Markov Decision Process – We model the environment as a Markov Decision Process (MDP) $\mathcal{M}$ [32], which is defined by $\langle \mathcal{S}, \mathcal{A}, R, T, \gamma, \rho_0 \rangle$. $\mathcal{S}$ and $\mathcal{A}$ denote the state space and action space, respectively. $R : \mathcal{S} \to \mathbb{R}$ is the reward function that tells the reward for a given state. $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ represents a deterministic transition function that gives the next state s' after applying the action a to current state s. $\gamma \in (0, 1)$ is the temporal discount factor. $\rho_0 : \mathcal{S} \to \mathbb{R}$ denotes the initial state distribution. A policy $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a mapping from states to probabilities over actions. We could generate a trajectory $\tau =< s_0, a_0, r_0, \cdots, s_t, a_t, r_t, \cdots >$ by executing the policy within the environment. The expected discounted return of one policy could be calculated by $J(\pi) = \mathbb{E}_{\tau\sim\pi} [\sum_{t=0}^{\infty} \gamma^t R(s_t)]$. The objective for RL is to find the optimal policy, $\pi^* = \arg\max_\pi J(\pi)$.

Inverse Reinforcement Learning – Inverse reinforcement learning (IRL) considers an MDP sans reward function and infers a reward function $\hat{R}$ from a set of demonstration trajectories, $\mathcal{D} = \{\tau_1, \tau_2, \cdots, \tau_N\}$. Our method is based on adversarial inverse reinforcement learning (AIRL) [33]. AIRL consists of a generator (i.e., a policy) to imitate the demonstrator and a discriminator to distinguish the generator's behavior from that of the demonstrator. The discriminator is defined as $D_\theta = e^{\{\hat{f}_\theta(s,a)\}}/e^{\{\hat{f}_\theta(s,a)\}+\pi_\phi(a|s)}$, where $\hat{f}_\theta(s, a)$ is the inferred advantage function and $\pi_\phi(a|s)$ is the learned policy parameterized by ϕ. The discriminator is trained to minimize a binary cross entropy loss, $\mathcal{L}_D$. The generator policy $\pi_\phi(a|s)$ is trained by optimizing the policy loss, $\mathcal{L}_{policy} = \max J(\pi)$, and to maximize the recovered reward function.

Control Barrier Function – Let $\mathcal{S}_s \subset \mathcal{S}$ be the safe states set, $\mathcal{S}_d = \mathcal{S} \backslash \mathcal{S}_s$ be the dangerous states set, and $\mathcal{S}_0$ be the set of initial states. A control barrier function, h, needs to satisfy the three requirements [28, 34]: **R1**: $\forall s \in \mathcal{S}_0$, $h(s) \geq 0$; **R2**: $\forall s \in \mathcal{S}_d$, $h(s) < 0$; and **R3**: $\forall s \in \{s|h(s) \geq 0\}$, $(h(T(s, \pi_\phi(s))) - h(s))/\Delta t + \alpha(h(s)) \geq 0$. Here, $\alpha(\cdot)$ is a class-$\mathcal{K}$ function (i.e., $\alpha(\cdot)$ is strictly increasing and $\alpha(0) = 0$). **R1** and **R3** ensure trajectories to stay inside the superlevel set $\mathcal{C}_h = \{s \in \mathcal{S} : h(s) \geq 0\}$. **R2** guarantees that unsafe states will never be visited under the policy π_ϕ. In order to obtain a safe policy $\pi_\phi(\cdot)$ and an $h(\cdot)$ to meet the three requirements, we formulate a similar optimization objective as Qin et al. [26]. We denote $\mathcal{P}$ and $\mathcal{H}$ as the function classes for $\pi_\phi(\cdot)$ and $h(\cdot)$, and $\mathcal{T}$ as the set of all trajectories. We assume initial states are safe, i.e. $\forall s \in \mathcal{S}_s$, $h(s) \geq 0$. We then define the function $y : \mathcal{H} \times \mathcal{P} \times \mathcal{T} \to \mathbb{R}$ as given by Equation 1.

