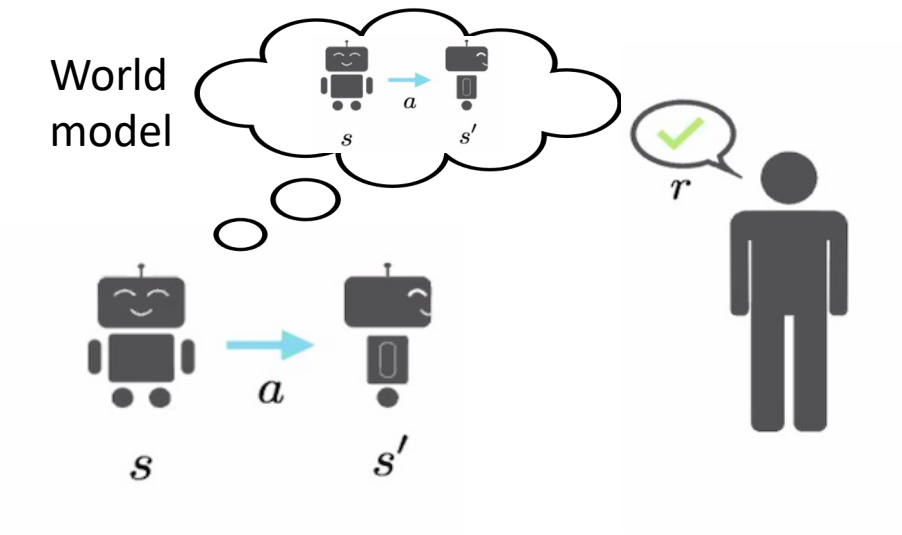
Research Discussion

Harshit Sikchi

Outline

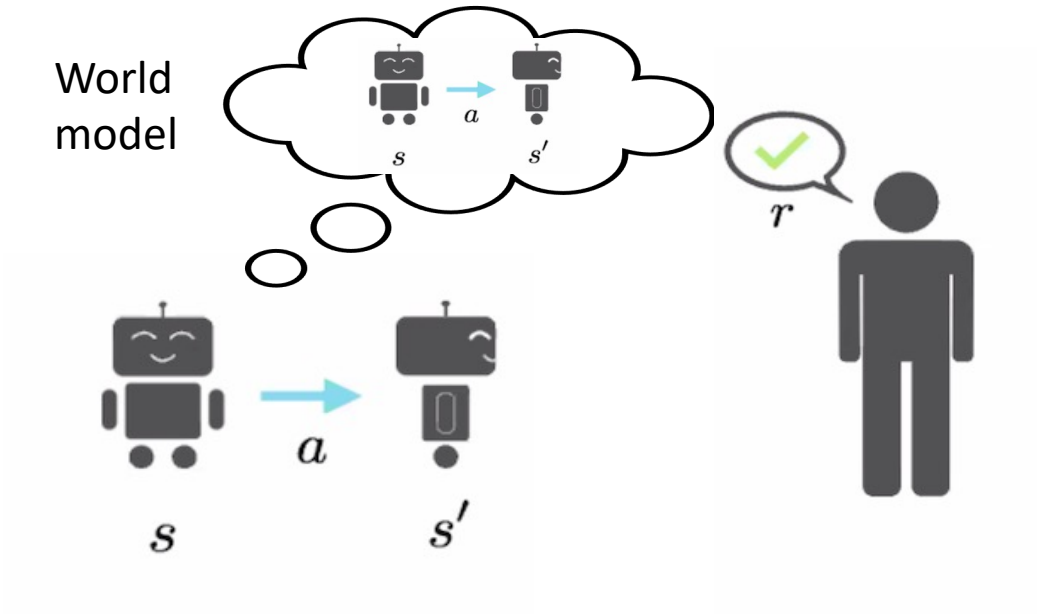


Model-based RL

Imitation Learning

LOOP: Learning off-policy with online planning

Harshit Sikchi, Wenxuan Zhou, David Held



Model-based RL

Prior Work: Combining model-based and model-free RL

Improving value targets

MBVE,
STEVE,
SVG,
SVG(H)

Model improves policy evaluation

Leveraging model generalization

(Dyna-style methods)
MBPO,
MAAC,
M2AC,
AMPO

Utilizing model for Trajectory Optimization

GPS,
PETS,
PDDM,
SMC,
POLO,
LOOP

Model improves policy optimization

Value learning with Function Approximation

Value learning via Bellman backups in model-free actor-critic methods for DRL has been plagued with a number of issues:

- Overestimation Bias[4,5]
- Delusional Bias[6]
- Rank Loss[3]
- Instability[1]
- Divergence[2]
- Further unknown issues?

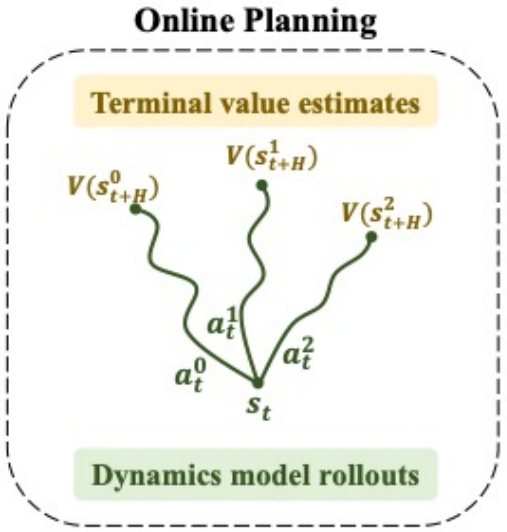
[1] Fu, J., Kumar, A., Soh, M., and Levine, S. Diagnosing bottlenecks in deep q-learning algorithms.
[2] Achiam, J., Knight, E., and Abbeel, P. Towards characterizing divergence in deep q-learning.392arXiv preprint arXiv:1903.08894, 2019
[3] Kumar, A., Agarwal, R., Ghosh, D., and Levine, S. Implicit under-parameterization inhibits data-efficient deep reinforcement learning
[4] Fujimoto, S., Van Hoof, H., and Meger, D. Addressing function approximation error in426actor-critic methods.
[5] Thrun, S. and Schwartz, A. Issues in using function approximation for reinforcement learning.
[6] Lu, T., Schuurmans, D., and Boutilier, C. Non-delusional q-learning and value-iteration. 2018

Solution: H-step lookahead policy

- Although each problem can be tackled and solved, one way is to be more robust to all value function errors arising from bellman backups.

- One such way of incorporating models is using **H-step lookahead policies**.

H-step lookahead policy



Benefits of H-step lookahead policies

- Previous work shows improved sample complexity and robustness to value function error using H-step lookahead policies with off-policy learning:

  with **tabular environments**: Efroni et al.[1]

  with **ground truth** dynamics model: Lowrey et al.[2]

- Offers a degree of interpretability missing in fully parametric methods.

- Allows to incorporate non-stationary constraints during deployment

[1] Y. Efroni, M. Ghavamzadeh, and S. Mannor. Online planning with lookahead policies. Advances in Neural Information Processing Systems, 33, 2020.
[2] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control.

H-step lookahead with Learned Model and Value Function

H-step lookahead Policy:

$$\pi_{H,\hat{V}}(s_0) = \operatorname*{argmax}_{a_0} \max_{a_1,..,a_{H-1}} \mathbb{E}_{s_0,s_{t+1}\sim\hat{\mathcal{M}}(.|s_t,a_t)}\left[L_{H,\hat{V}}(s_0)\right]$$

$$\text{where } L_{H,\hat{V}}(s_0) = \sum_{t=0}^{H-1} \gamma^t r(s_t,a_t) + \gamma^H \hat{V}(s_H)$$

Theorem 1. *(H-step lookahead policy) Suppose $\hat{M}$ is an approximate dynamics model with Total Variation distance bounded by ϵ_m. Let $\hat{V}$ be an approximate value function such that $\max_s |V^*(s) - \hat{V}(s)| \leq \epsilon_v$. Let the reward function $r(s,a)$ be bounded by $[0, R_{max}]$ and $\hat{V}$ be bounded by $[0, V_{max}]$. Then the performance of the H-step lookahead policy $\pi_{H,\hat{V}}$ can be bounded as:*

$$J^{\pi^*} - J^{\pi_{H,\hat{V}}} \leq \frac{2}{1-\gamma^H}[C(\epsilon_m, H, \gamma) + \gamma^H \epsilon_v] \tag{5}$$

where

$$C(\epsilon_m, H, \gamma) = R_{\max} \sum_{t=0}^{H-1} \gamma^t t \epsilon_m + \gamma^H H \epsilon_m V_{max}$$

H-step lookahead with Learned Model and Value Function

H-step lookahead Policy:

$$\pi_{H,\hat{V}}(s_0) = \operatorname*{argmax}_{a_0} \max_{a_1,..,a_{H-1}} \mathbb{E}_{s_0,s_{t+1}\sim\hat{\mathcal{M}}(.|s_t,a_t)}\left[L_{H,\hat{V}}(s_0)\right]$$

$$\text{where } L_{H,\hat{V}}(s_0) = \sum_{t=0}^{H-1} \gamma^t r(s_t,a_t) + \gamma^H \hat{V}(s_H)$$

Theorem 2. *Suppose $\hat{M}$ is an approximate dynamics model such that Assumption 1 holds. Let $\hat{V}$ be an approximate value function such that $\max_s |V^*(s) - \hat{V}(s)| \leq \epsilon_v$. Let the reward function by bounded in [0,R_{max}] and $\hat{V}$ be bounded in [0,V_{max}]. Let concentrability coefficient $\tilde{C}$ be such that $\forall s,a \ \frac{\nu(s,a)}{d^{\pi_D}(s,a)} \leq \tilde{C}$ where $\nu(s,a)$ is state-action distribution induced by any non-stationary policy. Then the performance of the H-step lookahead policy $\pi_{H,\hat{V}}$ obtained by running fitted-Q iteration on the learned model to convergence can be bounded as:*

$$J^{\pi^*} - J^{\pi_{H,\hat{V}}} \leq \frac{2}{1-\gamma^H}[C(\tilde{\epsilon}_m, \tilde{C}, H, \gamma) + \gamma^H \epsilon_v]$$

where

$$C(\tilde{\epsilon}_m, \tilde{C}, H, \gamma) = \frac{2(1-\gamma^H)}{1-\gamma}\left(\frac{1}{1-\gamma}\sqrt{2\tilde{\epsilon}_m \tilde{C}}\right)$$

H-step lookahead with Learned Model and Value Function

H-step lookahead Policy:

$$\pi_{H,\hat{V}}(s_0) = \operatorname*{argmax}_{a_0} \max_{a_1,..,a_{H-1}} \mathbb{E}_{s_0,s_{t+1}\sim\hat{\mathcal{M}}(.|s_t,a_t)}\left[L_{H,\hat{V}}(s_0)\right]$$

$$\text{where } L_{H,\hat{V}}(s_0) = \sum_{t=0}^{H-1} \gamma^t r(s_t,a_t) + \gamma^H \hat{V}(s_H)$$

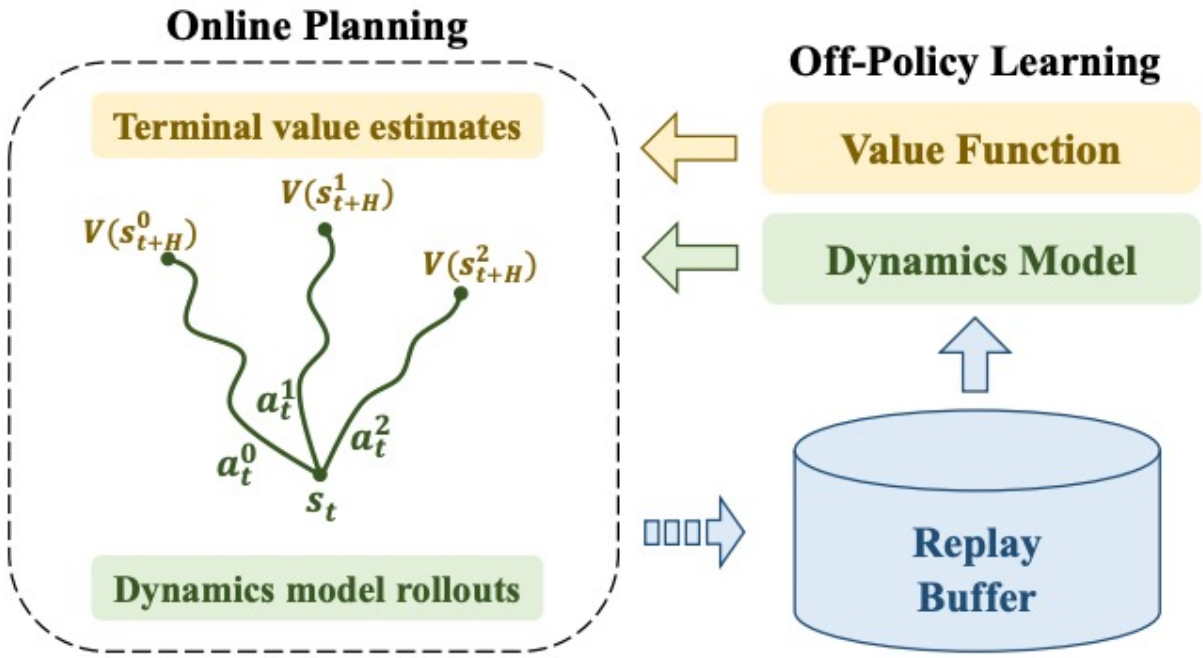
- Running H-step lookahead for computing value function targets is slow and computation can scale with batch size depending on resources.

