A Theory

We aim to show that `rank-game` has an equilibrium that bounds the f-divergence between the agent and the expert (Theorem A.1) in the imitation learning setting. For imitation learning, we have the vanilla implicit ranking $\rho^{agent} \preceq \rho^E$, between the behavior of agent and the expert. Later, we show that, the bounded f-divergence can be used to bound the performance gap with the expert under the expert's unknown reward function using a solution to Vajda's tight lower bound (Corollary 1). Our proof proceeds by first showing that minimizing the empirical ranking loss produces a reward function that is close to the reward function obtained by the true ranking loss. Then, we show that even under the presence of policy optimization errors maximizing the obtained reward function will lead to a bounded f-divergence with the expert.

Theorem A.1. *(Performance of the `rank-game` equilibrium pair) Consider an equilibrium of the imitation `rank-game` $(\hat{\pi}, \hat{R})$, such that $\hat{R}$ minimizes the empirical ranking-loss for dataset $D^{\hat{\pi}} = \{(\rho^{\hat{\pi}}, \rho^E)\}$ and the ranking-loss generalization error is bounded by $\epsilon_r' = 2R_{max}^2\epsilon_r$, and the policy $\hat{\pi}$ has bounded suboptimality with $J(\hat{R};\hat{\pi}) \geq J(\hat{R};\pi') - \epsilon_\pi \ \forall \pi'$, then we have that at this equilibrium pair:*

$$D_f\left(\rho^{\hat{\pi}}(s,a)||\rho^E(s,a)\right) \leq \frac{(1-\gamma)\epsilon_\pi + 4R_{max}\sqrt{2\epsilon_r}}{k} \tag{7}$$

where D_f is an f-divergence with the generator function $f(x) = \frac{1-x}{1+x}$ [60, 61, 62, 63].

Proof. Previous works [18, 64] characterize the equilibrium in imitation learning based on the *supremum* ranking loss/min-max adversarial setting under no error assumption. In this section, we consider the ranking loss function L_k and derive the equilibrium for the `rank-game` in presence of reward learning and policy optimization errors. L_k attempts to explain the rankings between the agent and the expert using their state-action visitations $\mathcal{D}^\pi = \{\rho^\pi(s,a), \rho^E(s,a)\}$ respectively, by attempting to induce a performance gap of k. With this dataset $\mathcal{D}^\pi$, L_k regresses the return of state or state-action pairs in the expert's visitation to a scalar k and the agent's visitation to a value of 0. Thus, we have:

$$L_k(\mathcal{D};R) = \mathbb{E}_{\rho^E(s,a)}\left[(R(s,a)-k)^2\right] + \mathbb{E}_{\rho^\pi(s,a)}\left[(R(s,a)-0)^2\right] \tag{8}$$

The above ranking loss is minimized ($\nabla L_k = 0$) pointwise when

$$R^*(s,a) = \frac{k(\rho^E(s,a))}{\rho^E(s,a) + \rho^\pi(s,a)} \tag{9}$$

In practice, we have finite samples from both the expert visitation distribution and the agent distribution so we minimize the following empirical ranking loss $\hat{L}_k(\mathcal{D};R)$:

$$\hat{L}_k(\mathcal{D};R) = \frac{\sum_{s,a\in\hat{\rho}^E}[(R(s,a)-k)^2]}{|\hat{\rho}^E|} + \frac{\sum_{s,a\in\hat{\rho}^\pi}[(R(s,a)-0)^2]}{|\hat{\rho}^\pi|} \tag{10}$$

where $\hat{\rho}^E$ and $\hat{\rho}^\pi$ are empirical state-action visitations respectively.

From empirical loss function to reward optimality: Since the reward function is trained with supervised learning, we can quantify the sample error in minimizing the empirical loss using concentration bounds [65] up to a constant with high probability. Since $0 < R(s,a) < R_{max}$ With high probability,

$$\forall R, \ \ |L_k(\mathcal{D};R) - \hat{L}_k(\mathcal{D};R)| \leq 2R_{max}^2\epsilon_r \tag{11}$$

where ϵ_r is the statistical estimation error that can be bounded using concentration bounds such as Hoeffding's. Let R^* belong to the optimal solution for $L_k(\mathcal{D};R)$ and $\hat{R}^*$ belong to the optimal minimizing solution for $\hat{L}_k(\mathcal{D};R)$. Therefore, we have that,

$$\forall R, \ \ \hat{L}_k(\mathcal{D};\hat{R}^*) \leq \hat{L}_k(\mathcal{D};R) \tag{12}$$

Using Eq 11 and Eq 12, we have

$$\begin{aligned}
\forall R, \ \ \hat{L}_k(\mathcal{D};\hat{R}^*) &\leq \hat{L}_k(\mathcal{D};R) && (13)\\
&\leq L_k(\mathcal{D};R) + 2R_{max}^2\epsilon_r && (14)\\
&\leq L_k(\mathcal{D};R^*) + 2R_{max}^2\epsilon_r && (15)
\end{aligned}$$

and similarly

$$\forall R, \;\; L_k(\mathcal{D}; R^*) \leq L_k(\mathcal{D}; R) \tag{16}$$
$$\leq \hat{L}_k(\mathcal{D}; R) + 2R_{max}^2\epsilon_r \tag{17}$$
$$\leq \hat{L}_k(\mathcal{D}; \hat{R}^*) + 2R_{max}^2\epsilon_r \tag{18}$$

Eq 15 and Eq 18 implies that $L_k(\mathcal{D}; R^*)$ and $\hat{L}_k(\mathcal{D}; \hat{R}^*)$ are bounded with high probability. i.e

$$|L_k(\mathcal{D}; R^*) - \hat{L}_k(\mathcal{D}; \hat{R}^*)| \leq 2R_{max}^2\epsilon_r \tag{19}$$

We will use Eq 19 to show that indeed $\hat{R}^*$ has a bounded loss compared to R^*.

$$\hat{L}_k(\mathcal{D}; \hat{R}^*) - L_k(\mathcal{D}; R^*) \leq 2R_{max}^2\epsilon_r \tag{20}$$
$$L_k(\mathcal{D}; \hat{R}^*) - 2R_{max}^2 - L_k(\mathcal{D}; R^*)\epsilon_r \leq 2R_{max}^2\epsilon_r \tag{21}$$
$$L_k(\mathcal{D}; \hat{R}^*) - L_k(\mathcal{D}; R^*) \leq 4R_{max}^2\epsilon_r \tag{22}$$

We consider the tabular MDP setting and overload R to denote a vector of reward values for the entire state-action space of size $|\mathcal{S} \times \mathcal{A}|$. Using the Taylor series expansion for loss function L_k, we have:

$$L_k(\mathcal{D}; \hat{R}^*) - L_k(\mathcal{D}; R^*) \leq 4R_{max}^2\epsilon_r \tag{23}$$
$$L_k(\mathcal{D}; R^*) + \langle \nabla_{R^*} L_k(\mathcal{D}; R^*), \hat{R}^* - R^* \rangle$$
$$+0.5(\hat{R}^* - R^*)^T H(\hat{R}^* - R^*) - L_k(\mathcal{D}; R^*) \leq 4R_{max}^2\epsilon_r \tag{24}$$
$$(\hat{R}^* - R^*)^T H(\hat{R}^* - R^*) \leq 8R_{max}^2\epsilon_r \tag{25}$$

where H denotes the hessian for the loss function w.r.t R and is given by $H = P^{\rho^\pi} + P^{\rho^E}$ where P^ρ is a matrix of size $|\mathcal{S} \times \mathcal{A}| \times |\mathcal{S} \times \mathcal{A}|$ with ρ vector of visitations as its diagonal and zero elsewhere.

$$(\hat{R}^* - R^*)^T H(\hat{R}^* - R^*) \leq 8R_{max}^2\epsilon_r \tag{26}$$
$$\mathbb{E}_{s\sim\rho^\pi}\left[(\hat{R}^*(s,a) - R^*(s,a))^2\right] + \mathbb{E}_{s\sim\rho^E}\left[(\hat{R}^*(s,a) - R^*(s,a))^2\right] \leq 8R_{max}^2\epsilon_r \tag{27}$$

Since both terms in the LHS are positive we have $\mathbb{E}_{s,a\sim\rho^\pi}\left[(\hat{R}^*(s,a) - R^*(s,a))^2\right] \leq 8R_{max}^2\epsilon_r$ and $\mathbb{E}_{s,a\sim\rho^E}\left[(\hat{R}^*(s,a) - R^*(s,a))^2\right] \leq 8R_{max}^2\epsilon_r$. Applying Jensen's inequality, we further have $(\mathbb{E}_{s,a\sim\rho^\pi}\left[\hat{R}^*(s,a) - R^*(s,a)\right])^2 \leq 8R_{max}^2\epsilon_r$ and $(\mathbb{E}_{s,a\sim\rho^E}\left[\hat{R}^*(s,a) - R^*(s,a)\right])^2 \leq 8R_{max}^2\epsilon_r$. Hence,

$$\left|\mathbb{E}_{s,a\sim\rho^\pi}\left[\hat{R}^*(s,a) - R^*(s,a)\right]\right| \leq R_{max}\sqrt{8\epsilon_r} \text{ , and} \tag{28}$$
$$\left|\mathbb{E}_{s,a\sim\rho^E}\left[\hat{R}^*(s,a) - R^*(s,a)\right]\right| \leq R_{max}\sqrt{8\epsilon_r} \tag{29}$$

At this point we have bounded the expected difference between the reward functions obtained from the empirical ranking loss and the true ranking loss. We will now characterize the equilibrium obtained by learning a policy on the reward function $\hat{R}^*$ that is optimal under the empirical ranking loss. Under a policy optimization error of ϵ_π we have:

$$J(\hat{R}^*; \hat{\pi}) \geq J(\hat{R}^*; \pi') - \epsilon_\pi \; \forall \pi^{'} \in \Pi \tag{30}$$

where $J(R; \pi)$ denotes the performance of policy π under reward function R.

Taking $\pi^{'} = \pi^E$, we can reduce the above expression as follows:

$$J(\hat{R}^*, \pi^E) - J(\hat{R}^*, \hat{\pi}) \leq \epsilon_\pi \tag{31}$$
$$\frac{1}{1-\gamma}\left[\mathbb{E}_{\rho^E(s,a)}\left[\hat{R}^*(s,a)\right] - \mathbb{E}_{\rho^\pi(s,a)}\left[\hat{R}^*(s,a)\right]\right] \leq \epsilon_\pi \tag{32}$$

Using Eq 28 and Eq 29 we can lower bound $\mathbb{E}_{\rho^E(s,a)}\left[\hat{R}^*(s,a)\right] - \mathbb{E}_{\rho^\pi(s,a)}\left[\hat{R}^*(s,a)\right]$ as follows:

$$\mathbb{E}_{\rho^E(s,a)}\left[\hat{R}^*(s,a)\right] \geq \mathbb{E}_{\rho^E(s,a)}[R^*(s,a)] - R_{max}\sqrt{8\epsilon_r} \tag{33}$$

$$\mathbb{E}_{\rho^\pi(s,a)}\left[\hat{R}^*(s,a)\right] \leq \mathbb{E}_{\rho^\pi(s,a)}[R^*(s,a)] + R_{max}\sqrt{8\epsilon_r} \tag{34}$$

where $R^*(s,a)$ is given by Equation 9.