$$y(h, \pi_\phi, \tau) := min\{\inf_{s\in\mathcal{S}_s} h(s), \inf_{s\in\mathcal{S}_d} -h(s), \inf_{\{s|h(s)\geq 0\}\cap\tau} (h(T(s, \pi_\phi(s))) - h(s))/\Delta t + \alpha(h) \geq 0\} \tag{1}$$

R1-**R3** are satisfied when we find $h(\cdot)$ and $\pi_\phi(\cdot)$ such that $y(h, \pi_\phi, \tau) > 0$ for $\forall \tau \in \mathcal{T}$. Thus, The optimization objective is given by Equation 2.

$$\text{Find } h(\cdot) \in \mathcal{H} \text{ and } \pi_\phi(\cdot) \in \mathcal{P}, \qquad s.t. \qquad \forall\tau \in \mathcal{T}, y(h, \pi_\phi, \tau) > 0 \tag{2}$$

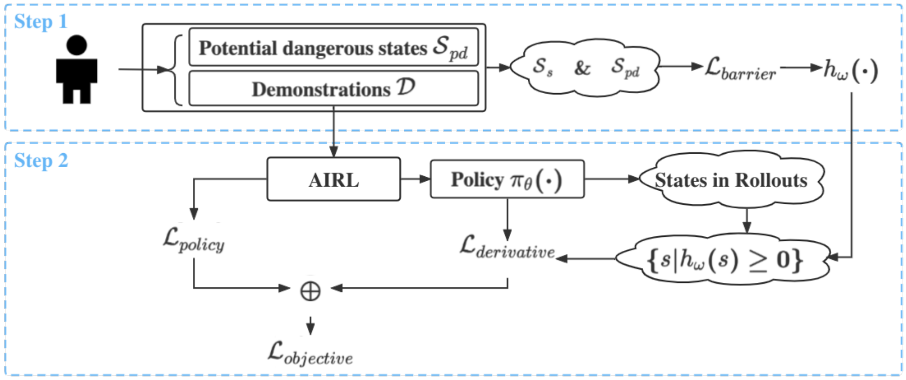

Figure 1: This figure depicts the architecture of CBFIRL.

3 Method

As shown in Figure 1, we combine CBF with AIRL in two steps: **Step 1)** Formulate a barrier loss function $\mathcal{L}_{barrier}$ to learn $h(\cdot)$ satisfying **R1** and **R2**; **Step 2)** leverage $h(\cdot)$ to formulate a derivative loss function $\mathcal{L}_{derivative}$ to synthesize a safer policy $\pi_\phi(\cdot)$ meeting **R3**. For Step 1, we formulate the $\mathcal{L}_{barrier}$ as shown in Equation 3 based on Equation 1, where $h_\omega(\cdot)$ is a neural network parameterized by ω. Terms in Equation 3 correspond to **R1** and **R2**. Intuitively, minimizing the $\mathcal{L}_{barrier}$ provides an $h_\omega(\cdot)$ that could discriminate safe states from dangerous ones. We collect the set $\mathcal{S}_s$ and $\hat{\mathcal{S}}_{pd}$ as described below in Section 3.1.

$$\mathcal{L}_{barrier} = \sum_{s \in \mathcal{S}_s} max(-h_\omega(s), 0) + \sum_{s \in \hat{\mathcal{S}}_{pd}} max(h_\omega(s), 0) \tag{3}$$

For Step 2, we formulate the $\mathcal{L}_{derivative}$ as shown in Equation 4, where the policy $\pi_\phi(\cdot)$ is a neural network parameterized by ϕ.

$$\mathcal{L}_{derivative} = \sum_{s \in \{s|h_\omega(s) \geq 0\}} max(-(h_\omega(T(s, \pi_\phi(s))) - h_\omega(s))/\Delta t - \alpha(h_\omega(s)), 0) \tag{4}$$