Learning Off-Policy with Online Planning

Efficient framework for using H-step lookahead policy

- H-step lookahead for deployment + *model-free off-policy algorithm* for learning a value function

$\pi_{H,\widehat{V}}$



$Q_\phi, \pi_\theta = \pi_{0,\widehat{V}}$

$\widehat{M}$

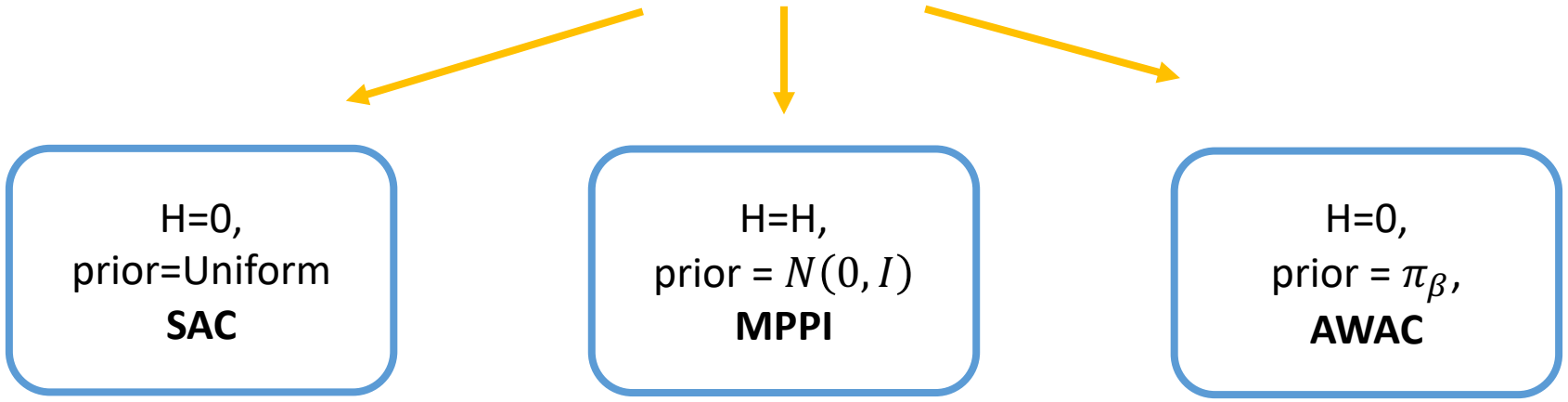
Actor Regularized Control (ARC)

Actor Divergence: $\pi_{H,\hat{V}}$ is used for deployment whereas π_θ is used for value learning.

- Solution: ARC

We note the general constrained optimization problem for policy improvement:

$$p^\tau_{opt} = \underset{p^\tau}{\operatorname{argmax}}\, \mathbb{E}_{p^\tau}\left[L_{H,\hat{V}}(s_t)\right] \text{ , s.t } D_{KL}(p^\tau || p^\tau_{prior}) \leq \epsilon$$



Actor Regularized Control (ARC)

Actor Divergence: $\pi_{H,\hat{V}}$ is used for deployment whereas π_θ is used for value learning.

$$p^\tau_{opt} = \underset{p^\tau}{\text{argmax}}\, \mathbb{E}_{p^\tau}[L_{H,\hat{V}}(s_t)]\ ,\ \text{s.t}\ D_{KL}(p^\tau || p^\tau_{prior}) \leq \epsilon$$

- Above constrained optimization admits a closed form: $p^\tau_{opt} \propto p^\tau_{prior} e^{\frac{1}{\eta} L_{H,\hat{V}}(s_t))}$

- For ARC, we set the prior to be equal to π_θ (the actor) and find the solution via Iterative Importance Sampling.

$$p^\tau_{opt} = \mathcal{N}(\mu_{opt}, \sigma_{opt})\ ,\ \mu_{opt} = \mathbb{E}_{\tau \sim p^\tau_{prior}, \hat{M}}\left[\frac{p^\tau_{opt}}{p^\tau_{prior}}\tau\right]\ ,\ \sigma_{opt} = \mathbb{E}_{\tau \sim p^\tau_{prior}, \hat{M}}\left[\frac{p^\tau_{opt}}{p^\tau_{prior}}(\tau - \mu)^2\right]$$

LOOP for Offline and Safe RL

Offline RL: ARC optimizes for the following uncertainty-pessimistic objective.

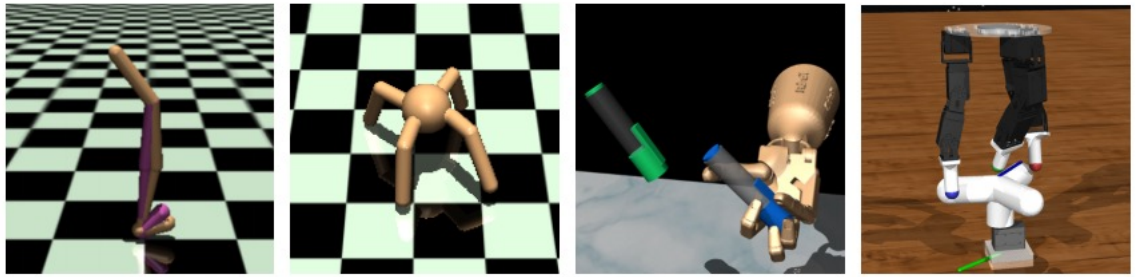
$$\text{mean}_{[K]}[L_{H,\hat{V}}(s_t)] - \beta \text{std}_{[K]}[L_{H,\hat{V}}(s_t)]$$

Safe RL: ARC optimizes for the following cost-pessimistic objective.

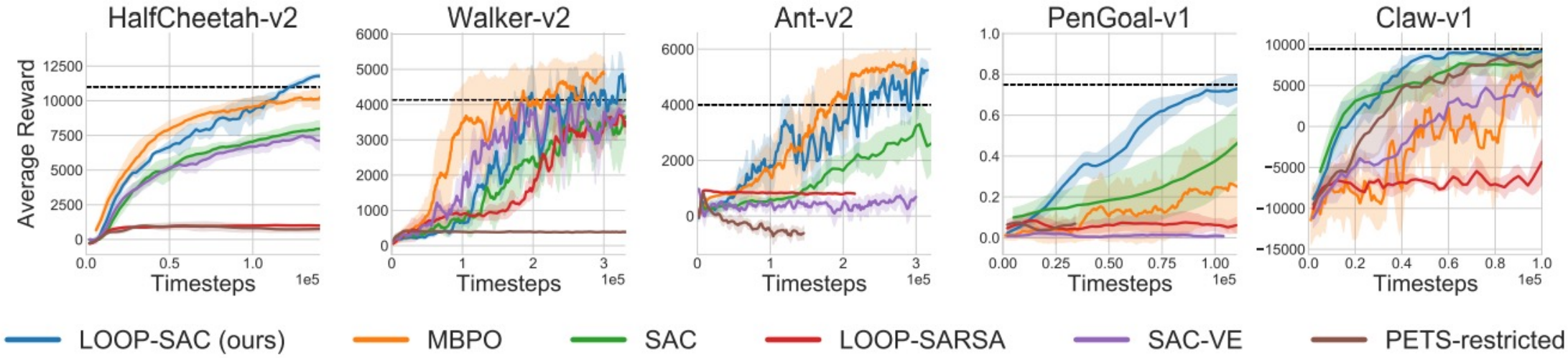
$$\text{argmax}_{a_t} \mathbb{E}_{\hat{M}}\left[L_{H,\hat{V}}(s_t)\right] \text{s.t. } \max_{[K]} \sum_{t=0}^{H} \gamma^t c(s_t, a_T) \leq d_0$$

Experiments

LOOP for Online RL



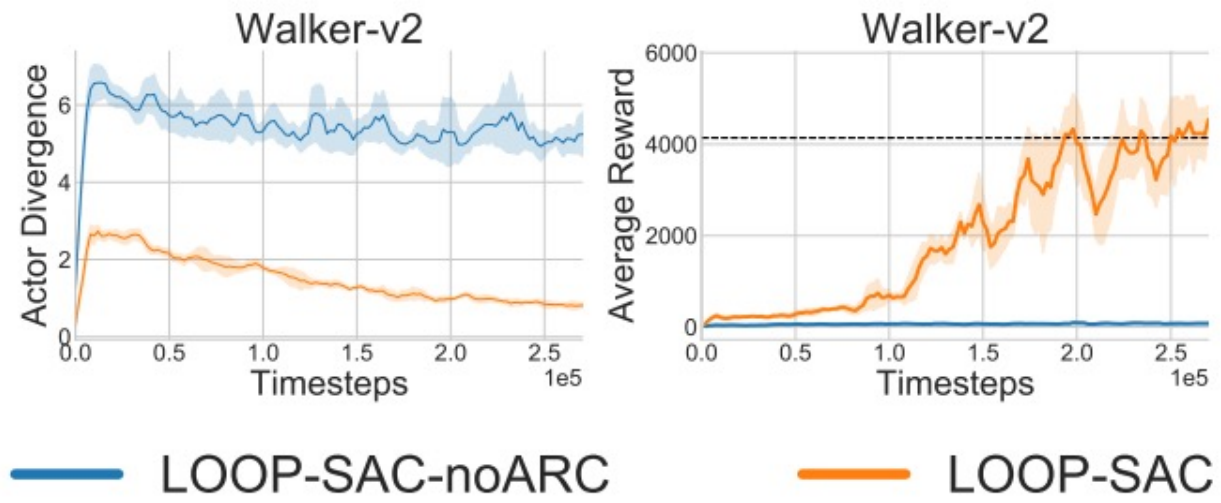
LOOP is sample-efficient and performs competitively to MBPO, outperforming it significantly in manipulation tasks.



Empirically demonstrates that model-error tradeoff with value-errors is indeed beneficial !

Necessity of ARC

ARC reduces actor-divergence and enable learning for LOOP

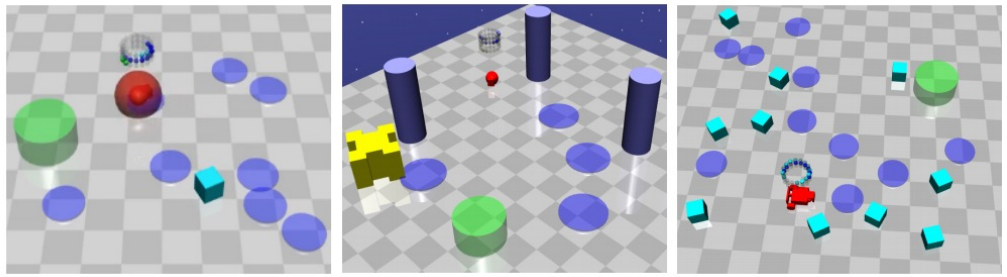


LOOP for Offline RL

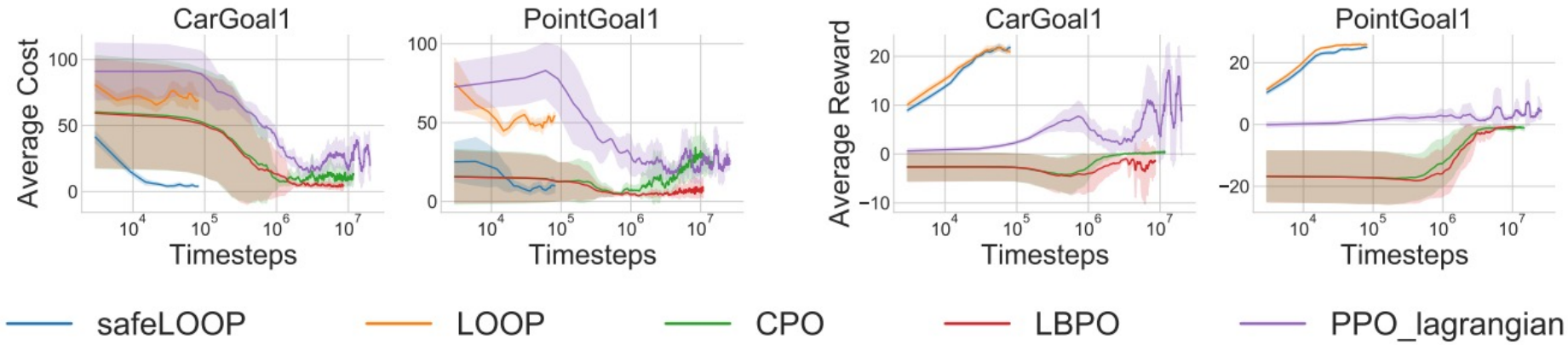
Test the non-parametric policy on a value function learned from offline data

Dataset	Env	CRR	LOOP CRR	Improve%	PLAS	LOOP PLAS	Improve%	MBOP
random	hopper	10.40	10.68	2.7	10.35	10.71	3.5	**10.8**
	halfcheetah	4.23	7.55	78.5	26.05	**26.14**	0.3	6.3
	walker2d	1.94	2.04	5.2	0.89	2.83	218.0	**8.1**
medium	hopper	65.73	**85.83**	30.6	32.08	56.47	76.0	48.8
	halfcheetah	41.14	41.54	1.0	39.33	39.54	0.5	**44.6**
	walker2d	69.98	**79.18**	13.1	46.20	52.66	14.0	41.0
med-replay	hopper	27.69	29.08	5.0	29.29	**31.29**	6.8	12.4
	halfcheetah	42.29	42.84	1.3	43.96	**44.25**	0.7	42.3
	walker2d	19.84	27.30	37.6	35.59	**41.16**	15.7	9.7
med-expert	hopper	112.02	113.71	1.5	110.95	**114.32**	3.0	55.1
	halfcheetah	21.48	24.19	12.6	93.08	98.16	5.5	**105.9**
	walker2d	103.77	**105.76**	1.9	90.07	99.03	9.9	70.2

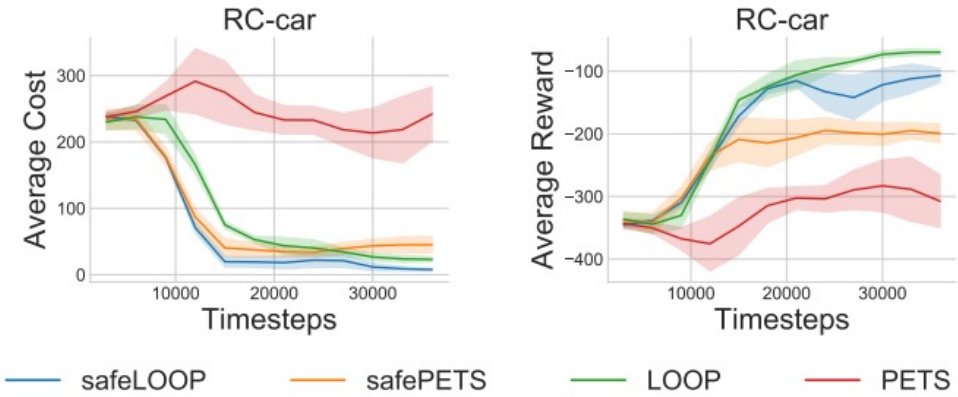
LOOP for Safe-RL



Safe Optimization in ARC leads to sample efficient training and low constraint violations.



Terminal value function helps in
learning long horizon behaviors like drifting



Conclusion and Discussion

Generalization or Stabilizing Value function?

