

# Inverse RL

Saurabh Gupta

# Where do rewards come from?

Computer Game



Robotic Setup



Very often, easier to provide expert demonstrations.

**Inverse RL:** Inferring reward functions from expert demonstrations.

# Inverse RL

## 1. Introduction

The inverse reinforcement learning (IRL) problem can be characterized informally as follows (Russell, 1998):

**Given** 1) measurements of an agent's behavior over time, in a variety of circumstances, 2) if needed, measurements of the sensory inputs to that agent; 3) if available, a model of the environment.

**Determine** the reward function being optimized.

# Inverse RL

- Focus is not on improving sample complexity
- Instead, focus is on recovering the reward function using as few expert demonstrations
- Methods assume access to the environment either via the model or via trajectory executions
- Can't we just use expert behavior as demonstrations, and do behavior cloning?

# Inverse RL

- There isn't one right reward function
- Heuristics for obtaining a reward function:
  - Reward functions that maximally differentiate expert policy from other policies
  - Can be solved via linear programming
- Settings
  - Known model, analytical access to the policy
  - Large MDPs w/ rewards that are linear functions of state features
  - Large MDPs with observer trajectories

# Inverse RL

$$\pi(s) \equiv a_1$$

**Theorem 1 (Bellman Equations)** *Let an MDP  $M = (S, A, \{P_{sa}\}, \gamma, R)$  and a policy  $\pi : S \mapsto A$  be given. Then, for all  $s \in S, a \in A, V^\pi$  and  $Q^\pi$  satisfy*

$$V^\pi(s) = R(s) + \gamma \sum_{s'} P_{s\pi(s)}(s') V^\pi(s') \quad (1)$$

$$Q^\pi(s, a) = R(s) + \gamma \sum_{s'} P_{sa}(s') V^\pi(s') \quad (2)$$

**Theorem 2 (Bellman Optimality)** *Let an MDP  $M = (S, A, \{P_{sa}\}, \gamma, R)$  and a policy  $\pi : S \mapsto A$  be given. Then  $\pi$  is an optimal policy for  $M$  if and only if, for all  $s \in S,$*

$$\pi(s) \in \arg \max_{a \in A} Q^\pi(s, a) \quad (3)$$

**Theorem 3** *Let a finite state space  $S,$  a set of actions  $A = \{a_1, \dots, a_k\},$  transition probability matrices  $\{P_a\},$  and a discount factor  $\gamma \in (0, 1)$  be given. Then the policy  $\pi$  given by  $\pi(s) \equiv a_1$  is optimal if and only if, for all  $a = a_2, \dots, a_k,$  the reward  $R$  satisfies*

$$(P_{a_1} - P_a)(I - \gamma P_{a_1})^{-1} R \succeq 0 \quad (4)$$

**Proof.** Since  $\pi(s) \equiv a_1,$  Equation (1) may be written  $V^\pi = R + \gamma P_{a_1} V^\pi.$  Thus,<sup>1</sup>

$$V^\pi = (I - \gamma P_{a_1})^{-1} R \quad (5)$$

Substituting Equation (2) into (3) from Theorem 2, we see that  $\pi \equiv a_1$  is optimal if and only if

$$a_1 \equiv \pi(s) \in \arg \max_{a \in A} \sum_{s'} P_{sa}(s') V^\pi(s') \quad \forall s \in S$$

$$\Leftrightarrow \sum_{s'} P_{sa_1}(s') V^\pi(s')$$

$$\geq \sum_{s'} P_{sa}(s') V^\pi(s') \quad \forall s \in S, a \in A$$

$$\Leftrightarrow P_{a_1} V^\pi \succeq P_a V^\pi \quad \forall a \in A \setminus a_1$$

$$\Leftrightarrow P_{a_1} (I - \gamma P_{a_1})^{-1} R$$

$$\succeq P_a (I - \gamma P_{a_1})^{-1} R \quad \forall a \in A \setminus a_1$$

where the last implication in this derivation used Equation (5). This completes the proof.  $\square$

# Inverse RL (LP Formulation, Penalty Terms)

- Maximize quality of optimal action over next-best action:

$$\bullet \sum_{s \in S} \left( Q^\pi(s, a_1) - \max_{a \in A \setminus a_1} Q^\pi(s, a) \right)$$

- Small rewards are simpler:  $-\lambda \|R\|_1$

$$\text{maximize } \sum_{i=1}^N \min_{a \in \{a_2, \dots, a_k\}} \left\{ (\mathbf{P}_{a_1}(i) - \mathbf{P}_a(i)) \right. \\ \left. (\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1} \mathbf{R} \right\} - \lambda \|\mathbf{R}\|_1$$

$$\text{s.t. } (\mathbf{P}_{a_1} - \mathbf{P}_a) (\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1} \mathbf{R} \succeq 0 \\ \forall a \in A \setminus a_1$$

$$\bullet \quad |\mathbf{R}_i| \leq R_{\max}, \quad i = 1, \dots, N$$



# Inverse RL (w/ linear function approximations)

- Linear approximation of reward:

- $$R(s) = \alpha_1 \phi_1(s) + \alpha_2 \phi_2(s) + \dots + \alpha_d \phi_d(s)$$

- $$V^\pi = \alpha_1 V_1^\pi + \dots + \alpha_d V_d^\pi.$$

- Linear program, with an appropriately chosen penalty function  $p$ :

$$\begin{aligned} & \text{maximize } \sum_{s \in S_0} \min_{a \in \{a_2, \dots, a_k\}} \{ \\ & \quad p(\mathbb{E}_{s' \sim P_{s a_1}} [V^\pi(s')] - \mathbb{E}_{s' \sim P_{s a}} [V^\pi(s')]) \} \\ & \text{s.t. } |\alpha_i| \leq 1, \quad i = 1, \dots, d \end{aligned}$$



