Simplifying Constraint Inference with Inverse Reinforcement Learning

Adriana Hugessen
Mila, Université de Montréal

Harley Wiltzer
Mila, McGill University

Glen Berseth
Mila, Université de Montréal

adriana.knatchbull-hugessen,wiltzerh,glen.berseth}@mila.quebec

Abstract

Learning safe policies has presented a longstanding challenge for the reinforcement learning (RL) community. Various formulations of safe RL have been proposed; however, fundamentally, tabula rasa RL must learn safety constraints through experience, which is problematic for real-world applications. Imitation learning, alternatively, is often preferred in real-world settings because the experts' safety preferences are embedded in the data the agent imitates. However, imitation learning is limited in its extensibility to new tasks, which can only be learned by providing the agent with expert trajectories. For safety-critical applications with sub-optimal or inexact expert data, it would be preferable to learn only the safety aspects of the policy through imitation, while still allowing for task learning with RL. The field of inverse constraint learning, which seeks to infer constraints from expert data, is a promising step in this direction. However, prior work in this area has relied on complex tri-level optimizations in order to infer safe behavior (constraints). This challenging optimization landscape leads to sub-optimal performance on several benchmark tasks. In this work, we present a simplified version of constraint inference that performs as well or better than prior work across benchmarks. Moreover, besides improving performance, this simplified framework more readily lends itself to various extensions of interest, such as offline constraint inference.

1 Introduction

Reinforcement learning (RL) has made significant advances in recent years, yet real-world applications of RL remain limited due to various challenges, including particularly safety concerns (Dulac-Arnold et al., 2021). One common setting where RL holds promise for real-world deployments is replacing existing, possibly sub-optimal, human-managed control policies. For example, in the field of power network control, decisions are often made by a combination of automatic control policies combined with careful human monitoring and intervention (Marot et al., 2022). However, concerns regarding the deployment of autonomous systems on safety-critical tasks have hindered the adoption of RL for replacing legacy control systems. This is particularly true for deep RL, which functions largely as a black box controller.

In the setting of replacing legacy controllers, however, the challenge is simplified due to access to data produced by the current control system. While existing control policies may be sub-optimal, they are nonetheless "safe" in the sense that they obey some explicitly or implicitly defined constraints corresponding to a human understanding of safety. Ideally, we would like to be able to learn an RL policy that can outperform the current system, in terms of optimizing the reward, while continuing to respect these constraints. When these constraints are unknown (i.e. defined implicitly through human actions), the implicit safety procedures that are being followed may not be explicitly known or have an obvious encoding. Thus, it would be beneficial to infer the constraints directly from trajectories produced by the existing control policy.

Typically, in settings with large-scale offline data, imitation learning (Schaal, 1996) or offline RL (Kumar et al., 2020; Kostrikov et al., 2022) might be used. However, these methods only extract a single policy that satisfies constraints and optimizes a reward function. Moreover, imitation learning generally cannot outperform the expert, which is prohibiting in cases where we wish to improve over current control policies. Offline RL can learn policies that outperform the demonstration (Kumar et al., 2020), however, offline RL cannot ensure that safety constraints remain satisfied without access to the constraint function or constraint violation annotations in the dataset.

Ideally, we would like to extract safety constraints from the data based on the expert behavior, which can then be used downstream to constrain task-specific learning. Learning constraints from expert trajectories is the purview of the field of constraint inference. Various methods have been proposed in this domain, with early work focusing on simple settings such as tabular MDPs (Scobee & Sastry, 2020). More generally, inverse constraint learning can infer constraints in continuous settings using parameterized constraint functions and adapting IRL methods such as maximum entropy IRL (Ziebart et al., 2008) to the constrained setting by solving a constrained MDP in the inner optimization. However, this tri-level optimization creates a challenging learning landscape.

In this work, we demonstrate that the constrained MDP inner loop is an unnecessary complication and that regular IRL techniques can recover as good and sometimes better solutions to constraint inference problems than more complicated methods. This result is significant because it allows simpler training dynamics for constraint inference and implies that advances in sub-domains of IRL can be directly applied to the constraint inference case. For example, recent progress in offline IRL (Yue et al., 2023; Kim et al., 2023) can be readily adapted to the constraint inference case to infer constraints entirely offline, significantly increasing the scope of applicability in real-world settings.

In particular, we make the following contributions. First, we show that inverse constrained RL and inverse RL are equivalent under certain classes of reward functions. Next, we experimentally validate this claim. Finally, we conduct ablations over various methods for regularizing deep RL to improve the stability and performance of constraint inference.

2 Related Work

Imitation learning and inverse reinforcement learning The problem of imitation learning (IL) is primarily concerned with learning a policy that produces similar behavior to a class of reference policies. Specifically, given transition data from "experts", the goal is to produce a new policy that, in a sense, generates a similar transition distribution. IL methods as discussed above can generally learn performant imitation policies, but provide no insight about the reward function that the expert is implicitly optimizing; this is the focus of inverse reinforcement learning, which will be useful for the purpose of learning transferable constraint functions. Inverse reinforcement learning (IRL) learns a policy that can mimic expert trajectories and also aims to infer a reward function that explains the expert behavior (Ng et al., 2006). The IRL problem is considerably more difficult than the IL problem; in particular, there is no unique reward function that maximally explains the expert behavior. The work of Abbeel & Ng (2004) alleviates this issue with a *maximum margin constraint* in the optimization. The work of Ziebart et al. (2008) resolves this problem by inferring the most likely reward function under a particular probabilistic model; this approach is widely known as *maximum entropy IRL* (MaxEnt IRL).

Inverse constrained reinforcement learning Early work in constraint inference focused primarily on particular settings such as convex constraints (Menner et al., 2019; Miryoosefi et al., 2019), or tabular RL (Scobee & Sastry, 2020; McPherson et al., 2021; Chou et al., 2020). More recent methods have integrated the power of deep learning within the constraint inference framework through adaptions of MaxEnt IRL to the constrained setting through inverse constrained reinforcement learning (ICRL) (Malik et al., 2021; Liu et al., 2023a; Kim et al., 2023). These methods essentially replace the forward RL inner loop of IRL with a constrained version of the problem by casting it as a constrained MDP (CMDP) and solving by Lagrangian or other methods. This adds

additional complexity to the IRL problem which is already difficult due to the bi-level optimization and identifiability issues (Gleave & Toyer, 2022). Though these prior works have sometimes considered simple IRL baselines (Malik et al., 2021; Liu et al., 2023a), none have specifically outlined the connections between IRL and ICRL, nor have they optimized IRL for constraint inference.