Leveraging model generalization

(Dyna-style methods)
MBPO,
MAAC,
M2AC,
AMPO

- Short-model rollouts are used to augment the replay buffer of a model-free off–policy algorithm.
- Short-model free rollouts allows for querying the model in more accurate places, bypassing the compounding error issue.
- **Claim is that model-generalization leads to improved sample efficiency of the complete algorithm.**

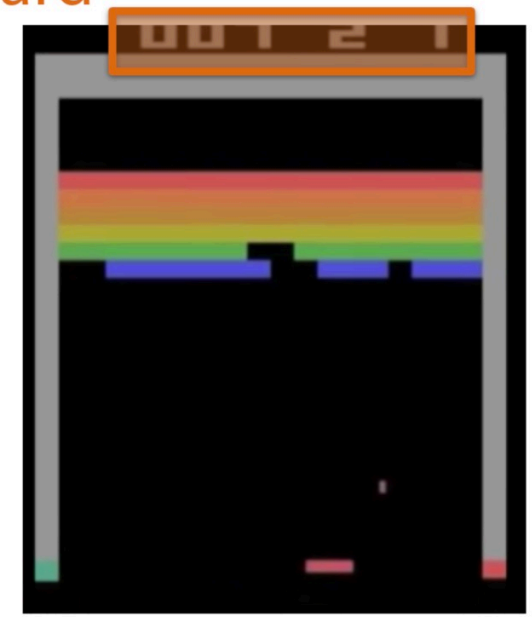
f-IRL: IRL via state marginal matching

Imitation Learning

Why imitate?

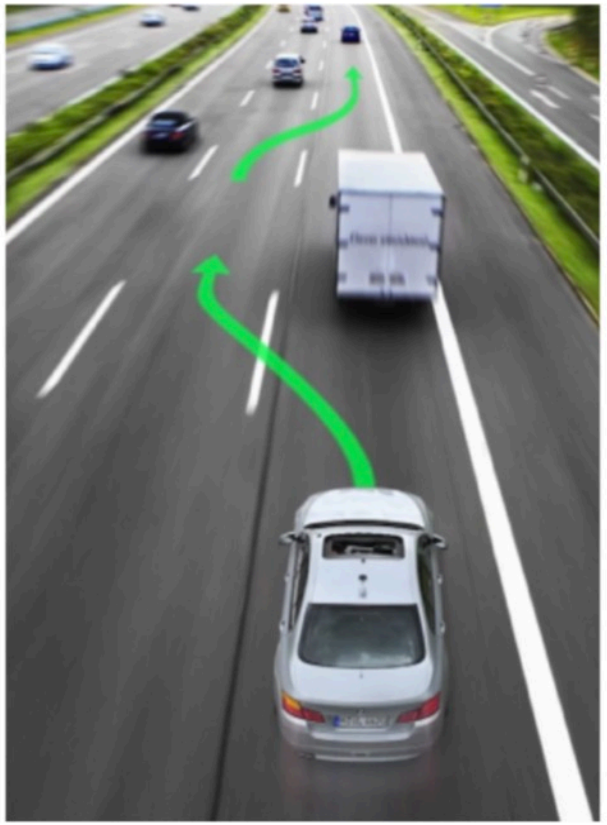
Designing Reward is tricky

reward



Mnih et al.'15

Ideal World



Real World – Where's the reward?

Reward Design

$$r_t^c = \exp\left[-10\left(\|c_t^* - c_t\|^2\right)\right]. \qquad r_t^p = \exp\left[-2\left(\sum_j \|q_{t,j}^* \ominus q_{t,j}\|^2\right)\right].$$

$$r_t^v = \exp\left[-0.1\left(\sum_j \|\dot{q}_{t,j}^* - \dot{q}_{t,j}\|^2\right)\right]. \quad r_t^e = \exp\left[-40\left(\sum_e \|p_{t,e}^* - p_{t,e}\|^2\right)\right].$$

$$r_t = w^p r_t^p + w^v r_t^v + w^e r_t^e + w^c r_t^c$$

$$w^p = 0.65, w^v = 0.1, w^e = 0.15, w^c = 0.1.$$

SFV reward function

$$r_{e_y} = e^{-k_1 e_y}$$

$$r_{e_\psi},\ r_{e_\beta} = f(x) = \begin{cases} e^{-k_2|x|} & |x| < 90^\circ \\ -e^{-k_2(180^\circ - x)} & x \geq 90^\circ \\ -e^{-k_2(180^\circ + x)} & x \leq -90^\circ \end{cases}$$

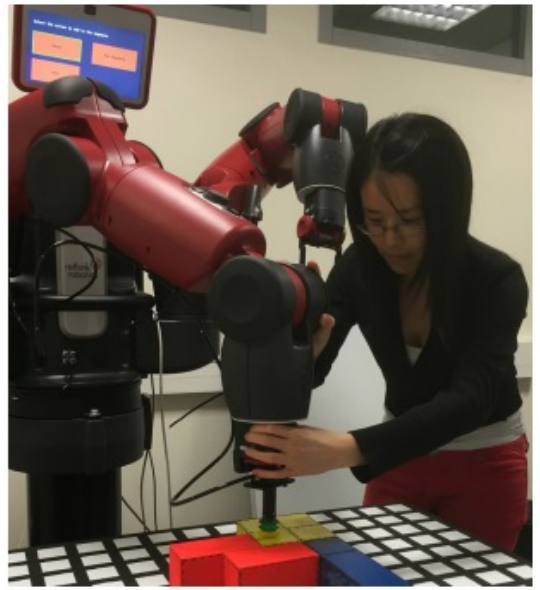
$$r = v(k_{e_y} r_{e_y} + k_{e_\psi} r_{e_\psi} + k_{e_\beta} r_{e_\beta}).$$

Drifting reward function

Expert Demonstrations

Can be specified using human skills

By simple kinesthetic demonstrations



Expert Demonstrations

Or even by dogs on treadmill

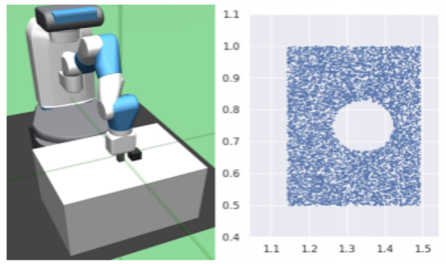
RSS ’20
best paper

Forms of expert demonstration

Trajectories
τ

State-Action pairs
$\rho_E(s, a)$

States
$\rho_E(s)$



Strictly Less Information

Imitation Learning as Divergence Minimization

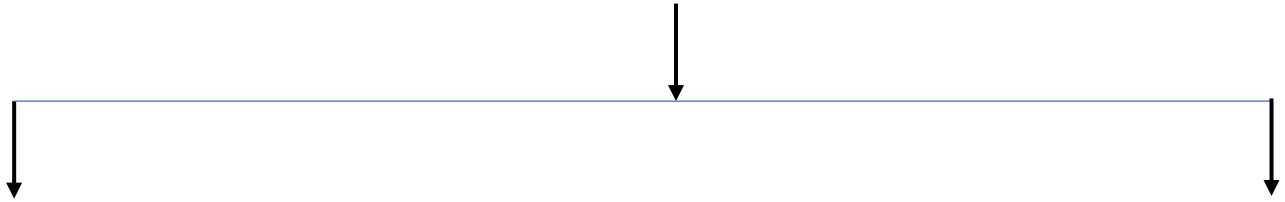
Given trajectories: $D_f\left(\rho_{expert}(\tau) \,||\, \rho_{agent}(\tau)\right)$

Given state-action pairs: $D_f\left(\rho_{expert}(s,a) \,||\, \rho_{agent}(s,a)\right)$

Given states: $D_f\left(\rho_{expert}(s) \,||\, \rho_{agent}(s)\right)$

A Divergence Minimization Perspective on Imitation Learning Methods – Ghasemipour et al 2019 [CoRL 2019 Best Paper]

Imitation Learning: Classification



Adversarial Imitation Learning

- Matches the divergences in trajectory, state-action and state space.
- Recovers the policy that minimizes the respective divergence
- Algorithms:
  - GAIL (Ho and Ermon ‘16)
  - F-MAX (Ghasemipour et al’ 19)

Inverse Reinforcement Learning

- Matches the divergences only in trajectory space and state space.
- Recovers the policy **and the reward** that minimizes the respective divergence
- Algorithms:
  - MaxEntIRL (Ziebart ‘08)
  - Deep MaxEntIRL (Wulfmeier’ 15)
  - AIRL (Fu et al’ 18)
  - f-IRL (Our method)

Bilevel Optimization

$$\max_{\phi} \min_{\theta} L(r_\theta, \pi_\phi)$$

$$\min_{\theta,\phi} L(r_\theta, \pi_\phi)$$

$$\text{s.t. } \phi \in \arg\max_{\phi} g(r_\theta, \pi_\phi)$$

Adversarial Imitation Learning (AIL)

A simple and unified way to understand AIL algorithms

- Learn a discriminator which classifies agent's current state-action(or state) distribution with expert's state-action(or state) distribution

- At convergence discriminator outputs the following ratio:

$$D^*(s,a) = \frac{\rho_{expert}(s,a)}{\rho_{expert}(s,a)+\rho_{agent}(s,a)}$$

Adversarial Imitation Learning (AIL)

A simple and unified way to understand AIL algorithms

$$D^*(s,a) = \frac{\rho_{expert}(s,a)}{\rho_{expert}(s,a) + \rho_{agent}(s,a)}$$

- Reverse KL between state-action distribution (**f-MAX** [Ghasemipour et al 2019])

$$\min E_{\rho_{agent}(s,a)}[\log \frac{\rho_{agent}(s,a)}{\rho_{expert}(s,a)}] = \min E_{\rho_{agent}(s,a)}[\log D^*(s,a) - \log(1 - D^*(s,a))]$$

- Jensen-Shannon between state-action distribution (**GAIL** [Ho and Ermon 2016])

$$\min E_{\rho_{agent}(s,a)}\left[\log \frac{\rho_{agent}(s,a)}{0.5\ \rho_{agent}(s,a) + 0.5\ \rho_{expert}(s,a)}\right] + E_{\rho_{expert}}[\log \frac{\rho_{expert}(s,a)}{0.5\ \rho_{agent}(s,a) + 0.5\ \rho_{expert}(s,a)}]$$

$$= \min\ -\log 4 + E_{\rho_{agent}(s,a)}[\log(1 - D^*(s,a))] + E_{\rho_{expert}}[\log D^*(s,a)]$$

Inverse Reinforcement Learning(IRL)

Learn a reward that explains the expert behavior.

We consider the Maximum Entropy Framework

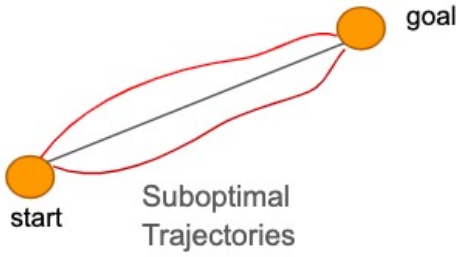
$$p_\theta(\tau) \propto p(s_0) \prod_{t=0}^{T-1} p(s_{t+1}|s_t, a_t) e^{r_\theta(s_t, a_t)}$$

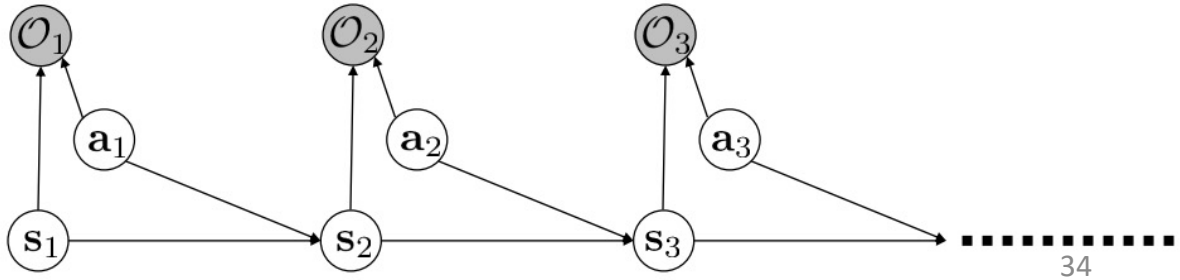
Initial state distribution

Environment dynamics

Exponentiated reward

Captures human notion of suboptimality*





Inverse Reinforcement Learning

MaxEnt IRL: Ziebart et al 2008

Objective:

$$\max_\theta J(\theta) = \max_\theta E_{\tau\sim\mathcal{D}}[\log p_\theta(\tau)]$$

where $p_\theta(\tau) \propto p(s_0)\prod_{t=0}^{T-1} p(s_{t+1}|s_t,a_t)e^{r_\theta(s_t,a_t)}$

Gradient takes an intuitive form

$$\frac{\partial}{\partial\theta}J(\theta) = E_{\mathcal{D}}[\frac{\partial}{\partial\theta}\log p_\theta(\tau)] == E_{\mathcal{D}}[\sum_{t=0}^{T}\frac{\partial}{\partial\theta}r_\theta(s_t,a_t)] - \frac{\partial}{\partial\theta}\log Z_\theta$$

$$= E_{\mathcal{D}}[\sum_{t=0}^{T}\frac{\partial}{\partial\theta}r_\theta(s_t,a_t)] - E_{p_\theta}[\sum_{t=0}^{T}\frac{\partial}{\partial\theta}r_\theta(s_t,a_t)]$$

Minimizes **forward KL** divergence between trajectory distribution of Expert and the Agent

f-IRL: Inverse Reinforcement Learning via State Marginal Matching

Tianwei Ni*, Harshit Sikchi*,
Yufei Wang*, Tejus Gupta*
Ben Eysenbach, Lisa Lee

* Equal Contribution, Dice rolling

Preliminaries: f-Divergence

Definition

Let P, Q be two distributions with density function p and q. For any convex, lower-semi-continuous function $f : R^+ \rightarrow R$, a statistical divergence can be defined as:

$$D_f(P||Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx$$

F-GAN: Training Generative Neural Networks using Variational Divergence Minimization. Nowozin et al.

Preliminaries: f-Divergence

$$D_f(P||Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx$$

Name	f-divergence $D_f(P \,\|\|\, Q)$	Generator $f(u)$	$h_f(u)$
FKL	$\int p(x) \log \frac{p(x)}{q(x)} dx$	$u \log u$	$-u$
RKL	$\int q(x) \log \frac{q(x)}{p(x)} dx$	$-\log u$	$1 - \log u$
JS	$\frac{1}{2} \int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} dx$	$u \log u - (1+u) \log \frac{1+u}{2}$	$-\log(1+u)$

Table 2: Selected list of f-divergences $D_f(P \,\|\, Q)$ with generator functions f and h_f defined in Theorem 4.1, where f is convex, lower-semicontinuous and $f(1) = 0$.

F-GAN: Training Generative Neural Networks using Variational Divergence Minimization. Nowozin et al.