Subtracting Equation 34 from Equation 33, we have

$$\mathbb{E}_{\rho^E(s,a)}\left[\hat{R}^*(s,a)\right] - \mathbb{E}_{\rho^\pi(s,a)}\left[\hat{R}^*(s,a)\right] \geq \mathbb{E}_{\rho^E(s,a)}[R^*(s,a)] - \mathbb{E}_{\rho^\pi(s,a)}[R^*(s,a)] - 2R_{max}\sqrt{8\epsilon_r} \tag{35}$$

Plugging in the lower bound from Equation 35 in Equation 32 we have:

$$\frac{1}{1-\gamma}[\mathbb{E}_{\rho^E(s,a)}[R^*(s,a)] - \mathbb{E}_{\rho^\pi(s,a)}[R^*(s,a)] - 2R_{max}\sqrt{8\epsilon_r}] \leq \epsilon_\pi \tag{36}$$

Replacing R^* using Equation 9 we get,

$$\frac{1}{1-\gamma}\left[\mathbb{E}_{\rho^E(s,a)}\left[\frac{k(\rho^E(s,a))}{\rho^E(s,a)+\rho^\pi(s,a)}\right] - \mathbb{E}_{\rho^\pi(s,a)}\left[\frac{k(\rho^E(s,a))}{\rho^E(s,a)+\rho^\pi(s,a)}\right] - 2R_{max}\sqrt{8\epsilon_r}\right] \leq \epsilon_\pi \tag{37}$$

$$\mathbb{E}_{\rho^E(s,a)}\left[\frac{k(\rho^E(s,a))}{\rho^E(s,a)+\rho^\pi(s,a)}\right] - \mathbb{E}_{\rho^\pi(s,a)}\left[\frac{k(\rho^E(s,a))}{\rho^E(s,a)+\rho^\pi(s,a)}\right] \leq (1-\gamma)\epsilon_\pi + 2R_{max}\sqrt{8\epsilon_r} \tag{38}$$

$$\mathbb{E}_{\rho^E(s,a)}\left[\frac{(\rho^E(s,a)-\rho^\pi(s,a))}{\rho^E(s,a)+\rho^\pi(s,a)}\right] \leq \frac{(1-\gamma)\epsilon_\pi + 2R_{max}\sqrt{8\epsilon_r}}{k} \tag{39}$$

The convex function $f(x) = \frac{1-x}{1+x}$ in $\mathbb{R}^+$ defines an f-divergence. Under this generator function, the LHS of Equation 39 defines an f-divergence between the state-visitations of the agent $\rho^\pi(s,a)$ and the expert $\rho^E(s,a)$. Hence, we have that

$$D_f[\rho^\pi(s,a), \rho^E(s,a)] \leq \frac{(1-\gamma)\epsilon_\pi + 4R_{max}\sqrt{2\epsilon_r}}{k} \tag{40}$$

This bound shows that the equilibrium of the ranking game is a near-optimal imitation learning solution when ranking loss target k trades off effectively with the policy optimization error ϵ_π and reward generalization error ϵ_r. We note that, since $k \leq R_{max}$ we can get the tightest bound when $k = R_{max}$. Now, in imitation learning both k and R_{max} are tunable hyperparameters. We vary k while keeping $k = R_{max}$ and observe in appendix D.9 that this hyperparameter can significantly affect learning performance.

□

Corollary 1. *(From f-divergence to performance gap) For the equilibrium of the* `rank-game` *$(\hat{\pi}, \hat{R})$ as described in Theorem A.1, we have that the performance gap of the expert policy with $\hat{\pi}$ is bounded under the unknown expert's reward function (r_{gt}) bounded in $[0, R^E_{max}]$ as follows:*

$$|J(\pi^E, r_{gt}) - J(\hat{\pi}, r_{gt})| \leq \frac{4R^E_{max}\sqrt{\frac{(1-\gamma)\epsilon_\pi + 4R_{max}\sqrt{2\epsilon_r}}{k}}}{1-\gamma} \tag{41}$$

Proof. In Theorem A.1, we show that the equilibrium of rank-game ensures that.the f-divergence of expert visitation and agent visitation is bounded with the generator function $f = \frac{1-x}{1+x}$. First we attempt to find a tight lower bound of our f-divergence in terms of the total variational distance

between the two distributions. Such a bound has been discussed in previous literature for the usual f-divergences like KL, Hellinger and χ^2. This problem of finding a tight lower bound in terms of variational distance for a general f-divergence was introduced in [66] and referred to as Vajda's tight lower bound and a solution for arbitrary f-divergences was proposed in [67]. The f-divergence with generator function $f = \frac{1-x}{1+x}$ satisfies that $f(t) = tf(\frac{1}{t}) + 2f'(1)(t-1)$ and hence the total variational bound for this f divergence takes the form $D_f \geq \frac{2-D_{TV}}{2} f(\frac{2+D_{TV}}{2-D_{TV}}) - f'(1)D_{TV}$. Plugging in the function definition $f = \frac{1-x}{1+x}$ the inequality simplifies to:

$$D_f(\rho^\pi(s,a)\|\rho^E(s,a)) \geq \frac{(D_{TV}(\rho^\pi(s,a)\|\rho^E(s,a))^2}{4} \tag{42}$$

We also note an upper bound for this f-divergence in TV distance, sandwiching this particular f-divergence with TV bounds:

$$D_f(\rho^\pi(s,a)\|\rho^E(s,a)) = \mathbb{E}_{\rho^E(s,a)}\left[\frac{\rho^E(s,a)}{\rho^E(s,a)+\rho^\pi(s,a)}\right] - \mathbb{E}_{\rho^\pi(s,a)}\left[\frac{\rho^E(s,a)}{\rho^E(s)+\rho^\pi(s,a)}\right] \tag{43}$$

$$\leq \sum_{s,a\in\mathcal{S}\times\mathcal{A}} \left|\rho^E(s,a) - \rho^\pi(s,a)\right| \left|\frac{\rho^E(s,a)}{\rho^E(s,a)+\rho^\pi(s,a)}\right| \tag{44}$$

$$\leq D_{TV}(\rho^\pi(s,a)\|\rho^E(s,a)) \tag{45}$$

So,

$$D_{TV}(\rho^\pi(s,a)\|\rho^E(s,a)) \geq D_f(\rho^\pi(s,a)\|\rho^E(s,a)) \geq \frac{(D_{TV}(\rho^\pi(s,a)\|\rho^E(s,a))^2}{4} \tag{46}$$

Therefore from Eq 40 we have that,

$$D_{TV}(\rho^\pi(s,a)||\rho^E(s,a)) \leq 2\sqrt{\frac{(1-\gamma)\epsilon_\pi + 4R_{max}\sqrt{2\epsilon_r}}{k}} \tag{47}$$

For any policy π, and experts unknown reward function r_{gt}, $J(\pi, r) = \frac{1}{1-\gamma}[\mathbb{E}_{s,a\sim\rho^\pi}[r(s,a)]]$. Therefore,

$$|J(\pi^E, r_{gt}) - J(\pi, r_{gt})| = \left|\frac{1}{1-\gamma}[\mathbb{E}_{s,a\sim\rho^E}[r_{gt}(s,a)]] - \frac{1}{1-\gamma}[\mathbb{E}_{s,a\sim\rho^\pi}[r_{gt}(s,a)]]\right| \quad \forall\pi \tag{48}$$

$$= \frac{1}{1-\gamma}\left|\sum_{s,a\in\mathcal{S}\times\mathcal{A}} |(\rho^E - \rho^\pi) r_{gt}(s,a)\right| \tag{49}$$

$$\leq \frac{R^E_{max}}{1-\gamma} \sum_{s,a\in\mathcal{S}\times\mathcal{A}} \left|(\rho^E - \rho^\pi)\right| \tag{50}$$

$$\leq \frac{2R^E_{max}}{1-\gamma} D_{TV}(\rho_E, \rho_\pi) \tag{51}$$

$$\tag{52}$$

where R^E_{max} is the upper bound for the expert's reward function. Under a worst case expert reward function which assigns finite reward values to the expert's visitation and $-\infty$ outside the visitation, even a small mistake (visiting any state outside the expert's visitation) by the policy can result in an infinite performance gap between expert and the agent. Thus, this parameter is decided by the expert and is not in control of the learning agent.

From Eq 47 and Eq 51 we have

$$|J(\pi^E, r_{gt}) - J(\hat{\pi}, r_{gt})| \leq \frac{4R^E_{max}\sqrt{\frac{(1-\gamma)\epsilon_\pi + 4R_{max}\sqrt{2\epsilon_r}}{k}}}{1-\gamma} \tag{53}$$

□

Lemma A.2. *(Regret bound under finite data assumptions) Let $\hat{M}_t$ denote the approximate transition model under the collected dataset of transitions until iteration t. Assume that the ground truth model M and the reward function are realizable. Under these assumptions the regret of* `rank-game` *at t^{th} iteration:*

$$V_M^{\pi^E} - V_M^{\pi^t} \leq \frac{2\gamma\epsilon_m^{\pi^t} R_{max}}{(1-\gamma)^2} + \frac{4R_{max}}{1-\gamma}\sqrt{D_f(\rho^{\pi^E}_{\hat{M}_t} \| \rho^{\pi^E}_M) + \frac{2\epsilon_{stat} + 4R_{max}\sqrt{\epsilon_r}}{k}} \tag{54}$$

where V_M^π denotes the performance of policy π under transition dynamics M, $\epsilon_m^{\pi^t}$ is expected model error under policy π^t's visitation, ρ_M^π is the visitation of policy π in transition dynamics M and ϵ_{stat} is the statistical error due to finite expert samples.

Proof. We use M to denote the ground truth model and $\hat{M}_t$ to denote the approximate transition model with collected data until the t^{th} iteration of `rank-game`. We are interested in solving the following optimization problem under finite data assumptions:

$$\max_\pi \mathbb{E}_{s,a\sim\rho^\pi_{\hat{M}_t}}\left[\hat{f}^*_\pi(s,a)\right] - \frac{\sum_{s,a\in\hat{\rho}^E}[\hat{f}^*_\pi(s,a)]}{|\hat{\rho}^E|} \;\; s.t \;\; \hat{f}^*_\pi = \arg\min_f(\hat{L}_k(f; D^\pi_{\hat{M}_t})) \tag{55}$$

where $\hat{\rho}^E$ is the empirical distribution generated from finite expert samples and $D^\pi_{\hat{M}_t} = \{(\hat{\rho}^\pi_{\hat{M}_t}, \hat{\rho}^E_M)\}$. Using standard concentration bounds such as Hoeffding's [68], we can bound the empirical estimate with true estimate $\forall\pi$ with high probability:

$$\left|\frac{\sum_{s,a\in\hat{\rho}^E}[f^*_\pi(s,a)]}{|\hat{\rho}^E|} - \mathbb{E}_{s\sim\rho^E_M}[f^*_\pi(s,a)]\right| \leq \epsilon_{stat} \tag{56}$$

Using the concentration bounds and the fact that π^t is the solution that maximizes the optimization problem Eq 55 at t-iteration,

$$\mathbb{E}_{s,a\sim\rho^{\pi^t}_{\hat{M}_t}}\left[\hat{f}^*_{\pi^t}(s,a)\right] - \frac{\sum_{s,a\in\hat{\rho}^E}[\hat{f}^*_{\pi^t}(s,a)]}{|\hat{\rho}^E|} \geq \mathbb{E}_{s,a\sim\rho^{\pi^E}_{\hat{M}_t}}\left[\hat{f}^*_{\pi^E}(s,a)\right] - \frac{\sum_{s,a\in\hat{\rho}^E}[\hat{f}^*_{\pi^E}(s,a)]}{|\hat{\rho}^E|} \tag{57}$$

$$\geq \mathbb{E}_{s,a\sim\rho^{\pi^E}_{\hat{M}_t}}\left[\hat{f}^*_{\pi^E}(s,a)\right] - \mathbb{E}_{s,a\sim\rho^E_M}\left[\hat{f}^*_{\pi^E}(s,a)\right] - \epsilon_{stat} \tag{58}$$

$\hat{f}^*_{\pi^t}$ is the reward function that minimizes the empirical ranking loss $\hat{L}_k$. Let $f^*_{\pi^t}$ be the solution to the true ranking loss L_k. As shown previously in Eq 28 and Eq 29, we can bound the expected values of these two quantities with high probability under agent or expert distribution.

