Implicit vs. Explicit Offline Inverse Reinforcement Learning: A Credit Assignment Perspective

Anonymous authors
Paper under double-blind review

Keywords: Inverse reinforcement learning, implicit reward models, model-based, credit assignment

Summary

Inverse reinforcement learning (IRL) alleviates the practical challenges of reward design by extracting reward functions from approximately rational demonstrators. Despite enjoying theoretical advantages, IRL has not received as much adoption as Behavior Cloning (BC) which does not require repeatedly solving a complex RL inner problem and is completely offline. Recently, a new class of IRL algorithms proposes an *implicit* reward function parameterization which enables directly updating the Q function without the RL inner loop or a reward model, making the algorithms more similar to BC, more memory efficient, and potentially easier to scale. In this paper, we aim to understand how implicit IRL differs from explicit IRL. We analyze their distinct learning dynamics, preference learning, and credit assignment mechanisms and suggest learning a dynamics model can overcome the dataset challenges of prior model-free approaches. We propose a new algorithm extending implicit IRL to the offline model-based setting to leverage suboptimal datasets without requiring online training. Using the D4RL MuJoCo benchmarks, we show that the proposed algorithm is competitive with explicit model-based offline IRL in matching expert performance with only a few demonstrations and enhances the performance of model-free baselines. Furthermore, our ablation experiments support the learning dynamics analysis of entangled preference learning and credit assignment mechanisms in implicit IRL and suggest a solution by prioritizing preference learning.

Contribution(s)

1. This paper presents a new algorithm for model-based offline inverse reinforcement learning with implicit reward models.
   Context: Implicit reward models have been proposed to improve and simplify IRL in the online and offline model-free settings (Garg et al., 2021; Sikchi et al., 2023; Al-Hafez et al., 2023). However, they have not been applied in the offline model-based setting and the associated design choices and learning dynamics are unknown.
2. This paper studies the difference between explicit and implicit IRL from a credit assignment perspective. The main observation is that implicit IRL entangles preference learning and credit assignment, and thus requires more delicate hyperparameter choices.
   Context: Prior work such as Sikchi et al. (2023); Al-Hafez et al. (2023) have studied implicit IRL from the distribution matching perspective. However, these works focus on the functional role of implicit IRL (i.e., why it works) but not its mechanistic role (i.e., how it works), such as its learning dynamics and how different algorithmic components interact.

Implicit vs. Explicit Offline Inverse Reinforcement Learning: A Credit Assignment Perspective

Anonymous authors
Paper under double-blind review

Abstract

Inverse reinforcement learning (IRL) alleviates the practical challenges of reward design by extracting reward functions from approximately rational demonstrators. Despite enjoying theoretical advantages, IRL has not received as much adoption as Behavior Cloning (BC) which does not require repeatedly solving a complex RL inner problem and is completely offline. Recently, a new class of IRL algorithms proposes an *implicit* reward function parameterization which enables directly updating the Q function without the RL inner loop or a reward model, making the algorithms more similar to BC, more memory efficient, and potentially easier to scale. In this paper, we aim to understand how implicit IRL differs from explicit IRL. We analyze their distinct learning dynamics, preference learning, and credit assignment mechanisms and suggest learning a dynamics model can overcome the dataset challenges of prior model-free approaches. We propose a new algorithm extending implicit IRL to the offline model-based setting to leverage suboptimal datasets without requiring online training. Using the D4RL MuJoCo benchmarks, we show that the proposed algorithm is competitive with explicit model-based offline IRL in matching expert performance with only a few demonstrations and enhances the performance of model-free baselines. Furthermore, our ablation experiments support the learning dynamics analysis of entangled preference learning and credit assignment mechanisms in implicit IRL and suggest a solution by prioritizing preference learning.

1 Introduction

Designing policies or reward functions that capture desired behavior is a very difficult task in practice. Failures of mis-specified reward functions are widely documented in the literature (Amodei et al., 2016; Knox et al., 2023; Gao et al., 2023). Inverse reinforcement learning (IRL; Ng et al., 2000) addresses this challenge by extracting reward functions from near-optimal or expert demonstrations. The extracted rewards can be used for not only training agent policies but also gaining insights into the demonstrated behavior for safety and scientific purposes (Bovenzi et al., 2024; Joselowitz et al., 2024; Ke et al., 2025; Muelling et al., 2014). Compared to its imitation learning counterpart Behavior Cloning (BC), IRL enjoys a number of theoretical advantages such as higher demonstration efficiency, robustness to distribution shift, and the ability to learn from suboptimal data (Spencer et al., 2021). However, in practice, BC has seen much wider adoption than IRL because of its simplicity and the ability to learn completely offline. How can we retain the advantages of IRL but make it more simple and scalable as BC?

The main contributor to IRL's complexity is its inherent bi-level structure as a result of modeling the expert as (approximately) reward optimal; the learner then has to search in the space of rewards, each time solving a RL problem so that the estimated policy can be compared with the expert. Recently, a new class of IRL algorithms starting from IQ-Learn proposes an alternative reward function parameterization by applying the inverse Bellman operator $\tilde{\mathcal{T}}^{\pi}$ on parameterized Q functions:

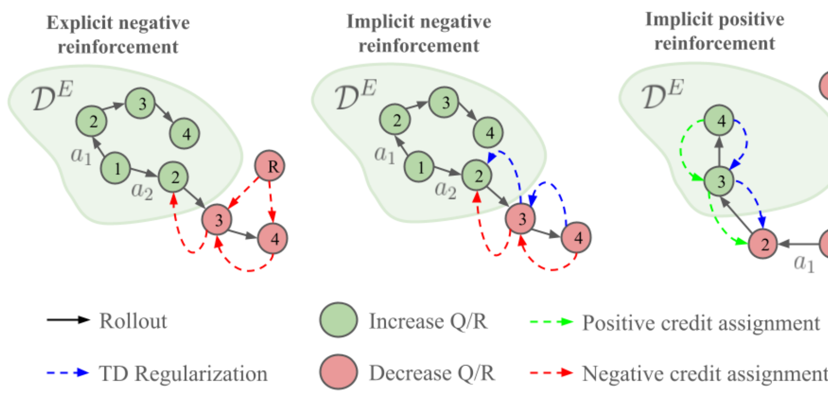

Figure 1: Illustration of preference learning and credit assignment mechanisms in explicit (left) and implicit (middle and right) IRL. At the first rollout step (circle labeled "1"), the learner chooses two actions a_1 and a_2 and simulates their effects forward. The action that takes or keeps the learner out of the expert distribution is negatively reinforced, via a decrease in Q value (i.e., negative preference learning) and backpropagation to preceding state-actions to assign negative credit. The action that takes or keeps the learner in distribution is positively reinforced. Explicit IRL decouples preference learning and credit assignment by training a separate reward model (circle labeled "R" in the left panel). Implicit IRL couples preference learning and credit assignment via TD regularization (blue arrow in middle and right panels).