3 Background

Maximum Entropy IRL As discussed in Section 2 there are several formulations of IRL. Here we describe maximum entropy IRL (Ziebart et al., 2008; Ziebart, 2010). Given a set of expert trajectories $D_e = \{\tau_e^{(i)}\}_{i=0}^N$, which follow a policy π_E, a policy π can be trained to produce similar trajectories by solving the min-max problem:

$$\min_{r \in \mathcal{R}} \max_{\pi} \left(\mathbb{E}_\pi \left[\sum_{t=0}^{T} r(s_t, a_t) \right] + H(\pi) - \mathbb{E}_{\pi_E} \left[\sum_{t=0}^{T} r(s_t, a_t) \right] \right) \tag{1}$$

where r is a learned reward function and $H(\pi)$ is a causal entropy term. Hence, IRL is searching for a reward function r that is high for the expert π_E and low for other policies.

Constrained RL One way to formulate constrained MDPs (CMDPs) is through trajectory-level constraints. In this formulation, the agent must solve the following objective:

$$\max_{\pi} \mathbb{E}_\pi \left[\sum_{t=0}^{T} r(s_t, a_t) \right] \text{s.t.} \quad \mathbb{E}_\pi \left[\sum_{t=0}^{T} c(s_t, a_t) \right] \leq \delta_i \quad \forall\, i \in \{1, ..., I\} \tag{2}$$

The most common method for solving this problem is Lagrangian variations of RL methods, such as SAC-Lag (Ha et al., 2020), where Equation (2) is converted to a min-max problem (See Equation (4)).

Inverse constrained RL Inverse constrained reinforcement learning (ICRL) is a class of methods used to infer constraints from expert trajectories. As in imitation learning and IRL, it is assumed that we have access to a set of expert trajectories $D_e = \{\tau_e^{(i)}\}_{i=0}^N$, which follow a policy π_E, however, it is now assumed that the expert policy is optimal under a CMDP instead of an MDP. It is generally assumed that the agent has access to the unconstrained (nominal) MDP with which to interact, and in particular that the reward is observed by the agent when interacting with the environment.

Kim et al. (2023) present a general formulation of ICRL as game-solving, though we note that their formulation was preceded by that of Malik et al. (2021), and, though motivated differently, generally matches it, up to some implementation details.

As shown in prior work (Swamy et al., 2021; Kim et al., 2023), the inverse RL problem can be cast as a two-player zero-sum game

$$\mathsf{Opt}_{\mathsf{IRL}}(\Pi, \mathcal{F}) = \min_{\pi \in \Pi} \sup_{f \in \mathcal{F}} J(\pi, f) - J(\pi_E, f) \tag{3}$$

where $\mathcal{F}$ is convex and compact and $J(\pi, f) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} f(s_t, a_t) \right]$

Similarly, the constrained RL problem in Equation (2) with a single constraint can also be cast as a two-player zero-sum game:

$$\min_{\pi \in \Pi} \max_{\lambda > 0} -J(\pi, r) + \lambda (J(\pi, c) - \delta) \tag{4}$$

Finally, Kim et al. (2023) show that for inverse constraint learning, these two games can be combined into a three-player game of the following form (where r is a given reward function in a class $\mathcal{F}_r$).

$$\mathsf{Opt}_{\mathsf{ICRL}}(\Pi, \mathcal{F}_r, \mathcal{F}_c) = \sup_{c \in \mathcal{F}_c} \max_{\lambda > 0} \min_{\pi \in \Pi} J(\pi_E, r - \lambda c) - J(\pi, r - \lambda c), \tag{5}$$

Practically speaking, solving this game involves running IRL where the inner loop optimization solves a constrained MDP using a Lagrangian version of RL, such as SAC-Lag (Ha et al., 2020).

4 Method

Starting from Equation (5), we now demonstrate that this tri-level optimization is equivalent to a simpler bi-level optimization under certain classes of constraint functions and that this bi-level optimization is equivalent to the IRL game-solving formulation in Equation (3).

4.1 Inverse constraint learning as IRL

Note that there are two outer maximizations in Equation (5), the first over constraints in the class $\mathcal{F}_c$ and the second over $\lambda > 0$. Our key insight is that, under a broad class of constraint functions $\mathcal{F}_c$, this tri-level optimization can be cast as a bi-level optimization, reducing ICRL to IRL.

Theorem 4.1 *Let $\pi_E \in \Pi$ and a reward function $r \in \mathcal{F}_r$ be given. Consider the objective*

$$\mathsf{Opt}_{\mathsf{S-ICRL}}(\Pi, \mathcal{F}_r, \mathcal{F}_c) = \max_{c \in \mathcal{F}_c} \min_{\pi \in \Pi} J(\pi_E, r - c) - J(\pi, r - c). \tag{6}$$

Then, if $\mathcal{F}_c$ is a convex cone (that is, $c \in \mathcal{F}_c \implies \lambda c \in \mathcal{F}_c$ for any $\lambda \geq 0$), it holds that $\mathsf{Opt}_{\mathsf{ICRL}}(\Pi, \mathcal{F}_r, \mathcal{F}_c) = \mathsf{Opt}_{\mathsf{S-ICRL}}(\Pi, \mathcal{F}_r, \mathcal{F}_c)$.

Moreover, suppose Π is compact and $\mathcal{F}_r = \mathcal{F}_c = \mathcal{F}$ is a vector space with elements $f : (s, a) \mapsto \langle \phi(s, a), w_f \rangle$, where $\phi : \mathcal{S} \times \mathcal{A} \to [0, 1]^d$ is a fixed (not necessarily known) feature map and $w_f \in \mathbb{R}^d$ identifies the elements of $\mathcal{F}$. Then any solution to $\mathsf{Opt}_{\mathsf{IRL}}(\Pi, \mathcal{F} - r)$[1] is a solution to $\mathsf{Opt}_{\mathsf{ICRL}}(\Pi, \mathcal{F}, \mathcal{F})$—that is, the ICRL problem is simply an instance of IRL.