We also have from concentration bound:

$$\mathbb{E}_{s,a\sim\rho^{\pi^t}_{\hat{M}_t}}\left[\hat{f}^*_{\pi^t}(s,a)\right] - \frac{\sum_{s,a\in\hat{\rho}^E}[\hat{f}^*_{\pi^t}(s,a)]}{|\hat{\rho}^E|} \leq \mathbb{E}_{s,a\sim\rho^{\pi^t}_{\hat{M}_t}}\left[\hat{f}^*_{\pi^t}(s,a)\right] - \mathbb{E}_{s,a\sim\rho^E_M}\left[\hat{f}^*_{\pi^t}(s,a)\right] + \epsilon_{stat} \tag{59}$$

Therefore, combining Eq 59 and Eq 57:

$$\mathbb{E}_{s,a\sim\rho^{\pi^t}_{\hat{M}_t}}\left[\hat{f}^*_{\pi^t}(s,a)\right] - \mathbb{E}_{s,a\sim\rho^E_M}\left[\hat{f}^*_{\pi^t}(s,a)\right] \geq \mathbb{E}_{s,a\sim\rho^{\pi^E}_{\hat{M}_t}}\left[\hat{f}^*_{\pi^E}(s,a)\right] - \mathbb{E}_{s,a\sim\rho^E_M}\left[\hat{f}^*_{\pi^E}(s,a)\right] - 2\epsilon_{stat} \tag{60}$$

The LHS of the Eq. 60 can be further upper bounded as follows:

$$\mathbb{E}_{s,a\sim\rho^{\pi^t}_{\hat{M}_t}}\left[\hat{f}^*_{\pi^t}(s,a)\right] - \mathbb{E}_{s,a\sim\rho^E_M}\left[\hat{f}^*_{\pi^t}(s,a)\right] \leq \mathbb{E}_{s,a\sim\rho^{\pi^t}_{\hat{M}_t}}[f^*_{\pi^t}(s,a)] - \mathbb{E}_{s,a\sim\rho^E_M}[f^*_{\pi^t}(s,a)] + 2R_{max}\sqrt{8\epsilon_r} \tag{61}$$

$$= \mathbb{E}_{s,a\sim\rho^{\pi^t}_{\hat{M}_t}}\left[\frac{k\rho^{\pi^E}_M(s,a)}{\rho^{\pi^E}_M(s,a) + \rho^{\pi^t}_{\hat{M}_t}(s,a)}\right] - \mathbb{E}_{s,a\sim\rho^E_M}\left[\frac{k\rho^{\pi^E}_M(s,a)}{\rho^{\pi^E}_M(s,a) + \rho^{\pi^t}_{\hat{M}_t}(s,a)}\right] + 2R_{max}\sqrt{8\epsilon_r} \tag{62}$$

$$= k\mathbb{E}_{s,a\sim\rho^{\pi^E}_M}\left[\frac{\rho^{\pi^t}_{\hat{M}_t}(s,a)-\rho^{\pi^E}_M(s,a)}{\rho^{\pi^t}_{\hat{M}_t}(s,a)+\rho^{\pi^E}_M(s,a)}\right]+2R_{max}\sqrt{8\epsilon_r} \tag{63}$$

$$= -kD_f(\rho^{\pi^t}_{\hat{M}_t}\|\rho^{\pi^E}_M)+2R_{max}\sqrt{8\epsilon_r} \tag{64}$$

Similarly the RHS of Eq 60 can be further lower bounded as follows:

$$\mathbb{E}_{s,a\sim\rho^{\pi^E}_{\hat{M}_t}}\left[\hat{f}^*_{\pi^E}(s,a)\right]-\mathbb{E}_{s,a\sim\rho^E_M}\left[\hat{f}^*_{\pi}(s,a)\right]-2\epsilon_{stat} \tag{65}$$

$$\geq \mathbb{E}_{s,a\sim\rho^{\pi^E}_{\hat{M}_t}}[f^*_{\pi^E}(s,a)]-\mathbb{E}_{s\sim\rho^E_M}[f^*_{\pi}(s,a)]-2\epsilon_{stat}-2R_{max}\sqrt{8\epsilon_r} \tag{66}$$

$$= k\mathbb{E}_{s,a\sim\rho^{\pi^E}_M}\left[\frac{\rho^{\pi^E}_{\hat{M}_t}(s,a)-\rho^{\pi^E}_M(s,a)}{\rho^{\pi^E}_{\hat{M}_t}(s,a)+\rho^{\pi^E}_M(s,a)}\right]-2\epsilon_{stat}-2R_{max}\sqrt{8\epsilon_r} \tag{67}$$

$$= -kD_f(\rho^E_{\hat{M}_t}\|\rho^E_M)-2\epsilon_{stat}-2R_{max}\sqrt{8\epsilon_r} \tag{68}$$

Plugging the relations obtained (Eq 68 and 64) back in Eq 60, we see that the f-divergence between the agent visitation in the learned MDP and the expert visitation in the ground truth MDP is bounded by the f-divergence of the expert policy's visitation on the learned vs. ground truth environment. We expect this term to be low if the dynamics are accurately learned at the transitions encountered in visitation of expert.

$$D_f(\rho^{\pi^t}_{\hat{M}_t}\|\rho^{\pi^E}_M)\leq D_f(\rho^{\pi^E}_{\hat{M}_t}\|\rho^{\pi^E}_M)+\frac{2\epsilon_{stat}+4R_{max}\sqrt{8\epsilon_r}}{k} \tag{69}$$

We can use the total-variation lower bound for this f-divergence to later obtain a performance bound between the policy in learned MDP and expert in ground-truth MDP.

$$D_{TV}(\rho^{\pi^t}_{\hat{M}_t}\|\rho^{\pi^E}_M)\leq 2\sqrt{D_f(\rho^{\pi^t}_{\hat{M}_t}\|\rho^{\pi^E}_M)}\leq 2\sqrt{D_f(\rho^{\pi^E}_{\hat{M}_t}\|\rho^{\pi^E}_M)+\frac{2\epsilon_{stat}+4R_{max}\sqrt{8\epsilon_r}}{k}} \tag{70}$$

Similar to Corollary 1, we can further get a performance bound:

$$|V^{\pi^E}_M-V^{\pi^t}_{\hat{M}}|\leq\frac{2R_{max}}{1-\gamma}D_{TV}(\rho^{\pi^t}_{\hat{M}_t}\|\rho^{\pi^E}_M)\leq\frac{4R_{max}}{1-\gamma}\sqrt{D_f(\rho^{\pi^E}_{\hat{M}_t}\|\rho^{\pi^E}_M)+\frac{2\epsilon_{stat}+4R_{max}\sqrt{8\epsilon_r}}{k}} \tag{71}$$

Let the local model error in the visitation of π^t be bounded by $\epsilon^{\pi^t}_m$, i.e $\mathbb{E}_{s,a\sim\rho^{\pi_t}}\left[D_{TV}(P_M(.|s,a)\|P_{\hat{M}}(.|s,a))\right]\leq\epsilon^{\pi^t}_m$. Using simulation lemma for local models [69, 70], we have:

$$|V^{\pi^t}_M-V^{\pi^t}_{\hat{M}}|\leq\frac{2\gamma\epsilon^{\pi^t}_m R_{max}}{(1-\gamma)^2} \tag{72}$$

We are interested in bounding the performance of the policy π^t in ground-truth MDP rather than the learned MDP.

$$V^{\pi^E}_M-V^{\pi^t}_M\leq V^{\pi^E}_M-V^{\pi^t}_{\hat{M}}+\frac{4R_{max}}{1-\gamma}\sqrt{D_f(\rho^{\pi^E}_{\hat{M}_t}\|\rho^{\pi^E}_M)+\frac{2\epsilon_{stat}+4R_{max}\sqrt{8\epsilon_r}}{k}} \tag{73}$$

$$\leq\frac{2\gamma\epsilon^{\pi^t}_m R_{max}}{(1-\gamma)^2}+\frac{4R_{max}}{1-\gamma}\sqrt{D_f(\rho^{\pi^E}_{\hat{M}_t}\|\rho^{\pi^E}_M)+\frac{2\epsilon_{stat}+4R_{max}\sqrt{8\epsilon_r}}{k}} \tag{74}$$

The regret of an algorithm with ranking-loss depends on the accuracy of the approximate transition model at the visitation of the output policy π_t and the expected accuracy of the approximate transition model at the transitions encountered in the visitation of expert. Using an exploratory policy optimization procedure, the regret grows sublinearly as shown in [2]. [2] uses an exploration bonus and shows that the RHS in the above regret simplifies to be information gain and for a number of MDP families the growth rate of information gain is mild. □

Potential imitation suboptimality with additional rankings

In this section, we consider how additional rankings can affect the intended performance gap as discussed in 4.2. Consider a tabular MDP setting in which we are given a set of rankings $\rho^{\pi} \preceq \rho^{1} \preceq .. \preceq \rho^{n} \preceq \rho^{E}$. In such a case, we regress the state-action pairs from their respective visitations to $[0, k_1, k_2, ..k_n, k]$ where $0 < k_1 < k_2 .. < k_n < k$. We will discuss in Appendix B.1.1 how this regression generalizes L_k . For this regression, the optimal reward function that minimizes the ranking loss pointwise is given by:

$$R^*(s,a) = \frac{\sum_{i=1}^{n} k_i \rho^{\pi^i}(s,a) + \rho^E(s,a)}{\rho^{\pi}(s,a) + \sum_{i=1}^{n} \rho^{\pi^i}(s,a) + \rho^E(s,a)} \tag{75}$$

We consider a surrogate ranking loss with regression target k_{eff} that achieves the same optimal reward when only $\rho \preceq \rho^E$ ranking is given. Therefore:

$$\frac{\sum_{i=1}^{n} k_i \rho^{i}(s,a) + k\rho^E(s,a)}{\rho^{\pi}(s,a) + \sum_{i=1}^{n} \rho^{i}(s,a) + \rho^E(s,a)} = \frac{k_{eff}\rho^E(s,a)}{\rho^E(s,a) + \rho^{\pi}(s,a)} \tag{76}$$

k' can be upper bounded as follows:

$$k_{eff} = \frac{\rho^E(s,a) + \rho^{\pi}(s,a)}{\rho^E(s,a)} \frac{\sum_{i=1}^{n} k_i \rho^{\pi^i}(s,a) + k\rho^E(s,a)}{\rho^{\pi}(s,a) + \sum_{i=1}^{n} \rho^{\pi^i}(s,a) + k\rho^E(s,a)} \tag{77}$$

$$\leq \frac{\rho^E(s,a) + \rho^{\pi}(s,a)}{\rho^E(s,a)} \frac{\sum_{i=1}^{n} k_i \rho^{\pi^i}(s,a) + k\rho^E(s,a)}{\rho^{\pi}(s,a) + \rho^E(s,a)} \tag{78}$$

$$= k + \sum_{i=1}^{n} k_i \frac{\rho^{\pi^i}(s,a)}{\rho^E(s,a)} \tag{79}$$

k_{eff} can be lower bounded by:

$$k_{eff} = \frac{\rho^E(s,a) + \rho^{\pi}(s,a)}{\rho^E(s,a)} \frac{\sum_{i=1}^{n} k_i \rho^{\pi^i}(s,a) + k\rho^E(s,a)}{\rho^{\pi}(s,a) + \sum_{i=1}^{n} \rho^{\pi^i}(s,a) + \rho^E(s,a)} \tag{80}$$

$$\geq \frac{\rho^E(s,a) + \rho^{\pi}(s,a)}{\rho^E(s,a)} \frac{k\rho^E(s,a)}{\rho^{\pi}(s,a) + \sum_{i=1}^{n} \rho^{\pi^i}(s,a) + \rho^E(s,a)} \tag{81}$$

$$= \frac{k}{1 + \frac{\sum_{i=1}^{n} \rho^{\pi^i}(s,a)}{\rho^{\pi}(s,a) + \rho^E(s,a)}} \tag{82}$$

Thus, k_{eff} can increase or decrease compared to k after augmenting the ranking dataset. We discuss the consequences of a decreased k in Section 4.2.