For the class-$\mathcal{K}$ function, we use a linear function $\alpha(h(s)) = \lambda h(s)$. Minimizing the $\mathcal{L}_{derivative}$ enforces the $\pi_\phi(\cdot)$ to generate actions that satisfy requirement **R3**, which provides a safe control policy. We now propose our combined loss function $\mathcal{L}_{combined}$ as shown in Equation 5, where the w is a trade-off coefficient between discriminator loss and derivative loss.

$$\mathcal{L}_{combined} = \mathcal{L}_{policy} + w * \mathcal{L}_{derivative} \tag{5}$$

By minimizing the $\mathcal{L}_{combined}$ via gradient descent, we could obtain a safer policy. The AIRL will be pre-trained to converge and provide a policy. The neural network h_ω pre-trained in Step 1 will be used to generate the set $\{s|h_\omega(s) \geq 0\}$, where the states s are explored by the policy.

3.1 Data Collection

To learn a CBF that enhances policy safety, we need to collect state sets $\mathcal{S}_s$, $\mathcal{S}_d$ before solving Equation 2. We assume the demonstration trajectories $\tau \in \mathcal{D}$ are safe and initialize $\mathcal{S}_s$ to be the set of states in the demonstrations. We cannot request demonstrators to take a risk of hurting themselves or damaging the robots to provide the dangerous states. Therefore, we define potentially dangerous states $\mathcal{S}_{pd}$ as a set that the agent has to pass before entering the $\mathcal{S}_d$. We design a new requirement "**R2**$'$: For $\forall s \in \mathcal{S}_{pd}$, $h(s) < 0$". As stated in the preliminaries, the agent cannot enter set $\mathcal{S}_{pd}$ if **R1**, **R2**$'$, and **R3** are satisfied. Hence, the agent cannot enter set $\mathcal{S}_d$ as well according to the definition of $\mathcal{S}_{pd}$. Therefore, **R2**$'$ is a more strict requirement that prevents the agent from entering dangerous states, and we could take the **R2**$'$ as **R2** and replace $\mathcal{S}_d$ in Equation 1 with $\mathcal{S}_{pd}$.

Table 1: This table shows the comparison between CBFIRL and AIRL on two domains.

Environment	Successful rate (Stdev)		Collision rate (Stdev)		
	AIRL	CBFIRL	AIRL	CBFIRL	Improvement
2D racecar - 8 obstacles	0.97 (0.02)	0.95 (0.03)	0.58 (0.07)	0.49 (0.09)	**15.52%**
2D racecar - 16 obstacles	0.63 (0.28)	0.64 (0.07)	1.00 (0.12)	0.80 (0.09)	**20.00%**
3D drone - 32 obstacles	0.34 (0.37)	0.30 (0.33)	0.61 (0.40)	0.31 (0.19)	**49.18%**

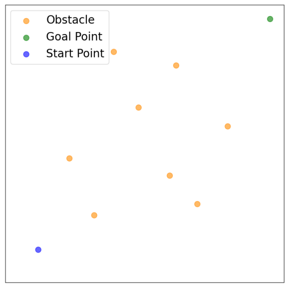

Figure 2: 2D racecar environment.

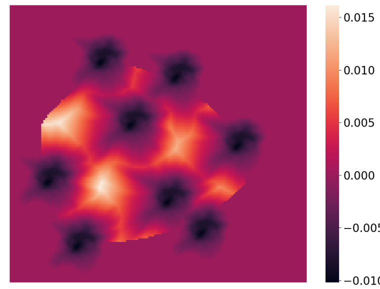

Figure 3: Heatmap of learned CBF.

Defining potentially dangerous states ensures that CBF learns information about dangerous states while being safe for users. Therefore, demonstrators can safely provide potentially dangerous states to a set $\hat{\mathcal{S}}_{pd}$, which acts to be an approximation to the $\mathcal{S}_{pd}$. To avoid losing all the feasible paths to the goal, we will request demonstrators to try to collect states close to dangerous states. One example of a good potential dangerous state close to dangerous state for a race car could be a position close to obstacles. We empirically show that the approximation $\hat{\mathcal{S}_{pd}}$ works well, as shown in Section 4.