Proof. See Appendix B. □

The intuition for this is the following. The only difference between solving a constrained MDP with Lagrange multiplier λ and solving an unconstrained MDP with a cost penalty term weighted by λ is that, in the former case λ is optimized to ensure constraint satisfaction. Indeed, given the correct λ a priori, constrained MDPs can be solved exactly by optimizing the later problem. This distinction is generally important when the cost function is fixed and unknown, however in the case of inverse learning where we *learn* the cost function from a class closed to scalar multiplication, it is possible to learn a cost that *directly* ensures constraint satisfaction, without scaling by the Lagrange multiplier. Note that optimizing over a convex cone $\mathcal{F}_c$ can pose challenges when exact optimization is required—for instance, convex cones violate a compactness condition required to prove regret bounds for inverse constrained RL given by Kim et al. (2023). However, in large scale applications that *approximately* optimize a constraint model represented by a deep neural network, one is often *already* in the setting where these regret bounds do not hold. Hence, when employing parameterized constraint inference through deep learning, it is sufficient to select an appropriate class as the output activation of our neural network and use a bi-level optimization to learn the scaled constraint.

4.2 Techniques for stabilizing constraint learning

Given that IRL and ICRL are mathematically equivalent under the class of constraint functions described above, the question of how best to perform constraint inference through inverse learning becomes largely a *practical* one. Is the optimization landscape smoother if we explicitly optimize for λ as a Lagrange multiplier or implicitly as part of the constraint function? Does optimizing

[1] The notation $\mathcal{F} - r$ refers to the set $\{f - r : f \in \mathcal{F}\}$.

the Lagrangian implicitly add additional algorithmic components that could explain the improved performance of ICRL over IRL in prior work? Moreover, are there *other* regularizations or modifications that we could consider? In this regard, we suggest the following practical modifications to IRL for the constraint learning case, which we will test in Section 5.

IRL with separate critics One component of prior methods for constraint inference that is implicitly added when utilizing a Lagrangian method for forward RL, is the use of separate critics for the reward and constraint functions. Separate critics may potentially be beneficial for learning, since separating the critics disentangles the value function learning of the reward (which is not changing during training) with the constraint function (which changes as the constraint is learnt). It is straightforward to implement separate critics in any IRL implementation without the additional Lagrangian optimization (please see Algorithm 1 in Appendix A.1).

Positive rewards and batch normalization One advantage of prior methods' use of a binary classifier for constraint representations is that it restricts costs to positive values (Malik et al., 2021). In practice, it is advantageous for interpretability and transferability to restrict the constraint function to be strictly positive, so that the learned constraint can only discourage the agent from visiting certain states not visited by the expert. Hence, in our experiments, we restrict values to the positive range by clipping the output of the constraint network. However, restricting the output of a neural network to positive values can be challenging due to the potential for vanishing gradients. Normalization layers such as batch normalization can be used to help prevent vanishing gradients (Ioffe & Szegedy, 2015). Hence, we test if adding batch normalization improves the convergence of IRL for constraint learning with output clipped to positive values.

Last-layer policy resetting Recent work has demonstrated that the primacy bias caused by plasticity loss can be a significant challenge for training deep RL agents (Nikishin et al., 2022). This challenge is exacerbated in the case of IRL and particularly ICRL since the learned cost function changes constantly during training while the true rewards are provided from the environment, potentially biasing early policy training towards learning the unconstrained policy. To combat this plasticity loss issue, we adopt the recommendations from Lyle et al. (2023) for MLPs, which advises periodically resetting the final layer of the network during training.

Reward scaling Finally, one potential practical advantage of the Lagrangian constraint inference methods is that the constraint function is learned independently of the reward scale and may be more robust to varying scales. To evaluate this, we consider normalizing the rewards using the rolling mean and variance, so that rewards are scaled to zero mean and unit variance.

5 Experiments

In this section, we conduct a series of experiments across several environments in order to answer the following questions: (1) How does IRL, without additional modifications, perform on constraint inference tasks compared to Lagrangian methods? and (2) How do various modifications or regularizations over vanilla IRL improve performance on constraint inference tasks?

Environments For our experiments, we consider the virtual environments for benchmarking inverse constraint learning, introduced by Liu et al. (2023a) since these were specially designed to test the performance of constraint inference tasks and also provide a recent baseline for Lagrangian-based constraint inference methods, including expert data. The environments include five Mujoco environments, modified to include constraints, which are primarily binary restrictions on the x-position of the agent. For more details, see Liu et al. (2023a).

Evaluation metrics Evaluating constrained RL is challenging due to the dual objectives of optimizing rewards and satisfying constraints, which are typically in conflict with one another. Hence, in our evaluation, we report two metrics, following Liu et al. (2023a), *feasible rewards* and *violation*

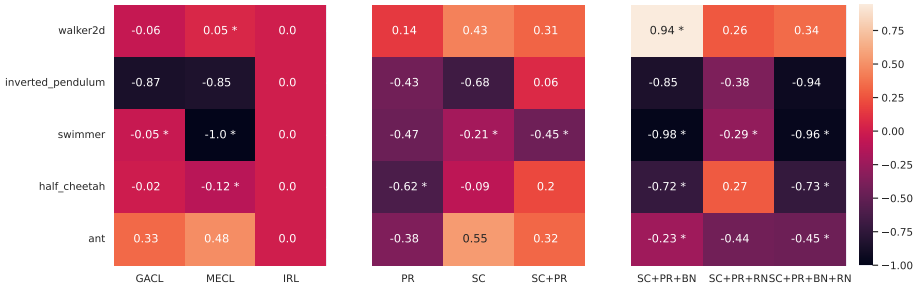


Figure 1: Heatmap of difference in feasible rewards over vanilla IRL, averaged over the last 100 testing episodes, normalized per environment (column-wise) by the range of values. Cases where the violation rate increased by more than 10% over vanilla IRL are indicated with a star. Positive values indicate a better result over vanilla IRL and negative values indicate a worse result.

rate. Feasible rewards are calculated as the total returns for an episode up to the point of the first constraint violation. The violation rate is the percentage of episodes with one or more constraint violations. Using these metrics allows us to compare the baselines along the same axis as prior work.