B Algorithm Details

B.1 Ranking Loss for the Reward Agent

Consider a dataset of behavior rankings $\mathcal{D} = \{(\rho_1^1 \preceq \rho_1^2), (\rho_2^1 \preceq \rho_2^2), ...(\rho_n^1 \preceq \rho_n^2)\}$, wherein for ρ_j^i — i denotes the comparison index within a pair of policies, j denotes the pair number, and $\rho_1^1 \preceq \rho_1^2$ denotes that ρ_1^2 is preferable in comparison to ρ_1^1 and in turn implies that ρ_1^2 has a higher return. Each pair of behavior comparisons in the dataset are between the state-action or state visitations. We will restrict our attention to a specific instantiation of the ranking loss (a regression loss) that attempts to explain the rankings between each pair of policies present in the dataset by a performance gap of at least k, i.e. $\mathbb{E}_{\rho^1}[R(s,a)] \leq \mathbb{E}_{\rho^2}[R(s,a)] - k$. Formally, the ranking loss is defined as follows:

$$\min_R L_k(\mathcal{D}; R) = \min_R \mathbb{E}_{(\rho^1,\rho^2)\sim\mathcal{D}}\left[\mathbb{E}_{s\sim\rho^1(s,a)}\left[(R(s,a)-0)^2\right] + \mathbb{E}_{s\sim\rho^2(s,a)}\left[(R(s,a)-k)^2\right]\right] \tag{83}$$

When k is set to 1 ($k = 1$), this loss function resembles the loss function used for SQIL [71]. Thus, SQIL can be understood as a special case.

Our work explores the setting of imitation learning given samples from state or state-action visitation ρ^E of the expert π^E. We will use π_m^{agent} to denote the m^{th} update of the agent in Algorithm 1. The updated agent generates a new visitation in the environment which is stored in an empty dataset $\mathcal{D}_m^{online}$ given by $\mathcal{D}_m^{online} = \{\rho^{\pi_m^{agent}} \preceq \rho^{\pi^E}\}$

B.1.1 Reward loss with automatically generated rankings (auto)

The ranking dataset $\mathcal{D}^p$ contains pairwise comparison between behaviors $\rho_i \preceq \rho_j$. First, we assume access to the trajectories that generate the behaviors, i.e $\rho^i = \{\tau_1^i, \tau_2^i ... \tau_n^i\}$ and $\rho^j = \{\tau_1^j, \tau_2^j ... \tau_m^j\}$ In this method we propose to automatically generate additional rankings using the following procedure: (a) Sample trajectory $\tau^i \sim \rho^i$ and $\tau^j \sim \rho^j$. Both trajectories are equal length because of our use of absorbing states (see Appendix C). (b) Generate an interpolation $\tau_{\lambda_p}^{ij}$ between trajectories depending on a parameter λ_p. A trajectory is a matrix of dimensions $H \times (|\mathcal{S}|+|\mathcal{A}|)$, where H is the horizon length of all the trajectories.

$$\tau_{\lambda_p}^{ij} = \lambda_p \tau_i + (1-\lambda_p)\tau_j \tag{84}$$

These intermediate interpolated trajectories lead to a ranking that matches the ranking under the expert reward function if the reward function is indeed linear in state features. We further note that τ can also be a trajectory of features rather than state-action pairs.

Next, we generate regression targets for the interpolated trajectories. For a trajectory $\tau_{\lambda_p}^{ij}$ the regression target is given by a vector $\lambda_p \mathbf{0} + (1-\lambda_p)k\mathbf{1}$, where vectors $\mathbf{0}, \mathbf{1}$ are given by [0,0,..0] and [1,1,...,1] of length H respectively. This procedure can be regarded as a form of mixup [46] in trajectory space. The set of obtained $\tau_{\lambda_p}^{ij}$ after expending the sampling budget forms our behavior $\rho_{\lambda_p}^{ij}$.

A generalized and computationally efficient interpolation strategy for `rank-game`

Once we have generated P interpolated rankings, we effectively have $O(P^2)$ rankings that we can use to augment our ranking dataset. Using them all naively would incur a high memory burden. Thus, we present another method for achieving the same objective of using automatically generated rankings in a more efficient and generalized way. For each pairwise ranking $\rho^i \preceq \rho^j$ in the dataset $\mathcal{D}^p$, we have the following new set of rankings $\rho^i \preceq \rho_{\lambda_1}^{ij} \preceq .. \preceq \rho_{\lambda_P}^{ij} \preceq \rho^j$. Using the $O(P^2)$ rankings in the ranking loss L_k, the ranking loss can be simplified to the following using basic algebraic manipulation:

$$\begin{aligned}
&(P+1)\mathbb{E}_{(s,a)\sim\rho^j}\left[(R(s,a)-k)^2\right] + (P)\mathbb{E}_{(s,a)\sim\rho_{\lambda_P}^{ij}}\left[(R(s,a)-k)^2\right] + .. + (1)\mathbb{E}_{(s,a)\sim\rho_{\lambda_1}^{ij}}\left[(R(s,a)-k)^2\right] \\
&+(P+1)\mathbb{E}_{(s,a)\sim\rho^i}\left[(R(s,a)-0)^2\right] + (P)\mathbb{E}_{(s,a)\sim\rho_{\lambda_1}^{ij}}\left[(R(s,a)-0)^2\right] + .. + (1)\mathbb{E}_{(s,a)\sim\rho_{\lambda_P}^{ij}}\left[(R(s,a)-0)^2\right]
\end{aligned} \tag{85}$$

The reward function that minimizes the above loss pointwise is given by:

$$R^*(s,a) = \frac{k[(P+1)\rho^j + P\rho^{ij}_{\lambda_P} + (P-1)\rho^{ij}_{\lambda_{P-1}} + .. + \rho^{ij}_{\lambda_1}]}{(P+1)(\rho^j + \rho^{ij}_{\lambda_P} + .. + \rho^{ij}_{\lambda_1} + \rho^i)} \tag{86}$$

$$= \frac{k[\rho^j + \frac{P}{P+1}\rho^{ij}_{\lambda_P} + \frac{P-1}{P+1}\rho^{ij}_{\lambda_{P-1}} + .. + \frac{1}{P+1}\rho^{ij}_{\lambda_1}]}{(\rho^j + \rho^{ij}_{\lambda_P} + .. + \rho^{ij}_{\lambda_1} + \rho^i)} \tag{87}$$

We consider a modification to the ranking loss objective (Equation 83) that increases flexibility in regression targets for ranking as well as reducing the computational burden from dealing with $O(P^2)$ rankings pairs to $O(P)$. In this modification we regress the current agent, the expert, and each of the intermediate interpolants $(\rho^i, \rho^{ij}_{\lambda_1}, ..., \rho^{ij}_{\lambda_P}, \rho^E)$ to a fixed scalar return $(k_0, k_1, ..., k_{P+1})$ where $k_0 \leq k_1 \leq ... \leq k_{P+1} = k$. The optimal reward function for this loss function is given by:

$$R^*(s,a) = \frac{k_{p+1}\rho^E(s,a) + k_p\rho^{ij}_{\lambda_P}(s,a) + k_{p-1}\rho^{ij}_{\lambda_{P-1}}(s,a) + .. + k_1\rho^{ij}_{\lambda_1}(s,a) + k_0\rho^\pi(s,a)}{(\rho^E(s,a) + \rho^{ij}_{\lambda_P}(s,a) + .. + \rho^{ij}_{\lambda_1}(s,a) + \rho^\pi)(s,a)} \tag{88}$$

This modified loss function generalizes Eq 86 and recovers it exactly when $[k_0, k_1.., k_{P+1}]$ is set to be $[0, k\frac{1}{P+1}, .., k\frac{P}{P+1}, k]$. We will call this reward loss function a *generalized ranking loss*.

Shaping the ranking loss: The generalized ranking loss contains a set of regression targets $(k_0, k_1, ..., k_{P+1})$ which needs to be decided apriori. We propose two strategies for deciding these regression targets.We consider two families of parameterized mappings: (1) linear in α ($k_\alpha = \alpha * k$) and (2) rate of increase in return exponential in α ($\frac{dk_\alpha}{d\alpha} \propto e^{\beta\alpha}$), where β is the temperature parameter and denote this family by exp-β. We also set $k_{\alpha=0} = 0$ (in agent's visitation) and $k_{\alpha=1} = k$ (in expert's visitation) under the reward function that is bounded in $[0, R_{max}]$. The shaped ranking regression loss, denoted by $SL_k(\mathcal{D}; R)$, that induces a performance gap between $p+2$ consecutive rankings ($\rho^i = \rho^{ij}_{\lambda_0}, \rho^{ij}_{\lambda_1}, ..., \rho^{ij}_{\lambda_P}, \rho^j = \rho^{ij}_{\lambda_{P+1}}$) is given by:

$$SL_k(\mathcal{D}; R) = \frac{1}{p+2}\sum_{i=0}^{p+1}\mathbb{E}_{s\sim\rho^{ij}_{\lambda_i}(s,a)}\left[(R(s,a) - k_i)^2\right] \tag{89}$$

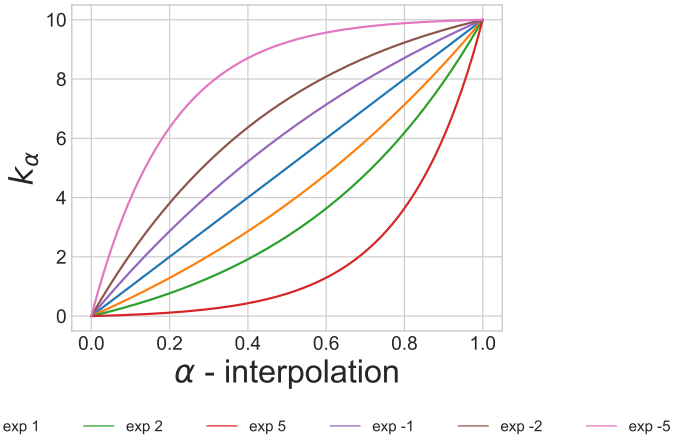

Figure 6: This figure shows the assignment of value k_α (intended return value) corresponding to values of α (degree of time-conditional interpolation between the visitation distribution of the agent and the expert). When the rate of increase is exponential with positive slope, we have a higher performance gap over comparisons closer to the expert and when the rate of increase is negative, the performance gap is higher for comparisons closer to the agent.

Figure 6 above shows the flexibility in reward shaping afforded by the two families of parameterized functions. The temperature parameter $\beta > 0$ encourages the initial preferences to have a smaller performance gap than the latter preferences. Conversely, $\beta < 0$ encourages the initial preferences to have a larger performance gap compared to the latter preferences. We ablate these choices of parameteric functions in Appendix D.5.

