Aux-AIRL: End-to-End Self-Supervised Reward Learning for Extrapolating beyond Suboptimal Demonstrations

Yuchen Cui [*1] **Bo Liu** [*1] **Akanksha Saran** [1] **Stephen Giguere** [1] **Peter Stone** [1 2] **Scott Niekum** [1]

Abstract

Real-world human demonstrations are often suboptimal. How to extrapolate beyond suboptimal demonstration is an important open research question. In this ongoing work, we analyze the success of a previous state-of-the-art self-supervised reward learning method that requires four sequential optimization steps, and propose a simple end-to-end imitation learning method Aux-ARIL that extrapolates from suboptimal demonstrations without requiring multiple optimization steps.

1. Introduction

The advent of autonomous agents in our homes and workplaces is contingent on their ability to adapt in novel, varied, and dynamic environments and learn new tasks from end-users in these environments. A natural approach is for end-users to teach learning agents by showing demonstrations of how a task should be performed, which is known as learning from demonstration (LfD) or imitation learning (Argall et al., 2009). Typically, LfD algorithms assume users provide near optimal demonstrations, which often does not hold true as novice end-users can provide suboptimal demonstrations.

Instead of discarding suboptimal demonstrations, a recent suite of self-supervised methods (Brown et al., 2019a;b; Chen et al., 2020) have shown how to leverage this suboptimal data to learn reward functions that can induce behaviors extrapolating beyond the demonstrator's performance. Brown et al. (2019b) propose disturbance-ranked reward extrapolation (D-REX), which bootstraps off suboptimal demonstrations to synthesize noise-injected trajectory rollouts. These synthesized trajectories with varying levels of noise are then used to train an idealized reward function, which is then used with reinforcement learning (RL) (Sutton et al., 1998) to learn the final policy. By improving upon this approach, Chen et al. (2020) propose self-supervised reward regression (SSRR), which leverages adversarial inverse reinforcement learning (AIRL) (Fu et al., 2017) to generate synthetic demonstrations and then learn a reward function that is aware of the amount of noise injected to the policy during self-supervision. SSRR first fits the noise-performance curve with a sigmoid function, and then regresses a reward function to the resultant noise-performance curve. They show that training a RL policy on this regressed noise-aware reward function outperforms D-REX in three Mujoco environments (Todorov et al., 2012).

In this work, we perform an in-depth study on the mechanisms of SSRR. While we observe that most steps of SSRR are essential to its success, we also find that the criterion used to approach reward regression in this work may not be the most optimal. Chen et al. (2020) demonstrate empirically that fitting a sigmoid function to the noise-performance curve generates a regression target for reward functions that will induce better-than-demonstrator agents, outperforming D-REX. However, we learn that the sigmoid is not the only function that can give a high extrapolation performance and at the same time fitting the function parameters to the AIRL reward is not a necessary step. We hypothesize what is critical for extrapolation is to ensure is a steep drop of reward estimation when noise is injected. We propose to directly apply an auxiliary loss on AIRL (Fu et al., 2017), which enforces trajectories without noise to have higher rewards than trajectories with noise. We show that this simple auxiliary objective of creating a separation between the predicted performance of policies with and without noise, can extrapolate beyond suboptimal demonstrations in a more efficient manner— replacing the multi-stage training process of SSRR with one single step.

2. Background

In this section we introduce the problem setting and give an overview of three related works.

2.1. Problem Setup

We consider sequential decision making problems modeled as Markov Decision Processes (MDPs). An MDP is given

[*]Equal contribution [1]Department of Computer Science, University of Texas at Austin, Austin, Texas, USA [2]Sony AI, USA. Correspondence to: Yuchen Cui <yuchencui@utexas.edu>, Bo Liu <bliu@cs.utexas.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