4 Experimental Results

We evaluate CBFIRL on two simulated control environments: a 2D racecar and a 3D drone [26]. For both environments, the agent travels across the map to reach the blue target from the green start point without colliding with the yellow obstacles that are moving. We define the state as the combination of the position and velocity of the agent and the nearest K obstacles. The episode terminates after 100 timesteps for the 2D racecar and 400 for the 3D drone. We test two levels of difficulty in racecar (8 and 16 obstacles) and a setting of 32 obstacles for the drone domain.

Two metrics are designed to evaluate the performance of the CBFIRL: "Successful rate" which measures the ratio of reaching the goal and "Collision rate" which shows the ratio of collision out of the 100 trajectories. We evaluate the two metrics on 100 trajectories to test CBFIRL's task success and safety against AIRL. We summarize the comparison in Table 1. Across the three environments, CBFIRL achieves a smaller collision rate than AIRL, which indicates a safer policy. Meanwhile, CBFIRL achieves a similar success rate as AIRL, which shows that our method has a good balance between safety and performance without being over-conservative to stand still.

To evaluate the learned control barrier function $h(\cdot)$ in discriminating the safe set, $\mathcal{S}_s$, from the potentially dangerous states, $\hat{\mathcal{S}}_{pd}$, we visualize the $h(\cdot)$ for one 2D racecar state through the heatmap in Figure 3. In generating the heatmap, We fix the positions of all obstacles and only move the agent over the map, which provides us with the corresponding $h(s)$ for varied s to build the heatmap. Figure 3 show $h(s) < 0$ (darker) in the area where the agent is close to the obstacles (Shown in Figure 2) and provides qualitative evidence that the set $\hat{\mathcal{S}}_{pd}$ works well as an approximation of $\mathcal{S}_{pd}$.

5 Conclusion

In this paper, we develop a novel framework, CBFIRL, to learn a safe policy from demonstrations by embedding the safety property of CBF into IRL methods. We transform the optimization problem of satisfying CBF conditions into a learning framework where the loss functions could be used to enhance the safety of IRL. We empirically validate that the proposed CBFIRL has less collisions for the two agile-robot domains than just IRL.

References

[1] Z. Kootbally, C. Schlenoff, B. Antonishek, F. Proctor, T. Kramer, W. Harrison, A. Downs, and S. Gupta. Enabling robot agility in manufacturing kitting applications. *Integrated Computer-Aided Engineering*, 25(2):193–212, 2018.

[2] Q. Yuan and I. S. W. Leong. Human-robot coordination in agile robot manipulation. In *International Conference on Social Robotics*, pages 532–540. Springer, 2021.

[3] H. Chae, C. M. Kang, B. Kim, J. Kim, C. C. Chung, and J. W. Choi. Autonomous braking system via deep reinforcement learning. In *2017 IEEE 20th International conference on intelligent transportation systems (ITSC)*, pages 1–6. IEEE, 2017.

[4] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura. Navigating occluded intersections with autonomous vehicles using deep reinforcement learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2034–2039. IEEE, 2018.

[5] S. Yang, W. Wang, C. Liu, W. Deng, and J. K. Hedrick. Feature analysis and selection for training an end-to-end autonomous vehicle controller using deep learning approach. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1033–1038. IEEE, 2017.

[6] S. Wang, D. Jia, and X. Weng. Deep reinforcement learning for autonomous driving. *arXiv preprint arXiv:1811.11329*, 2018.

[7] H. X. Pham, H. M. La, D. Feil-Seifer, and A. Nefian. Cooperative and distributed reinforcement learning of drones for field coverage. *arXiv preprint arXiv:1803.07250*, 2018.

[8] M. A. Akhloufi, S. Arola, and A. Bonnet. Drones chasing drones: Reinforcement learning and deep search area proposal. *Drones*, 3(3):58, 2019.

