Inverse RL

Saurabh Gupta

Solving a RL Problem

Better Reward Signals

Better Optimization

Convert into a Supervised Training Problem

Solve a Related but Supervision-rich Problem

Build Models and Plan with Them

Model-free RL with sparse rewards

Known reward, known model. Model-based RL

Solving a RL Problem

Better Reward Signals

Better Optimization

Convert into a Supervised Training Problem

Solve a Related but Supervision-rich Problem

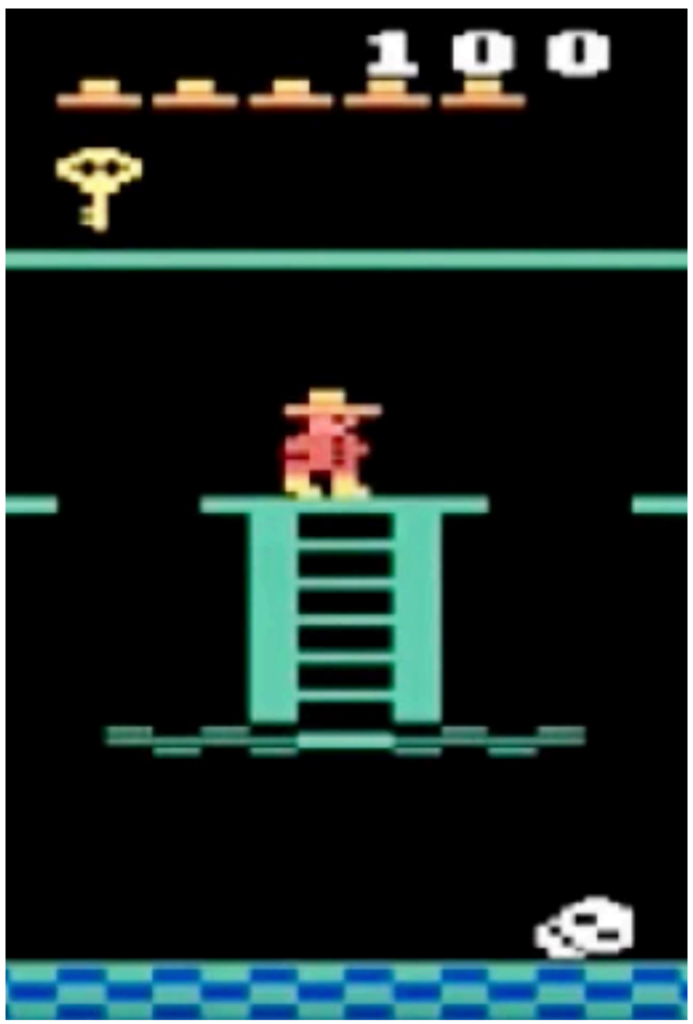
Build Models and Plan with Them

Model-free RL with sparse **rewards**

Known **reward**, known model. Model-based RL

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.

Ng and Russell. ICML 2000. **Algorithms for Inverse RL.**

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?

Ng and Russell. ICML 2000. **Algorithms for Inverse RL.**

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

Ng and Russell. ICML 2000. **Algorithms for Inverse RL.**

Inverse RL

2.2 Basic Properties of MDPs

For our solution to the IRL problem, we will need two of the classical results concerning MDPs (Sutton & Barto, 1998; Bertsekas & Tsitsiklis, 1996).

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^π and Q^π 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 π is an optimal policy for M if and only if, for all $s \in S$,*

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

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

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

$$(\boldsymbol{P}_{a_1} - \boldsymbol{P}_a)(\boldsymbol{I} - \gamma \boldsymbol{P}_{a_1})^{-1} \boldsymbol{R} \succeq 0 \quad (4)$$

Proof. Since $\pi(s) \equiv a_1$, Equation (1) may be written $\boldsymbol{V}^\pi = \boldsymbol{R} + \gamma \boldsymbol{P}_{a_1} \boldsymbol{V}^\pi$. Thus,[1]