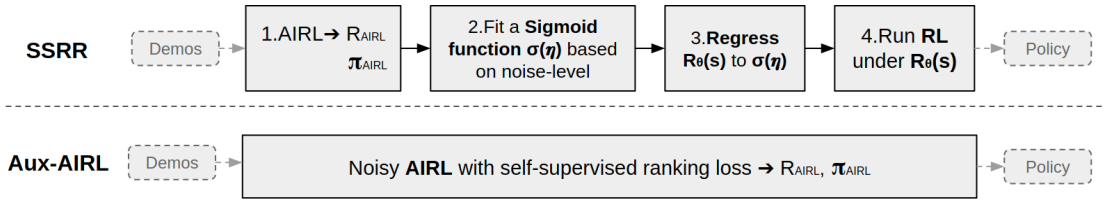


Figure 1. Comparison between SSRR and Aux-AIRL: SSRR requires four optimization steps, two involves interacting with the environment; Aux-AIRL leverages an auxiliary loss function and is trained end-to-end.

by the tuple $\langle S, A, T, R, \gamma \rangle$, where: S is a set of states; A is a set of actions; $T : S \times A \times S \to [0, 1]$ is the transition dynamics; $R : S \to \mathbb{R}$ is a reward function; $\gamma \in [0, 1)$ is the discount factor. A policy, π maps from states to a probability distribution over actions. The expected value of a policy π under reward function R is the expected return of that policy and is denoted as $V_R^\pi = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi]$.

In the problem of learning from demonstrations (LfD), an MDP with out reward function is given, together with a set of demonstration trajectories $\mathcal{D} = \{\tau_1, \tau_2, \ldots, \tau_M\}$, where $\tau_i = (s_0^{(i)}, a_0^{(i)}, s_1^{(i)}, \ldots)$ stores the consecutive states and actions the demonstrator encounter. The objective is to find a policy π that performs well under some unknown reward R^* that is only observable to the demonstrator. As R^* is hidden from the learner, typical LfD methods imitate what the demonstrator does on states from $\mathcal{D}$. When $\mathcal{D}$ is suboptimal, it is in general impossible to extrapolate beyond the demonstrator without any assumption about the data.

2.2. AIRL

Adversarial inverse reinforcement learning (AIRL) (Fu et al., 2017) approaches imitation learning from optimal demonstrations as part of a generative adversarial framework. AIRL consists of a generator in the form of a policy optimized using RL methods, and a discriminator that serves as the reward function. The policy optimization follows the standard RL technique. Let $\pi(a|s)$ denote the RL policy and $f_\theta(s, a)$ be the reward function parameterized by θ. The discriminator D is given by $D_\theta(s, a) = \frac{e^{f_\theta(s,a)}}{e^{f_\theta(s,a)} + \pi(a|s)}$. D_θ is trained with a loss function to differentiate the demonstration and the behavior generated from π:

$$L_D(\theta) = \mathbb{E}_{(s,a)\sim\pi}[\log(1 - D_\theta(s, a))] - \mathbb{E}_{(s,a)\sim\mathcal{D}}[\log D_\theta(s, a)] \tag{1}$$

The policy $\pi(a|s)$ is alternatively trained along with D_θ to imitate the expert by maximizing the psuedo-reward function given by $\hat{R} = f_\theta(s, a)$.

2.3. D-REX

Disturbance-based reward extrapolation (D-REX) (Brown et al., 2019a) is a deep inverse reinforcement learning method leveraging ranking-based reward learning techniques to extrapolate performance by automatically generating ranked trajectories. D-REX first learns a policy π_{BC} via behavioral cloning (BC) (Bain & Sommut, 1999; Ross et al., 2011; Daftry et al., 2016) from given demonstrations. Next, it adds noise to π_{BC} to create noisy trajectories, as given by Equation 2, where U is the uniform distribution and η is the proportion of noise injection.

$$\pi^\eta(a|s) = \eta U(a) + (1 - \eta)\pi_{BC}(a|s) \tag{2}$$

Finally, D-REX learns a reward function via supervised learning over trajectory pairs using the pairwise ranking loss according to Luce-Shepard rule (Luce, 2012).

$$L(\theta) = -\frac{1}{|\mathcal{P}|} \sum_{(i,j)\in\mathcal{P}} \log \frac{e^{\sum_{s\in\tau_i} R_\theta(s)}}{e^{\sum_{s\in\tau_i} R_\theta(s)} + e^{\sum_{s\in\tau_j} R_\theta(s)}}. \tag{3}$$

The Luce-Shepard rule enforces that a trajectory with higher noise level to be ranked lower. However, it does not account for the amount of noise injection during reward learning or the extent of suboptimality but rather just optimizes for an ordinal relationship among suboptimal demonstrations.