B.1.2 Reward loss with offline annotated rankings (pref)

Automatically generated rankings are generated without any additional supervision and can be understood as a form of data augmentation. By contrast, with offline annotated rankings, we are given a fixed dataset of comparisons which is a form of additional supervision for the reward function. Automatically generated rankings can only help by making the reward landscape easier to optimize, but offline rankings can help reduce the exploration burden by informing the agent about counterfactuals that it had no information about. This can, for instance, help the agent avoid unnecessary exploration by providing a dense improvement signal. The offline rankings are either provided by a human or extracted from a set of trajectories for which ground truth reward is known. In our work, we extract offline preferences by uniformly sampling p trajectories from an offline dataset obtained from a training run of an RL method (SAC) [53] with ground truth reward.

For imitation learning with offline annotated rankings, at every iteration m of Algorithm 1 we have a new dataset of rankings given by $\mathcal{D}_m^{online} = \{\rho_m^{agent} \preceq \rho^E\}$ along with a fixed offline dataset containing rankings of the form ($\mathcal{D}^{offline} = \{\rho^1 \preceq \rho^2 ... \preceq \rho^p\}$). We always ground the offline preferences by expert's visitation in our experiments, i.e $\rho^p \preceq \rho^E$. We incorporate the offline rankings as a soft constraint in reward learning by combining the ranking loss L_k between the policy agent and the expert, with a shaped ranking loss SL_k over offline trajectories:

$$L_k^{offline}(\mathcal{D}_,^{online}, \mathcal{D}^{offline}; R) = \lambda L_k(\mathcal{D}^{online}; R) + (1-\lambda) * SL_k(\mathcal{D}^{offline}; R) \tag{90}$$

where SL_k is the smooth ranking loss from Equation 89. Here, instead of the consecutive rankings being interpolants, they are offline rankings. The videos attached in the supplementary show the benefit of using preferences in imitation learning. The policy learned without preferences in the pen environment drops the pen frequently and in the door environment is unable to successfully open the door.

B.2 Stackelberg Game Instantiation

A Stackelberg game view of optimizing the two-player game with a dataset of behavior rankings leads to two methods: PAL (Policy as Leader) and RAL (Reward as Leader) (refer Section 4.3). PAL uses a fast reward update step and we simulate this step by training the reward function until convergence (using a validation set) on the dataset of rankings. We simulate a slow update step of the policy by using a few iterations of the SAC [53] update for the policy. RAL uses a slow reward update which we approximate by dataset aggregation — aggregating all the datasets of rankings generated by the agent in each previous iteration enforces the reward function to update slowly. A fast policy update is simulated by using more iterations of SAC. Since SAC does not perform well with a high update to environment step ratio, more iterations of SAC would imply more environment steps under a fixed reward function. This was observed to lead to reduced learning efficiency, and an intermediate value of SAC updates was observed to perform best (Table 5).

B.2.1 Policy as Leader

Algorithm 2 presents psuedocode for a practical instantiation of the PAL methods - RANK-PAL (vanilla), RANK-PAL (auto) and RANK-PAL (pref) that we use in our work. Recall that (vanilla) variant uses no additional rankings, whereas (auto) uses automatically generated rankings and (pref) uses offline annotated ranking.

B.2.2 Reward as Leader

Algorithm 3 presents psuedocode for a practical instantiation of the RAL methods - RANK-RAL (vanilla), RANK-RAL (auto).

C Implementation and Experiment Details

Environments: Figure 7 shows some of the environments we use in this work. For benchmarking we use 6 MuJoCo (licensed under CC BY 4.0) locomotion environments. We also test our method on manipulation environments - Door opening environment from Robosuite [55] (licensed under MIT License) and the Pen-v0 environment from mjrl [56] (licensed under Apache License 2.0).

Algorithm 2 Policy As Leader (PAL) practical instantiation

Initialize: Policy network π_θ, reward network R_ϕ, replay buffer $\mathcal{R}$
Hyperparameters: *Common*: Policy update steps n_{pol}, Reward update steps n_{rew}, Performance gap k, empty ranking dataset $\mathcal{D}^{online}$, *RANK-PAL (auto)*: number of interpolations P, *RANK-PAL(pref)*: Offline annotated rankings $\mathcal{D}^{offline}$.
for $m = 0, 1, 2, \ldots$ **do**
  Collect transitions in the environment and add to replay buffer $\mathcal{R}$. Run policy update step: π_θ^m = Soft Actor-Critic($R_\phi^{m-1}; \pi_\theta^{m-1}$) with transitions relabelled with reward obtained from R_ϕ^{m-1}. `// call` n_{pol} `times`
  Add absorbing state/state-actions to all early-terminated trajectories collected in the current n_{pol} policy update steps to make them full horizon and collect in $\mathcal{D}_m^{online}$. $\mathcal{D}^{online} = \mathcal{D}_m^{online}$ (discard old data).
  (for RANK-PAL(auto)) Generate interpolations for rankings in the dataset $\mathcal{D}^{online}$ and collect in $\mathcal{D}_{auto}^{online}$
  Reward Update step: `// call` n_{rew} `times`

$$R_\phi^m = \begin{cases} \min L_k(\mathcal{D}^{online}; R_\phi^{m-1}), & \text{RANK-PAL (vanilla) (Equation 83)} \\ \min SL_k(\mathcal{D}_{auto}^{online}; R_\phi^{m-1}), & \text{RANK-PAL (auto) (Equation 89)} \\ \min L_k^{offline}(\mathcal{D}^{online}, \mathcal{D}^{offline}; R), & \text{RANK-PAL (pref) (Equation 90)} \end{cases}$$

end for

Algorithm 3 Reward As Leader (RAL) practical instantiation

Initialize: Policy network π_θ, reward network R_ϕ, replay buffer $\mathcal{R}$, trajectory buffer D
Hyperparameters: *Common*: Policy update steps n_{pol}, Reward update steps n_{rew}, Performance gap k, empty ranking dataset $\mathcal{D}^{online}$, *RANK-PAL (auto)*: number of interpolations P, *RANK-PAL (pref)*: Offline annotated rankings $\mathcal{D}^{offline}$.
for $m = 0, 1, 2, \ldots$ **do**
  Collect transitions in the environment and add to replay buffer $\mathcal{R}$. Run policy update step: π_θ^m = Soft Actor-Critic($R_\phi^{m-1}; \pi_\theta^{m-1}$) with transitions relabelled with reward obtained from R_ϕ^{m-1}. `// call` n_{pol} `times`
  Add absorbing state/state-actions to all early-terminated trajectories collected in the current n_{pol} policy update steps to make them full horizon and collect in D_m^{online}. Aggregate data in $\mathcal{D}^{online} = \mathcal{D}_m^{online} \cup \mathcal{D}^{online}$.
  (for RANK-RAL(auto)) Generate interpolations for rankings in the dataset $\mathcal{D}^{online}$ and collect in $\mathcal{D}_{auto}^{online}$
  Reward Update step: `// call` n_{rew} `times`

$$R_\phi^m = \begin{cases} \min L_k(\mathcal{D}^{online}; R_\phi^{m-1}), & \text{RANK-RAL (vanilla) (Equation 83)} \\ \min SL_k(\mathcal{D}_{auto}^{online}, R_\phi^{m-1}), & \text{RANK-RAL(auto) (Equation 89)} \end{cases}$$

end for

Expert data: For all environments, we obtain expert data by a policy trained until convergence using SAC [53] with ground truth rewards.

Baselines: We compare our proposed methods against 6 representative LfO approaches that cover a spectrum of on-policy and off-policy, model-free methods from prior work: GAIfO [8, 51], DACfO [52], BCO [30], f-IRL [28], OPOLO [26] and IQ-Learn [54]. GAIfO [51] is a modification of the adversarial GAIL method [8], in which the discriminator is trained to distinguish between state-distributions rather than state-action distributions. DAC-fO [52] is an off-policy modification of GAIfO [51], in which the discriminator distinguishes the expert states with respect to the entire replay buffer of the agent's previously visited states, with additional implementation details such as added absorbing states to early-terminated trajectories. BCO [30] learns an inverse dynamics model, iteratively using the state-action-next state visitation in the environment and using it to predict the actions that generate the expert state trajectory. OPOLO [26] is a recent method which presents

Env	Swimmer	Hopper	HalfCheetah	Walker	Ant	Humanoid
BCO	102.76±0.90	20.10±2.15	5.12±3.82	4.00±1.25	12.80±1.26	3.90±1.24
GaiFO	99.04±1.61	81.13± 9.99	13.54±7.24	83.83±2.55	20.10±24.41	3.93±1.81
DACfO	95.09±6.14	94.73±3.63	85.03±5.09	54.70±44.64	86.45±1.67	19.31±32.19
f -IRL	103.89±2.37	97.45± 0.61	96.06±4.63	**101.16±1.25**	71.18±19.80	77.93±6.372
OPOLO	98.64±0.14	89.56±5.46	88.92±3.20	79.19±24.35	93.37± 3.78	24.87±17.04
IMIT-PAL (ours)	**105.93±3.12**	86.47± 7.66	90.65±15.17	75.60±1.90	82.40±9.05	94.49±3.21
IMIT-RAL (ours)	100.35±3.6	92.34±8.63	96.80±2.45	94.41±2.94	78.06±4.24	91.27±9.33
RANK-PAL (ours)	98.83±0.09	87.14± 16.14	94.05±3.59	93.88±0.72	**98.93±1.83**	**96.84±3.28**
RANK-RAL (ours)	99.31±1.50	**99.34±0.20**	**101.14±7.45**	93.24±1.25	93.21±2.98	94.45±4.13
Expert	100.00± 0	100.00± 0	100.00± 0	100.00± 0	100.00± 0	100.00± 0
$(\lvert\mathcal{S}\rvert, \lvert\mathcal{A}\rvert)$	$(8, 2)$	$(11, 3)$	$(17, 6)$	$(17, 6)$	$(111, 8)$	$(376, 17)$

Table 3: Asymptotic normalized performance of LfO methods at 2 million timesteps on MuJoCo locomotion tasks. The results in this Table also include evaluations for the IMIT-{PAL, RAL} methods.

Env	Swimmer	Hopper	HalfCheetah	Walker	Ant	Humanoid
BCO	210.22±3.43	721.92±89.89	410.83±238.02	224.58±71.42	704.88±13.49	324.94±44.39
GAIfO	202.66±4.87	2871.47±365.73	1532.57±693.72	4666.31±143.75	1141.66±1400.11	326.69±13.26
DACfO	194.65±14.08	3350.55±141.69	11057.54±407.26	3045.21±2485.33	5112.15±38.01	1165.40±1867.61
f -IRL	212.50±6.43	3446.33±35.66	12527.24±344.95	**5630.32±71.35**	4200.48±1124.17	4362.46±459.72
OPOLO	210.84±1.31	3168.35±206.26	11576.12±155.09	4407.70±1356.39	5529.44±164.94	1468.90± 1041.853
IMIT-PAL (ours)	**216.64±7.95**	3059.43±283.85	11806.47± 1750.24	4208.17±107.41	4872.39±480.23	5265.60±287.44
IMIT-RAL (ours)	205.33±8.92	3266.28±318.03	12626.18±54.71	5254.54±165.19	4612.8±192.06	5089.88±621.07
RANK-PAL (ours)	202.24±1.80	3082.98±582.59	12259.06± 206.82	5225.49±42.02	**5862.42±47.68**	**5393.45±291.16**
RANK-RAL (ours)	203.20±4.65	**3512.67±21.09**	**13204.49±721.77**	5189.51±71.27	5520.14±116.77	5262.96±337.44
Expert	204.6 ± 0	3535.88 ± 0	13051.46 ± 0	5456.91 ± 0	5926.17 ± 0	5565.53 ± 0
$(\lvert\mathcal{S}\rvert, \lvert\mathcal{A}\rvert)$	$(8, 2)$	$(11, 3)$	$(17, 6)$	$(17, 6)$	$(111, 8)$	$(376, 17)$

Table 4: Asymptotic performance of LfO methods at 2 million timesteps on MuJoCo locomotion tasks. The results in this Table also include evaluations for the IMIT-{PAL, RAL} methods.

a principled off-policy approach for imitation learning by minimizing an upper-bound of the state marginal matching objective. IQ-Learn [54] proposes to make imitation learning non-adversarial by directly optimizing the Q-function and removing the need to learn a reward as a subproblem. All the approaches only have access to expert state-trajectories.