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

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

$$\begin{aligned}
& a_1 \equiv \pi(s) \in \arg\max_{a \in A} \sum_{s'} P_{sa}(s') V^\pi(s') \qquad \forall s \in S \\
\Leftrightarrow & \sum_{s'} P_{sa_1}(s') V^\pi(s') \\
& \qquad \geq \sum_{s'} P_{sa}(s') V^\pi(s') \qquad \forall s \in S, a \in A \\
\Leftrightarrow & \boldsymbol{P}_{a_1} \boldsymbol{V}^\pi \succeq \boldsymbol{P}_a \boldsymbol{V}^\pi \qquad \forall a \in A \setminus a_1 \\
\Leftrightarrow & \boldsymbol{P}_{a_1} (\boldsymbol{I} - \gamma \boldsymbol{P}_{a_1})^{-1} \boldsymbol{R} \\
& \qquad \succeq \boldsymbol{P}_a (\boldsymbol{I} - \gamma \boldsymbol{P}_{a_1})^{-1} \boldsymbol{R} \qquad \forall a \in A \setminus a_1
\end{aligned}$$

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

Ng and Russell. ICML 2000. **Algorithms for Inverse RL.**

Inverse RL (LP Formulation, Penalty Terms)

- Maximize quality of optimal action over next-best action:
- $$\sum_{s \in S} \left(Q^\pi(s, a_1) - \max_{a \in A \setminus a_1} Q^\pi(s, a) \right)$$
- Small rewards are simpler: $-\lambda ||R||_1$

$$\begin{aligned} \text{maximize} \quad & \textstyle\sum_{i=1}^{N} \min_{a \in \{a_2, \ldots, a_k\}} \{(\boldsymbol{P}_{a_1}(i) - \boldsymbol{P}_a(i)) \\ & \quad (\boldsymbol{I} - \gamma \boldsymbol{P}_{a_1})^{-1} \boldsymbol{R}\} - \lambda ||\boldsymbol{R}||_1 \\ \text{s.t.} \quad & (\boldsymbol{P}_{a_1} - \boldsymbol{P}_a)(\boldsymbol{I} - \gamma \boldsymbol{P}_{a_1})^{-1} \boldsymbol{R} \succeq 0 \\ & \qquad \forall a \in A \setminus a_1 \end{aligned}$$

- $$|\boldsymbol{R}_i| \le R_{\max}, \quad i = 1, \ldots, N$$

Ng and Russell. ICML 2000. **Algorithms for Inverse RL.**

Inverse RL (w/ linear function approximations)

- Linear approximation of reward:
  - $R(s) = \alpha_1 \phi_1(s) + \alpha_2 \phi_2(s) + \cdots + \alpha_d \phi_d(s)$
- $V^\pi = \alpha_1 V_1^\pi + \cdots + \alpha_d V_d^\pi.$
- Linear program, with an appropriately chosen penalty function p:

$$\begin{aligned} &\text{maximize} \textstyle\sum_{s \in S_0} \min_{a \in \{a_2, \ldots, a_k\}} \{ \\ &\qquad\qquad p(\mathrm{E}_{s' \sim P_{sa_1}} [V^\pi(s')] - \mathrm{E}_{s' \sim P_{sa}} [V^\pi(s')])\} \\ &\qquad \text{s.t. } |\alpha_i| \leq 1, \quad i = 1, \ldots, d \end{aligned}$$

Ng and Russell. ICML 2000. **Algorithms for Inverse RL.**

Inverse RL (Sampled Trajectories)

- Monte-carlo estimate of values:
  - $\hat{V}_i^{\pi}(s_0) = \phi_i(s_0) + \gamma\phi_i(s_1) + \gamma^2\phi_i(s_2) + \cdots$

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

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

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

- $\alpha_d\phi_d$. We then find a policy π_{k+1} that maximizes $V^{\pi}(s_0)$ under R, add π_{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").