2.4. SSRR

Chen et al. (2020) argue that the Luce-Shepard rule D-REX uses results in a counterproductive inductive bias. They show that a BC-based rollout generation for synthesizing ranked, suboptimal demonstrations results in a reward function that is brittle against covariate shift. Instead of BC, they use AIRL (Fu et al., 2017) to obtain an initial estimate of a reward function and a policy. They overcome the limitations of D-REX (Brown et al., 2019b) by specifically characterizing the relationship between the amount of noise injection and policy performance (obtained via the reward function of AIRL). The SSRR pipeline consists of four distinct optimizations (shown in Figure 1): (1) training a policy π_{AIRL} and reward function R_{AIRL} (discriminator) via noisy AIRL and generating data in a self-supervised manner. Noisy AIRL injects noise into the AIRL generator π_{AIRL} to expose the reward function to a broader state space as visited by noisy demonstrations; (2) characterizing noise-performance by fitting a sigmoid function $\sigma(\eta)$, where performance is computed using R_{AIRL}; (3) learning a reward function $R_\theta(s)$

via regression to the sigmoid function from (2); and (4) training a reinforcement learning agent using $R_\theta(s)$. Chen et al. (2020) find that the noise-performance relationship is well-characterized by a four parameter sigmoid function:

$$\sigma(\eta) = \frac{c}{1 + \exp(-k(\eta - x_0))} + y_0. \quad (4)$$

The accuracy of the sigmoid fit is measured by finding a correlation between the ground truth reward and the learned reward function. Chen et al. (2020) find higher correlation of their learned reward with ground truth ($M = 0.996, SD = 0.004$) compared to that learned via D-REX ($M = 0.812, SD = 0.153$).

3. Methodology

3.1. In-depth study of SSRR

SSRR consists of a pipeline of optimization steps. To get a better understanding of how the different design choices of SSRR contribute to its performance gain, we conducted an in-depth study on the different modules of the SSRR pipeline. Chen et al. (2020) empirically demonstrated SSRR's effectiveness for extrapolating suboptimal demonstrations. However, it is unclear why a sigmoid function is the best choice as the regression target. We conducted experiments in two continuous control tasks environments in Mujoco (Todorov et al., 2012) and designed them to specifically answer the following questions: **Q1.** Is it important to fit a noise-based function as the target for regressing the reward? Can we directly use R_{AIRL} as the regression target? **Q2.** Can we replace the sigmoid function with different functional forms? **Q3.** Is it necessary to fit function parameters to R_{AIRL}?

Q1. SSRR leverages R_{AIRL} for fitting the noise-performance curve $\sigma(\eta)$ and then regresses a reward $R_\theta(s)$ to the curve using a supervised loss on sub-trajectory-level returns. The supervised training step itself may reshape the reward function. To understand whether it is important to regress to a noise-level-based target function, we conducted experiments skipping step (2) of SSRR and directly train a reward function $R_{\text{prop.}}$ that regresses to the returns from R_{AIRL}, such that the learned reward function is proportional to R_{AIRL} at the return level. The resulting RL agents' performances in Hopper-V3 did not extrapolate beyond demonstrations while the performances in HalfCheetah-V3 was worse than π_{AIRL} (see Table 4 in Appendix). The results of this experiment show that using a noise-level based reward target is important for extrapolation and we hypothesize that it is because the learned AIRL reward from sub-optimal demonstrations is too noisy, especially when evaluated on unseen states. A T-SNE visualization of R_{AIRL} based on state samples is shown in the Appendix (Figure 2). The supervised learning technique does reshaped the reward distribution of $R_{\text{propo.}}$ from R_{AIRL}, however, it is "extrapolating" in a wrong direction.