Prior methods for comparison We compare our results against two baselines from Liu et al. (2023a). The first is their implementation of MaxEnt ICRL (Malik et al., 2021) (denoted **MECL**), which is the canonical method for Lagrangian-based constraint inference that provides the primary baseline for our method. The second is their implementation of GAIL for constraint inference (denoted **GACL**), introduced as a baseline for **MECL** in Malik et al. (2021). This is a basic version of IRL for constraint inference, which uses the GAIL algorithm (Ho & Ermon, 2016) with an additive log constraint term. For our implementation of the maximum entropy IRL method, we use SAC as the forward RL algorithm, similar to Kim et al. (2023), whereas Malik et al. (2021) and Liu et al. (2023a) use PPO with an entropy regularization term. Though we have attempted to reimplement Malik et al. (2021) using SAC, our performance is low compared to Liu et al. (2023a). To compensate we instead include the reported results from Liu et al. (2023a) directly in the figures [2] and include full results of our SAC implementation in Appendix C.1.

Experimental Setup As mentioned, we use SAC for policy optimization. As the IRL algorithm, we utilize a version of maximum entropy IRL as implemented in Zeng et al. (2022). The learnt constraint function is parameterized as an MLP with a linear output activation clipped to the positive range. To provide the most fair assessment of our method, we do not tune hyperparameters specifically for each environment. Instead, we adopt the hyperparameters for SAC as used in Achiam (2018), except that we use automatic α-tuning (Haarnoja et al., 2018). All the hyperparameters for IRL were set to those used in Zeng et al. (2022). We train all variations in all environments for 5M environment steps across three seeds. Full configuration details are included in Appendix A.2.

5.1 IRL versus ICRL

First, we compare our IRL implementation to the baseline methods. In Figure 1 we present a summary of our findings across environments, showing a heatmap of the difference in feasible rewards versus vanilla IRL for various configurations, normalized by the within-environment range. The modifications proposed in Section 4.2 are labeled as **SC** for separate critics, **PR** for policy reset, **BN** for batch normalization, and **RN** for reward normalization.

Notably, vanilla IRL (**IRL**) performs quite favorably versus **MECL**. While **MECL** does perform slightly better on *Walker2d*, the effect is marginal and comes at the expense of a greater than 10%

[2] We take the results directly from the tables provided in Liu et al. (2023a) when available or make a best estimate based on figures when not. Notably, we double the reported performance on the *Half Cheetah* environment in all our comparisons, as it appears to us, based on the expert performance, that the authors used episodes of only 500 steps rather than the 1000 steps they report.

increase in the violation rate. Only in the *Ant* environment does **MECL** significantly outperform vanilla **IRL**, though we note that including the best modification for the *Ant* environment (**SC**, see Section 5.2) increases the performance of IRL to slightly exceed **MECL**. The base IRL method performs similarly to the previous IRL baseline method **GACL**, though it outperforms in *Inverted Pendulum* and underperforms in *Ant*. Including the best overall modifications across environments **SC+PR**, we see considerable improvement over **MECL** in *Half Cheetah*, *Inverted Pendulum* and *Walker2d*. Considering the best modifications in each environment, we see considerable improvement over **MECL** in all environments.

5.1.1 Sub-optimal Expert Trajectories

Previous work has found that constraint inference with a Lagrangian inner loop is more robust to suboptimal expert trajectories whereas simple IRL methods cannot learn effectively (Liu et al., 2023a). We conduct experiments on the suboptimal dataset for *Half Cheetah* provided in Liu et al. (2023a) to test whether this remains true with the added modifications. For these experiments, we use the best modification found in the *Half Cheetah* environment (see Section 5.2 for further discussion), which is separate critics with policy reset and reward normalization (**SC+PR+RN**).

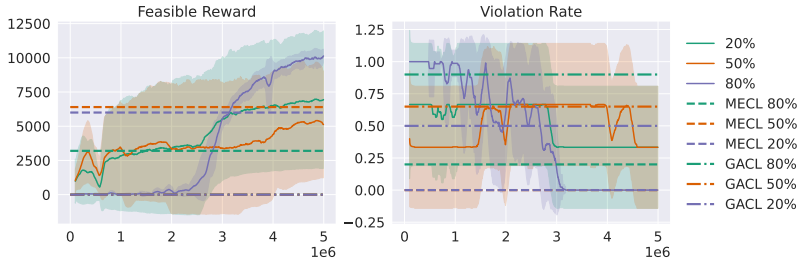


Figure 2: Training curves for *Half Cheetah* with suboptimal trajectories. The suboptimal datasets are constructed to contain 20%, 50% and 80% unsafe expert trajectories, i.e. expert trajectories with one or more constraint violations. Error bands show standard deviation across three seeds and a smoothing window of 200 testing episodes.

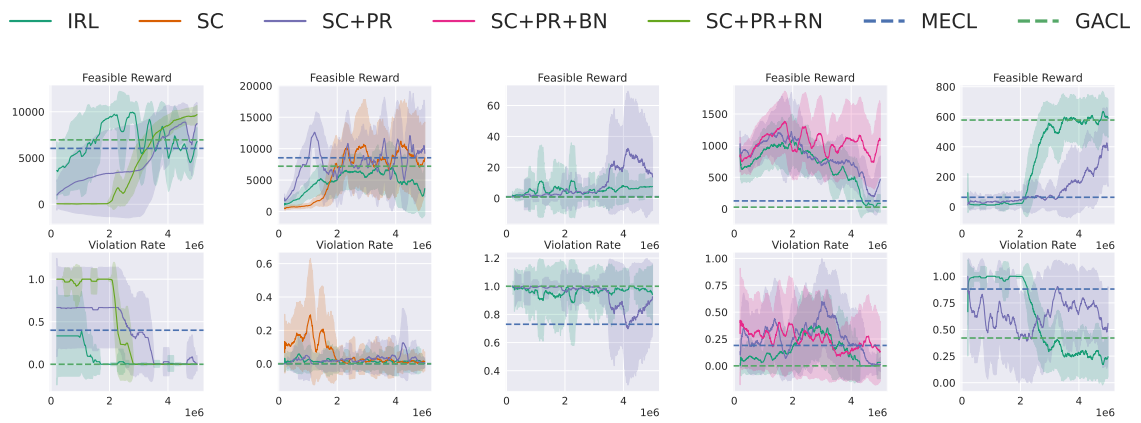
In all cases, our method performs much better than the simple IRL baseline **GACL**, which does not achieve any feasible rewards. Our method also outperforms **MECL** by a wide margin in two of the three variations and achieves similar performance in the third, though with notably high variance.