$R(s,a) = \tilde{\mathcal{T}}^\pi[Q](s,a) := Q(s,a) - \gamma\mathbb{E}_{P(s'|s,a)}[V(s')]$ (Garg et al., 2021). This *implicit* reward parameterization allows bypassing the inner loop RL step because after each implicit reward update step, the optimal policy can be extracted either in closed form or easily from the Q function (e.g., by training an actor).

Theoretically, implicit IRL has been studied from the perspectives of reparameterization, distribution matching, and regularized behavior cloning (Garg et al., 2021; Sikchi et al., 2023; Al-Hafez et al., 2023). The last perspective highlights its connection with BC and potential for simplified implementation and better scalability by being "RL-free". However, these perspectives focus on studying implicit IRL functionally (i.e., why it works) but not mechanistically (i.e., how it works), which may be important for algorithmic design choices. We address the latter with a mechanistic comparison of explicit and implicit IRL's learning dynamics summarized in Fig. 1. We focus on *preference learning* (i.e., how the agent learns to prefer some state-action pairs and avoid others) and *credit assignment* (i.e., how such preferences are generalized and reinforced across the state-action space) and highlight the differences in these two mechanisms between explicit and implicit IRL, especially the entanglement of these mechanisms in the latter. The learning dynamics also suggests datasets with certain branching structure are desirable.

Practically, implicit IRL has mostly been studied in the online model-free setting. Recently, Ma et al. (2022); Sikchi et al. (2023) extended it to the offline model-free setting, allowing the agent to learn from additional large non-expert datasets. Yet, these methods struggled when few expert demonstrations exist in the offline dataset. Particularly, implicit IRL has, to our knowledge, not been studied in the offline model-based setting, where explicit IRL methods have demonstrated strong performance and robustness to dataset quality (Zeng et al., 2023; Chang et al., 2021). We propose a new implicit IRL algorithm in this setting based on the framework of Wei et al. (2023), which simultaneously trains the reward and an adversarial dynamics model. This simply requires replacing the explicit reward model with an implicit one, and the resulting algorithm becomes a straightforward extension of the offline model-free algorithm of Sikchi et al. (2023). Furthermore, learning a dynamics model enables the agent to generate dataset with desired branching structures from the mechanistic analysis and, for the analysis of implicit IRL algorithms, provides a new intervention mode of model rollout designs in addition to changing offline data mixtures.

Using the D4RL MuJoCo datasets, we show that the proposed algorithm is competitive with explicit model-based offline IRL algorithms while enhancing the performance of model-free baselines. Our ablation experiments validate the mechanistic explanations and contribute to better understanding of implicit IRL algorithms and design choices.

2 Background

Markov decision process We denote an *entropy-regularized* Markov decision process with $(\mathcal{S}, \mathcal{A}, P, R, d_0, \gamma, \alpha)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $P(s'|s,a) \in \Delta(\mathcal{S})$ the transition dynamics, $R(s,a) \in \mathbb{R}$ the reward function, $d_0 \in \Delta(\mathcal{S})$ the initial state distribution, $\gamma \in (0,1)$ the discount factor, $\alpha > 0$ the temperature parameter or entropy regularization weight. We denote the discounted occupancy measure as $\rho_P^\pi(s,a) = \mathbb{E}_{d_0,P,\pi}[\sum_{t=0}^\infty \gamma^t \Pr(s_t = s, a_t = a)]$ and the marginal state-action distribution as $d_P^\pi(s,a) = (1-\gamma)\rho_P^\pi(s,a)$. The optimal policy maximizes expected discounted cumulative rewards plus policy entropy: $\max_\pi \mathbb{E}_{d_0,P,\pi}[\sum_{t=0}^\infty \gamma^t (R(s_t,a_t) + \alpha\mathbb{H}[\pi(\cdot|s_t)])]$, where $\mathbb{H}[\pi(\cdot|s)] := -\sum_a \pi(a|s)\log\pi(a|s)$, and is known to have a softmax form (Haarnoja et al., 2018).

Inverse reinforcement learning The goal of IRL is to estimate the reward function optimized by an expert from a dataset of trajectories $\mathcal{D}^E = \{(s_{0:T}, a_{0:T})_{n=1}^N\}$ generated from expert interaction with the environment which we denote as d^E. It is well known in the literature that IRL aims to solve the following max-min optimization problem:

$$\max_{R\in\mathcal{R}} \min_{\pi\in\Pi} \Big(-\mathbb{E}_{(s,a)\sim d^\pi}[R(s,a)] - \alpha\mathbb{H}[\pi] \Big) + \mathbb{E}_{(s,a)\sim d^E}[R(s,a)] - \beta\psi(R)\,, \tag{1}$$

where $\mathbb{H}[\pi] := \mathbb{E}_{s\sim d^\pi}[\mathbb{H}[\pi(\cdot|s)]]$, $\psi : \mathbb{R}^{|\mathcal{R}|} \to \mathbb{R}$ is a convex regularizer on the reward function, and $\beta > 0$ is the regularization weight. This objective can be motivated from either the maximum entropy (Ziebart et al., 2008) or maximum likelihood perspective (Zeng et al., 2022).

Implicit reward models Recently, a family of IRL algorithms proposed an implicit reward parameterization using parameterized Q functions: $R(s,a) = \tilde{\mathcal{T}}^\pi[Q](s,a) := Q(s,a) - \gamma\mathbb{E}_{P(s'|s,a)}[V(s')]$, where $V(s) = \alpha\log\sum_a \exp(Q(s,a)/\alpha)$ (Garg et al., 2021). This method bypasses the inner loop RL problem because, after each update, the optimal policy can be found either easily or in closed form using $\pi(a|s) \propto \exp(Q(s,a)/\alpha)$. Depending on the chosen reward regularizer, the algorithm can be understood as minimizing different divergence measures with the expert via convex duality, which provides a principled way to incorporate non-expert offline datasets (Sikchi et al., 2023).

Model-based offline IRL In model-based offline IRL, we estimate a dynamics model $M(s'|s,a)$ from expert data and optionally a non-expert transition dataset to help with reward learning. A key concern is avoiding distribution shift caused by model inaccuracy. We adopt the framework of Wei et al. (2023) which proposed a Bayesian model for simultaneous estimation of reward and dynamics called RMIRL. Using a prior belief over dynamics model that enforces accuracy on the expert (and optionally transition) dataset, a maximum a posteriori estimate of the reward and dynamics is the solution to the following max-min optimization problem:

$$\max_{\substack{R\in\mathcal{R}\\ M\in\mathcal{M}}} \min_{\pi\in\Pi} \Big(-\mathbb{E}_{(s,a)\sim d_M^\pi}[R(s,a)] - \alpha\mathbb{H}[\pi] \Big) + \mathbb{E}_{(s,a,s')\sim d^E}[R(s,a) + \lambda\log M(s'|s,a)] - \beta\psi(R)\,, \tag{2}$$

with $\lambda \gg 0$. In words, the dynamics model is trained adversarially to the policy, which helps mitigate distribution shift. With explicitly parameterized reward, we can view the inner loop as solving a robust RL problem (Rigter et al., 2022).