Ng and Russell. ICML 2000. **Algorithms for Inverse RL.**

Inverse RL (Experiments)

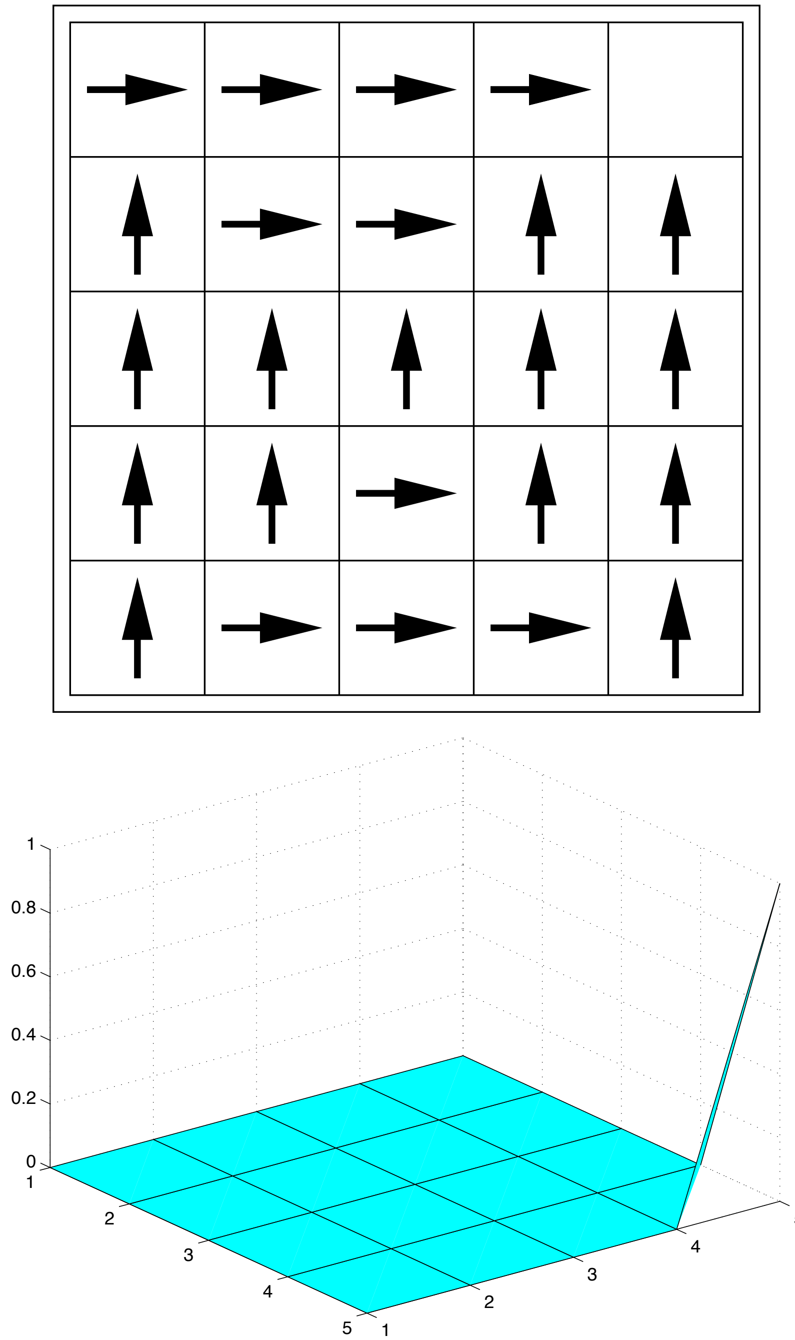


Figure 1. Top: 5x5 grid world with optimal policy. *Bottom*: True reward function.