Q2. To test whether using a sigmoid function is necessary to get the performance boost, we tested the SSRR pipeline with different forms of the target regression function:

- Linear: $l(\eta) = a\eta + b$
- Quadratic: $q(\eta) = a\eta^2 + b\eta + c$
- Exponential: $e(\eta) = a\exp(-b\eta) + c$

We fit each of these functions to the noisy AIRL reward as in SSRR. We found that these non-sigmoid functions also give comparable performance for extrapolating from the suboptimal demonstrations in both domains we tested. We observe that the performance of SSRR has large variance across runs. This instability comes from both the first step of training AIRL and the last step of running RL. Therefore, for our experiments, we fixed R_{AIRL} and π_{AIRL} after training once, generated a single set of synthetic trajectories, and only varied the regression target function to use in step (2). The authors of SSRR selected one model's performance to report from 5 runs by testing the learned reward's correlation with ground-truth reward (Chen et al., 2020). However, this approach is not practical since ground-truth reward is not available for real-world tasks. Therefore, in our experiments, we repeat each experiment three times and report the average performance. As shown in Table 1, non-sigmoid functions can generate comparable performances. See Appendix for detailed results for each run (Tables 5-8).

Q3. To evaluate whether fitting function parameters is necessary for the success of SSRR, we tested hand-picked functions as the regression target. Specifically, we used (x_0=0,y_0=2,k=5,c=-2) for sigmoid, (a=-1,b=1) for linear, (a=1,b=-2,c=1) for quadratic, and (a=1,b=5,c=0) for exponential function (see Figure 6a in Appendix for the shape of the curves). We found these regression targets result in comparable (often better) performance than fitted noise curves in HalfCheetah-V3 (Table 2), indicating that fitting the function parameters to R_{AIRL} is not necessary for extrapolating the performance from suboptimal demonstrations.

3.2. Aux-AIRL

Based on our study of SSRR, we found that the particular form of the noise curve function is not critical, nor is the step of fitting the function parameters to the noise-performance curve. We hypothesize that the major reason SSRR (and D-REX) can extrapolate is that the re-learned reward function is constrained by the noise-performance curve, enforcing the reward function to extrapolate in the direction that ensures "noisier is worse", while the reward learned just with noisy AIRL can "extrapolate" in any arbitrary direction. Specifically, the learned reward function guarantees to assign a policy π a higher value than π^η with noise level $\eta > 0$

Domain	Metric	Target Regression Function Form (fitted)			
		Sigmoid (SSRR)	**Linear**	**Quadratic**	**Exponential**
HalfCheetah-V3	Avg. Return	**1148.98 ±945.36**	821.34 ±208.60	774.66 ±418.97	476.42 ±881.49
	GT Corr.	**0.965**	0.934	0.956	0.952
Hopper-V3	Avg. Ret.	1916.72 ±102.36	2447.04 ±199.35	1630.26 ±339.61	**2529.09 ±315.39**
	GT Corr.	0.948	**0.966**	0.949	**0.966**

Table 1. Ground truth reward (with standard error) and correlation coefficients of different target regression functions.

Target Regression Function Form (hand-picked)			
Sigmoid	**Linear**	**Quadratic**	**Exponential**
3375.67	**4946.85**	2520.87	2045.53
±638.00	±514.95	±798.11	±1228.59
0.977	0.978	0.958	**0.983**

Table 2. Ground truth reward (with standard error) and correlation coefficients of different hand-picked regression functions (no fitting) in HalfCheetah-V3.

($V^{\pi} > V^{\pi^{\eta}}$). Therefore, a natural next step of investigation is to directly apply this constraint on a reward learning algorithm. To this end, we propose an auxiliary loss function that leads to maximization of the performance gap between a learning policy and its noisier variant. In particular, by fixing a noise level η_0, the objective is expressed as:

$$L_{\text{aux}}(\theta) = \mathbb{E}_{\eta \geq \eta_0}\left[V_\theta^{\pi^\eta}\right] - \mathbb{E}_{\eta < \eta_0}\left[V_\theta^{\pi^\eta}\right]. \tag{5}$$