3 Model-based offline IRL with implicit rewards

In this section, we propose an extension of RMIRL by replacing the explicit reward model in (2) with an implicit one, which we refer to as implicit-RMIRL (i-RMIRL). We set the regularizer as the squared implicit reward value with penalty weight $\beta > 0$ on a mixture distribution of the expert dataset and rollout data generated by the policy $\overline{\pi}$ and dynamics $\overline{M}$ from the previous iteration: $\mathcal{D}^{mix} := \mathcal{D}^{E} \bigcup \mathcal{D}^{\overline{\pi}}_{\overline{M}}$. From RMIRL's Bayesian view, the TD regularizer can be seen as a Gaussian prior over the reward magnitude.

Using a semi-gradient update rule, the critic and dynamics objective functions are the following (see derivation and practical algorithm in Appendix B):

$$\begin{aligned}
&\max_{Q \in \mathcal{Q}} \quad \underbrace{\mathbb{E}_{(s,a)\sim \mathcal{D}^E}[Q(s,a)] - \mathbb{E}_{(s,a)\sim d^{\pi}_{M}}[Q(s,a)]}_{\text{Preference learning}} - \underbrace{\beta \mathbb{E}_{(s,a,s')\sim \mathcal{D}^{mix}}\left[(Q(s,a) - \gamma V(s'))^2\right]}_{\text{Credit assignment}}, \\
&\min_{M \in \mathcal{M}} \quad \underbrace{\mathbb{E}_{(s,a)\sim d^{\pi}_{M}}[Q(s,a)]}_{\text{Adversarial training}} - \lambda \mathbb{E}_{(s,a,s')\sim \mathcal{D}^E}[\log M(s'|s,a)]\,.
\end{aligned} \tag{3}$$

The policy uses the SAC objective (Haarnoja et al., 2018) and is unchanged. The first two terms of the critic objective performs preference learning by contrasting the values of expert and learner state-action pairs, and the last term performs credit assignment by setting the values of all state-action pairs to that of their subsequent γ-discounted state using a temporal difference (TD) regularization. The critic objective has the same form as the RECOIL algorithm from Sikchi et al. (2023), which can be seen as minimizing χ^2 divergence with the expert distribution. The difference is the data mixture used for contrastive learning and TD regularization is augmented by model-based samples. Without the TD regularizer, the algorithm reduces to implicit behavior cloning (Florence et al., 2022).

4 Credit assignment analysis

The dominant theoretical view of implicit IRL is distribution matching with the expert dataset, which upon convergence should have matching performance with the expert. However, the empirical performance of these algorithms decreases substantially when only a few expert demonstrations exist in the offline dataset (Sikchi et al., 2023). In contrast, explicit model-based offline IRL upholds strong performance in the few-expert data setting (Zeng et al., 2023; Wei et al., 2023). In this section, we analyze this phenomenon from a credit assignment perspective to understand how model-based methods address this gap and shed light on algorithm design and dataset selection choices.

Learning dynamics & modes Our main argument, as summarized in Fig. 1, is that explicit and implicit IRL differ in their learning dynamics, which alternates between two main steps. In the *preference learning* step, the values of state-action pairs outside the expert distribution are decreased. In the *credit assignment* step, the values of state-action pairs are propagated upstream to preceding state-action pairs. The main difference between explicit and implicit IRL is that, whereas preference learning and credit assignment are decoupled and performed by two different networks in explicit IRL, these two steps are coupled and performed by a single network in implicit IRL. Furthermore, implicit IRL assigns credit not by accumulating future rewards but rather by directly setting the value of a state-action pair to (γ times) that of its subsequent state using TD regularization. Depending on the TD regularization strength, preference learning may be inhibited by credit assignment in implicit IRL (as we show later in the experiments), which does not occur in explicit IRL.

We also observe two possible credit assignment modes: a *negative reinforcement* mode and a *positive reinforcement* mode, which may occur in both explicit and implicit IRL. In the negative mode (Fig. 1 left and middle), credit assignment is based on identifying key states from which good actions keep the learner in distribution while bad actions take the learner out of distribution, and the learner policy learns to avoid bad actions. In the positive mode (Fig. 1 right), credit assignment relies

Table 1: D4RL MuJoCo benchmark performance. We use 10 expert trajectories for RMIRL and i-RMIRL and 20 expert trajectoreis for the rest. Each row reports the mean and standard deviation of the inter-quartile mean of normalized returns over 3 random seeds. We use pink to highlight settings that underperform substantially from expert level.

Environment	Dataset	BC	IBC	RECOIL	RMIRL	i-RMIRL (ours)
HalfCheetah	Medium-expert	40.02 ± 16.98	29.31 ± 9.26	104.14 ± 0.93	106.67 ± 1.05	103.07 ± 0.80
HalfCheetah	Medium-replay	40.02 ± 16.98	29.31 ± 9.26	77.80 ± 9.22	100.04 ± 1.49	94.50 ± 3.05
Hopper	Medium-expert	89.76 ± 10.85	56.78 ± 6.31	98.92 ± 3.21	96.82 ± 5.30	96.74 ± 4.02
Hopper	Medium-replay	89.76 ± 10.85	56.78 ± 6.31	81.28 ± 15.77	99.12 ± 0.30	100.18 ± 0.28
Walker2D	Medium-expert	99.46 ± 0.22	78.35 ± 18.42	100.27 ± 0.20	99.14 ± 0.33	99.98 ± 0.32
Walker2D	Medium-replay	99.46 ± 0.22	78.35 ± 18.42	99.95 ± 0.34	95.56 ± 8.15	99.23 ± 0.58

on states in the dataset self-correcting and returning to the expert distribution, so that in-distribution state values at later time steps are propagated to corrective actions in earlier time steps, even if these "good" actions and their associated state are not in the expert dataset. Intuitively, the positive mode requires exploration and is less likely in general because we would expect only expert policies to self-correct.

Design insights For offline IRL algorithms, the analysis suggests that having datasets exhibiting the branching structure in the negative reinforcement mode or the self-correcting structure in the positive reinforcement mode is crucial, albeit the latter is more challenging to acquire and verify. One empirical observation that supports this argument is that adding more expert trajectories to the offline dataset (without labeling them as experts) leads to better performance (Sikchi et al., 2023).

One way to overcome the offline dataset limitations is to train a dynamics model and rollout from expert states to generate negative reinforcement data. This is similar to recent expert-reset based methods to accelerate explicit IRL by avoiding solving a globally optimal policy (Swamy et al., 2023).

Finally, one can address objective inhibition in implicit IRL by prioritizing preference learning and reducing the TD regularization weight.