# Inverse RL (Sampled Trajectories)

- Monte-carlo estimate of values:

- $$\hat{V}_i^\pi(s_0) = \phi_i(s_0) + \gamma\phi_i(s_1) + \gamma^2\phi_i(s_2) + \dots$$

The “inductive step” of the algorithm is as follows: We have some set of policies  $\{\pi_1, \dots, \pi_k\}$ , and want to find a setting of the  $\alpha_i$ s so that the resulting reward function (hopefully) satisfies

- $$V^{\pi^*}(s_0) \geq V^{\pi_i}(s_0), \quad i = 1, \dots, k \quad (12)$$

- $$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^k p \left( \hat{V}^{\pi^*}(s_0) - \hat{V}^{\pi_i}(s_0) \right) \\ \text{s.t.} \quad & |\alpha_i| \leq 1, \quad i = 1, \dots, d \end{aligned}$$

- $\alpha_d \phi_d$ . We then find a policy  $\pi_{k+1}$  that maximizes  $V^\pi(s_0)$  under  $R$ , add  $\pi_{k+1}$  to the current set of policies, and continue (for some large number of iterations, until we find an  $R$  with which we are “satisfied”).

# Inverse RL (Experiments)

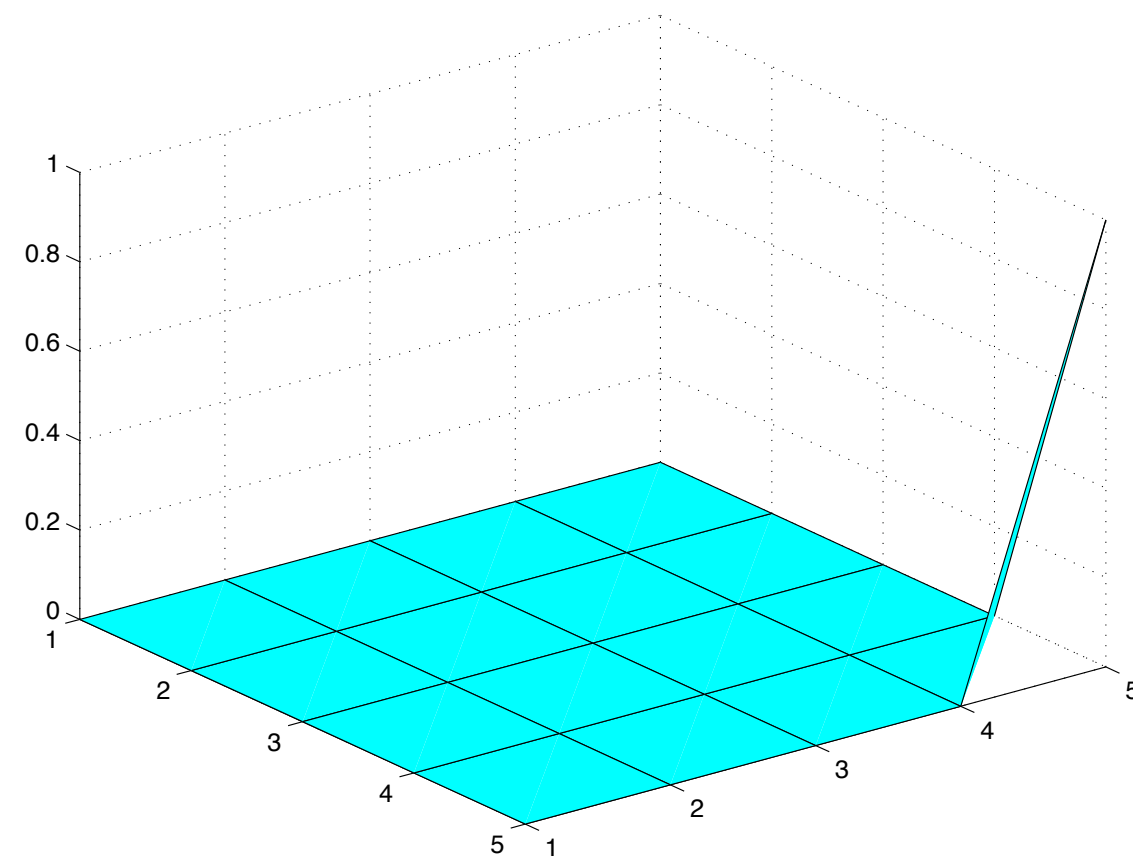
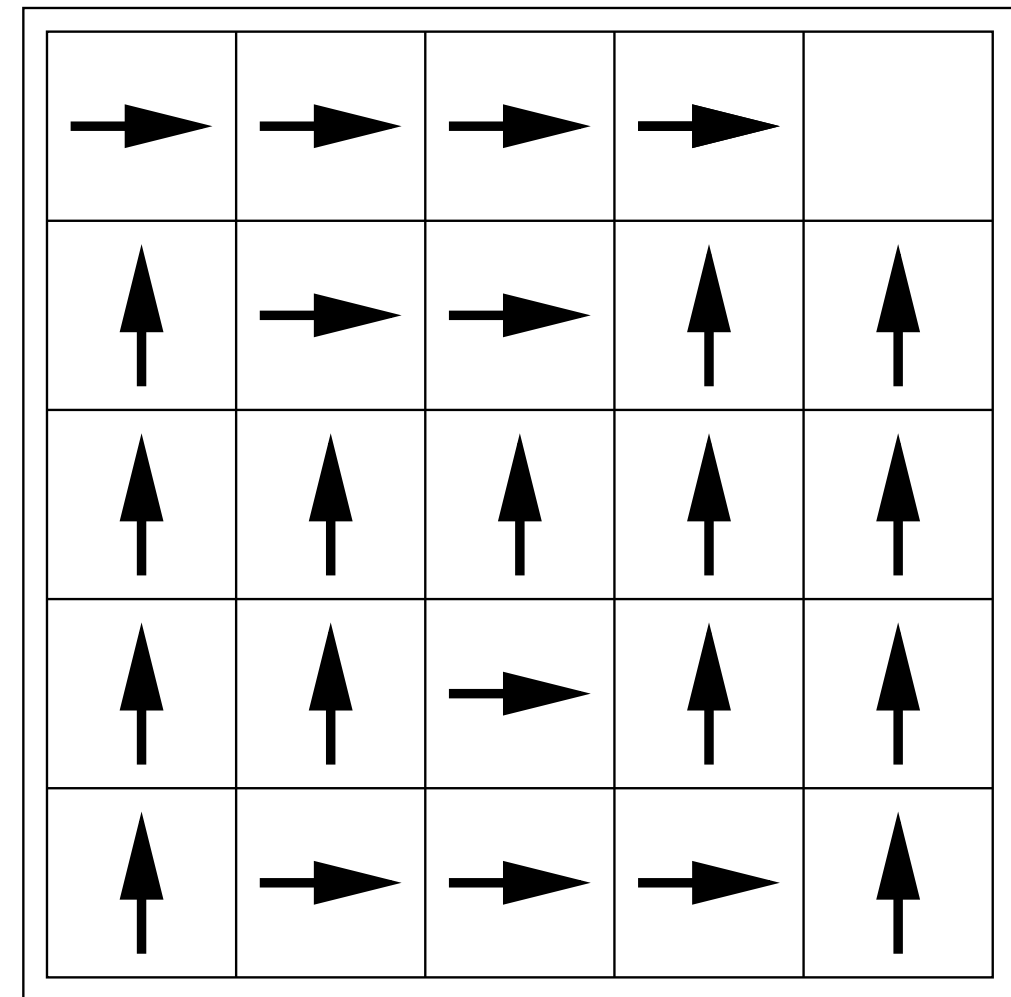