5.2 Impact of Modifications

Here, we more closely examine the impact of the proposed modifications (Section 4.2) on constraint inference with IRL. Overall, the impact of the modifications varies considerably across environments, with each environment achieving maximum rewards under different modifications. However, certain trends emerge from this analysis that indicate which modifications may be most useful in general.

Policy reset and separate critics Overall, the most consistent modification across environments is separate critics combined with last-layer policy reset (**SC+PR**). This modification increases feasible rewards over the base case IRL in all but one environment (*Swimmer*). Moreover, it is the best or second best modification by feasible reward improvement in three of the five environments (*Half Cheetah*, *Ant*, *Inverted Pendulum*). And in *Walker2d*, it still improves feasible reward performance versus IRL and decreases the violation rate. Even though it decreases performance in *Swimmer*, it still learns a reasonable policy as can be seen in the Figure 3.

Interestingly, neither the separate critics modification nor the policy reset modification alone produces good results, suggesting it is the interaction of these two modifications that are beneficial. For example, policy reset alone hurts performance in four of the five environments, but when combined with separate critics increases performance in four of the five environments. Notably, using separate



(a) Half Cheetah (b) Ant (c) Inv. Pendulum (d) Walker (e) Swimmer

Figure 3: Comparison of training curves of (1) Vanilla IRL (2) IRL with policy reset and separate critics, and (3) Best modifications on a per-environment basis. See Appendix C.2 for full results. Error bands show standard deviation across three seeds and a smoothing window of 200 test episodes.

critics allows us to reset the last layer of only the constraint critic and not the reward critic, which may contribute to the improved performance of policy reset when combined with separate critics.

Batch normalization and reward normalization Selecting the **SC+PR** modification as the base case, we then add batch normalization (**BN**) and reward normalization (**RN**) to test for further improvements. However, overall, we find these modifications are generally more harmful than beneficial. Including batch normalization, reward normalization, or both, on top of separate critics and policy reset hurt performance over basic IRL in a majority of environments. However, we note that in certain environments, such as *Walker2d* and *Half Cheetah*, adding these modifications results in the highest feasible rewards of all combinations, suggesting that these modifications may be helpful on an environment-specific basis.

6 Discussion and Conclusions

Overall, we find that Lagrangian-based methods for constraint inference do not confer significant performance benefits over vanilla IRL and the simple modification of separate critics and policy reset can improve the performance of IRL over previous Lagrangian-based ICRL methods. Moreover, certain combinations of additional simple regularization such as batch normalization and reward normalization can produce significantly better results in several environments, though the necessary modifications need to be tuned on an environment level.

Advantages of IRL over ICRL Besides some modest performance improvements, using IRL for constraint inference confers several benefits to the field. Primarily, this could facilitate applying IRL to several extensions of constraint inference that have not yet been fully explored, such as the offline constraint inference using offline IRL techniques (Yue et al., 2023). We also highlight that we did not have to perform any hyperparameter tuning to achieve decent results in all environments.

Limitations and Future Work While we show the feasibility of IRL for constraint inference as an alternative to ICRL, we note that there is substantial variance in the solutions produced by IRL as can be observed in Figures 2, 3 and 5. Reducing this variance should be a priority of future work. Moroever, ongoing work is still needed to improve the performance of constraint inference across more challenging environments such as *Inverted Pendulum*. An exciting extension for future work would be to explore various subdomains of IRL in the context of constraint inference, particularly the offline IRL case, and to apply these techniques to real datasets.

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. Spinning Up in Deep Reinforcement Learning. 2018.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Glen Chou, Dmitry Berenson, and Necmiye Ozay. Learning constraints from demonstrations. In *Algorithmic Foundations of Robotics XIII: Proceedings of the 13th Workshop on the Algorithmic Foundations of Robotics 13*, pp. 228–245. Springer, 2020.

Robert Dadashi, Adrien Ali Taiga, Nicolas Le Roux, Dale Schuurmans, and Marc G Bellemare. The value function polytope in reinforcement learning. In *International Conference on Machine Learning*, pp. 1486–1495. PMLR, 2019.

Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021.

Adam Gleave and Sam Toyer. A primer on maximum causal entropy inverse reinforcement learning. *arXiv preprint arXiv:2203.11409*, 2022.

Sehoon Ha, Peng Xu, Zhenyu Tan, Sergey Levine, and Jie Tan. Learning to walk in the real world with minimal human effort. *arXiv preprint arXiv:2002.08550*, 2020.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. pmlr, 2015.

Konwoo Kim, Gokul Swamy, Zuxin Liu, Ding Zhao, Sanjiban Choudhury, and Steven Z Wu. Learning shared safety constraints from multi-task demonstrations. *Advances in Neural Information Processing Systems*, 36, 2023.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=68n2s9ZJWF8.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

Guiliang Liu, Yudong Luo, Ashish Gaurav, Kasra Rezaee, and Pascal Poupart. Benchmarking constraint inference in inverse reinforcement learning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023a. URL https://openreview.net/pdf?id=vINj_Hv9szL.

Zuxin Liu, Zijian Guo, Haohong Lin, Yihang Yao, Jiacheng Zhu, Zhepeng Cen, Hanjiang Hu, Wenhao Yu, Tingnan Zhang, Jie Tan, et al. Datasets and benchmarks for offline safe reinforcement learning. *arXiv preprint arXiv:2306.09303*, 2023b.

Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. In *International Conference on Machine Learning*, pp. 23190–23211. PMLR, 2023.

Shehryar Malik, Usman Anwar, Alireza Aghasi, and Ali Ahmed. Inverse constrained reinforcement learning. In *International conference on machine learning*, pp. 7390–7399. PMLR, 2021.

Antoine Marot, Adrian Kelly, Matija Naglic, Vincent Barbesant, Jochen Cremer, Alexandru Stefanov, and Jan Viebahn. Perspectives on future power system control centers for energy transition. *Journal of Modern Power Systems and Clean Energy*, 10(2):328–344, 2022. doi: 10.35833/MPCE.2021.000673.

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.

Marcel Menner, Peter Worsnop, and Melanie N Zeilinger. Constrained inverse optimal control with application to a human manipulation task. *IEEE Transactions on Control Systems Technology*, 29(2):826–834, 2019.