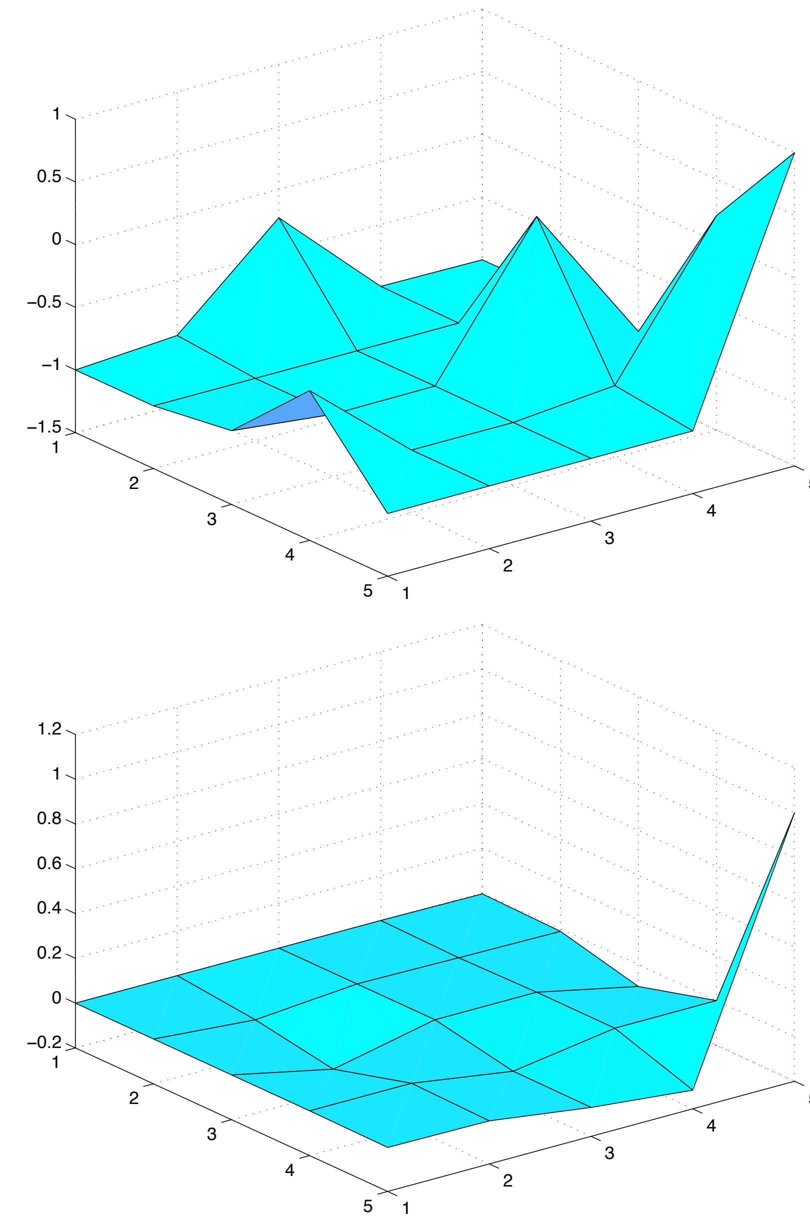
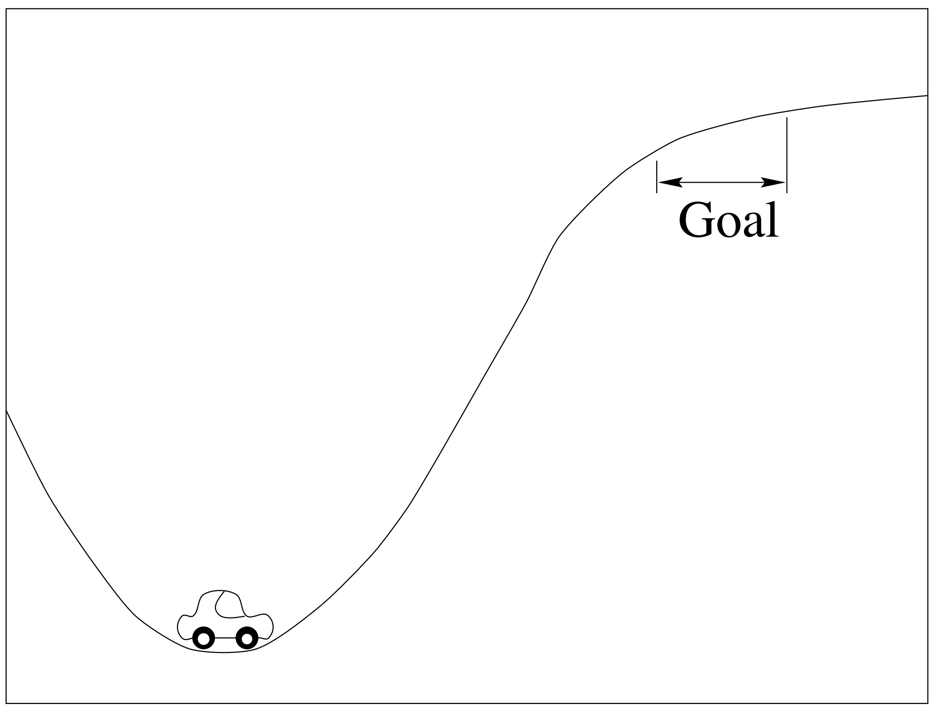