Figure 1. Top: 5x5 grid world with optimal policy. Bottom: True reward function.

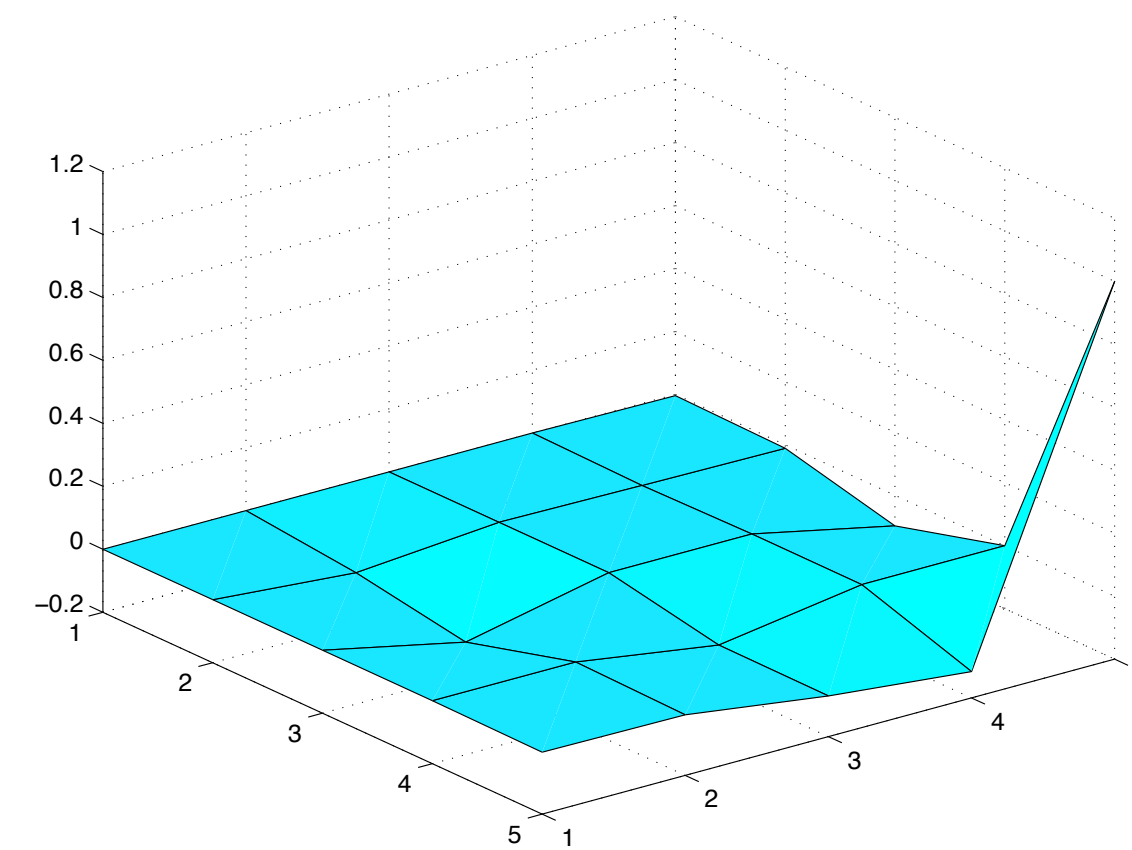
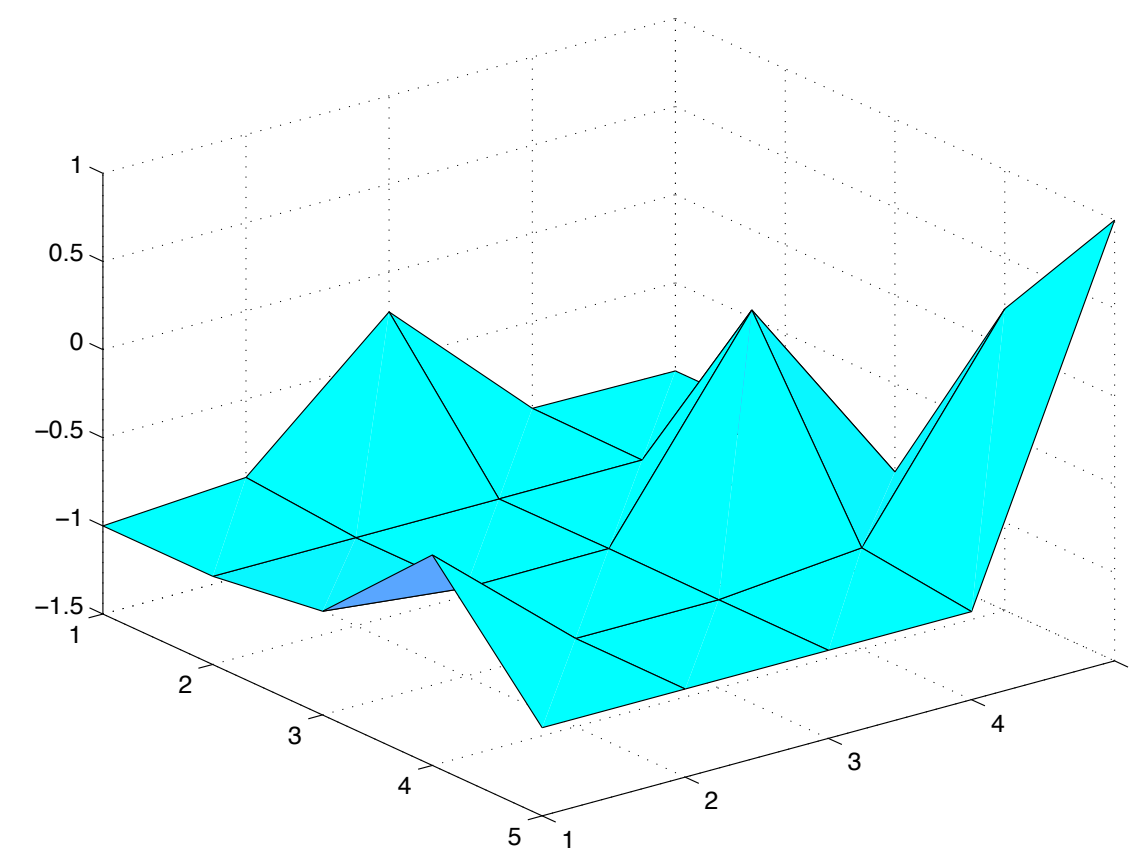
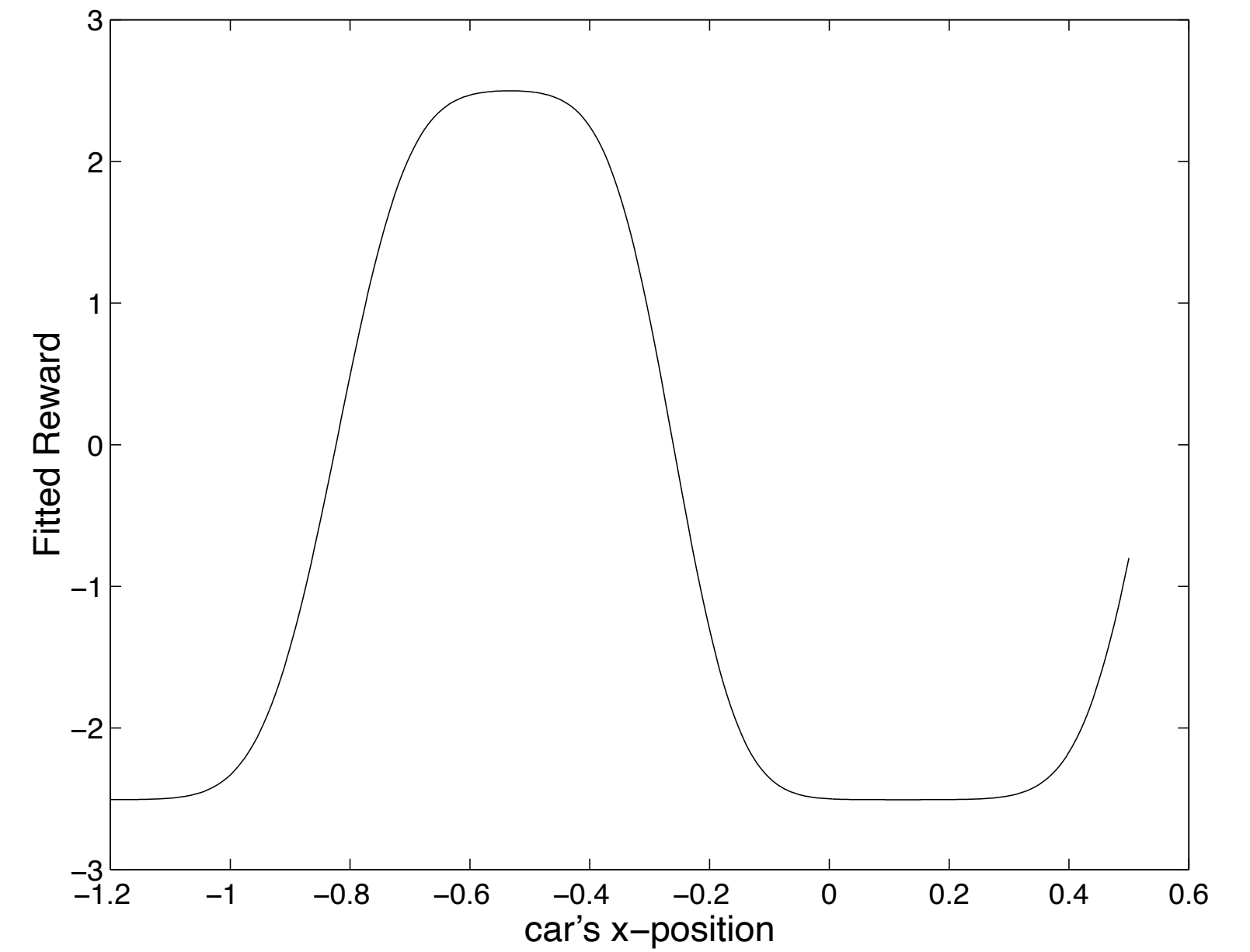