According to Kakade & Langford (2002), we know that:

$$\forall \pi_1, \pi_2, \quad V^{\pi_1} - V^{\pi_2} = \mathbb{E}_{\tau \sim \pi_1}\left[\sum_{t=0}^{\infty} \gamma^t A^{\pi_2}(s,a)\right], \tag{6}$$

where $\tau = (s_0, a_0, s_1, a_1, \dots)$ denotes any roll-out trajectory. Therefore, we can modify the objective in (5) to:

$$\begin{aligned} L_{\text{aux}}(\theta) &= \mathbb{E}_{\eta \geq \eta_0}\left[V_\theta^{\pi^\eta} - V_\theta^{\pi}\right] - \mathbb{E}_{\eta < \eta_0}\left[V_\theta^{\pi^\eta} - V_\theta^{\pi}\right] \\ &= \mathbb{E}_{\eta \geq \eta_0, \tau \sim \pi^\eta}\Big[\sum_{t=0}^{\infty} \gamma^t A_\theta^{\pi}(s,a)\Big] \\ &\quad - \mathbb{E}_{\eta < \eta_0, \tau \sim \pi^\eta}\Big[\sum_{t=0}^{\infty} \gamma^t A_\theta^{\pi}(s,a)\Big]. \end{aligned} \tag{7}$$

Under the maximum causal entropy reinforcement learning, as mentioned in AIRL (Fu et al., 2017), we have:

$$\begin{aligned} A_\theta^{\pi}(s,a) &= \mathbb{E}_{s' \sim T(\cdot|s,a)}[f_{\xi,\phi}(s,a,s')] \\ &= \mathbb{E}_{s' \sim T(\cdot|s,a)}[g_\xi(s,a) + \gamma h_\phi(s') - h_\phi(s)]. \end{aligned} \tag{8}$$

Here, g_ξ and h_ϕ are learnable models for predicting the reward as mentioned in Sec. 2.2 (e.g. $\theta = (\xi, \phi)$). Plugging the value of A^{π} back to equation (7) results in the final auxiliary objective.

Method	HalfCheetah-v3	Hopper-v3
Demonstration	1085	1130
AIRL (Fu et al., 2017)	1872.81±87.13	1188.93±31.00
Aux-AIRL	**2191.64**±103.34	**1453.61**±15.09

Table 3. Imitation learning performance of Aux-AIRL and AIRL evaluated on the ground-truth reward throughout the training.

3.3. Preliminary Results

We tested our proposed simple one-step Aux-AIRL algorithm in the same Mujoco task environments to compare its performance with basline AIRL and SSRR. Table 3 shows the resulting performance of Aux-AIRL comparing with AIRL. Aux-AIRL extrapolates the demonstrations and outperforms the AIRL baseline in both tasks. Aux-AIRL's performance is also comparable with SSRR in the HalfCheetah-V3 domain (shown in Table 1). An experiment with varying number of demonstrations in Hopper-V3 is reported in the Appendix.

4. Conclusion and Future Work

In this paper we described an ongoing work, in which we conducted an in-depth analysis of the state-of-the-art self-supervised reward learning method SSRR. Using the insights from our analysis, we propose an end-to-end method Aux-AIRL that introduces a simple auxiliary loss in the learning objective of AIRL's discriminator. AIRL by itself is designed to learn only from optimal demonstrations. Adding our auxiliary loss enforces a gap between optimal and noisy rollouts of the learning agent's policy and adapts AIRL to leverage suboptimal demonstrations. Aux-AIRL is simpler than SSRR and outperforms the baseline AIRL method in two high-dimensional control tasks.

This work has several limitations to be addressed in future work: (1) The task domains used share a similar objective of 'running faster' and therefore is intuitively easy to extrapolate. We plan to conduct a comprehensive analysis of SSRR and Aux-AIRL on more domains with different characteristics. (2) Despite the empirical success of Aux-AIRL, we also notice that the *extrapolation* objective inherently conflicts with AIRL's *imitation* objective and it is important to find a principled way to address such conflict. (3) Currently we consider a hard noise level for Aux-AIRL and maximize the performance gap between the learned policy

and its noisy variant. The amount of noise-level is not being leveraged in the loss as SSRR does. Further incorporating insights from the analysis of SSRR and design of an end-to-end imitation learning method that extrapolates still remains an open research question.