5 Experiments

We conduct experiments on the D4RL MuJoCo datasets to validate the proposed algorithm and observations in the credit assignment analysis. Specifically, we aim to answer the following questions: **1) Does i-RMIRL improve over model-free baselines and is it competitive with SOTA offline IRL algorithms? 2) How does coupled preference learning and credit assignment affect i-RMIRL and whether prioritizing preference learning improves performance? 3) How do positive and negative reinforcement modes affect i-RMIRL?**

For Q1, we use RMIRL (Wei et al., 2023) as the SOTA comparison. Our main goal is to improve upon RECOIL (Sikchi et al., 2023) which is model-free but able to learn from suboptimal data using the same value and policy objective as (3). We also include BC and IBC (Florence et al., 2022), which cannot learn from suboptimal offline data, as additional baselines for bottom-line performance. We replace the Langevin action sampler in IBC with a SAC style policy (Haarnoja et al., 2018) to unify implementations across all algorithms. RECOIL does not work well with the SAC loss and instead uses advantage weighted regression (Peng et al., 2019). We use 10 expert trajectories (10k steps) for RMIRL and i-RMIRL and 20 expert trajectories for the rest because they become much more unstable with 10 expert trajectories. For RECOIL, RMIRL, and i-RMIRL, we use a maximum of 1M steps from the suboptimal offline datasets. We discuss more implementation details in Appendix C.

To answer Q2 and Q3, we conduct the following ablations. First, we study the preference learning-credit assignment trade off by varying the TD regularization weight between $[0.5, 1, 2]$. Lower TD weight prioritizes preference learning. To understand the credit assignment modes, we vary the

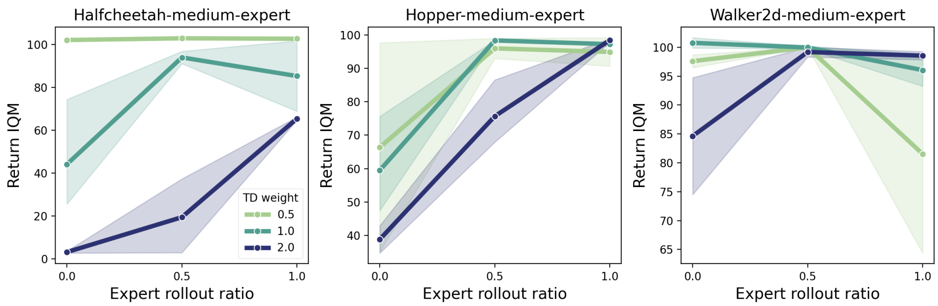

Figure 2: Effects of TD weight and expert rollout ratio on normalized return IQM. Higher TD weight generally hurts performance by inhibiting preference learning. Higher expert rollout ratio generally improves performance by generating more (branching) negative reinforcement data. Lower expert rollout ratio responds more negatively to high TD weight.

expert rollout ratio between $[0, 0.5, 1]$, which refers to the ratio of expert states to initiate model rollouts. Lower expert ratio prohibits negative reinforcement by reducing the amount of such data.

Overall performance We measure the overall performance of each algorithm using the interquartile means (Agarwal et al., 2021) of normalized returns of 30 evaluation runs averaged over 3 seeds. The results are listed in Table 1. All algorithms presented here use a single set of hyperparameters. The best overall configuration for i-RMIRL is 0.5 expert rollout ratio and 0.5 TD regularization weight. Although BC underperforms in halfcheetah and hopper, their learning curves in Appendix D suggests this is mainly due to performance decrements towards the end of training, likely due to overfitting. RECOIL only underperforms with much higher return variance on the medium-replay datasets of halfcheetah and hopper compared to RMIRL. i-RMIRL enhances RECOIL's performance particularly in these settings and reaches the performance of RMIRL.

Compute-wise, i-RMIRL has fewer parameters than RMIRL because it does not need a reward model, however, it has longer training time because the critic objective in (3) requires more evaluations of the Q networks for the preference learning loss (see Table 3 in Appendix C). This presents a memory-time efficiency trade off. In preliminary experiments, we found that replacing the double Q network with a single Q network achieves similar performance with significant time speed up in some environments. But we did not fully investigate this choice and leave it to future work.

Ablations Fig. 2 shows the performances of i-RMIRL for different TD regularization weights and expert rollout ratios on the medium-expert datasets. Higher TD weight substantially decreases performance with TD weight of 2 leading to nearly zero performance in halfcheetah with 0 expert ratio. This confirms our observation of the inhibition between preference learning and credit assignment, although the effect of TD inhibition is environment dependent. On the other hand, higher expert rollout ratio generally improves performance for all TD weights, where even having all rollouts initiated from expert data at ratio 1 can lead to expert performance, and low expert rollout ratio generally leads to decreased performance. The exception is walker with 0.5 TD weight. A likely reason for this is overfitting to expert state distribution. Still, this highlights the role of the negative reinforcement mode in the learning dynamics. However, this does not provide evidence for the existence or the effect of the positive reinforcement mode, which is more challenging to study and we leave to future work. Finally, TD weight and expert ratio interact with each other with lower expert ratio responding more negatively to high TD weight.

6 Conclusion

In this paper, we study implicit IRL from a credit assignment perspective. We first bring implicit IRL to the offline, model-based setting by extending a prior SOTA algorithm. Having access to a learned model allowed us to perform ablation experiments to validate our observations of the entangled preference learning and credit assignment mechanisms in implicit IRL. Our results show that prioritizing preference learning over credit assignment benefits implicit IRL in the offline, model-based setting, leading to matching performance with its explicit counterpart and the expert. Overall, while implicit and explicit IRL each have its own pros and cons, with the former being simpler and more memory efficient and the latter being less sensitive to hyperparameters, our results show that both can excel when chosen for the right domains and properly tuned.

A limitation is we did not investigate in depth the positive reinforcement mode of the IRL learning dynamics. However, its requirement on the offline dataset is much higher and may not be practical in realistic settings. We also did not investigate alternative ways to decouple preference learning and credit assignment, such as orthogonal gradient methods (Mao et al., 2024). We leave these to future work.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Firas Al-Hafez, Davide Tateo, Oleg Arenz, Guoping Zhao, and Jan Peters. Ls-iq: Implicit reward regularization for inverse reinforcement learning. *arXiv preprint arXiv:2303.00599*, 2023.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Inko Bovenzi, Adi Carmel, Michael Hu, Rebecca M Hurwitz, Fiona McBride, Leo Benac, José Roberto Tello Ayala, and Finale Doshi-Velez. Pruning the path to optimal care: Identifying systematically suboptimal medical decision-making with inverse reinforcement learning. *arXiv preprint arXiv:2411.05237*, 2024.

Alex J Chan and Mihaela van der Schaar. Scalable bayesian inverse reinforcement learning. *arXiv preprint arXiv:2102.06483*, 2021.