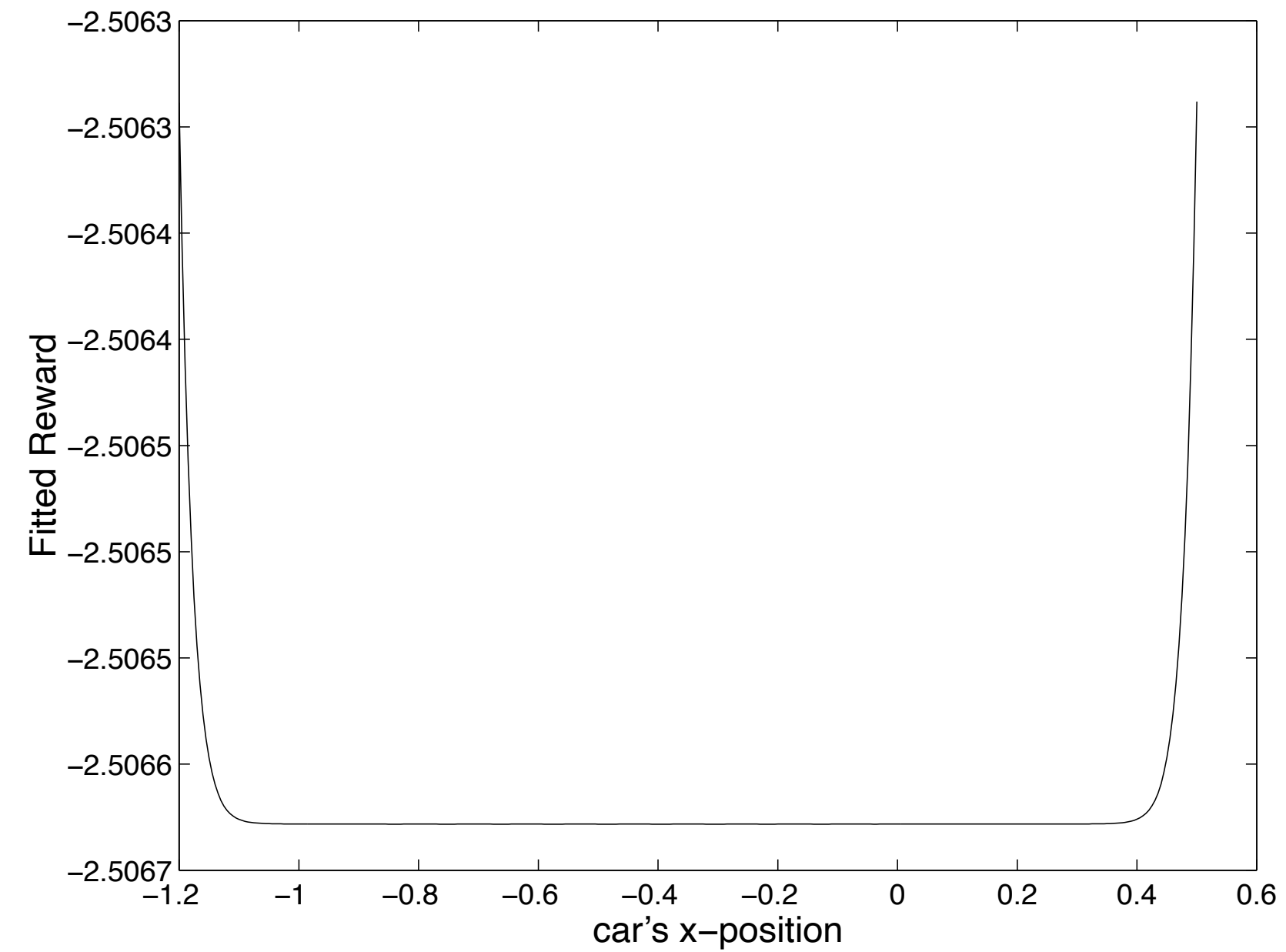
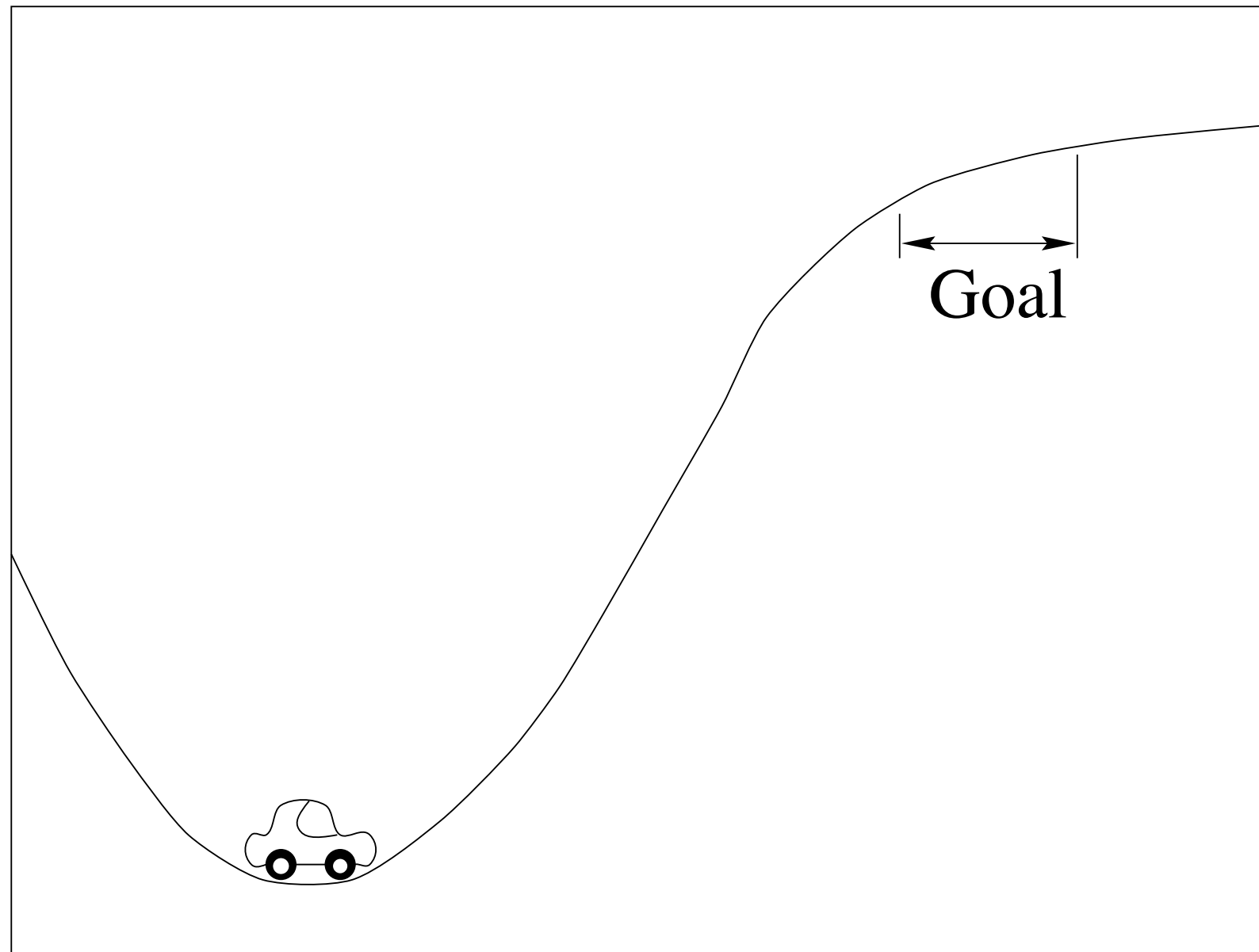