References

Argall, B. D., Chernova, S., Veloso, M., and Browning, B. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

Bain, M. and Sommut, C. A framework for behavioural claning. *Machine intelligence*, 15(15):103, 1999.

Brown, D. S., Goo, W., Nagarajan, P., and Niekum, S. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. *arXiv preprint arXiv:1904.06387*, 2019a.

Brown, D. S., Goo, W., and Niekum, S. Ranking-based reward extrapolation without rankings. *arXiv preprint arXiv:1907.03976*, 2019b.

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

Daftry, S., Bagnell, J. A., and Hebert, M. Learning transferable policies for monocular reactive mav control. In *International Symposium on Experimental Robotics*, pp. 3–11. Springer, 2016.

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

Kakade, S. and Langford, J. Approximately optimal approximate reinforcement learning. In *In Proc. 19th International Conference on Machine Learning*. Citeseer, 2002.

Luce, R. D. *Individual choice behavior: A theoretical analysis*. Courier Corporation, 2012.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635, 2011.

Sutton, R. S., Barto, A. G., et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

	Reward	GT Corr.
HalfCheetah-V3	-334.10 $\pm$ 0.62	0.21072828
	-272.82 $\pm$ 0.60	0.5978017841
	-248.42 $\pm$ 0.95	0.09366589742
Hopper-V3	1007.34 $\pm$ 0.60	0.9388336024
	1023.48 $\pm$ 0.56	0.8237243993
	1024.13 $\pm$ 0.55	0.8824499667

Table 4. Performance of SSRR-ablation (without noise-curve fitting) directly regressing to the AIRL reward (from three different runs).

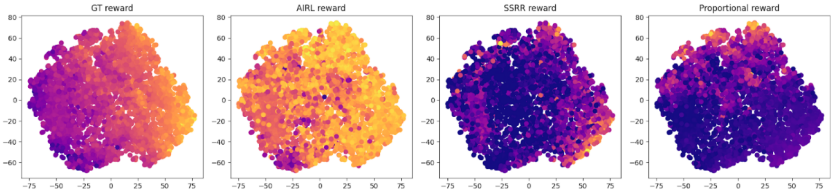


Figure 2. T-SNE visualization of reward values for sampled states from HalfCheetah-V3 experiment of using R_{AIRL} directly as regression target (brighter yellow means higher reward). Ground truth reward is added as part of state representation such that the visualization is smooth w.r.t. ground truth.

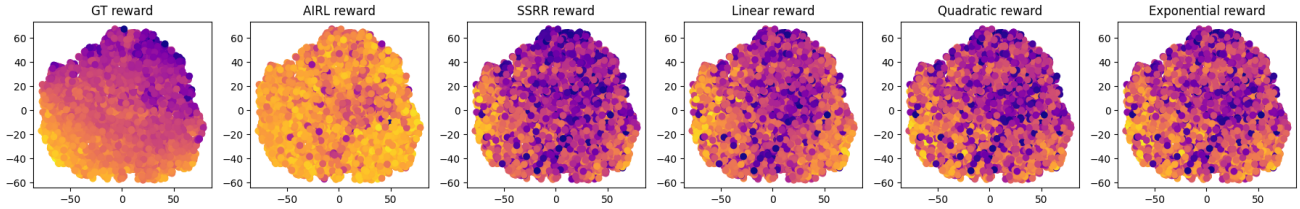


Figure 3. T-SNE visualization of different reward values for sampled states from HalfCheetah-V3 exp1 of using fitted regression target functions (brighter yellow means higher reward). Ground truth reward is added as part of state representation such that the visualization is smooth w.r.t. ground truth.