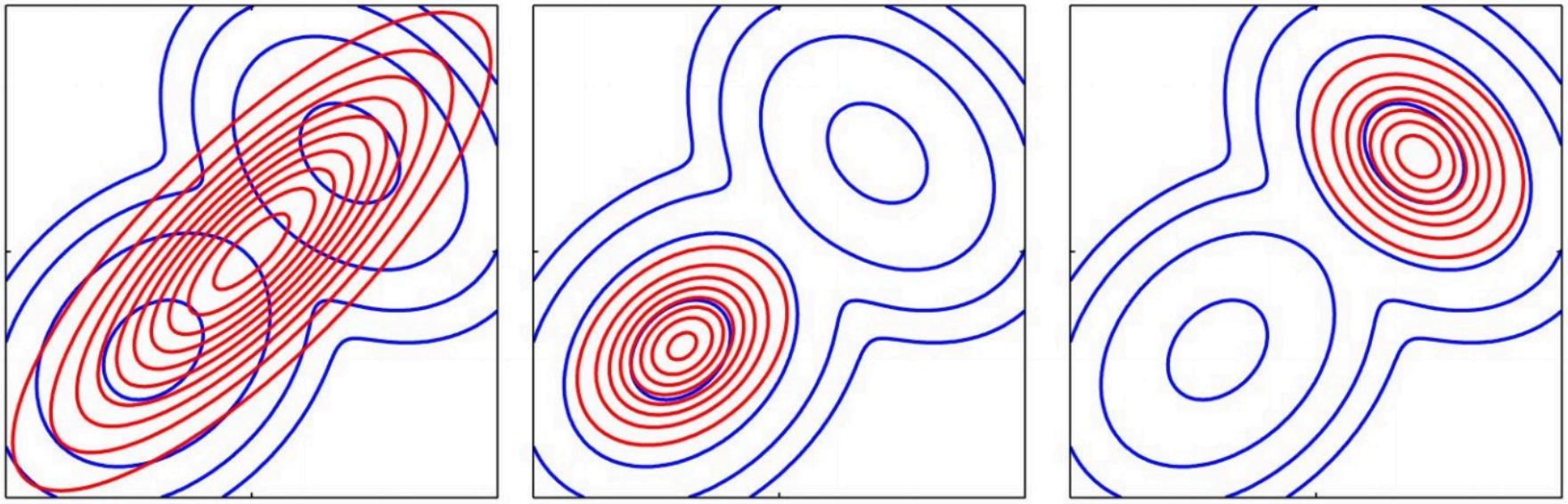
Preliminaries: KL divergences

- **Forward KL**: Max Likelihood, mode-covering
  - Application: **risk-seeking** behavior
- **Reverse KL**: mode-seeking
  - Application: **risk-averse** behavior

$$KL(p||q) = \sum_x p(x) log \frac{p(x)}{q(x)}$$

$$KL(q||p) = \sum_x q(x) log \frac{q(x)}{p(x)}$$

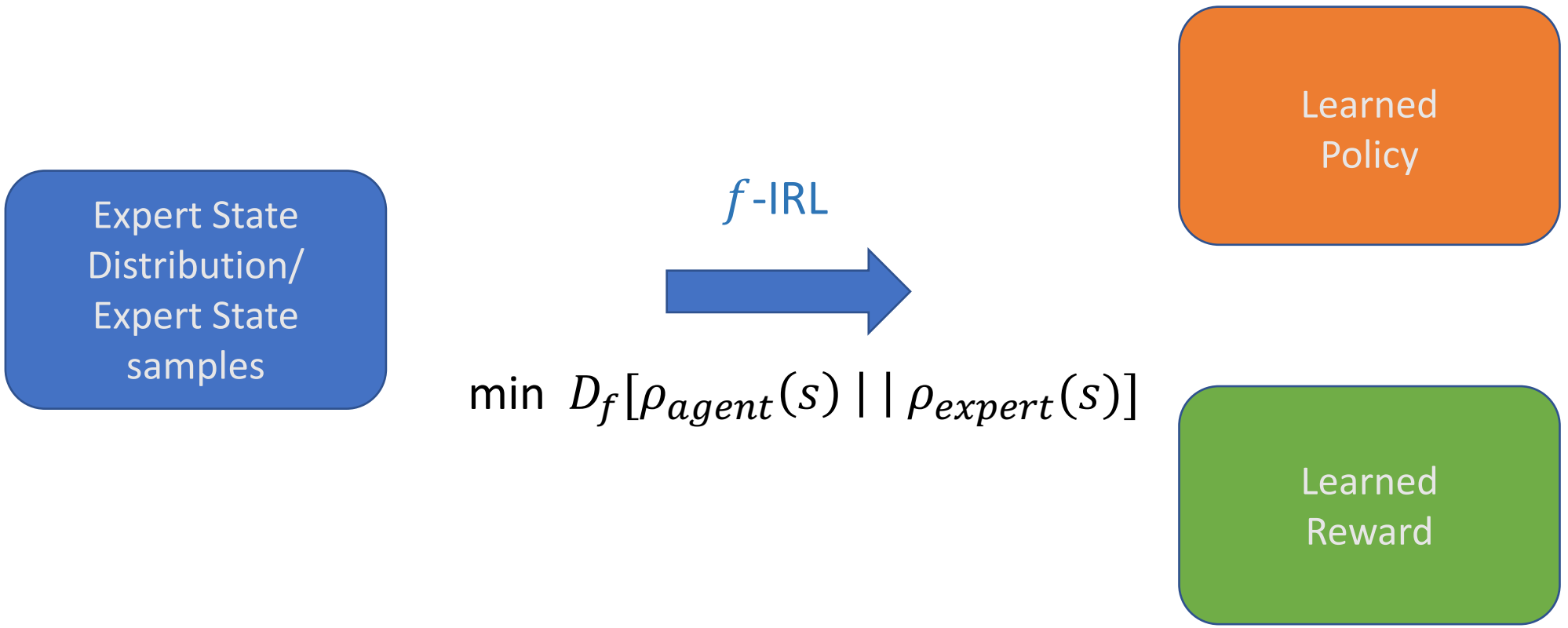
[Image courtesy of Bishop's book.]



Forward-KL

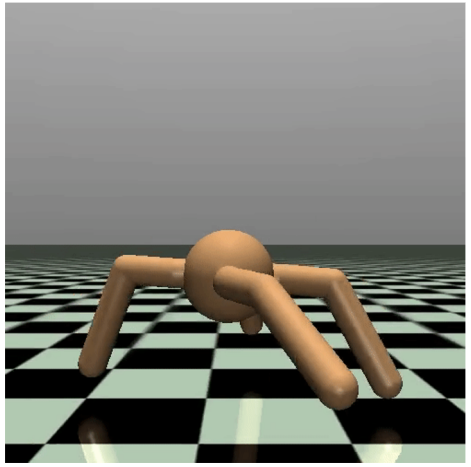
Reverse-KL

Learning a policy and reward function from expert state distribution

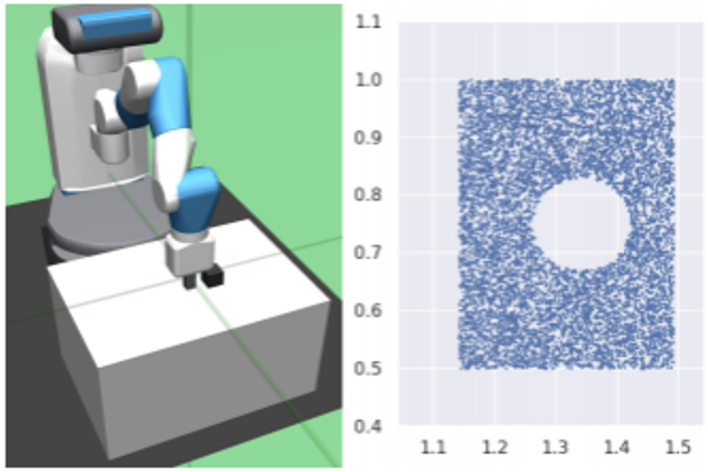


Why do State Marginal Matching for Inverse RL?

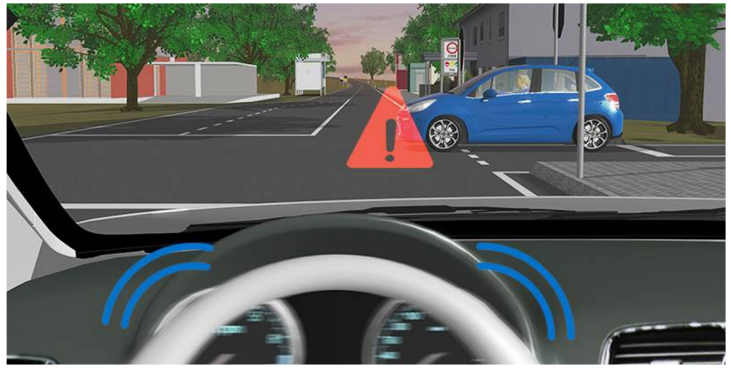
$r_\theta(s, a)$ ← Inverse RL — Expert Samples or Target State Density



Tough to specify demonstrations



Uniform Exploration



Safety

How our work differs from Previous Work?

Choice of f divergence

Can the method learn reward?

IL/IRL Method	Space	f-Divergence	Recover Reward?
MaxEntIRL (Ziebart et al.), GCL (Finn et al.)	τ	Forward Kullback-Leibler	✓
GAN-GCL (Finn et al.)	τ	Forward Kullback-Leibler*	✓
AIRL (Fu et al.), EAIRL (Qureshi et al)	τ	Forward Kullback-Leibler*	✓
EBIL (Liu et al.)	τ	Reverse Kullback-Leibler	✓
GAIL (Ho and Ermon)	s, a	Jensen-Shannon	×
f-MAX (Ghasemipour et al)	s, a	f-divergence	×
SMM (Lee et al.)	s	Reverse Kullback-Leibler	×
f-IRL (Our method)	s	f-divergence	✓

Works for **general** f-divergences and recovers policy **as well as the reward**

f-IRL : Objective

f-IRL derives the following surrogate:

$$\tilde{L}_f(\theta) = \frac{1}{\alpha T}\text{cov}_{\tau\sim\rho_{\hat{\theta}}(\tau)}\left(\sum_{t=1}^{T} h_f\left(\frac{\rho_E(s_t)}{\rho_{\hat{\theta}}(s_t)}\right), \sum_{t=1}^{T} r_\theta(s_t)\right)$$

Proof: Refer to Theorem 4.1 of f-IRL

Intuitively, the objective increases the reward for the trajectories that have higher sum of h-transformation of density ratios. Different choices of f leads to the corresponding h:

1.Forward KL divergence h(x) = -x
2.Reverse KL divergence h(x) = 1 - log(x)
3.Jensen Shannon Divergence h(x) = -log(1+x)

f-IRL : Objective

Surrogate: $$\tilde{L}_f(\theta) = \frac{1}{\alpha T}\text{cov}_{\tau\sim\rho_{\hat{\theta}}(\tau)}\left(\sum_{t=1}^{T} h_f\left(\frac{\rho_E(s_t)}{\rho_{\hat{\theta}}(s_t)}\right), \sum_{t=1}^{T} r_\theta(s_t)\right)$$

Algorithm 1: Inverse RL via State Marginal Matching (f-IRL)

Input : Expert state density $\rho_E(s)$ or expert observations s_E , f-divergence
Output: Learned reward r_θ, Policy π_θ
Initialize r_θ, and density estimation model (provided $\rho_E(s)$) or disciminator D_ω (provided s_E)
for $i \leftarrow 1$ **to** $Iter$ **do**
  $\pi_\theta \leftarrow$ MaxEntRL(r_θ) and collect agent trajectories τ_θ
  if *provided* $\rho_E(s)$ **then**
    Fit the density model $\hat{\rho}_\theta(s)$ to the state samples from τ_θ
  // provided s_E
  else
    Fit the discriminator D_ω by Eq. 4 using expert and agent state samples from s_E and τ_θ
  Compute sample gradient $\hat{\nabla}_\theta L_f(\theta)$ for Eq. 3 over τ_θ
  $\theta \leftarrow \theta - \lambda\hat{\nabla}_\theta L_f(\theta)$

f-IRL : Recovering Disentangled Rewards

A policy invariant class of reward functions

Ng et al (1999) : **Only** class of reward transformations that exhibit policy invariance

$$\hat{r}(s, a, s') = r_{\text{gt}}(s, a, s') + \underbrace{\gamma\Phi(s') - \Phi(s)}$$

Reward Shaping Term

We wish to learn a reward without the extra reward shaping term:

- Allows for recovering the ground truth reward function
- Allows for better transfer across change in dynamics

f-IRL : Recovering Disentangled Rewards

Theorem A.1. *Let $r_{gt}(s)$ be the expert reward, and T be a dynamics satisfying the decomposability condition as defined in [19]. Suppose f-IRL learns a reward r_{IRL} such that it produces an optimal policy in T: $Q^*_{r_{IRL},T}(s,a) = Q^*_{r_{gt},T}(s,a) - f(s)$,where $f(s)$ is an arbitrary function of the state. Then we have:*

$r_{IRL}(s) = r_{gt}(s) + C$ for some constant C, and thus $r_{IRL}(s)$ is robust to all dynamics.