[9] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. Eslami, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

[10] X. B. Peng, P. Abbeel, S. Levine, and M. Van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)*, 37(4):1–14, 2018.

[11] L. Liu and J. Hodgins. Learning basketball dribbling skills using trajectory optimization and deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

[12] L. Chen, R. Paleja, and M. Gombolay. Learning from suboptimal demonstration via self-supervised reward regression. *arXiv preprint arXiv:2010.11723*, 2020.

[13] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020.

[14] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard. Recent advances in robot learning from demonstration. *Annual review of control, robotics, and autonomous systems*, 3: 297–330, 2020.

[15] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots. Agile autonomous driving using end-to-end deep imitation learning. *arXiv preprint arXiv:1709.07174*, 2017.

[16] L. Chen, R. Paleja, M. Ghuy, and M. Gombolay. Joint goal and strategy inference across heterogeneous demonstrators via reward network distillation. In *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, pages 659–668, 2020.

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

[18] D. Brown, R. Coleman, R. Srinivasan, and S. Niekum. Safe imitation learning via fast bayesian reward inference from preferences. In *International Conference on Machine Learning*, pages 1165–1177. PMLR, 2020.

[19] D. Brown and S. Niekum. Efficient probabilistic performance bounds for inverse reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[20] D. S. Brown, W. Goo, and S. Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on robot learning*, pages 330–359. PMLR, 2020.

[21] D. S. Brown and S. Niekum. Toward probabilistic safety bounds for robot learning from demonstration. In *2017 AAAI Fall Symposium Series*, 2017.

[22] D. S. Brown and S. Niekum. Deep bayesian reward learning from preferences. *arXiv preprint arXiv:1912.04472*, 2019.

[23] Z. Marvi and B. Kiumarsi. Safe reinforcement learning: A control barrier function optimization approach. *International Journal of Robust and Nonlinear Control*, 31(6):1923–1940, 2021.

[24] J. Choi, F. Castaneda, C. J. Tomlin, and K. Sreenath. Reinforcement learning for safety-critical control under model uncertainty, using control lyapunov functions and control barrier functions. *arXiv preprint arXiv:2004.07584*, 2020.

[25] H. Ma, J. Chen, S. Eben, Z. Lin, Y. Guan, Y. Ren, and S. Zheng. Model-based constrained reinforcement learning using generalized control barrier function. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4552–4559. IEEE, 2021.

[26] Z. Qin, K. Zhang, Y. Chen, J. Chen, and C. Fan. Learning safe multi-agent control with decentralized neural barrier certificates. *arXiv preprint arXiv:2101.05436*, 2021.

[27] A. D. Ames, J. W. Grizzle, and P. Tabuada. Control barrier function based quadratic programs with application to adaptive cruise control. In *53rd IEEE Conference on Decision and Control*, pages 6271–6278. IEEE, 2014.

[28] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pages 3420–3431. IEEE, 2019.

[29] L. Lindemann, H. Hu, A. Robey, H. Zhang, D. V. Dimarogonas, S. Tu, and N. Matni. Learning hybrid control barrier functions from data. *arXiv preprint arXiv:2011.04112*, 2020.

[30] A. Robey, H. Hu, L. Lindemann, H. Zhang, D. V. Dimarogonas, S. Tu, and N. Matni. Learning control barrier functions from expert demonstrations. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 3717–3724. IEEE, 2020.

[31] M. Srinivasan, A. Dabholkar, S. Coogan, and P. A. Vela. Synthesis of control barrier functions using a supervised machine learning approach. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7139–7145. IEEE, 2020.

[32] D. J. White. A survey of applications of markov decision processes. *Journal of the operational research society*, 44(11):1073–1096, 1993.

[33] J. Fu, K. Luo, and S. Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

[34] Y. Luo and T. Ma. Learning barrier certificates: Towards safe reinforcement learning with zero training-time violations. *Advances in Neural Information Processing Systems*, 34:25621–25632, 2021.