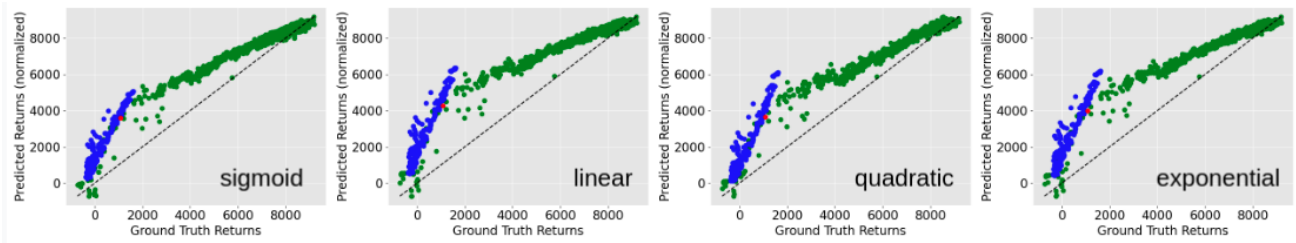


Figure 4. Correlation of learned reward functions with ground truth in HalfCheetah-v3 with a fixed AIRL reward function (exp1). Blue dots are noise-injected training trajectories and green dots are unseen trajectories (same as used in SSRR paper).

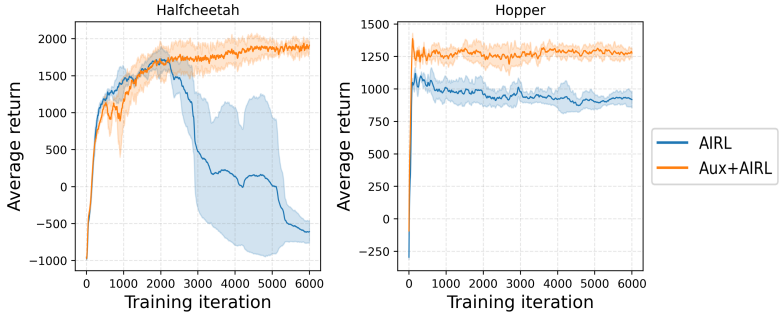


Figure 5. The learning curves of AIRL and Aux-AIRL evaluated on the ground-truth reward throughout 6000 iterations of training. Each experiment includes 3 independent runs and the shaded area indicates the standard error.

	sigmoid (SSRR)	linear	quadratic	exponential
exp1	1727.36 ± 45.69	499.03 ± 26.62	-59.69 ± 16.75	**2236.85 ± 130.08**
exp2	**2418.70 ± 65.30**	1211.92 ± 419.90	1259.00 ± 52.12	-485.88 ± 25.10
exp3	-699.12 ± 277.14	753.08 ± 82.01	**1124.66 ± 53.06**	-321.72 ± 5.76

Table 5. HalfCheetah performances for different fitting functions for each run of the experiment.

	sigmoid (SSRR)	linear	quadratic	exponential
exp1	**0.970758729**	0.9658376226	0.9592082657	0.9528530128
exp2	0**.9547071844**	0.91553707	0.9513416796	0.9478297329
exp3	**0.9688677222**	0.9201615698	0.9571505663	0.9558878951

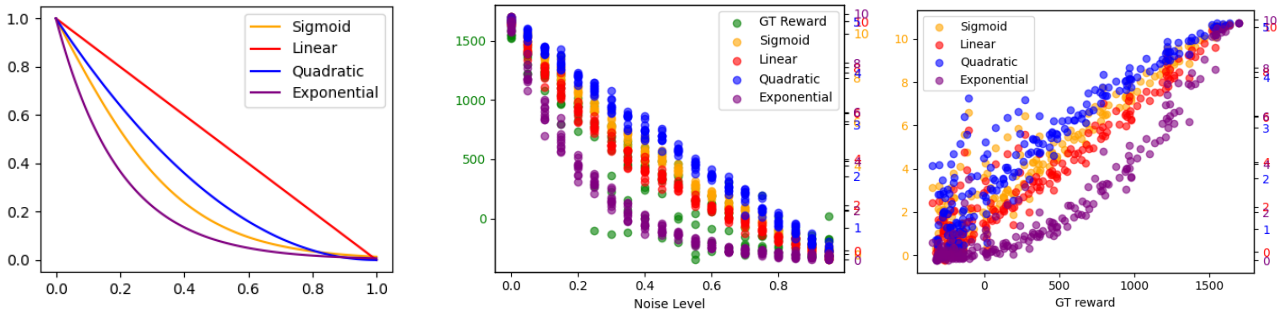
Table 6. Reward correlations for HalfCheetah for each run of the experiment.