Figure 2. Inverse RL on the  $5 \times 5$  grid. Top:  $\lambda = 0$ . Bottom:  $\lambda = 1.05$ .

resulting in moving in a random direction instead. As

# Inverse RL (Experiments)



Reward is only a function of car x-position, state features are linear combinations of 26 evenly spaced Gaussian shaped basis functions.

# Apprenticeship Learning

---

## Apprenticeship Learning via Inverse Reinforcement Learning

---

Pieter Abbeel

Andrew Y. Ng

Computer Science Department, Stanford University, Stanford, CA 94305, USA

PABBEEL@CS.STANFORD.EDU

ANG@CS.STANFORD.EDU

# Apprenticeship Learning

- Directly optimize for a policy that gives the same reward as expert:

$$\begin{aligned} E_{s_0 \sim D}[V^\pi(s_0)] &= E[\sum_{t=0}^{\infty} \gamma^t R(s_t) | \pi] \\ &= E[\sum_{t=0}^{\infty} \gamma^t w \cdot \phi(s_t) | \pi] \\ &= w \cdot E[\sum_{t=0}^{\infty} \gamma^t \phi(s_t) | \pi] \end{aligned}$$

- $\mu(\pi) = E[\sum_{t=0}^{\infty} \gamma^t \phi(s_t) | \pi] \in \mathbb{R}^k$ .

- $E_{s_0 \sim D}[V^\pi(s_0)] = w \cdot \mu(\pi)$ .

- If  $\|\mu(\tilde{\pi}) - \mu_E\|_2 \leq \epsilon$ .