We use the author's open-source implementations of baselines OPOLO, DACfO, GAIfO, BCO available at `https://github.com/illidanlab/opolo-code`. We use the author-provided hyperparameters (similar to those used in [26]) for all MuJoCo locomotion environments. For f-IRL, we use the author implementation available at `https://github.com/twni2016/f-IRL` and use the author provided hyperparameters. IQ-Learn was tested on our expert dataset by following authors implementation found here: `https://github.com/Div99/IQ-Learn`. We tested two IQ-Learn loss variants: 'v0' and 'value' as found in their hyperparameter configurations and took the best out of the two runs.

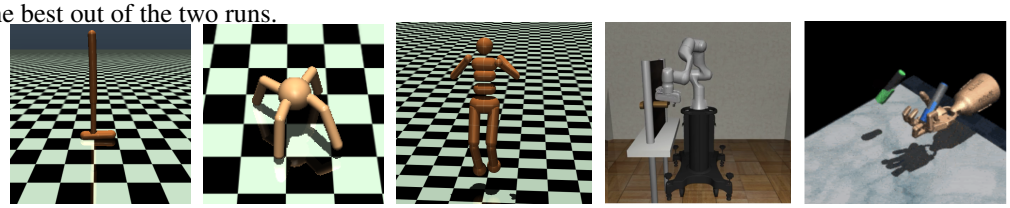

Figure 7: We evaluate `rank-game` over environments including Hopper-v2, Ant-v2, Humanoid-v2, Door, and Pen-v0.

Policy Optimization: We implement RANK-PAL and RANK-RAL with policy learning using SAC [53]. We build upon the SAC code [72] (https://github.com/openai/spinningup) without changing any hyperparameters.

Reward Learning: For reward learning, we use an MLP parameterized by two hidden layers of 64 dimensions each. Furthermore, we clip the outputs of the reward network between $[-10, 10]$ range to keep the range of rewards bounded while also adding an L2 regularization of 0.01. We add absorbing states to early terminated agent trajectories following [35]. For training the ranking loss

until convergence in both update strategies (PAL and RAL), we used evaluation on a holdout set that is 0.1 the total dataset size as a proxy for convergence.

Data sharing between players: We rely on data sharing between players to utilize the same collected transitions for both players' gradient updates. The reward learning objective in RANK-PAL and RANK-RAL requires rolling out the current policy. This makes using an off-policy routine for training the policy player quite inefficient, since off-policy model-free algorithms update a policy frequently even when executing a trajectory. To remedy this, we reuse the data collected with a mixture of policies obtained during the previous off-policy policy learning step for training the reward player. This allows us to reuse the same data for policy learning as well as reward learning at each iteration.

Ranking loss for reward shaping via offline annotated rankings: In practice for the (pref) setting (Section 4.2), to increase supervision and prevent overfitting, we augment the offline dataset by regressing the snippets (length l) of each offline trajectory τ^i for behavior ρ^i to $k * l$, in addition to regressing the rewards for each state to k. The snippets are generated as contiguous subsequence from the trajectory, similar to [3].

C.1 Hyperparameters

Hyperparameters for RANK-{PAL,RAL} (vanilla,auto and pref) methods are shown in Table 5. For RANK-PAL, we found the following hyperparameters to give best results: $n_{pol} = H$ and $n_{rew} =$ ('validation' or H/b), where H is the environment horizon (usually set to 1000 for MuJoCo locomotion tasks) and b is the batch size used for the reward update. For RANK-RAL, we found $n_{pol} = H$ and $n_{rew} =$ ('validation' or $|D|/b$), where $|D|$ indicates the cumulative size of the ranking dataset. We found that scaling reward updates proportionally to the size of the dataset also performs well and is a computationally effective alternative to training the reward until convergence (see Section D.7).

Hyperparameter	**Value**
Policy updates n_{pol}	H
Reward batch size(b)	1024
Reward gradient updates n_{rew}	val or |D|/1024
Reward learning rate	1e-3
Reward clamp range	[-10,10]
Reward l2 weight decay	0.0001
Number of interpolations [auto]	5
Reward shaping parameterization [auto]	exp-[-1]
Offline rankings loss weight (λ) [pref]	0.3
Snippet length l [pref]	10

Table 5: Common hyperparameters for the RANK-GAME algorithms. Square brackets in the left column indicate which hyperparameters that are specific to 'auto' and 'pref' methods.

D Additional Experiments

D.1 Complete evaluation of LfO with `rank-game`(auto)

Figure 8 shows a comparison of RANK-PAL(auto) and RANK-RAL(auto) for the LfO setting on the Mujoco benchmark tasks: `Swimmer-v2, Hopper-v2, HalfCheetah-v2, Walker2d-v2, Ant-v2` and `Humanoid-v2`. This section provides complete results for Section 5.1 in the main paper.

D.2 Evaluation of LfD with `rank-game`(auto)

`rank-game` is a general framework for both LfD(with expert states and actions) and LfO (with only expert states/observations). We compare performance of `rank-game` compared to LfD baselines: IQ-Learn [54], DAC [52] and BC [11].

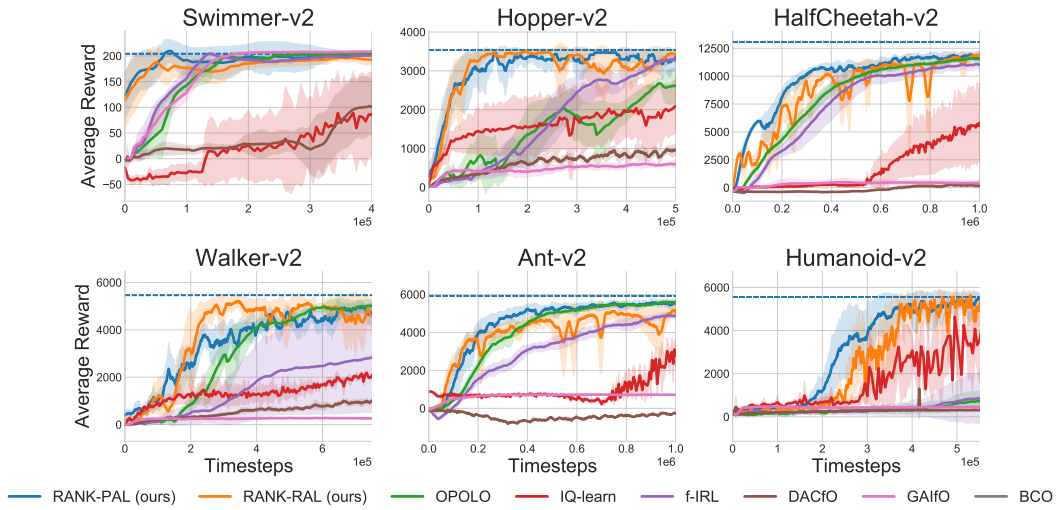

Figure 8: Comparison of performance on OpenAI gym benchmark tasks. The shaded region represents standard deviation across 5 random runs. RANK-PAL and RANK-RAL substantially outperform the baselines in sample efficiency. Dotted blue line shows the expert's performance.

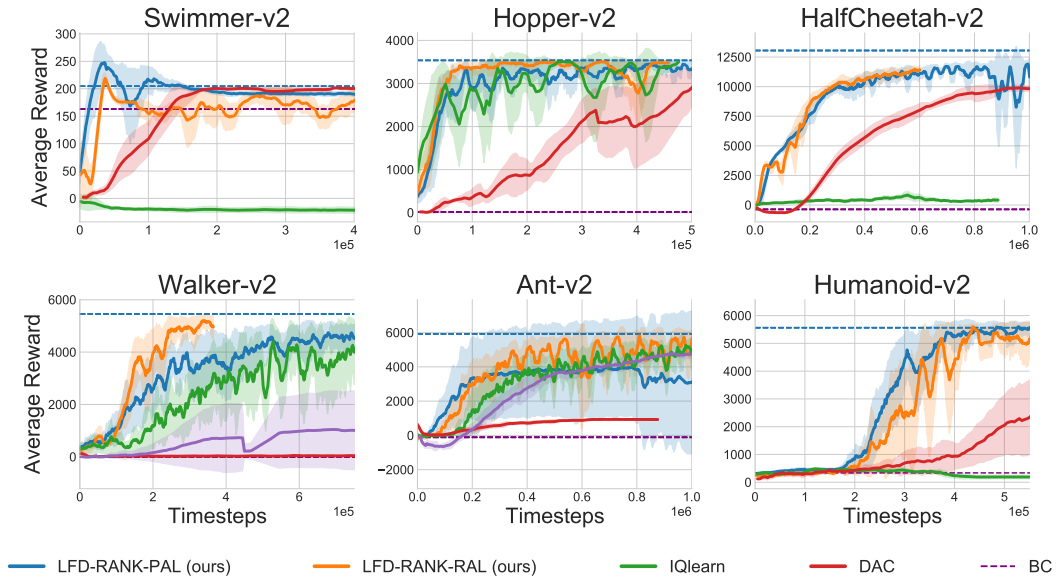

Figure 9: Comparison of `rank-game` methods with baselines in the LfD setting (expert actions are available). RANK-{PAL,RAL} are competitive to state of the art methods.

In figure 9, we observe that `rank-game` is among the most sample efficient methods for learning from demonstrations. IQlearn shows poor learning performance on some tasks which we suspect is due to the low number of expert trajectories we use in our experiments compared to the original work. DAC was tuned using the guidelines from [50] to ensure fair comparison.

D.3 Utility of automatically generated rankings in `rank-game`(auto)

We investigate the question of how much the automatically generated rankings actually help in this experiment. To do that, we keep all the hyperparameters same and compare RANK-GAME (vanilla) with RANK-GAME (auto). RANK-GAME (vanilla) uses no additional ranking information and L_k is used as the reward loss.

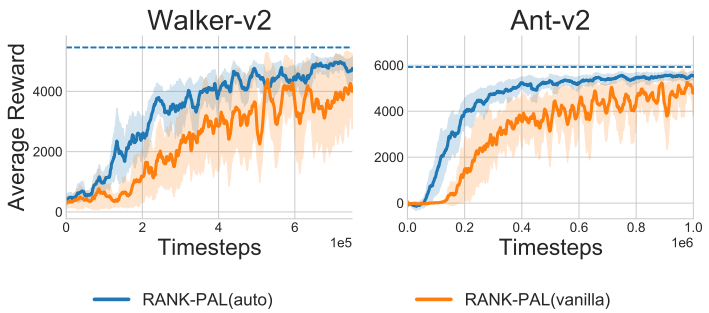

Figure 10: RANK-PAL(vanilla) has high variance learning curves with lower sample efficiency compared to RANK-PAL(auto).

Figure 10 shows that in RANK-PAL (auto) has lower variance throughout training (more stable) and is more sample efficient compared to RANK-PAL(vanilla).