	sigmoid (SSRR)	linear	quadratic	exponential
exp1	2106.03 ± 6.24	2173.72 ± 9.61	981.47 ± 0.58	**2813.03 ± 2.95**
exp2	1889.56 ± 2.21	**2835.08 ± 2.33**	2128.72 ± 4.29	1899.33 ± 75.19
exp3	1754.58 ± 6.24	2332.32 ± 3.18	1780.60 ± 9.51	**2874.91 ± 6.87**

Table 7. Hopper performances for different fitting functions for each run of the experiment.

	sigmoid (SSRR)	linear	quadratic	exponential
exp1	0.9472535399	0.9601988176	0.9459690465	**0.9602076292**
exp2	0.9383132994	0.9660400965	0.9408339161	**0.9660585813**
exp3	0.9592074637	0.9724836503	0.9587196361	**0.972498264**

Table 8. Reward correlations for Hopper for each run of the experiment.



(a) Regression target functions. (b) Noise-performance profile. (c) Correlation with ground truth reward.

Figure 6. Characteristic plots (evaluating on unseen synthetic trajectories) for exp1 with hand-picked noise-curves in HalfCheetah-V3.

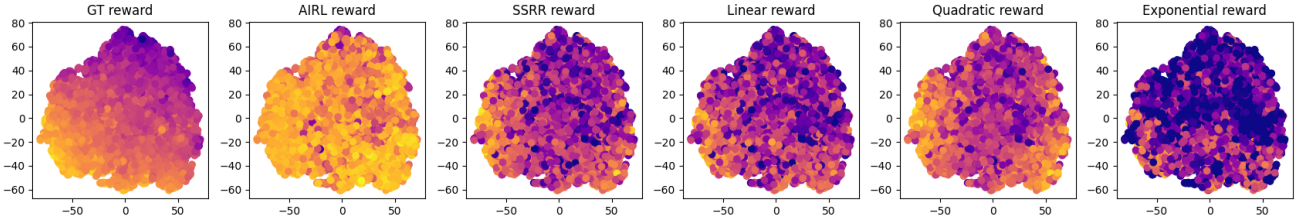


Figure 7. T-SNE visualization of learned rewards across states in synthetically generated trajectories for HalfCheetah-V3 exp1 using hand-picked regression target functions (brighter yellow means higher reward).

	sigmoid (hand-picked)	linear (hand-picked)	quadratic (hand-picked)	exponential (hand-picked)
exp1	2409.36 ± 61.85	**3919.41 ± 77.24**	1422.98 ± 73.82	3033.82 ± 35.66
exp2	4580.51 ± 74.98	**5398.97 ± 179.38**	4073.27 ± 116.02	-396.89 ± 0.90
exp3	3137.15 ± 132.75	**5522.18 ± 147.15**	2066.36 ± 164.03	3499.65 ± 48.41

Table 9. Ground truth reward for different hand-picked regression functions across three runs in HalfCheetah-V3.

	sigmoid (hand-picked)	linear (hand-picked)	quadratic (hand-picked)	exponential (hand-picked)
exp1	0.9688677222	0.9708442371	0.9576270162	**0.9803470237**
exp2	**0.9885478353**	0.9843574358	0.9530998765	0.9829318125
exp3	0.9727246482	0.979322846	0.962053522	**0.9866936429**

Table 10. Correlation coefficients with ground truth for different hand-picked regression functions across three runs in HalfCheetah-V3.

#demo trajectories	1	2	3	4
Noisy-AIRL	1308.53	1175.82	1184.19	1188.93
Aux-AIRL	**1667.66**	**1423.07**	**1468.55**	**1453.61**

Table 11. Varying number of sub-optimal demos in Hopper-V3.