Jonathan Chang, Masatoshi Uehara, Dhruv Sreenivas, Rahul Kidambi, and Wen Sun. Mitigating covariate shift in imitation learning via offline data with partial coverage. *Advances in Neural Information Processing Systems*, 34:965–979, 2021.

Angelos Filos, Clare Lyle, Yarin Gal, Sergey Levine, Natasha Jaques, and Gregory Farquhar. Psiphi-learning: Reinforcement learning with demonstrations using successor features and inverse temporal difference learning. In *International Conference on Machine Learning*, pp. 3305–3317. PMLR, 2021.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on robot learning*, pp. 158–168. PMLR, 2022.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pp. 10835–10866. PMLR, 2023.

Divyansh Garg, Shuvam Chakraborty, Chris Cundy, Jiaming Song, and Stefano Ermon. Iq-learn: Inverse soft-q learning for imitation. *Advances in Neural Information Processing Systems*, 34: 4028–4039, 2021.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

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

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

Jared Joselowitz, Arjun Jagota, Satyapriya Krishna, and Sonali Parbhoo. Insights from the inverse: Reconstructing llm training goals through inverse rl. *arXiv preprint arXiv:2410.12491*, 2024.

Jingyang Ke, Feiyang Wu, Jiyi Wang, Jeffrey Markowitz, and Anqi Wu. Inverse reinforcement learning with switching rewards and history dependency for characterizing animal behaviors. *arXiv preprint arXiv:2501.12633*, 2025.

Edouard Klein, Matthieu Geist, Bilal Piot, and Olivier Pietquin. Inverse reinforcement learning through structured classification. *Advances in neural information processing systems*, 25, 2012.

W Bradley Knox, Alessandro Allievi, Holger Banzhaf, Felix Schmitt, and Peter Stone. Reward (mis) design for autonomous driving. *Artificial Intelligence*, 316:103829, 2023.

Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. *arXiv preprint arXiv:1809.02925*, 2018.

Donghun Lee, Srivatsan Srinivasan, and Finale Doshi-Velez. Truly batch apprenticeship learning with deep successor features. *arXiv preprint arXiv:1903.10077*, 2019.

Yecheng Ma, Andrew Shen, Dinesh Jayaraman, and Osbert Bastani. Versatile offline imitation from observations and examples via regularized state-occupancy matching. In *International Conference on Machine Learning*, pp. 14639–14663. PMLR, 2022.

Liyuan Mao, Haoran Xu, Weinan Zhang, and Xianyuan Zhan. Odice: Revealing the mystery of distribution correction estimation via orthogonal-gradient update. *arXiv preprint arXiv:2402.00348*, 2024.

Katharina Muelling, Abdeslam Boularias, Betty Mohler, Bernhard Schölkopf, and Jan Peters. Learning strategies in table tennis using inverse reinforcement learning. *Biological cybernetics*, 108: 603–619, 2014.

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

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Sid Reddy, Anca Dragan, and Sergey Levine. Where do you think you're going?: Inferring beliefs about dynamics from behavior. *Advances in Neural Information Processing Systems*, 31, 2018.

Marc Rigter, Bruno Lacerda, and Nick Hawes. Rambo-rl: Robust adversarial model-based offline reinforcement learning. *Advances in neural information processing systems*, 35:16082–16097, 2022.

Harshit Sikchi, Qinqing Zheng, Amy Zhang, and Scott Niekum. Dual rl: Unification and new methods for reinforcement and imitation learning. *arXiv preprint arXiv:2302.08560*, 2023.

Jonathan Spencer, Sanjiban Choudhury, Arun Venkatraman, Brian Ziebart, and J Andrew Bagnell. Feedback in imitation learning: The three regimes of covariate shift. *arXiv preprint arXiv:2102.02872*, 2021.

Gokul Swamy, David Wu, Sanjiban Choudhury, Drew Bagnell, and Steven Wu. Inverse reinforcement learning without reinforcement learning. In *International Conference on Machine Learning*, pp. 33299–33318. PMLR, 2023.

Ran Wei, Siliang Zeng, Chenliang Li, Alfredo Garcia, Anthony D McDonald, and Mingyi Hong. A bayesian approach to robust inverse reinforcement learning. In *Conference on Robot Learning*, pp. 2304–2322. PMLR, 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.

Siliang Zeng, Chenliang Li, Alfredo Garcia, and Mingyi Hong. When demonstrations meet generative world models: A maximum likelihood framework for offline inverse reinforcement learning. *Advances in Neural Information Processing Systems*, 36:65531–65565, 2023.

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.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Related work

In this section, we discuss prior works bridging IRL and BC. The common thread among all these works is directly learning the Q function to bypass the inner RL problem. However, the exact approaches differed. In the linear reward setting, Klein et al. (2012) proposed estimating the successor feature of the expert policy $\Psi^{\pi^E}(s,a)$ such that behavior cloning can be formulated as classification with a structured Q function: $Q_\theta(s,a) = \theta^\intercal \Psi^{\pi^E}(s,a)$ where $\theta \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$ is the linear reward weights. However, the Monte Carlo successor feature estimator only worked for simple environments. Lee et al. (2019) extended this idea to more complex continuous environments using deep RL based successor feature estimator and estimate the successor feature of the learner policy $\Psi^\pi(s,a)$ instead of the expert policy $\Psi^{\pi^E}(s,a)$. A similar algorithm was used by Filos et al. (2021) for opponent modeling in multi-agent RL.

Another line of work trains neural network parameterized Q functions using the behavior cloning loss: $\log\pi(a|s) = Q(s,a) - \log\sum_{\tilde{a}}\exp(Q(s,\tilde{a}))$ under constraints or regularizations on the Q function. Reddy et al. (2018) used this method to learn the "internal" dynamics of human users in assistive applications, where the constraint is the squared TD error with a known reward function averaged over uniformly sampled states and actions. Perhaps the first to identify the implicit reward parameterization, Chan & van der Schaar (2021) used the same behavior cloning loss with regularization on the squared implicit reward value averaged over the expert dataset. However, this direct parameterization approach only worked for discrete actions. IQ-learn (Garg et al., 2021) and follow up works (Al-Hafez et al., 2023; Sikchi et al., 2023) arguably extended this to the continuous action setting using actor-critic algorithms along with regularizing the implicit reward on non-expert data distribution which we showed is crucial. Sikchi et al. (2023) showed that implicit behavior cloning (Florence et al., 2022) which substantially enhanced the expressivity of BC policies and performance on robotics manipulation tasks using an energy-based model loss on expert-only data can be seen as an instance of this family of algorithms with a different regularization. The maximum likelihood and Bayesian formulations of (Zeng et al., 2022; Wei et al., 2023) can also be seen as attempts to formulate IRL with the BC loss function. However, due to the maximum entropy RL constraint in the formulation, the resulting algorithm resembled adversarial IRL (Ho & Ermon, 2016; Fu et al., 2017) and required solving the inner loop RL problem.