Figure 2. Inverse RL on the 5×5 grid. *Top*: $\lambda = 0$. *Bottom*: $\lambda = 1.05$.

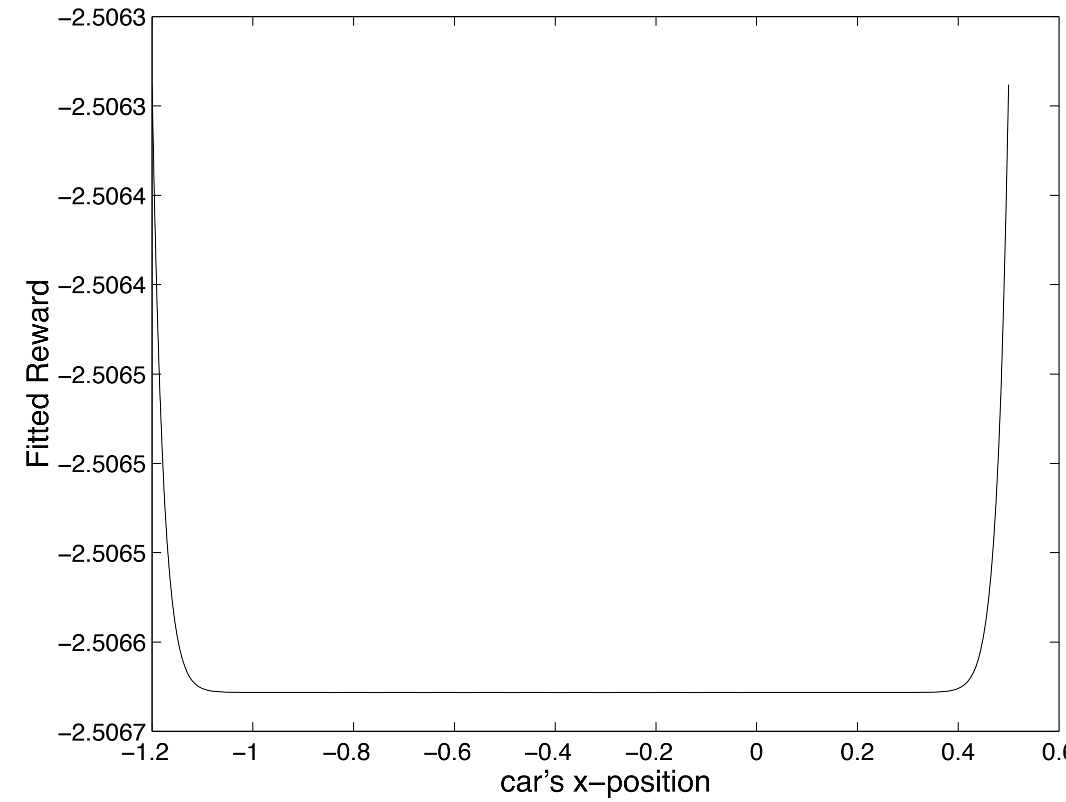
resulting in moving in a random direction instead. An