D.4 Comparison of `imit-game` and `rank-game` methods

Imitation learning algorithms, particularly adversarial methods, have a number of implementation components that can affect learning performance. In this experiment, we aim to further reduce any implementation/hyperparameter gap between adversarial imitation learning (AIL) methods that are based on the *supremum*-loss (described in section 3) function and `rank-game` to bring out the obtained algorithmic improvements. To achieve this, we swap out the ranking loss L_k based on regression with a *supremum*-loss and call this method IMIT-{PAL,RAL}. This results in all the other hyperparameters such as batch size, reward clipping, policy and reward learning iterations, and optimizer iterations to be held constant across experiments.

We present a comparison of RANK-{PAL, RAL} and IMIT-{PAL, RAL} in terms of asymptotic performance in Table 3 and their sample efficiency in Figure 11. Note that Table 3 shows normalized returns that are mean-shifted and scaled between [0-100] using the performance of a uniform random policy and the expert policy. The expert returns are given in Table 4 and we use the following performance values from random policies for normalization: { Hopper= 13.828, HalfCheetah= -271.93, Walker= 1.53, Ant= -62.01, Humanoid= 112.19}. Table 4 shows unnormalized asymptotic performance of the different methods.

In terms of sample efficiency, we notice IMIT-{PAL, RAL} methods compare favorably to other regularized *supremum*-loss counterparts like GAIL and DAC but are outperformed by RANK-{PAL, RAL} (auto) methods. We hypothesize that better learning efficiency in L_k compared to *supremum*-loss is due to regression to fixed targets being a simpler optimization than maximizing the expected performance gap under two distributions.

D.5 Effect of parameterized reward shaping in `rank-game` (auto)

We experiment with different ways of shaping the regression targets (Appendix B) for automatically generated interpolations in RANK-GAME (auto) in Figure 12. In the two left-most plots for RANK-PAL (auto), we see that reward shaping instantiations (exponential with negative temperature) which learns a higher performance gap for pairs of interpolants closer to the agent lead to higher sample efficiency. We note that decreasing the temperature too much leads to a fall in sample efficiency. The same behavior is observed in RANK-RAL (two right-most plots) methods but we find them to be more robust to parameterized shaping than PAL methods. We use the following interpolation scheme: exponential with temperature=-1 for our experiments in the main paper.

D.6 On the rank preserving nature of SL_k

The ranking loss SL_k (Appendix B, Eq 89) regresses the ρ^i, ρ^j and each of the intermediate interpolants ($\rho^i = \rho^{ij}_{\lambda_0}, \rho^{ij}_{\lambda_1}, ..., \rho^{ij}_{\lambda_P}, \rho^j = \rho^{ij}_{\lambda_{P+1}}$) to fixed scalar returns $(k_0, k_1, ..., k_{P+1})$ where

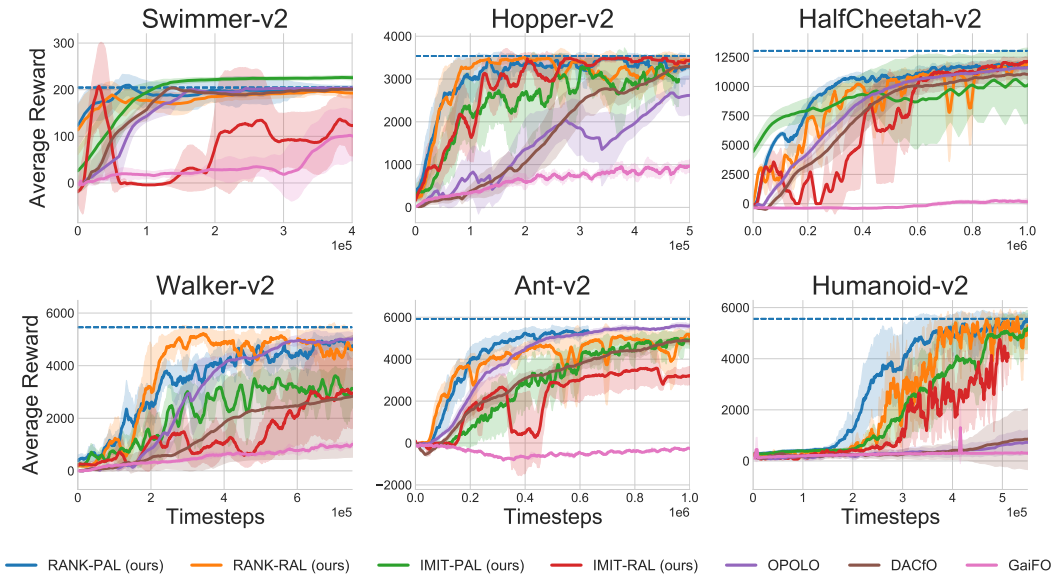

Figure 11: Comparison of performance on OpenAI gym benchmark tasks. Specifically, we seek to compare RANK-{PAL, RAL} methods to IMIT-{PAL, RAL} methods and IMIT-{PAL, RAL} methods to their non-Stackelberg counterparts GAIfO and DACfO. The shaded region represents standard deviation across 5 random runs. RANK-PAL and RANK-RAL substantially outperform the baselines in sample efficiency and IMIT-{PAL, RAL} is competitive to the strongest prior baseline OPOLO.

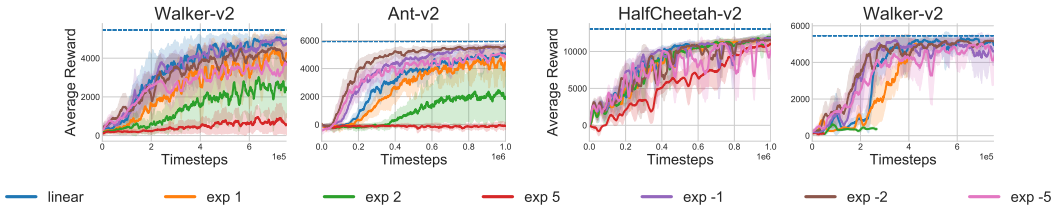

Figure 12: The two left-most plots show the effect of reward shaping in RANK-PAL (auto) methods using linear and exponential shaping functions. The two right-most plots show the same effect of reward shaping in RANK-RAL (auto) methods. Reward shaping instantiations which induce a higher performance gap between pairs of interpolants closer to the agent perform better and RAL is more robust to reward shaping variants than PAL.

$k_0 \leq k_1 \leq ... \leq k_{p+1} = k$. The ranking loss SL_k is given by:

$$SL_k(\mathcal{D}; R) = \frac{1}{p+2} \sum_{i=0}^{P+1} \mathbb{E}_{s \sim \rho^{ij}_{\lambda_i}(s,a)} \left[(R(s,a) - k_i)^2 \right] \tag{91}$$

SL_k provides a dense reward assignment for the reward agent but does not guarantee that minimizing SL_k would lead to the performance ordering between rankings, i.e $\mathbb{E}_{\rho^1}[f(s)] < \mathbb{E}_{\rho^2}[f(s)] < \mathbb{E}_{\rho^3}[f(s)] < .. < \mathbb{E}_{\rho^{P+1}}[f(s)]$. An ideal loss function for this task regresses the expected return under each behavior to scalar values indicative of ranking, but needs to solve a complex credit assignment problem. Formally, we can write the ideal loss function for reward agent as follows

$$SL^{ideal}_k(\mathcal{D}; R) = \frac{1}{p+2} \sum_{i=0}^{P+1} \left[\mathbb{E}_{s \sim \rho^{ij}_{\lambda_i}(s,a)} [R(s,a)] - k_i \right]^2 \tag{92}$$

We note that the SL_k upper bounds SL^{ideal}_k using Jensen's inequality and thus is a reasonable target for optimization. In this section we wish to further understand if SL_k has a rank-preserving policy. SL_k is a family of loss function for ranking that assigns a scalar reward value for each states of a

particular state visitation corresponding to its ranking. Ideally, given a ranking between behaviors $\rho^0 \preceq \rho^1 \preceq \rho^2 ... \preceq \rho^{P+1}$ we aim to learn a reward function f that satisfies $\mathbb{E}_{\rho^0}[f(s)] < \mathbb{E}_{\rho^1}[f(s)] < \mathbb{E}_{\rho^2}[f(s)] < .. < \mathbb{E}_{\rho^{P+1}}[f(s)]$. We empirically test the ability of the ranking loss function SL_k to facilitate the desired behavior in performance ranking. We consider a finite state space $\mathcal{S}$ and number of rankings P. We uniformly sample $P+1$ possible state visitations and the intermediate regression targets $\{k_i\}_{i=1}^n \; s.t \; k_i \leq k_{i+1}$. To evaluate the rank-preserving ability of our proposed loss function we study the fraction of comparisons the optimization solution that minimizes SL_k is able to get correct. Note that $P+1$ sequential ranking induces $P(P+1)/2$ comparisons.

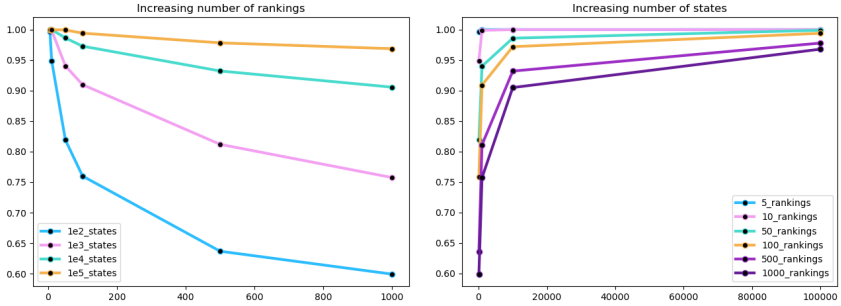

Figure 13: Increasing the state size of the domain increases the rank consistency afforded by SL_k and increasing the number of rankings decreases the rank consistency.

Figure 13 shows that with large state spaces SL_k is almost rank preserving and the rank preserving ability degrades with increasing number of rankings to be satisfied.

D.7 Stackelberg game design

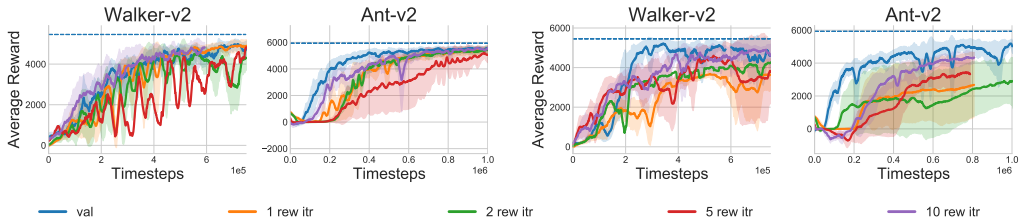

Figure 14: The left two plots use PAL strategy and the right two plots use RAL strategy. Reward learning using a validation loss on a holdout set leads to improved learning performance compared to hand designed reward learning iterations.

We consider the sensitivity of the two-player game with respect to policy update iterations and reward update iterations. Our results (Figure 14) draw analogous conclusions to [36] where we find that using a validation loss for training reward function on on-policy and aggregate dataset in PAL and RAL respectively works best. Despite its good performance, validation loss based training can be wall-clock inefficient. We found a substitute method to perform similarly while giving improvements in wall-clock time - make number of iterations of reward learning scale proportionally to the dataset set size. A proportionality constant of (1/batch-size) worked as well as validation loss in practice. Contrary to [36] where the policy is updated by obtaining policy visitation samples from the learned model, our ability to increase the policy update is hindered due to unavailability of a learned model and requires costly real-environment interactions. We tune the policy iteration parameter (Figure 15) and observe the increasing the number of policy updates can hinder learning performance.