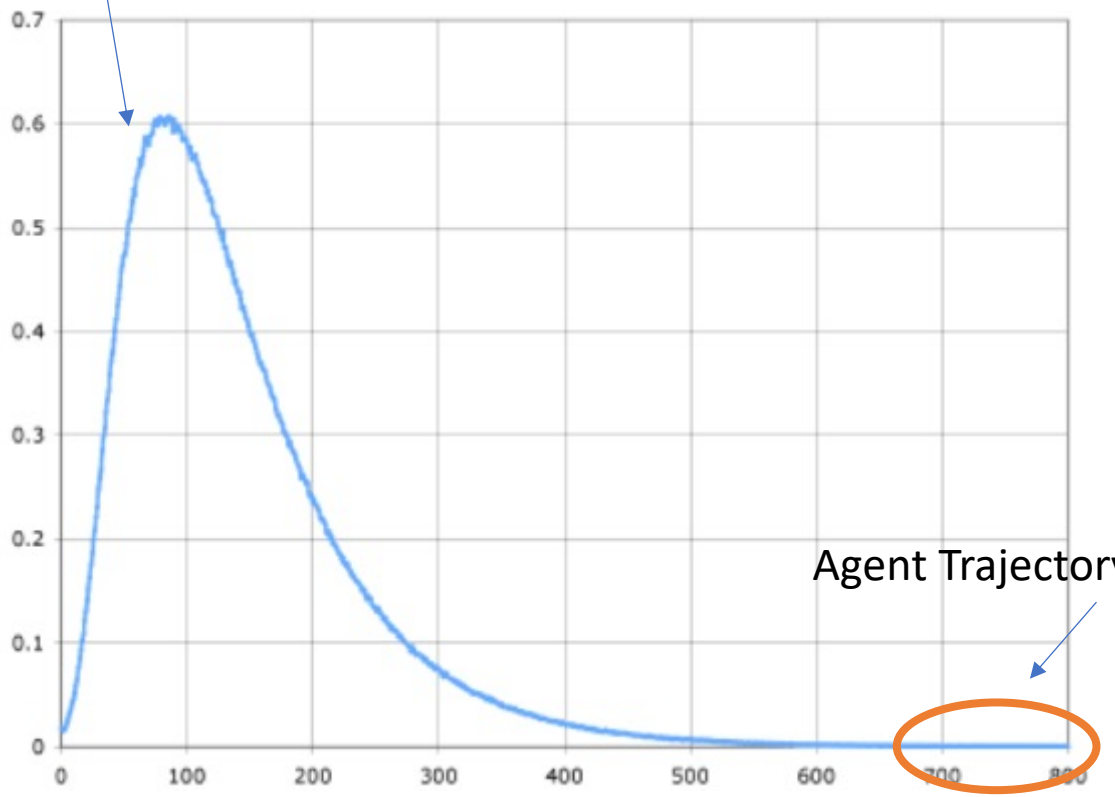
Takeaway:

If the ground truth reward function is only a function of states, f-IRL recovers ground truth reward up to a constant.

f-IRL : Practical Optimization Issues

$$\tilde{L}_f(\theta) = \frac{1}{\alpha T}\text{cov}_{\tau\sim\rho_{\hat{\theta}}(\tau)}\left(\sum_{t=1}^{T} h_f\left(\frac{\rho_E(s_t)}{\rho_{\hat{\theta}}(s_t)}\right), \sum_{t=1}^{T} r_\theta(s_t)\right)$$

Expert Trajectory Distribution



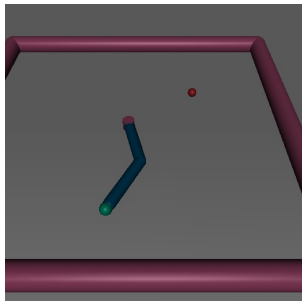
Agent Trajectory Distribution

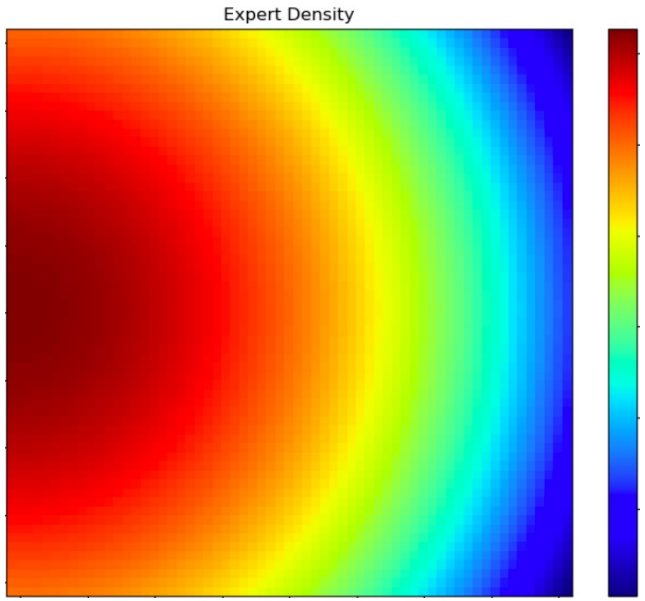
- Poor gradients when Agent's distribution is far-off the Expert's state marginal distribution.
- Use a heuristic fix where we sample trajectories from a mixture distribution of Agent and Expert, rather than Agent alone.
- For a more discussion and a principled solution to this issue:
  Refer: f-EBM Yu, Stephano Ermon 2020

Experiments

Reward function learned by f-IRL

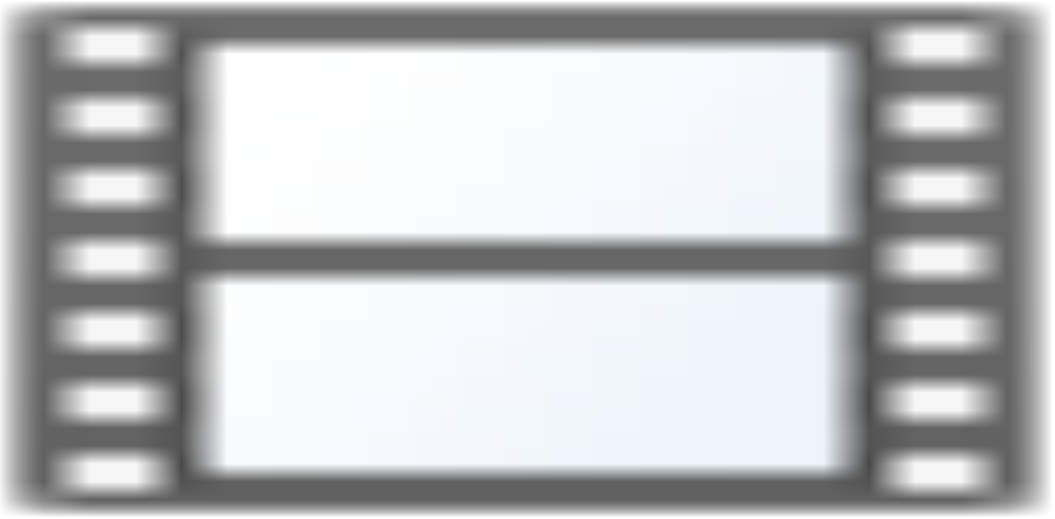
Reacher-v2





Higher

Expert State Density

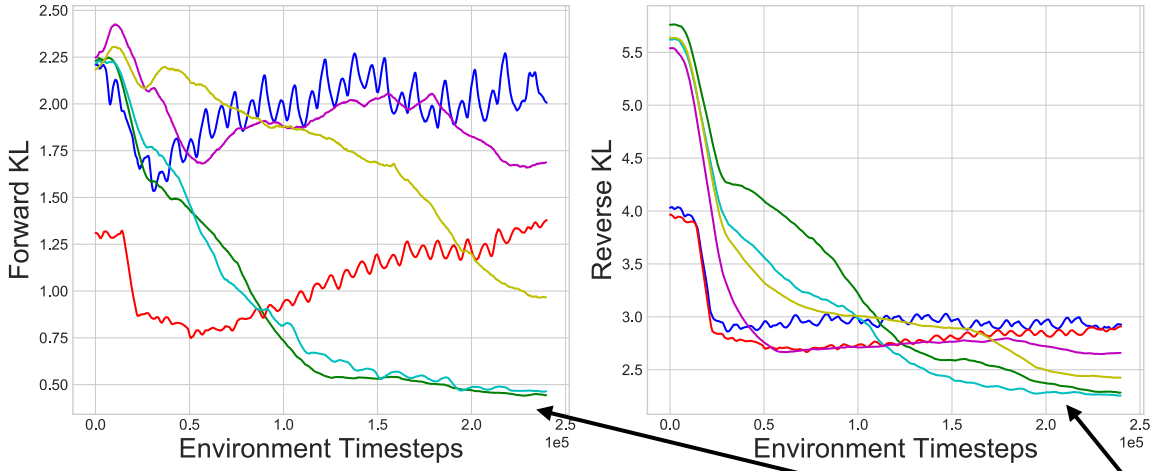


Rewards and trajectory evolution during training

f-IRL is more sample efficient than baselines and produces policies which are closer to expert

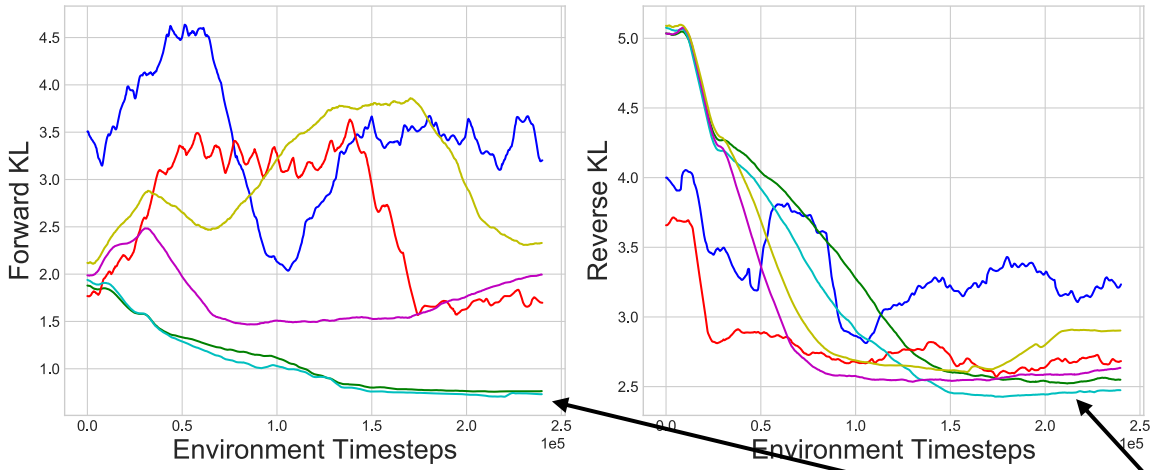
State Marginal Matching Performance (Reacher-v2)

Expert Density: Gaussian



f-IRL (Ours)

Expert Density: Mixture of gaussian

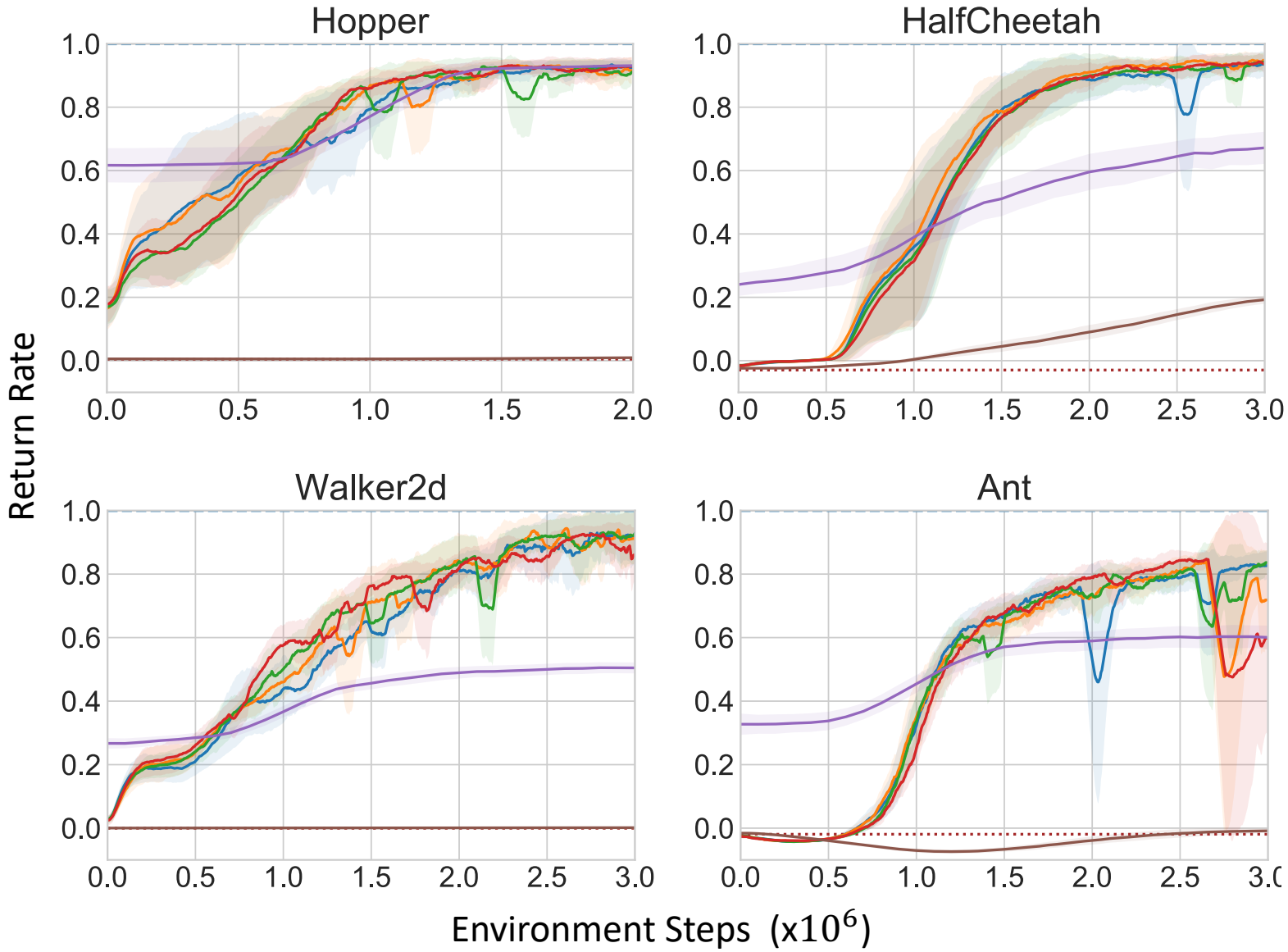


f-IRL (Ours)

f-max-rkl fkl (f-IRL) GAIL js (f-IRL) MaxEnt IRL rkl (f-IRL)

Performance on Imitation Learning Benchmarks

(1 expert trajectory)



Quality of Learned Rewards

(1 expert trajectory)

Method	Hopper	Walker2d	HalfCheetah	Ant
AIRL	-	-	-0.03	-
MaxEnt IRL	**0.93**	**0.92**	0.96	0.79
f-IRL	**0.93**	0.88	**1.02**	**0.82**

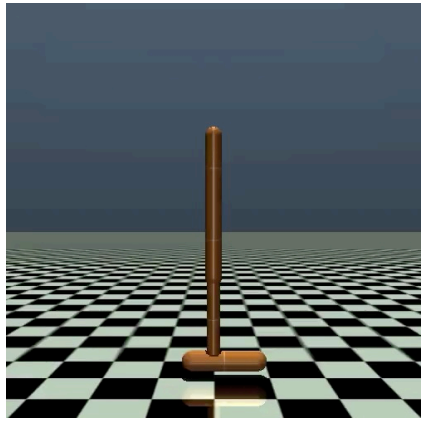
Table 4: The ratios of final return of the obtained policy against expert return across IRL methods. We average f-IRL over FKL, RKL, and JS.