Sobhan Miryoosefi, Kianté Brantley, Hal Daume III, Miro Dudik, and Robert E Schapire. Reinforcement learning with convex constraints. *Advances in Neural Information Processing Systems*, 32, 2019.

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

Andrew Y Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger, and Eric Liang. Autonomous inverted helicopter flight via reinforcement learning. In *Experimental robotics IX: The 9th international symposium on experimental robotics*, pp. 363–372. Springer, 2006.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *International conference on machine learning*, pp. 16828–16847. PMLR, 2022.

Stefan Schaal. Learning from demonstration. *Advances in neural information processing systems*, 9, 1996.

Dexter R. R. Scobee and S. Shankar Sastry. Maximum likelihood constraint inference for inverse reinforcement learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=BJliakStvH.

Gokul Swamy, Sanjiban Choudhury, J Andrew Bagnell, and Steven Wu. Of moments and matching: A game-theoretic framework for closing the imitation gap. In *International Conference on Machine Learning*, pp. 10022–10032. PMLR, 2021.

Jiayi Weng, Huayu Chen, Dong Yan, Kaichao You, Alexis Duburcq, Minghao Zhang, Yi Su, Hang Su, and Jun Zhu. Tianshou: A highly modularized deep reinforcement learning library. *Journal of Machine Learning Research*, 23(267):1–6, 2022. URL http://jmlr.org/papers/v23/21-1127.html.

S Yue, G Wang, W Shao, Z Zhang, S Lin, J Ren, and J Zhang. Clare: Conservative model-based reward learning for offline inverse reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2023.

Siliang Zeng, Chenliang Li, Alfredo Garcia, and Mingyi Hong. Maximum-likelihood inverse reinforcement learning with finite-time guarantees. *Advances in Neural Information Processing Systems*, 35:10122–10135, 2022.

Brian D Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.

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.

A Algorithm and Experiment Details

A.1 Algorithm

Here we provide an algorithm for IRL with separate critics for the constraint and reward functions when the reward signal r_t is observed in the environment and the constraint function R_ϕ is inferred.

Algorithm 1 Simple ICRL - Separate Critics

Initialize cost network parameters ϕ, policy parameters ψ and cost and reward critic parameters θ and α. Let M be the number of cost function updates and N be the number of policy updates.
for $m = 0, 1 \ldots,$ M **do**
  for $n = 0, \ldots, N$ **do**
    Collect experience (s_t, a_t, s_{t-1}, r_t) from nominal environment and add to buffer β
    Sample batch $\{(s_i, a_i, s_{i+1}, r_i)\}_{i=1}^{B} \sim \beta$.
    Compute learned cost for batch $c_i = R_\phi(s_i, a_i) \;\; \forall i \in B$
    Update policy parameters using critic $Q(s,a) = Q_\alpha(s,a) - Q_\theta(s,a)$
    Update cost critic parameters from c_i and reward critic parameters from r_i
  end for
  Update cost function using loss $\mathbf{E}_{\pi_E}[R_\phi(s,a)] - \mathbf{E}_{\pi_\psi}[R_\phi(s,a)]$
end for

A.2 Hyperparameters

All of our code is based on the Tianshou (Weng et al., 2022) and FSRL (Liu et al., 2023b) implementations of SAC and SAC-Lagrangian, respectively. Here we include all the hyperparameter configurations for our experiments. Any hyperparameters not listed here use the default hyperparameters in their respective libraries (Tianshou version 1.0.0 and FSRL version 0.1.0).

SAC Parameters	
τ	0.005
γ	0.99
α LR	0.0003
α optimizer	Adam
Critic hidden layers	[256, 256]
Critic optimizer	Adam
Critic learning rate	0.001
Actor hidden layers	[256, 256]
Actor learning rate	0.001
Actor optimizer	Adam
n-step returns	1

Constraint Learning Parameters	
Output clip range	[-20, 0]
Output activation	Linear
Hidden layers	[64, 64]
Learning rate	0.0001
Optimizer	Adam
Weight decay	0.001
Batch size	5000
Gradient steps per epoch	1
Regularization coefficient	0

Training Parameters	
Max epochs	1000
Batch size	128
Environment steps per epoch	5000
Gradient steps per environment step	1
Test episodes per epoch	5
Episodes between gradient updates	1
Buffer size	1000000

A.3 Computational Resources

Each run (consisting of three seeds) was trained on a node with a single GPU (varying GPU resources were used), 6 CPUs and 6GB of RAM per CPU. A single run across three seeds under this configuration took approximately 1.5 days to complete training.

B Proof of Theorem 4.1

We prove both claims individually.

B.1 Step 1: $\mathsf{Opt}_{\mathsf{ICRL}} = \mathsf{Opt}_{\mathsf{S-ICRL}}$

We begin by proving the first claim, that $\mathsf{Opt}_{\mathsf{ICRL}}(\Pi, \mathcal{F}_r, \mathcal{F}_c) = \mathsf{Opt}_{\mathsf{S-ICRL}}(\Pi, \mathcal{F}_r, \mathcal{F}_c)$ when $\mathcal{F}_c$ is a convex cone. Firstly, it is simple to show that $\mathsf{Opt}_{\mathsf{ICRL}}(\Pi, \mathcal{F}_r, \mathcal{F}_c) \geq \mathsf{Opt}_{\mathsf{S-ICRL}}(\Pi, \mathcal{F}_r, \mathcal{F}_c)$,

$$\begin{aligned}
\mathsf{Opt}_{\mathsf{ICRL}}(\Pi, \mathcal{F}_r, \mathcal{F}_c) &= \max_{c \in \mathcal{F}_c} \max_{\lambda \geq 0} \min_{\pi \in \Pi} J(\pi_E, r - \lambda c) - J(\pi, r - \lambda c) \\
&\geq \max_{c \in \mathcal{F}_c} \min_{\pi \in \Pi} J(\pi_E, r - 1 \cdot c) - J(\pi, r - 1 \cdot c) \\
&= \mathsf{Opt}_{\mathsf{S-ICRL}}(\Pi, \mathcal{F}_r, \mathcal{F}_c).
\end{aligned}$$