- then,

$$\begin{aligned} &|E[\sum_{t=0}^{\infty} \gamma^t R(s_t) | \pi_E] - E[\sum_{t=0}^{\infty} \gamma^t R(s_t) | \tilde{\pi}]| \\ &= |w^T \mu(\tilde{\pi}) - w^T \mu_E| \\ &\leq \|w\|_2 \|\mu(\tilde{\pi}) - \mu_E\|_2 \\ &\leq 1 \cdot \epsilon = \epsilon \end{aligned}$$

# Apprenticeship Learning

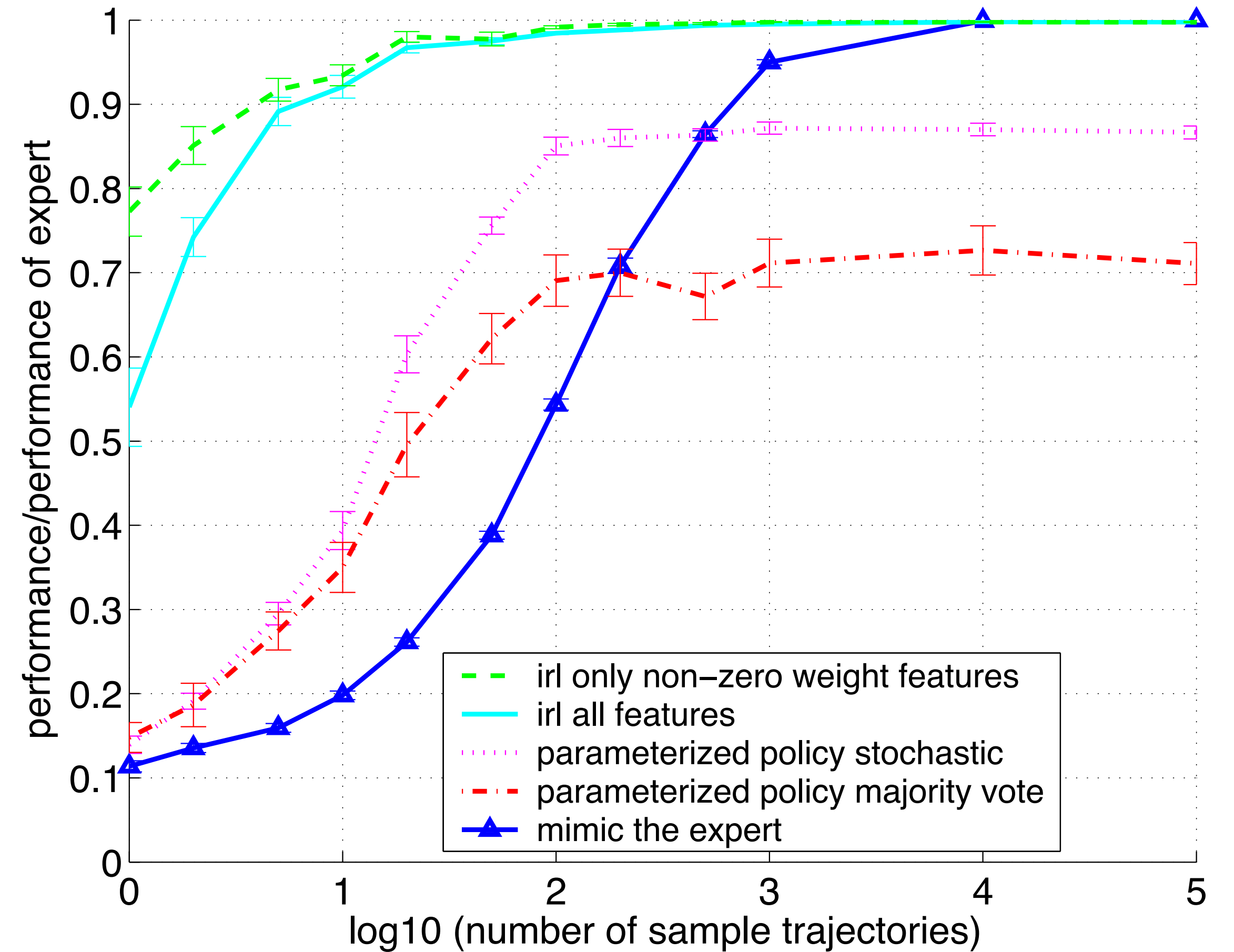
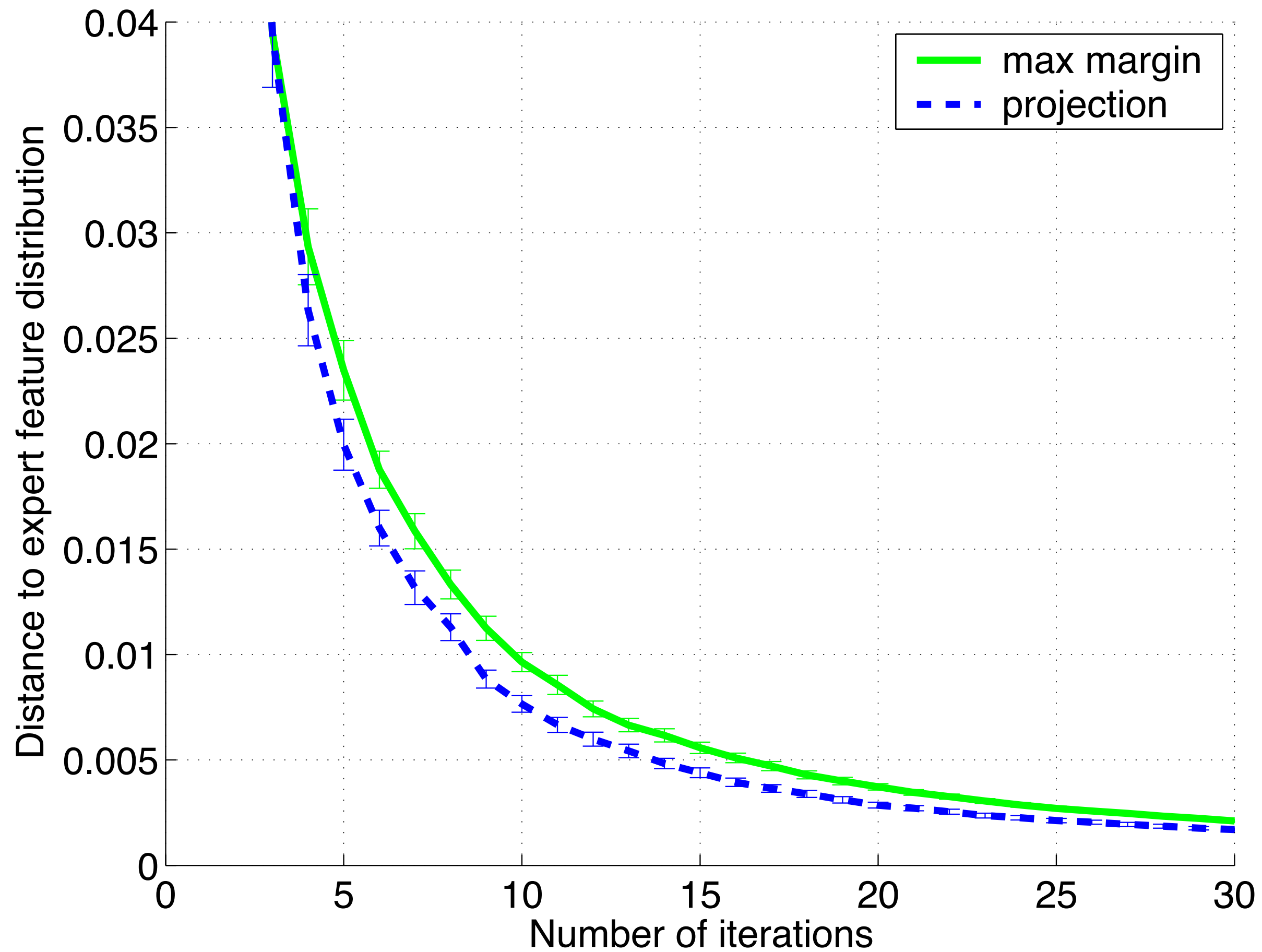
1. Randomly pick some policy  $\pi^{(0)}$ , compute (or approximate via Monte Carlo)  $\mu^{(0)} = \mu(\pi^{(0)})$ , and set  $i = 1$ .
2. Compute  $t^{(i)} = \max_{w: \|w\|_2 \leq 1} \min_{j \in \{0..(i-1)\}} w^T (\mu_E - \mu^{(j)})$ , and let  $w^{(i)}$  be the value of  $w$  that attains this maximum.
3. If  $t^{(i)} \leq \epsilon$ , then terminate.
4. Using the RL algorithm, compute the optimal policy  $\pi^{(i)}$  for the MDP using rewards  $R = (w^{(i)})^T \phi$ .
5. Compute (or estimate)  $\mu^{(i)} = \mu(\pi^{(i)})$ .
6. Set  $i = i + 1$ , and go back to step 2.