Comparison of Learned Policies

Environment: Hopper-v2

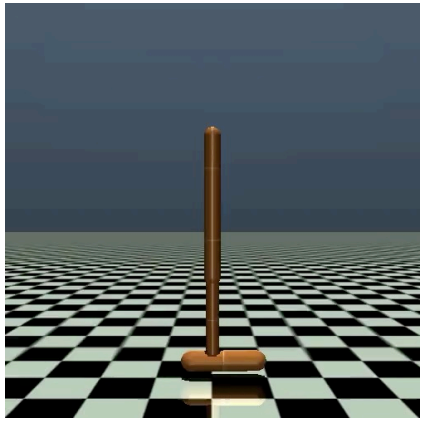
Expert Trajectories available = 1

Return
3592.63



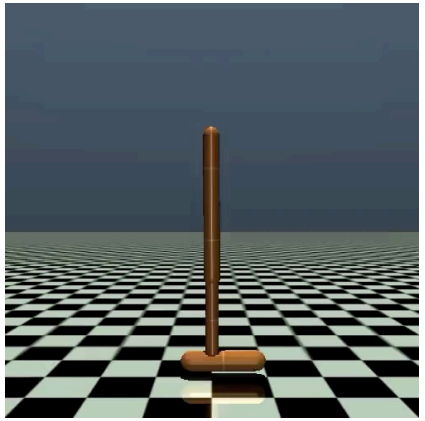
Expert agent

Return
3349.62



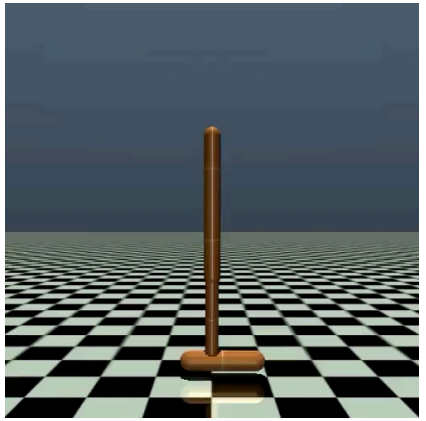
f-MAX
(Ghasemipour et
al. 2019)

Return
3329.94



F-IRL (Our method)

Return
3341.15



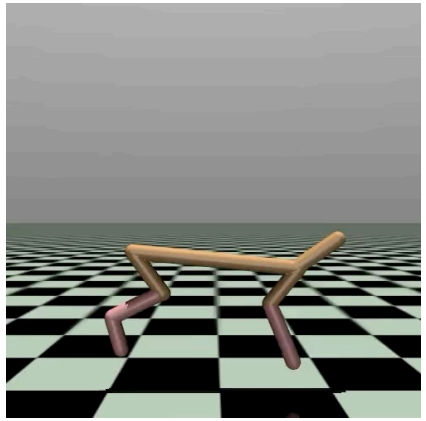
F-IRL (Our method)
– Trained from
scratch on obtained
reward

Comparison of Learned Policies

Environment: HalfCheetah-v2

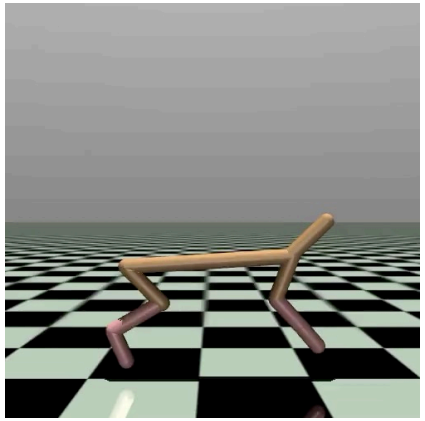
Expert Trajectories available = 1

Return
12427.49



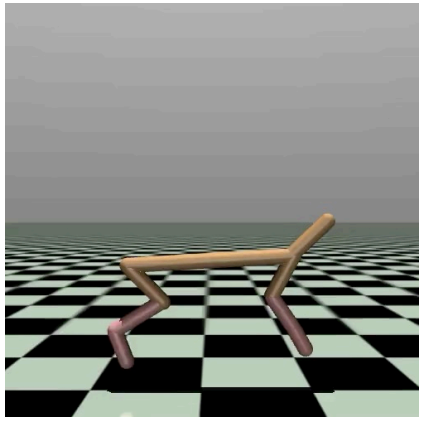
Expert agent

Return
8688.37



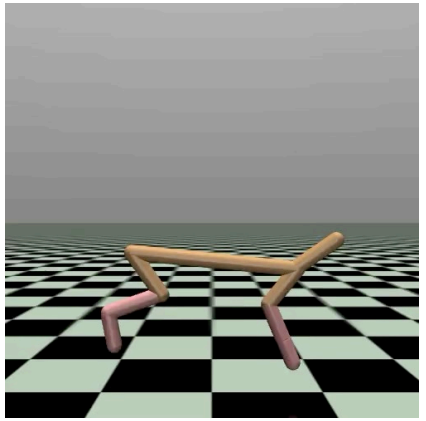
f-MAX
(Ghasemipour et al. 2019)

Return
11612.46



F-IRL (Our method)

Return
12676.03



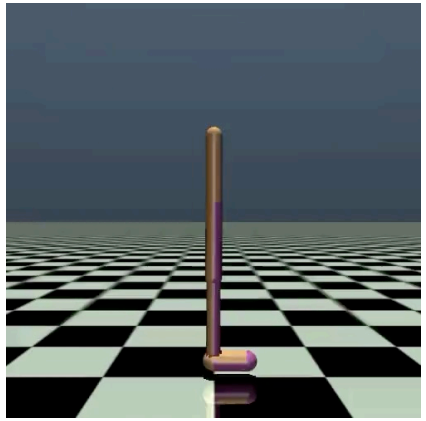
F-IRL (Our method) – Trained from scratch on obtained reward

Comparison of Learned Policies

Environment: Walker2d-v2

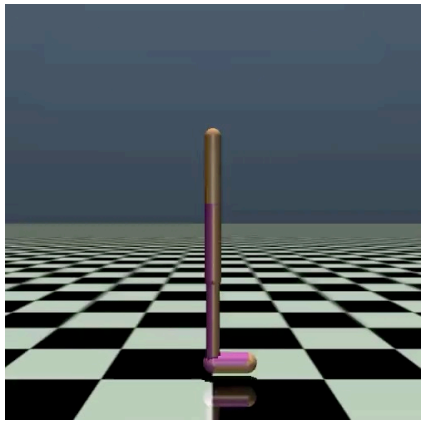
Expert Trajectories available = 1

Return
5344.21



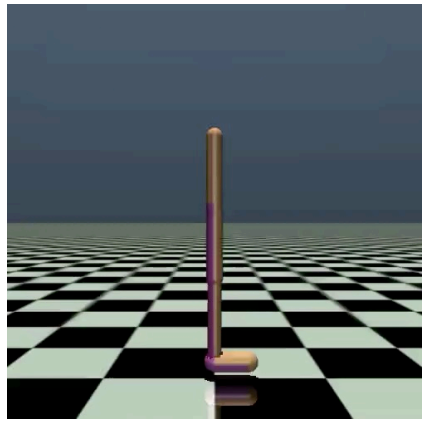
Expert agent

Return
2683.11



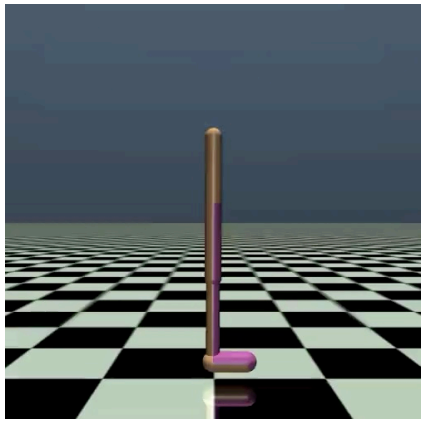
f-MAX
(Ghasemipour et al. 2019)

Return
4927.02



F-IRL (Our method)

Return
4916.67



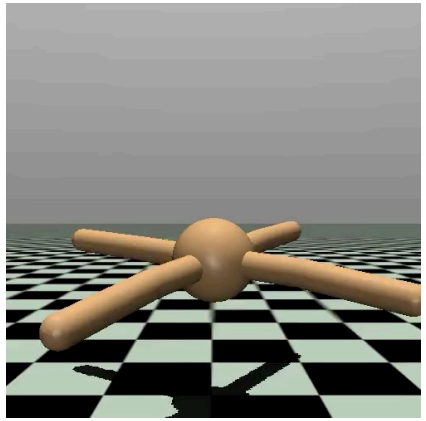
F-IRL (Our method) – Trained from scratch on obtained reward

Comparison of Learned Policies

Environment: Ant-v2

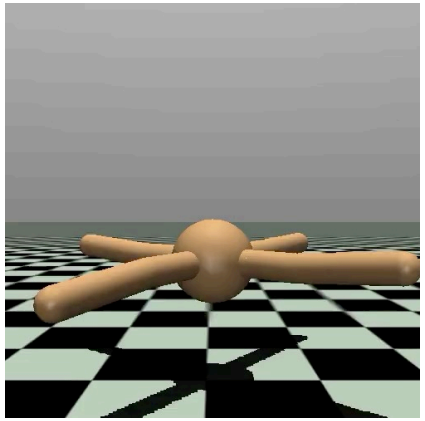
Expert Trajectories available = 1

Return
5926.18



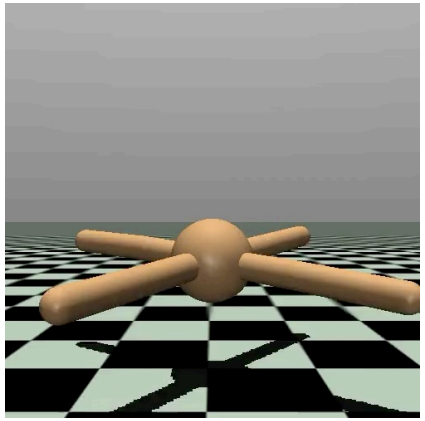
Expert agent

Return
3585.03



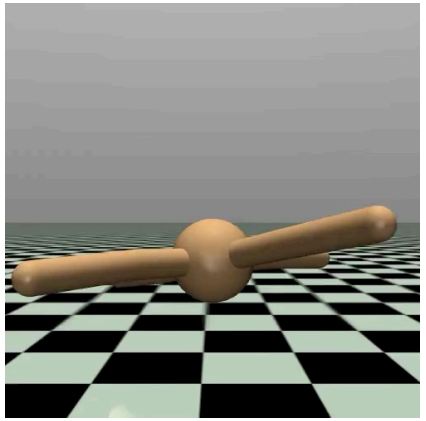
f-MAX
(Ghasemipour et al. 2019)

Return
4859.86



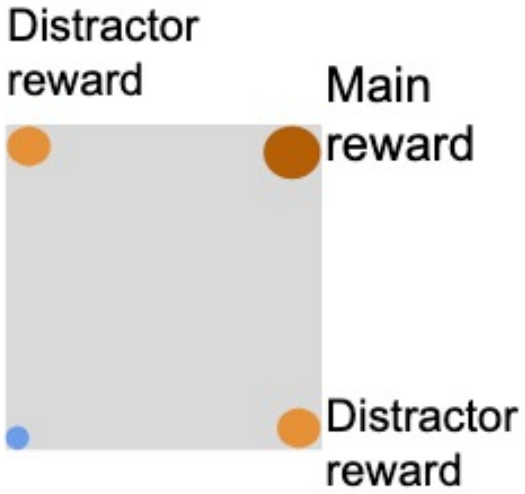
F-IRL (Our method)

Return
4859.46

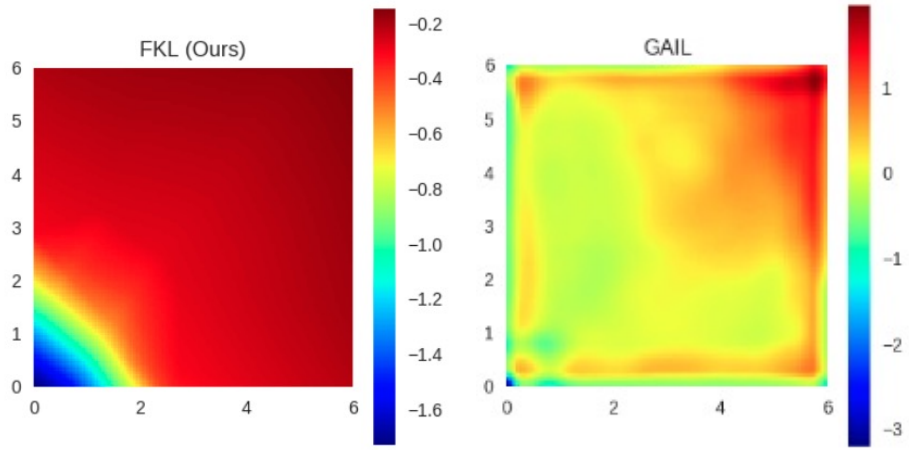


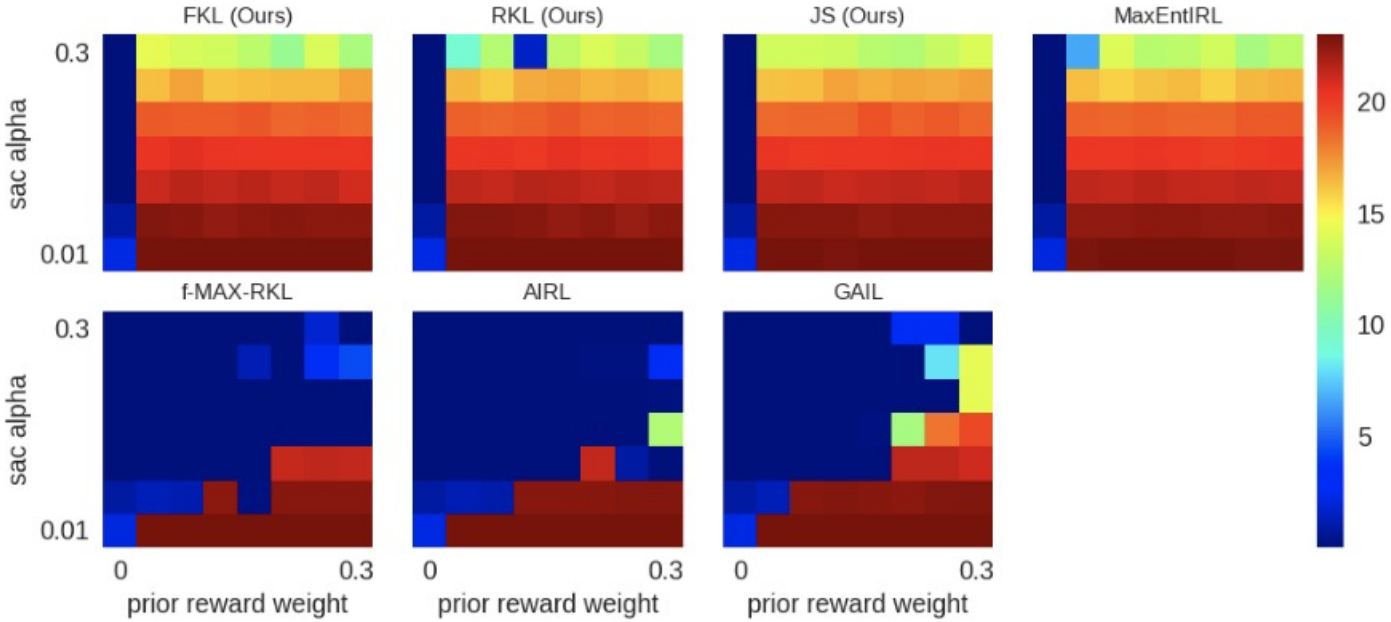
F-IRL (Our method) – Trained from scratch on obtained reward