It remains to show that $\mathsf{Opt}_{\mathsf{ICRL}}(\Pi, \mathcal{F}_r, \mathcal{F}_c) \leq \mathsf{Opt}_{\mathsf{S-ICRL}}(\Pi, \mathcal{F}_r, \mathcal{F}_c)$. We have

$$\begin{aligned}
\mathsf{Opt}_{\mathsf{S-ICRL}}(\Pi, \mathcal{F}_r, \mathcal{F}_c) &= \max_{c \in \mathcal{F}_c} \min_{\pi \in \Pi} J(\pi_E, r - c) - J(\pi, r - c) \\
&\geq \max_{c \in \mathcal{F}_c} \min_{\pi \in \Pi} J(\pi_E, r - \lambda c) - J(\pi, r - \lambda c) \qquad \forall \lambda \geq 0 \\
&\geq \max_{c \in \mathcal{F}_c} \max_{\lambda \geq 0} \min_{\pi \in \Pi} J(\pi_E, r - \lambda c) - J(\pi, r - \lambda c) \\
&= \mathsf{Opt}_{\mathsf{ICRL}}(\Pi, \mathcal{F}_r, \mathcal{F}_c),
\end{aligned}$$

where the first inequality holds since $\lambda c \in \mathcal{F}_c$ whenever $c \in \mathcal{F}_c$ by the hypothesis that $\mathcal{F}_c$ is a convex cone. It follows that $\mathsf{Opt}_{\mathsf{ICRL}}(\Pi, \mathcal{F}_r, \mathcal{F}_c) = \mathsf{Opt}_{\mathsf{S-ICRL}}(\Pi, \mathcal{F}_r, \mathcal{F}_c)$. ■

B.2 Step 2: IRL Solves ICRL

We now prove the second claim. For any policy π, we define the *successor features* (Ng et al., 2000; Ziebart et al., 2008; Barreto et al., 2017) $\psi^\pi : \mathcal{S} \to \mathbb{R}^d$ according to

$$\psi^\pi(s) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t \phi(S_t, A_t) \,\middle|\, S_0 = s\right] \tag{7}$$

where $A_t \sim \pi(\cdot \mid S_t)$. Then, for any initial state distribution μ_0 and $r \in \mathcal{F}$, we have $J(\pi, r) = \mathbb{E}_{s_0 \sim \mu}[\langle \psi^\pi(s_0), w_r \rangle]$, where $r(x, a) = \langle \phi(x, a), w_r \rangle$ — this holds by the linearity of expectation.

Now, we define $\Psi = \{\mathbb{E}_{s_0 \sim \mu} \psi^\pi(s_0) : \pi \in \Pi\} \subset \mathbf{R}^d$. Note that $s \mapsto \psi_i^\pi(s)$ is equivalent to the value function under policy π for the reward function ϕ_i, for any $i \in [d]$. Thus, since Π is assumed to be compact and ϕ is bounded, Dadashi et al. (2019) shows that the set $\mathcal{V}_i = \{\psi_i^\pi : \pi \in \Pi\}$ is convex and compact for each $i \in [d]$. By continuity, $\overline{\mathcal{V}_i} = \{\mathbb{E}_{s_0 \in \mu} \psi_i^\pi(s_0) : \pi \in \Pi\}$ is convex compact. Therefore, it holds that as a product of convex and compact sets, $\Psi = \bigotimes_{i=1}^d \overline{\mathcal{V}_i}$ is convex and compact.

Next, suppose $(\pi^\star, c^\star - r)$ realizes $\mathsf{Opt}_{\mathsf{IRL}}(\Pi, \mathcal{F} - r)$. Since Ψ is compact and convex and $\mathcal{F}$ is convex, we apply Sion's minimax theorem and observe that, for $\psi_E := \mathbb{E}_{s_0 \sim \mu_0} \psi^{\pi_E}(s_0)$,

$$\begin{aligned}
J(\pi^\star, c^\star - r) - J(\pi_E, c^\star - r) &= \mathsf{Opt}_{\mathsf{IRL}}(\Pi, \mathcal{F} - r) \\
&= \min_{\pi \in \Pi} \sup_{c \in \mathcal{F}} J(\pi, c - r) - J(\pi_E, c - r) \\
&= \min_{\psi \in \Psi} \sup_{w_c \in \mathbb{R}^d} \langle \psi - \psi_E, w_c - w_r \rangle \\
&= \min_{\psi \in \Psi} \sup_{w_c \in \mathbb{R}^d} \langle \psi_E - \psi, w_r - w_c \rangle \\
&= \sup_{w_c \in \mathbb{R}^d} \min_{\psi \in \Psi} \langle \psi_E - \psi, w_r - w_c \rangle && \text{Sion's minimax theorem} \\
&= \sup_{c \in \mathcal{F}} \min_{\pi \in \Pi} J(\pi_E, r - c) - J(\pi, r - c) \\
&= \mathsf{Opt}_{\mathsf{S-ICRL}}(\Pi, \mathcal{F}, \mathcal{F}) \\
&= \mathsf{Opt}_{\mathsf{ICRL}}(\Pi, \mathcal{F}, \mathcal{F}) && \text{Step 1} \\
\therefore J(\pi_E, r - c^\star) - J(\pi^\star, r - c^\star) &= \mathsf{Opt}_{\mathsf{ICRL}}(\Pi, \mathcal{F}, \mathcal{F}),
\end{aligned}$$

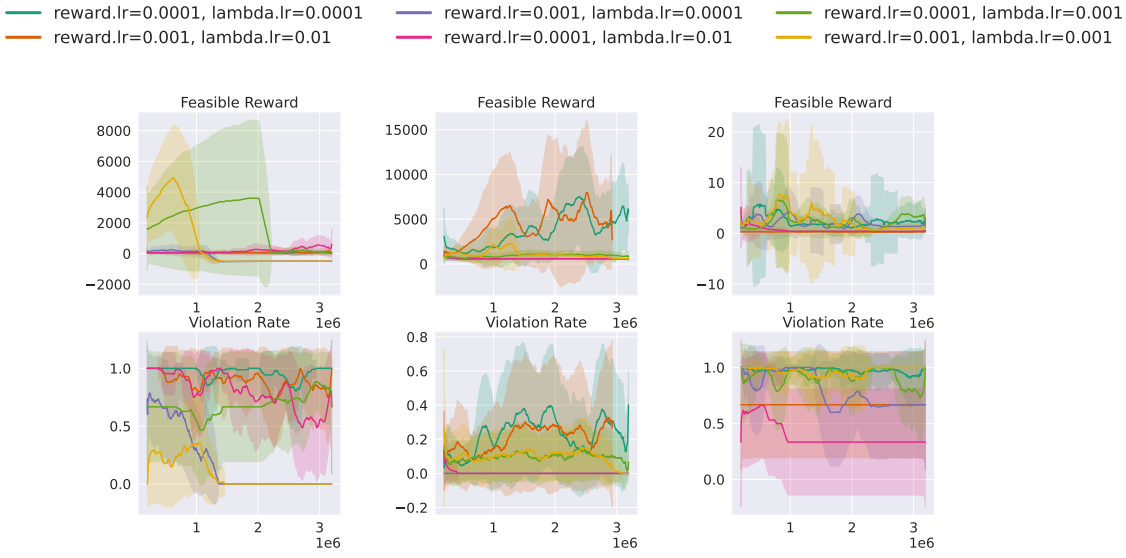
where the penultimate step holds because $\mathcal{F}$, as a vector space, is a convex cone. Thus, we conclude that solving the IRL problem over the reward class $\mathcal{F} - r$ and policy class Π solves the inverse constrained RL problem of interest. ■