B Model-based offline IRL derivation

In this section, we derive the implicit-RMIRL formulation in (2) and (3).

B.1 Bayesian formulation

Let us denote the dataset with $\mathcal{D}^E = \{\tau_{i:N}\}, \tau = (s_{0:T}, a_{0:T}) \sim P^E(\tau)$. Starting from the Bayesian formulation of Wei et al. (2023), we denote the posterior over reward and dynamics as:

$$P(R,M|\mathcal{D}) \propto P(\mathcal{D}|R,M)P(R)P(M) = \prod_{i=1}^{N}\prod_{t=0}^{T}\pi(a_{i,t}|s_{i,t},R,M)P(R)P(M)\,, \tag{4}$$

where

$$P(R) \propto \exp(\beta\psi(R)), \quad P(M) \propto \exp\left(\lambda\sum_{i=1}^{N}\sum_{t=0}^{T-1}\log M(s_{i,t+1}|s_{i,t},a_{i,t})\right)\,. \tag{5}$$

Inverse scaling by the dataset size NT, the MAP estimator maximizes the follow objective:

$$\begin{aligned} L(R, M) &= \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=0}^{T}\Big[\log \pi(a_{i,t}|s_{i,t}, R, M) + \lambda \log M(s_{i,t+1}|s_{i,t}, a_{i,t}) + \beta\psi(R)\Big] \\ &\approx \mathbb{E}_{(s,a,s')\sim d^E}[\log \pi(a|s, R, M) + \lambda \log M(s'|s, a)] + \frac{\beta}{NT}\psi(R)\,, \end{aligned} \tag{6}$$

under the constraint that π is the optimal entropy-regularized policy w.r.t. R, M:

$$\text{s.t.} \quad \pi = \arg\max_{\pi\in\Pi} \mathbb{E}_{d_0,M,\pi}\left[\sum_{t=0}^{\infty}\gamma^t\left(R(s_t, a_t) + \alpha\mathbb{H}[\pi(\cdot|s_t)]\right)\right]. \tag{7}$$

We now expand the likelihood term:

$$\begin{aligned} &\mathbb{E}_{(s,a)\sim d^E}[\log \pi(a|s, R, M)] \\ &= (1-\gamma)\mathbb{E}_{P^E(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t \log \pi(a_t|s_t, R, M)\right] \\ &= (1-\gamma)\mathbb{E}_{P^E(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t\left(Q(s_t, a_t) - V(s_t)\right)\right] \\ &= (1-\gamma)\left\{\mathbb{E}_{P^E(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t(R(s_t, a_t) + \gamma\mathbb{E}_{M(s'|s_t,a_t)}[V(s')])\right] - \mathbb{E}_{P^E(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t V(s_t)\right]\right\} \\ &= (1-\gamma)\left\{\mathbb{E}_{P^E(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t, a_t)\right] + \sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{d^E(s_t,a_t)}[\gamma\mathbb{E}_{M(s'|s_t,a_t)}[V(s')]]\right. \\ &\quad \left. - \mathbb{E}_{d_0(s_0)}[V(s_0)] - \sum_{t=1}^{\infty}\gamma^t\mathbb{E}_{d^E(s_t)}[V(s_t)]\right\} \\ &= (1-\gamma)\left\{\mathbb{E}_{P^E(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t, a_t)\right] - \mathbb{E}_{d_0(s_0)}[V(s_0)]\right. \\ &\quad \left. + \sum_{t=0}^{\infty}\gamma^{t+1}\mathbb{E}_{d^E(s_t,a_t)}[\mathbb{E}_{M(s'|s_t,a_t)}[V(s')] - \mathbb{E}_{P(s''|s_t,a_t)}[V(s'')]]\right\} \\ &= \mathbb{E}_{d^E(s,a)}[R(s, a)] - \mathbb{E}_{d_M^\pi(s,a)}[R(s, a)] + \underbrace{\gamma\mathbb{E}_{d^E(s,a)}[\mathbb{E}_{M(s'|s,a)}[V(s')] - \mathbb{E}_{P(s''|s,a)}[V(s'')]]}_{\mathbf{T1}}. \end{aligned} \tag{8}$$

Wei et al. (2023); Zeng et al. (2023) showed that with a sufficiently accurate dynamics model M under the *expert* data distribution, $\mathbf{T1}$ can be ignored.

Thus, dropping $\mathbf{T1}$ and adding the regularizations and the policy entropy objective, we get the final RMIRL objective:

$$\max_{\substack{R\in\mathcal{R}\\ M\in\mathcal{M}}}\min_{\pi\in\Pi}\left(-\mathbb{E}_{(s,a)\sim d_M^\pi}[R(s, a)] - \alpha\mathbb{H}[\pi]\right) + \mathbb{E}_{(s,a,s')\sim d^E}[R(s, a) + \lambda \log M(s'|s, a)] - \tilde{\beta}\psi(R)\,, \tag{9}$$

where $\tilde{\beta} = \beta/(NT)$.

Algorithm 1 Implicit Robust Model-based IRL (i-RMIRL)

Require: Expert dataset $\mathcal{D}^E$, suboptimal dataset $\mathcal{D}^S$, dynamics model $M(s'|s,a)$, critic $Q(s,a)$, actor $\pi(a|s)$, expert rollout ratio κ, TD weight β, dynamics accuracy weight λ.

```
for k = 1 : K do
    Rollout dynamics model M and policy π from s ~ D_κ^{ES} and add to buffer
    Sample expert state-action pairs from D^E
    Sample learner state-action pairs from buffer
    Evaluate (13) and take an actor-critic gradient step
    if k mod 1000 = 0 then
        Sample (s, a, s') ~ D^{ES} for dynamics model training
        Evaluate (14) and take a few dynamics gradient steps
    end if
end for
```

B.2 Implicit reward parameterization

With the above formulation, it is easy to replace the explicit reward with an implicit one. Recall the implicit reward is defined as:

$$R(s,a) = \tilde{\mathcal{T}}^{\pi}[Q](s,a) := Q(s,a) - \gamma \mathbb{E}_{P(s'|s,a)}[V(s')] \,. \tag{10}$$

Furthermore, we define the regularizer as the squared reward values averaged over the dataset $\mathcal{D}^{mix} := \mathcal{D}^E \bigcup \mathcal{D}_{\overline{M}}^{\overline{\pi}}$, $\mathcal{D}_{\overline{M}}^{\overline{\pi}} \sim d_{\overline{M}}^{\overline{\pi}}$ is the rollout dataset generated by policy $\overline{\pi}$ and dynamics $\overline{M}$ from the previous iteration. The regularizer can be written as:

$$\psi(R) = \mathbb{E}_{(s,a,s') \sim \mathcal{D}^{mix}} \left[(Q(s,a) - \gamma V(s'))^2 \right] \,. \tag{11}$$

This can be understood as independent Gaussian priors over $R(s,a)$.