Reward as a prior for Hard Exploration Tasks



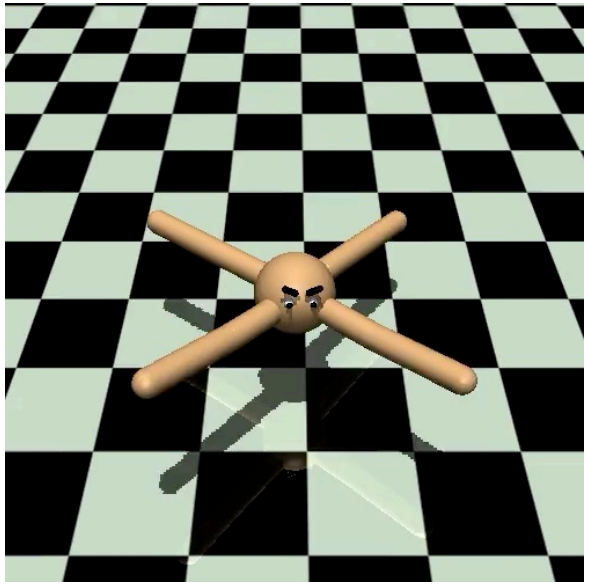
$$r(s) = r_{\text{task}}(s) + \lambda(\gamma r_{\text{prior}}(s') - r_{\text{prior}}(s))$$





Learned rewards comparison

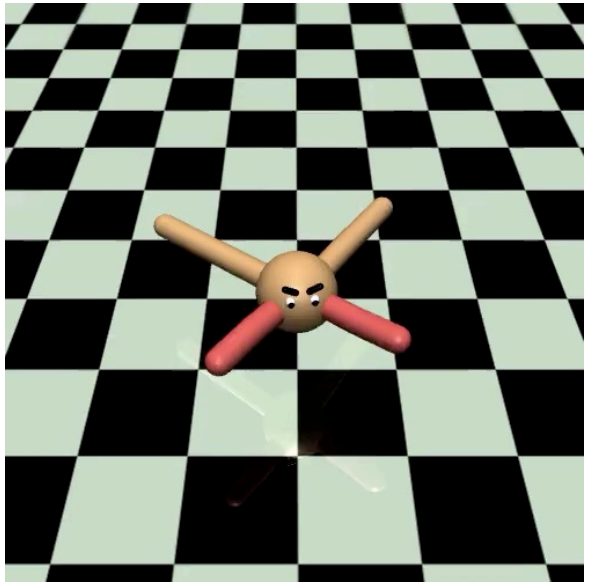
Transferring learned rewards



Healthy Ant
(Expert)

Extract a reward function using f-IRL

Learned Reward



Disabled Ant
(Learned policy)

Train SAC agent on the Disabled Ant

Transferring learned rewards - Quantitatively

32 expert trajectories

Policy Transfer using GAIL	AIRL	MaxEnt IRL	f-IRL	Ground-truth Reward
-29.9	130.3	**145.5**	141.1	315.5

Table 5: Returns obtained after transferring the policy/reward on modified Ant environment using different IL methods.

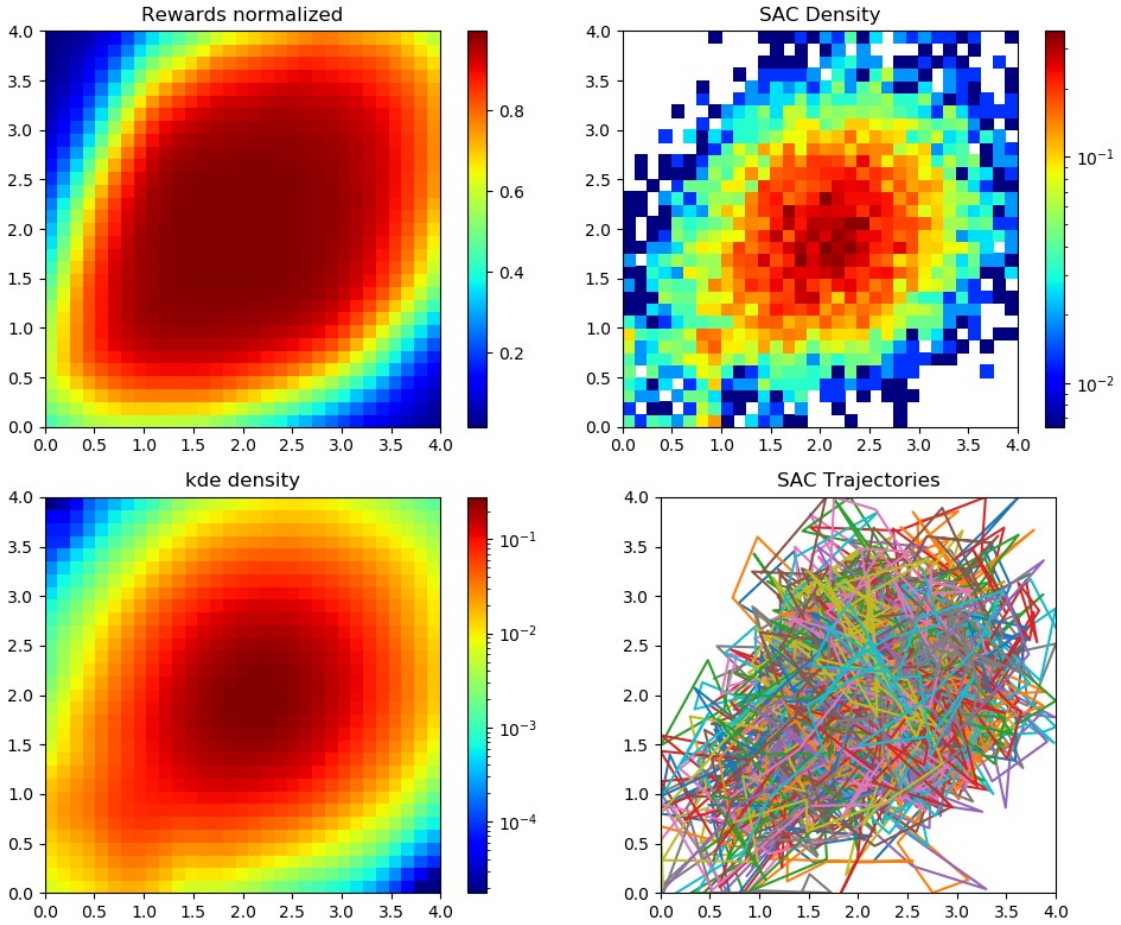
50 expert trajectories

Policy Transfer using GAIL	AIRL	MaxEntIRL	f-IRL	Ground-truth Reward
-29.9	130.3	145.5	**232.5**	315.5

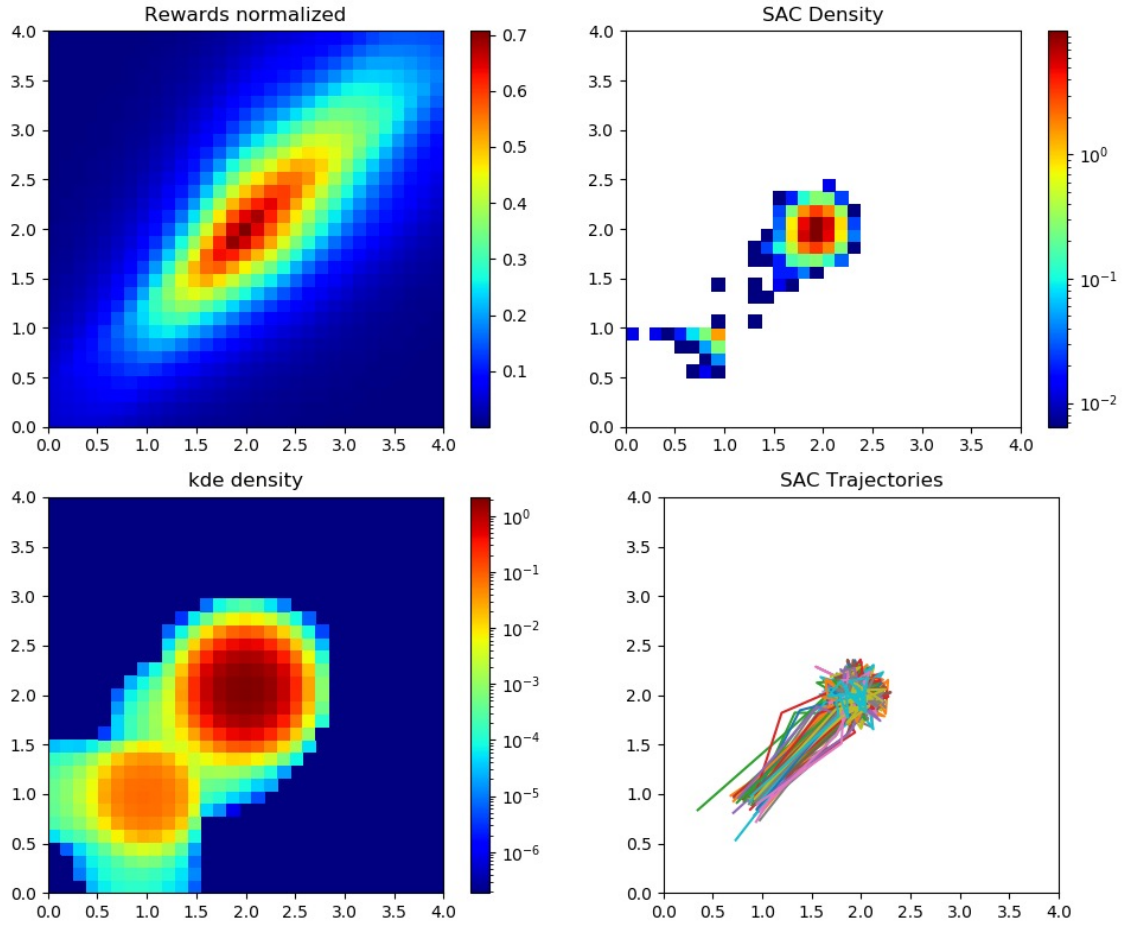
Table 8: Returns obtained after transferring the policy/reward on modified Ant environment using different IL methods. In this case, we report the performance of best seed with a maximum of 50 expert trajectories.