$$\begin{aligned} \max_{t,w} \quad & t \\ \text{s.t.} \quad & w^T \mu_E \geq w^T \mu^{(j)} + t, \quad j = 0, \dots, i - 1 \\ & \|w\|_2 \leq 1 \end{aligned}$$

At termination,

$$\forall w \text{ with } \|w\|_2 \leq 1 \exists i \text{ s.t. } w^T \mu^{(i)} \geq w^T \mu_E - \epsilon.$$

# Apprenticeship Learning





# MaxEnt IRL

- Even when matching expectation of state features, there is still ambiguity among policies
- Using the maximum entropy principle to decide what policy

# MaxEnt IRL

- Consider path  $\zeta$ , with features  $f$ .
- Reward weights are  $\theta$
- Reward on path  $\zeta = \theta^T f(\zeta) = \sum_{s \in \zeta} \theta^T f(s)$
- Apprenticeship learning paper tells us we should match the feature counts for expert trajectories,  $\tilde{f}$ .
- Max entropy principle suggests among trajectory distributions that match the feature counts, we should prefer ones that have the maximum entropy:  
$$\max H(P(\zeta_i)) \text{ s.t. } \tilde{f} = \sum_{\zeta_i} P(\zeta_i) f_{\zeta_i}$$

# GAIL

---

**Algorithm 1** Generative adversarial imitation learning

---

- 1: **Input:** Expert trajectories  $\tau_E \sim \pi_E$ , initial policy and discriminator parameters  $\theta_0, w_0$
- 2: **for**  $i = 0, 1, 2, \dots$  **do**
- 3:     Sample trajectories  $\tau_i \sim \pi_{\theta_i}$
- 4:     Update the discriminator parameters from  $w_i$  to  $w_{i+1}$  with the gradient

$$\hat{\mathbb{E}}_{\tau_i} [\nabla_w \log(D_w(s, a))] + \hat{\mathbb{E}}_{\tau_E} [\nabla_w \log(1 - D_w(s, a))] \quad (17)$$

- 5:     Take a policy step from  $\theta_i$  to  $\theta_{i+1}$ , using the TRPO rule with cost function  $\log(D_{w_{i+1}}(s, a))$ . Specifically, take a KL-constrained natural gradient step with

$$\hat{\mathbb{E}}_{\tau_i} [\nabla_{\theta} \log \pi_{\theta}(a|s) Q(s, a)] - \lambda \nabla_{\theta} H(\pi_{\theta}), \quad (18)$$

where  $Q(\bar{s}, \bar{a}) = \hat{\mathbb{E}}_{\tau_i} [\log(D_{w_{i+1}}(s, a)) \mid s_0 = \bar{s}, a_0 = \bar{a}]$

- 6: **end for**
-