We can then write the implicit-RMIRL objective as:

$$\begin{aligned}\max_{\substack{Q \in \mathcal{Q} \\ M \in \mathcal{M}}} \min_{\pi \in \Pi} \quad & \Big(- \mathbb{E}_{(s,a,s') \sim d_M^{\pi}}[Q(s,a) - \gamma V(s')] - \alpha \mathbb{H}[\pi] \Big) + \mathbb{E}_{(s,a,s') \sim d^E}[Q(s,a) - \gamma V(s')] \\ & + \lambda \mathbb{E}_{(s,a,s') \sim d^E}[\log M(s'|s,a)] - \tilde{\beta} \mathbb{E}_{(s,a,s') \sim \mathcal{D}^{mix}} \left[(Q(s,a) - \gamma V(s'))^2 \right] \,.\end{aligned} \tag{12}$$

B.3 Practical algorithm

Our algorithm 1 follows the design of Wei et al. (2023) where we alternate between actor-critic training and dynamics model training. To construct d_M^{π}, we start model rollouts from a mixture of expert-suboptimal datasets with expert ratio κ. We denote the κ-mixed dataset as $\mathcal{D}_{\kappa}^{ES}$ and raw concatenation as $\mathcal{D}^{ES}$. We then take a semi-gradient approach to update the critic, this reduces (12) to:

$$\begin{aligned}\max_{Q \in \mathcal{Q}} \min_{\pi \in \Pi} \quad & \Big(- \mathbb{E}_{(s,a) \sim d_M^{\pi}}[Q(s,a)] - \alpha \mathbb{H}[\pi] \Big) + \mathbb{E}_{(s,a) \sim d^E}[Q(s,a)] \\ & - \tilde{\beta} \mathbb{E}_{(s,a,s') \sim \mathcal{D}^{mix}} \left[(Q(s,a) - \gamma V(s'))^2 \right] \,.\end{aligned} \tag{13}$$

Then every 1000 steps, we train the dynamics model adversarially while maximizing log likelihood on the combined expert-suboptimal dataset $\mathcal{D}^{ES}$ using the following objective:

$$\min_{M \in \mathcal{M}} \quad \frac{1}{\lambda} \mathbb{E}_{(s,a) \sim d_M^{\pi}}[Q(s,a)] - \mathbb{E}_{(s,a,s') \sim \mathcal{D}^{ES}}[\log M(s'|s,a)] \,. \tag{14}$$

We use the branched rollout method from Rigter et al. (2022) to approximate d_M^π for adversarial model training. We estimate the dynamics model gradient using REINFORCE with baseline:

$$\begin{aligned}\nabla_M \mathbb{E}_{(s,a)\sim d_M^\pi}[Q(s,a)] &\propto \nabla_M \mathbb{E}_{(s,a)\sim d_M^\pi}\left[\gamma \mathbb{E}_{s'\sim M(\cdot|s,a), a'\sim\pi(a'|s')}[Q(s',a')]\right] \\ &= \mathbb{E}_{(s,a,s')\sim d_M^\pi}\left[\left(\gamma \mathbb{E}_{a'\sim\pi(a'|s')}[Q(s',a')] - b(s,a)\right)\nabla_M \log M(s'|s,a)\right] ,\end{aligned} \tag{15}$$

where we set the baseline to $b(s,a) = Q(s,a)$.

B.4 Connection with implicit behavior cloning

Depending on the learner rollout distribution, the objective (12) can and often does contain an IBC term. To show this, let us assume the learner distribution can be expressed as a mixture of expert and suboptimal state distributions $d_M^\pi(s) = \delta d^E(s) + (1-\delta)d^S(s)$, where $\delta \in (0,1)$ is the mixing weight. This can be achieved by rolling out the model from expert states as described in the previous section. We can then write the semi-gradient contrastive loss as:

$$\begin{aligned}&\mathbb{E}_{(s,a)\sim d^E}[Q(s,a)] - \mathbb{E}_{(s,a)\sim d_M^\pi}[Q(s,a)] \\ &= \delta\mathbb{E}_{(s,a)\sim d^E}[Q(s,a)] - \delta\mathbb{E}_{s\sim d^E, a\sim\pi}[Q(s,a)] + (1-\delta)\mathbb{E}_{(s,a)\sim d^E}[Q(s,a)] - (1-\delta)\mathbb{E}_{s\sim d^S, a\sim\pi}[Q(s,a)] \\ &= \delta\underbrace{\mathbb{E}_{(s,a)\sim d^E}[\log\pi(a|s)]}_{\text{Behavior cloning}} + (1-\delta)\left(\mathbb{E}_{(s,a)\sim d^E}[Q(s,a)] - \mathbb{E}_{s\sim d^S, a\sim\pi}[Q(s,a)]\right).\end{aligned} \tag{16}$$

C Implementation details

We use our own implementations of all baseline algorithms, which as shown in Table 1 are tuned to expert level whenever possible. Our implementations largely follow prior works and released code bases. Policy, critic, and dynamics model architectures are shared between different algorithms. We discuss necessary details below.

Dynamics model pre-training Following Janner et al. (2019), we use ensemble MLP dynamics models with 3 hidden layers of 200 units each and SiLU activation. Each ensemble member predicts a Gaussian distribution over the difference between the next state and the current state. More details can be found in the appendix of Wei et al. (2023).

For pre-training, we sample 10 expert trajectories (a total of 10k steps) and combine with a maximum of 1M steps from the offline dataset in the D4RL MuJoCo suit as the dynamics model training data. Medium-replay datasets are on the order of 200k, which is much less than 1M. Wei et al. (2023) did not add expert trajectories to the dynamics model training dataset, which is likely the reason why their reported performances on the medium-replay datasets are not as good as ours. Our results show that explicit model-based offline IRL performance could be much stronger than previously known.

Policy and critic We use the standard TanhNormal MLP policy with 3 hidden layers of 256 units each and SiLU activation. RMIRL uses automatic entropy tuning. For all other algorithms, the entropy coefficient α is fixed to 0.1. For RMIRL, i-RMIRL, and RECOIL, we use double Q network by default. IBC uses a single Q network. All Q networks have the same architecture as the policy.

Gradient penalty We use the IBC gradient penalty to improve reward or critic training stability. Applying gradient penalty to the critic in the case of i-RMIRL is more tricky than applying it to the reward. Gradient norm target that's too small hurts performance in certain environments. In those cases, we remove the gradient penalty.

Terminal state handling Several works have suggested properly handling terminal states can lead to better performance in imitation learning (Kostrikov et al., 2018; Al-Hafez et al., 2023). We found that using terminal state flags hurts stability for RMIRL and i-RMIRL. This is likely because in the offline setting model error causes incorrect terminal flags.