Bonus: Reward maps for Different Divergences

Forward-KL

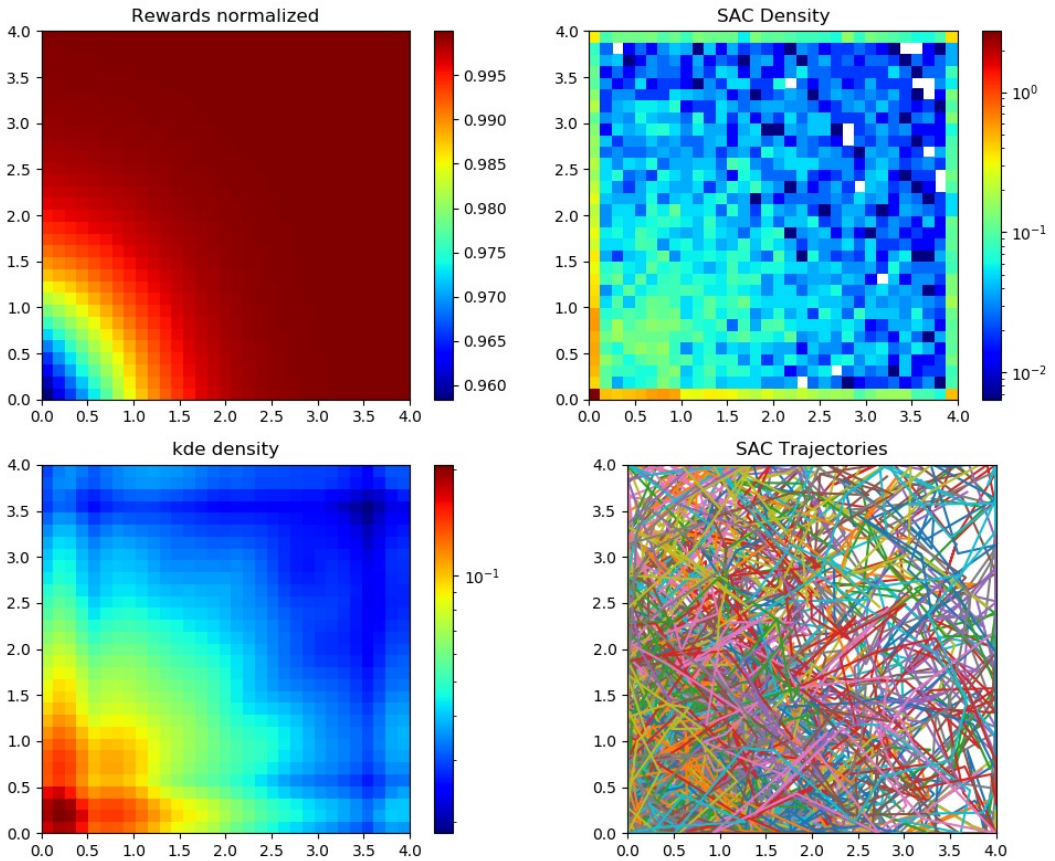


Reverse-KL

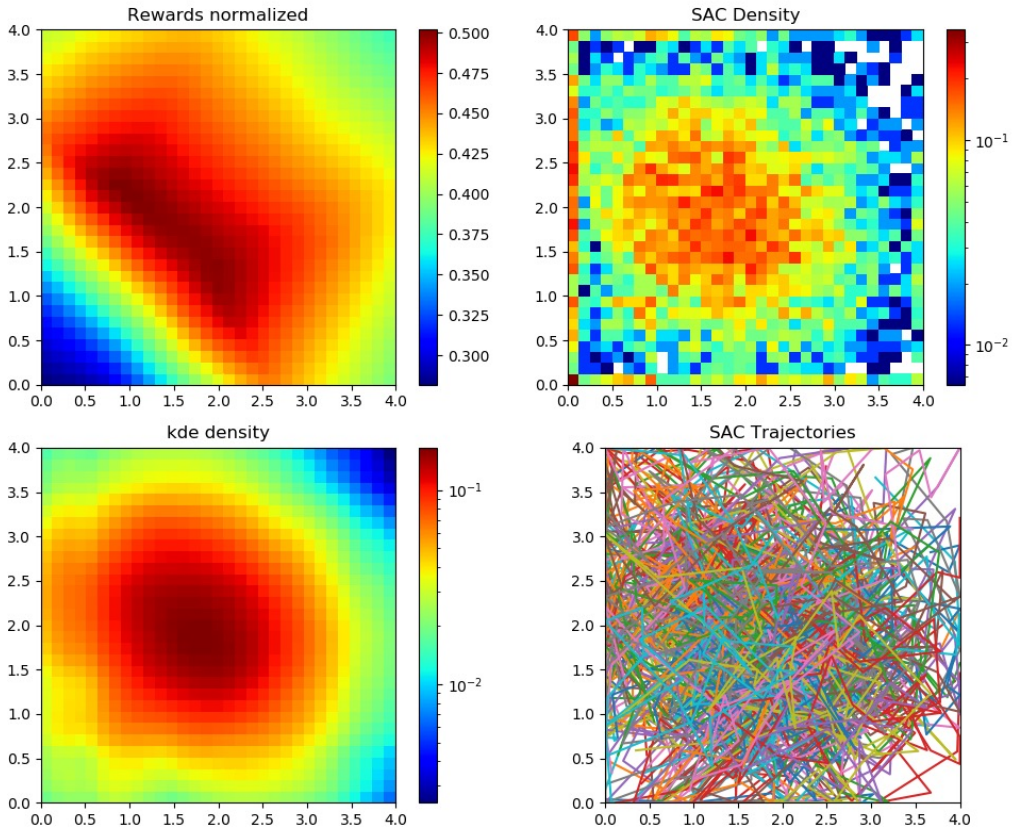


Bonus: Reward maps for Different Divergences

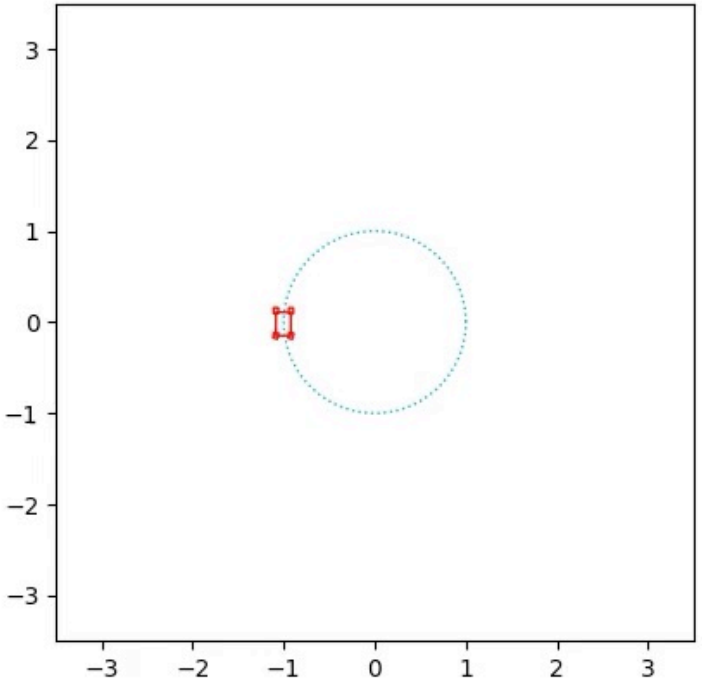
Forward-KL



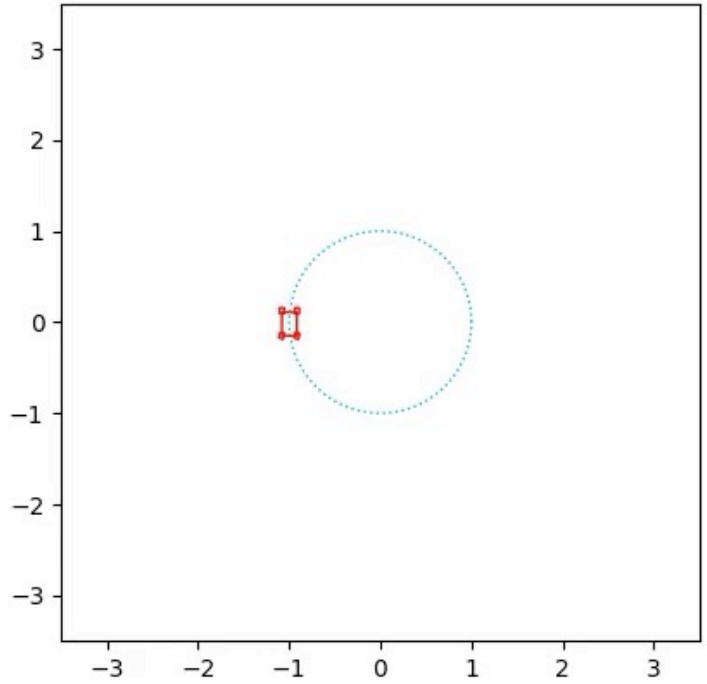
Reverse-KL



Bonus: Drifting



Normal Speed



High Speed

*Work in Progress

Characteristics of a good Imitation Algorithm

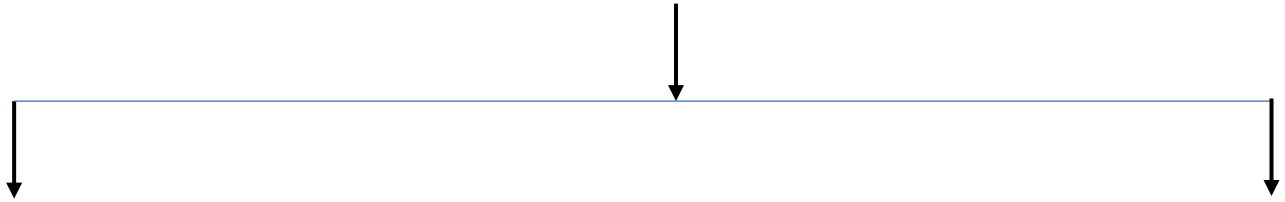
Sample Efficiency in environment timesteps:

We want the agent to learn a policy close to expert with as **few environment steps** as possible.

Sample Efficiency in expert trajectories:

We want the agent to learn a policy close to expert with as **few expert samples** as possible.

Imitation Learning: Another Classification



On-Policy Imitation

- Gradient for policy update/ reward update requires samples from current policy
- All the methods we have seen till now.
- Algorithms:
  - MaxEntIRL (Ziebart '08)
  - GAIL (Ho and Ermon '16)
  - GCL, GAN-GCL (Finn' 16, 17)
  - AIRL (Fu '18)
  - F-MAX (Ghasemipour et al' 19)
  - f-IRL (Our method)

Off-Policy Imitation

- Utilize all the samples we have seen till now for the reward/policy update.
  Use a replay buffer!
- Algorithms:
  - Discriminator Actor-Critic (Kostrikov et al 18)
    SQIL (Reddy et al 19)
  - **ValueDICE** (Kostrikov et al 20)
  - DRIL (Brantley et al 20)

None of them recover reward!

DRIL: Disagreement Regularized Imitation Learning

Algorithm:

- Train an ensemble of policies using Behavior cloning on the dataset.
- Train a new policy with

$$J_{\text{alg}}(\pi) = \underbrace{\mathbb{E}_{s\sim d_{\pi^\star}}\left[\|\pi^\star(\cdot|s) - \pi(\cdot|s)\|\right]}_{J_{\text{BC}}(\pi)} + \underbrace{\mathbb{E}_{s\sim d_\pi, a\sim\pi(\cdot|s)}\left[C_{\text{U}}(s,a)\right]}_{J_{\text{U}}(\pi)}$$

where

$$C_{\text{U}}(s,a) = \text{Var}_{\pi\sim\Pi_{\text{E}}}(\pi(a|s)) = \frac{1}{E}\sum_{i=1}^{E}\left(\pi_i(a|s) - \frac{1}{E}\sum_{i=1}^{E}\pi_i(a|s)\right)^2$$

ValueDICE

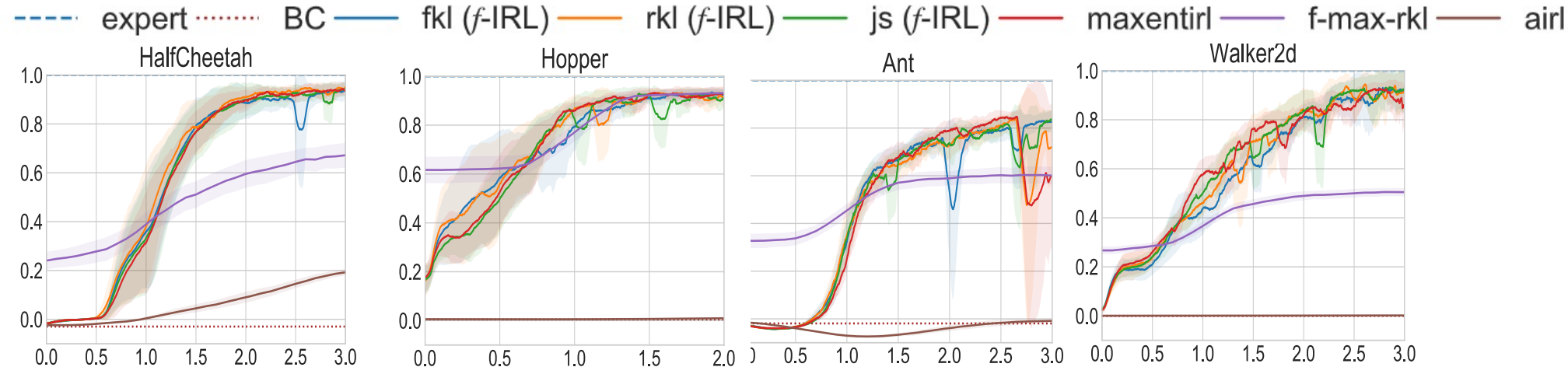
- Variational Representation (Donsker-Varadhan) of KL divergence.
- Objective

$$\max_{\pi} \min_{\nu:\mathcal{S}\times\mathcal{A}\to\mathbb{R}} J_{\text{DICE}}(\pi,\nu) := \log \mathop{\mathbb{E}}_{(s,a)\sim d^{\text{exp}}}[e^{\nu(s,a)-\mathcal{B}^{\pi}\nu(s,a)}] - (1-\gamma)\cdot \mathop{\mathbb{E}}_{\substack{s_0\sim p_0(\cdot),\\ a_0\sim\pi(\cdot|s_0)}}[\nu(s_0,a_0)].$$

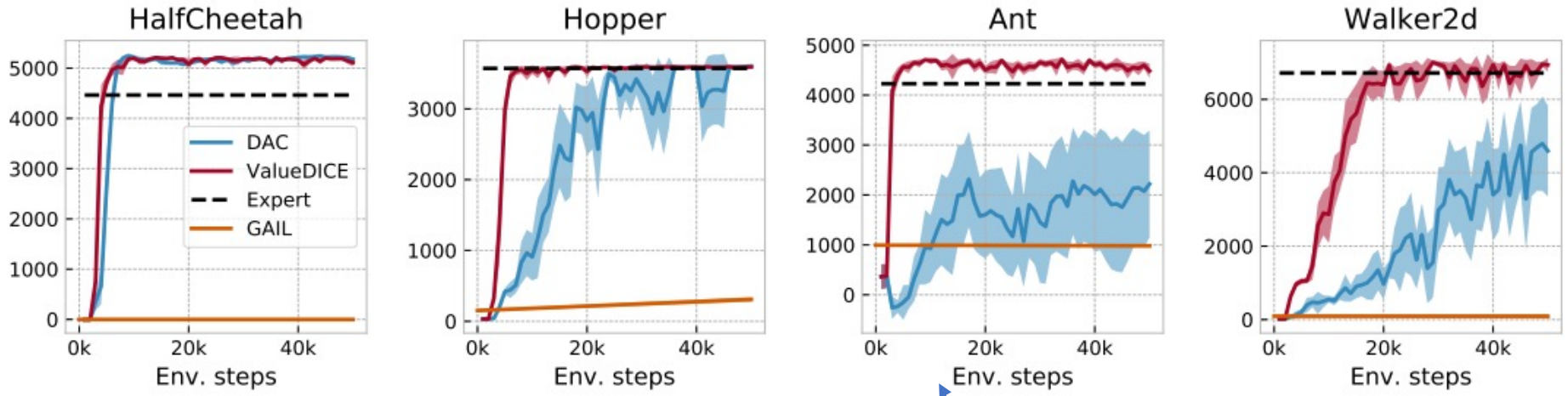
- A series of algorithms relying on Convex Programming instead of Dynamics Programming.
  - DualDICE (Nachum et al. 2019) - Off-Policy Evaluation
  - AlgaeDICE (Nachum et al. 2019) - Off-Policy RL (competitive to SAC)
  - ValueDICE (Kostrikov et al. 2020) - Off-Policy Imitation Learning
- Recommended talk on convex duality for RL: Link

Progress in Sample Efficiency (environment timesteps)

On-policy Imitation Algorithms



Off-policy Imitation Algorithm



Imitating complex agents in Maybe 4 or 5 episodes? Huge Leap!

Conclusions: Which algorithm to use?

Do you have expert supervision in entire state-space?

-> Can use Dagger and get linear regret O(ϵT) instead of Behavior cloning O(ϵT^2)

Do you need to extract a reward function?

-> Can use Efficient Inverse Reinforcement Learning methods

-> MaxEntIRL is you have trajectory data / f-IRL if you have only observation data

If only a policy is required:

-> Can use efficient off-policy Imitation algorithms

-> Ex: valueDICE, DIRL

Discussion

- What are the utilities of extracting a reward function beside the ones shown in the presentation?
- What is the best mode to transfer behavior from one agent to another?
  Policy ? Reward Function? State Marginals?
  Is there a way to characterize when each one is useful?
- Applications of state marginal matching?
- What's next?

  - Off-Policy Inverse Reinforcement Learning

  - Efficient Model-based IL/IRL

  - Batch Imitation Learning

Discussion

- Some claims in the f-IRL paper:

1. GAN-GCL , EAIRL, AIRL are **NOT** minimizing forward KL divergence in trajectory space.

2. AIRL is **NOT** minimizing reverse KL divergence in State-action marginal space.

A connection between GAN, IRL and Energy based models – Finn et al
Learning robust rewards with Adversarial Inverse Reinforcement Learning – Fu et al
Adversarial imitation by variational inverse reinforcement learning – Qureshi et al.