D.8 Sensitivity of reward range for the ranking loss L_k

In Section 4.2, we discussed how the scale of learned reward function can have an effect on learning performance. We validate the hypothesis here, where we set $R_{max} = k$ and test the learning performance of RANK-PAL (auto) on various different values of k. Our results in figure D.9 show that the hyperparameter k has a large effect on learning performance and intermediate values of k works well with $k = 10$ performing the best.

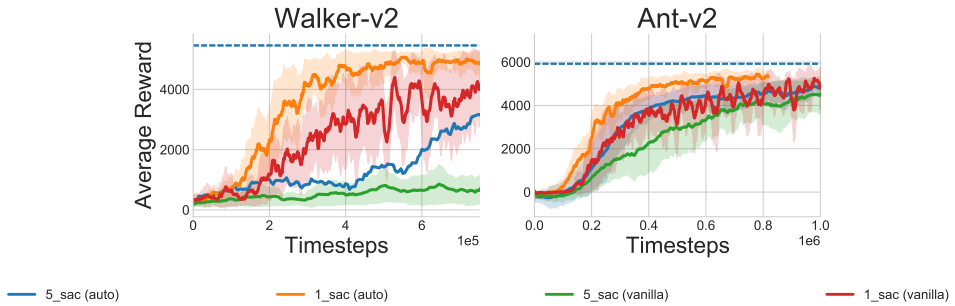

Figure 15: Small number of policy updates are useful for good learning performance in the PAL setting here.

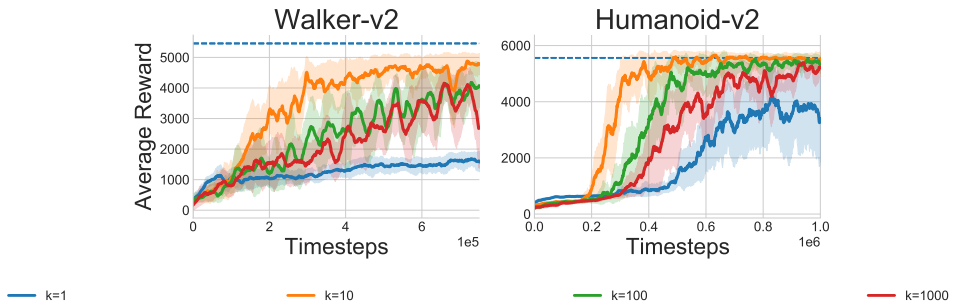

Figure 16: Intermediate values of k work best in practice.

D.9 Effect of regularizer for rank-game

`rank-game`(auto) incorporates automatically generated rankings which can be understood as a form of regularization, particularly mixup [46] in trajectory space. In this experiment, we work in the PAL setting with ranking loss L_k and compare the performances of other regularizers: Weight-decay (wd), Spectral normalization (sn), state-based mixup to (auto). Contrary to trajectory based mixup (auto) where we interpolate trajectories, in state-based mixup we sample states randomly from the behaviors which are pairwise ranked and interpolate between them.

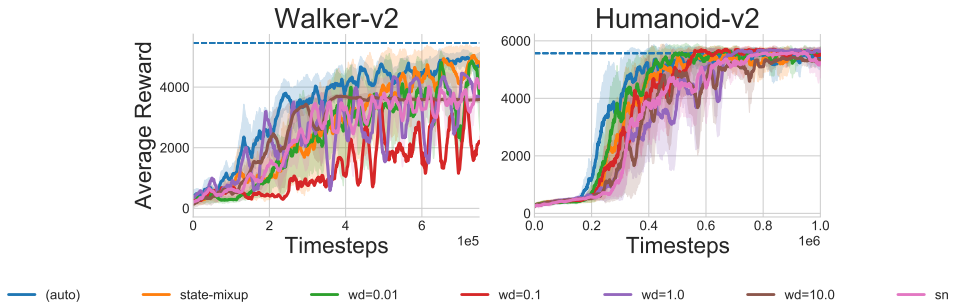

Figure 17: (auto) regularization outperforms other forms of regularization in `rank-game`

Figure 17 shows learning with (auto) regularizer is more efficient and stable compared to other regularizers.

D.10 Ablation analysis summary

We have ablated the following components for our method: Automatically-generated rankings D.3, Ranking loss D.4, Parameterized reward shaping D.5, Stackelberg game design D.7 and range of the bounded reward D.9. Our analysis above (Figure 11,16 and 14) shows quantitatively that the key improvements over baselines are driven by using the proposed ranking loss, controlling the reward range and the reward/policy update frequency in the Stackelberg framework. Parameterized reward shaping (best hyperparameter : exp -1 compare to unshaped/linear shaping) and automatically-generated rankings contribute to relatively small improvements. We note that a *single hyperparameter* combination (Table 5) works well across all tasks demonstrating robustness of the method to environment changes.

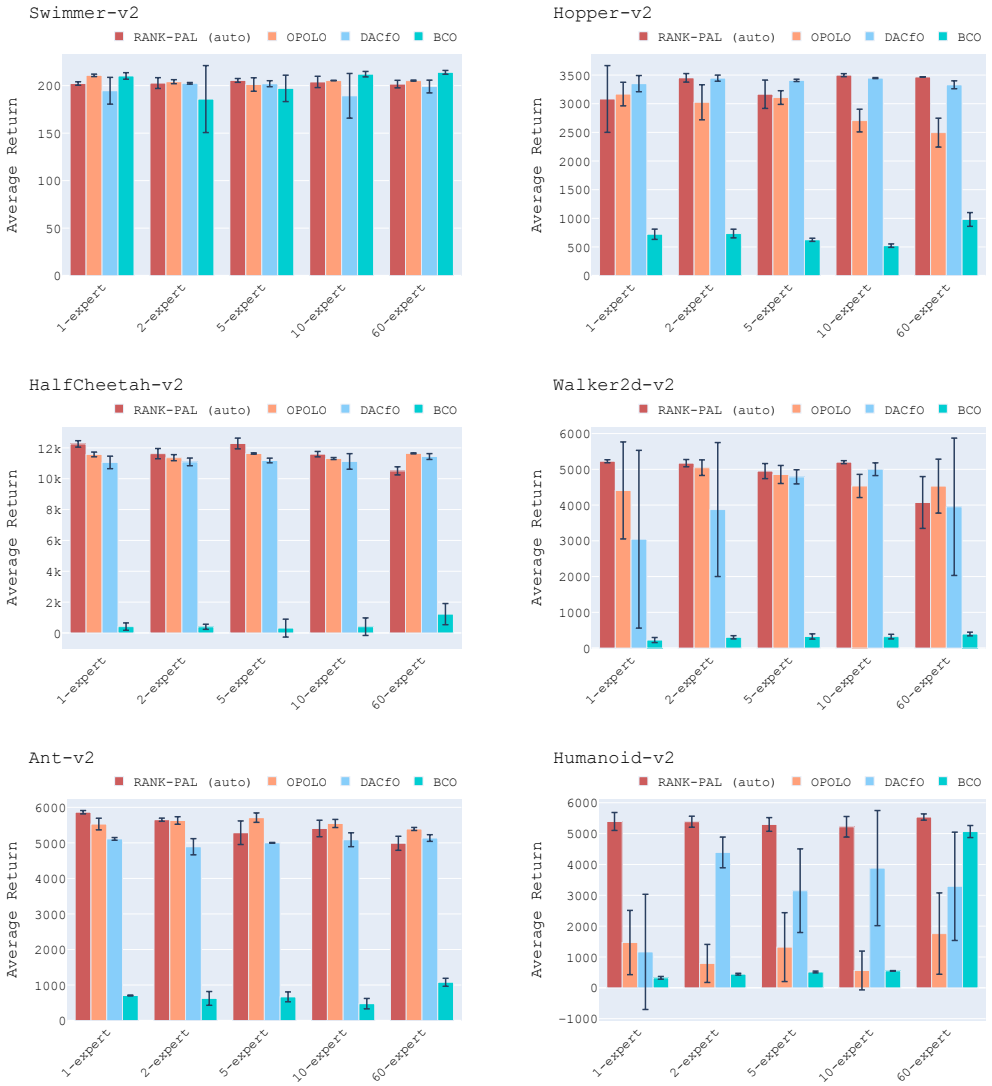

Figure 18: Performance analysis of different algorithms in the LfO setting with varying number of expert trajectories. RANK-PAL (auto) compares favorably to other methods

D.11 Varying number of expert trajectories for imitation learning

In the main text, we considered experiment settings where the agent is provided with only 1 expert trajectory. In this section, we test how our methods perform compared to baselines as we increase the number of available expert observation trajectories. We note that these experiments are in the LfO setting. Figure 18 shows that RANK-GAME compares favorably to other methods for a varying number of expert demonstrations/observations trajectories.

D.12 Robustness to noisy preferences

In this section, we investigate the effect of noisy preferences on imitation learning. We consider the setting of Section 5.2 where we attempt to solve hard exploration problems for LfO setting by leveraging trajectory snippet comparisons. In this experiment, we consider a setting similar to [3] where we inject varying level of noise, i.e flip $x\%$ of trajectory snippet at random. Figure 19 shows that RANK-PAL(pref) is robust in learning near-expert behavior upto 60 percent noise in the Door environment. We hypothesize that this robustness to noise is possible because the preferences are only used to shape reward functions and does not change the optimality of expert.

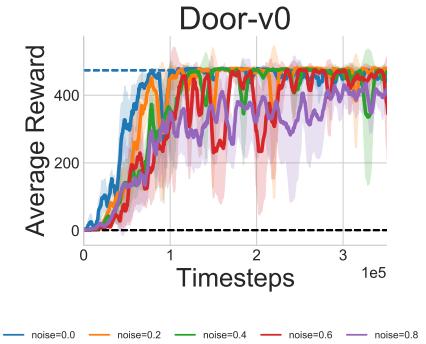

Figure 19: We investigate learning from expert observation+offline preferences where the offline preferences are noisy. RANK-PAL shows considerable robustness to noisy preferences.

D.13 Learning purely from offline rankings in manipulation environments

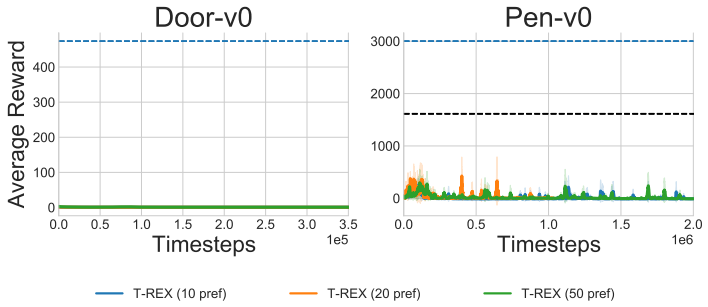

Figure 20: Testing with 10, 20 and 50 suboptimal preferences uniformly sampled from a replay buffer of SAC trained from pre-specified reward we see that TREX is not able to solve these tasks. The black dotted line shows asymptotic performance of RANK-PAL (auto) method.

In section 5.2, we saw that offline annotated preferences can help solve complex manipulation tasks via imitation. Now, we compare with the ability of a prior method—TREX [3] that learns purely from suboptimal preferences—under increasing numbers of preferences. We test on two manipulation tasks: Pen-v0 and Door-v0 given varying number of suboptimal preferences: 10, 20, 50. These preferences are uniformly sampled from a replay buffer of SAC trained until convergence under a pre-specified reward, obtained via D4RL (licensed under CC BY) .We observe in Figure 20 that T-REX is unable to solve these tasks under any selected number of suboptimal preferences.