IBC As mentioned in the main text, we train a policy to sample from the energy-based model parameterized by the critic for IBC rather than using Langevin dynamics to sample actions. The policy is trained simultaneously with the critic as in standard actor-critic training. The critic is trained using the info-NCE loss function. For the negative samples, we draw 1 sample from the policy and 3 samples uniformly at random from within the action bounds.

RMIRL Our RMIRL implementation makes several additional modifications to the original implementation to enhance stability and performance. First, we apply the expert rollout ratio idea to RMIRL and sample half of the rollout batch from expert data and the other half from offline data. Second, instead of generating a separate rollout batch for each reward update step and then discarding the data and updating the reward at a much slower scale of every 1000 policy steps, we follow standard adversarial IRL algorithm designs and update the reward model using data from the policy training buffer at a faster scale of every 10 policy steps (the reason for not updating every 1 policy step is to reduce training time). For adversarial dynamics model training, rather than letting the model rollout for all specified steps, we terminate rollouts when the max observation norm exceeds a threshold (30 of the normalized observation scale) and refill with new samples from the expert-offline buffer mixture to maintain a constant batch size. The latter two modifications prevent loss blow-ups in the middle of training and performance collapse at the end of training.

RECOIL Our RECOIL implementation adopts two implementation tricks from the official implementation. First, we train the policy using advantage-weighted regression (AWR) (Peng et al., 2019) rather than the SAC loss because the latter did not work well in our initial experiments. The AWR objective is defined as:

$$\max_{\pi \in \Pi} \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[e^{(Q(s,a)-V(s))/\alpha}\log\pi(a|s)\right], \quad V(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[Q(s,a)]\,, \tag{17}$$

where the value baseline doesn't include the entropy bonus. We also used a l2 loss to regress expert state-action pairs onto a target value. Different from the original implementation, for the TD regularizer loss, we did not train a separate value network using implicit maximization. Rather, we approximate state values by sampling actions from the policy as in standard actor-critic. We also added a small action noise of 0.1 to the AWR loss to prevent overfitting dataset actions.

i-RMIRL Our i-RMIRL implementation adapts the implementation of RMIRL and RECOIL. Different from RECOIL, we did not use AWR policy loss or the l2 loss to regress expert state-action pairs onto a target value. AWR could potentially make i-RMIRL even more stable than the SAC loss, however, our goal here is to make the implementation consistent with RMIRL. Following Al-Hafez et al. (2023), we clip the critic value to a range, which we set to $[-1000, 1000]$. To stabilize training, we use cosine annealing of the critic learning rate from $3e-4$ to $1e-5$ on top of gradient penalty. In preliminary experiments, we found that gradient penalty and double Q network were not needed for halfcheetah and hopper. However, we did not systematically investigate the effects of these hyperparameters. The best hyperparameters across all environments from our searches are listed in Table 2.

Computational efficiency The number of parameters in total and per module for RMIRL and i-RMIRL are listed in Table 3. i-RMIRL uses fewer parameters because it does not have a reward model. However, the approximate training times for RMIRL and i-RMIRL are 2.97 hours and 3.75 hours respectively on a MacBook Pro M3 with 18 GB unified memory. This is because evaluating the loss in (13) requires more queries of the double Q network than RMIRL due to the contrastive terms and the gradient penalty is computed on the double Q network rather than a single reward

Table 2: Best hyperparameters for i-RMIRL.

	Hyparameter	i-RMIRL
Rollout	model rollout expert ratio (κ)	0.5
	model rollout batch size	5000
	model rollout steps	20
	model rollout every steps	250
	model retain epochs	5
Actor-critic	actor learning rate	3e-4
	critic learning rate	3e-4
	min critic learning rate	1e-5
	critic warmup epochs	300
	discount factor (γ)	0.99
	soft target update parameter (τ)	5e-3
	temperature (α)	0.1
	TD regularization (β)	0.5
	batch size	256
	training epochs	1000
	steps per epoch	1000
Dynamics	# model networks	7
	# elites	5
	adv. rollout batch size	256
	adv. loss weighting ($1/\lambda$)	0.05
	learning rate	1e-4
	adv. update steps	50

model. As mentioned before, removing gradient penalty and replacing the double Q network with a single Q network can significantly speed up training. However, we did not fully validate the stability of this setup in all environments.

Table 3: Model parameter counts in the halfcheetah environment.

Module	RMIRL	i-RMIRL
Actor	139,276	139,276
Critic	275,970	275,970
Dynamics	460,204	460,204
Reward	137,985	-
Total	1,013,435	875,450

D Additional results

Fig. 3 shows the normalized return IQM over the number of policy update steps for all algorithms except IBC. We train BC for 200k steps, IBC and RECOIL for 500k steps, and others for 1000k steps.

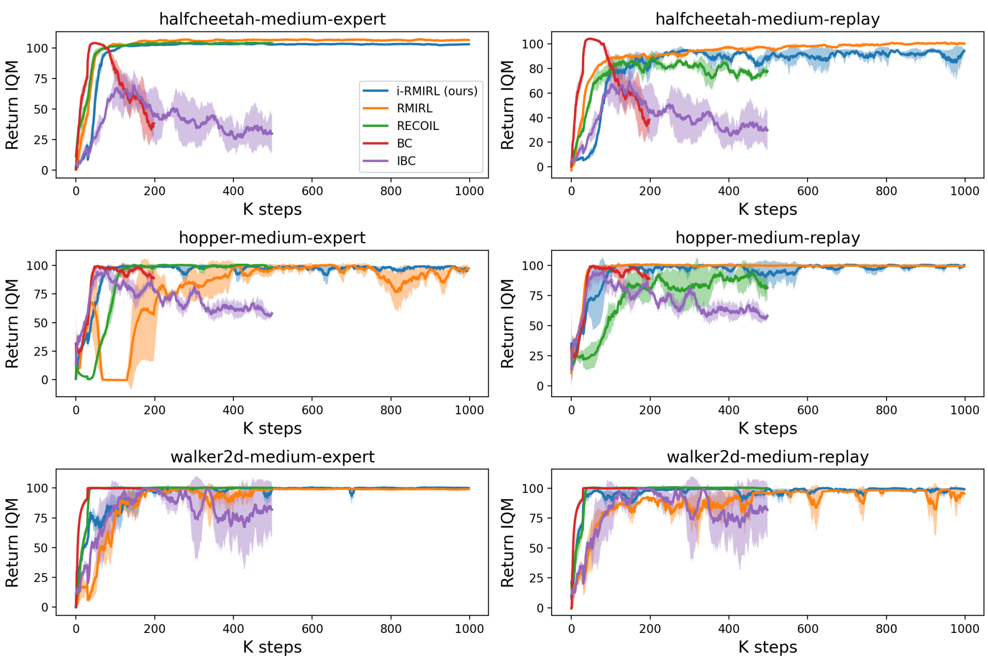

Figure 3: Normalized return IQM vs. the number of thousand (K) policy update steps.