C Additional Results

C.1 ICRL SAC Implementation Results

We attempted to reproduce the results of Liu et al. (2023a) using SAC for policy learning. We tested six settings for the Lagrangian and reward learning rates shown in Figure 4. All other hyperparameters were kept the same as Appendix A.2. Due to computational constraints, we ran each configuration for 3M timesteps. Notably, we had considerable difficulty reproducing the results of Liu et al. (2023a) using the SAC algorithm. While Kim et al. (2023) provided a SAC implementation of a similar algorithm, we note that their implementation added various additional components

to the algorithm to improve convergence, including in particular, interleaving reward learning with behavior cloning. We did not use these techniques here.



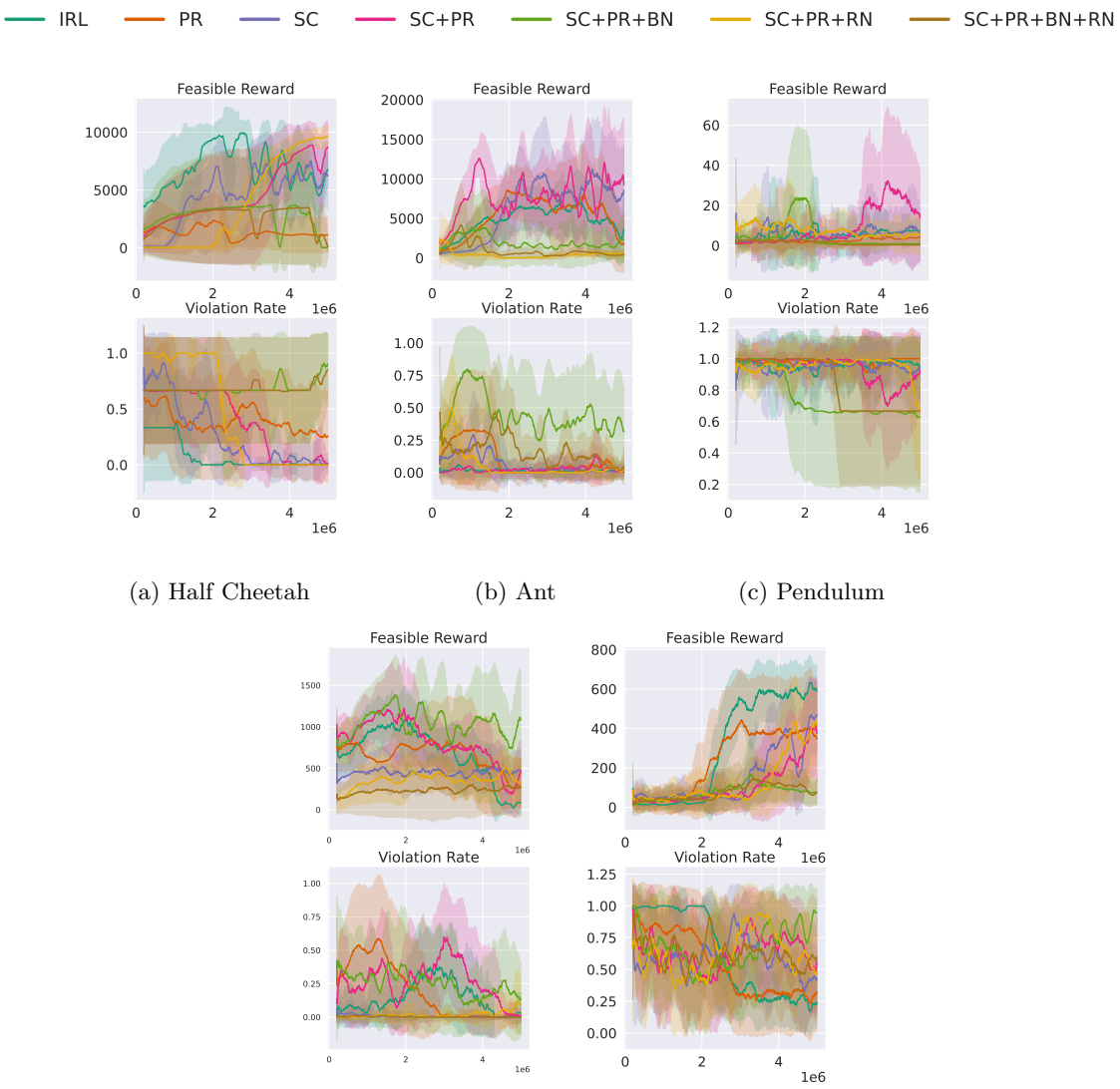
(a) Half Cheetah (b) Ant (c) Pendulum

(d) Walker (e) Swimmer

Figure 4: Training curves for feasible rewards (top) and violation rate (below) for ICRL with SAC with varying learning rates across the five test MuJoCo environments. Error bands show standard deviation across three seeds and smoothing window of 200 testing episodes.

C.2 Results of all modifications

Here we provide the results of all combinations of modifications on the five MuJoCo environments.



(a) Half Cheetah (b) Ant (c) Pendulum

(d) Walker (e) Swimmer

Figure 5: Training curves for feasible rewards (top) and violation rate (below) for IRL and all variations across the five test MuJoCo environments. Error bands show standard deviation across three seeds and a smoothing window of 200 testing episodes.