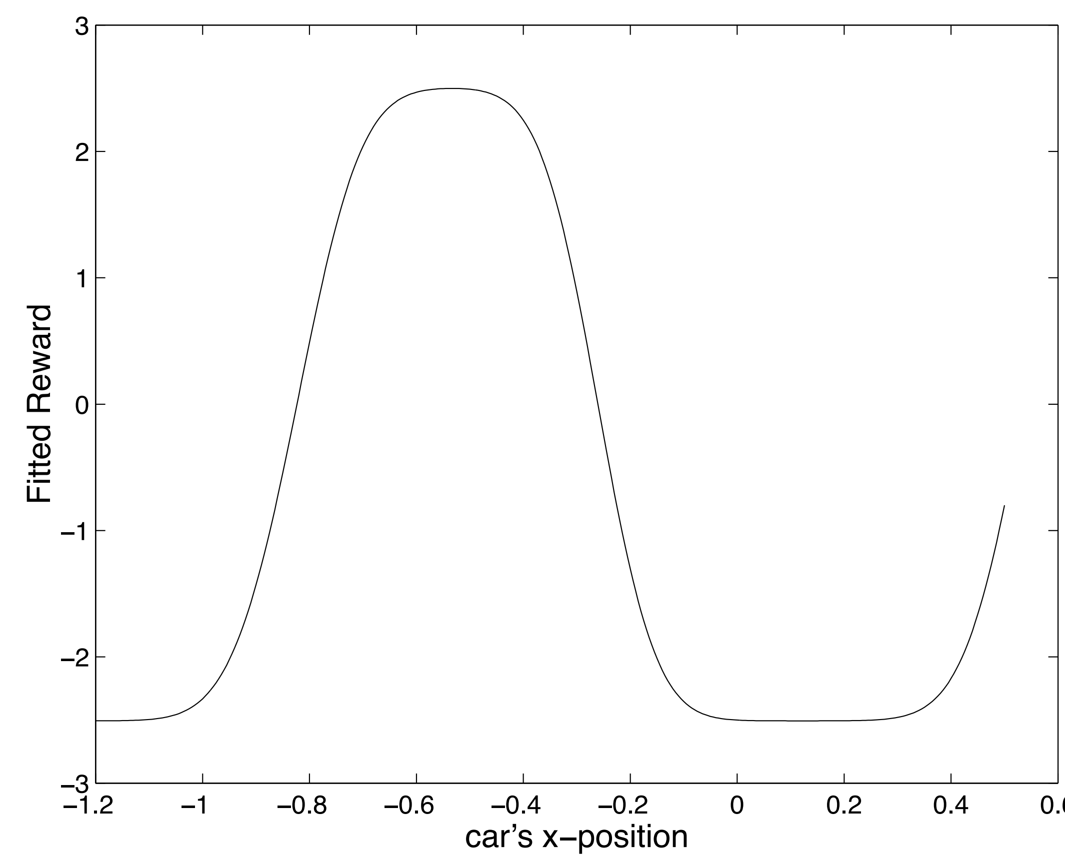
Ng and Russell. ICML 2000. **Algorithms for Inverse RL.**

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.





Ng and Russell. ICML 2000. **Algorithms for Inverse RL.**

Apprenticeship Learning via Inverse Reinforcement Learning

Pieter Abbeel PABBEEL@CS.STANFORD.EDU
Andrew Y. Ng ANG@CS.STANFORD.EDU
Computer Science Department, Stanford University, Stanford, CA 94305, USA

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,

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

Abbeel and Ng. ICML 2004. **Apprenticeship Learning via Inverse RL.**

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 $t^{(i)} = \max_{w:\|w\|_2 \le 1} \min_{j \in \{0..(i-1)\}} w^T(\mu_E - \mu^{(j)})$, and let $w^{(i)}$ be the value of w that attains this maximum.
3. If $t^{(i)} \le \epsilon$, 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.

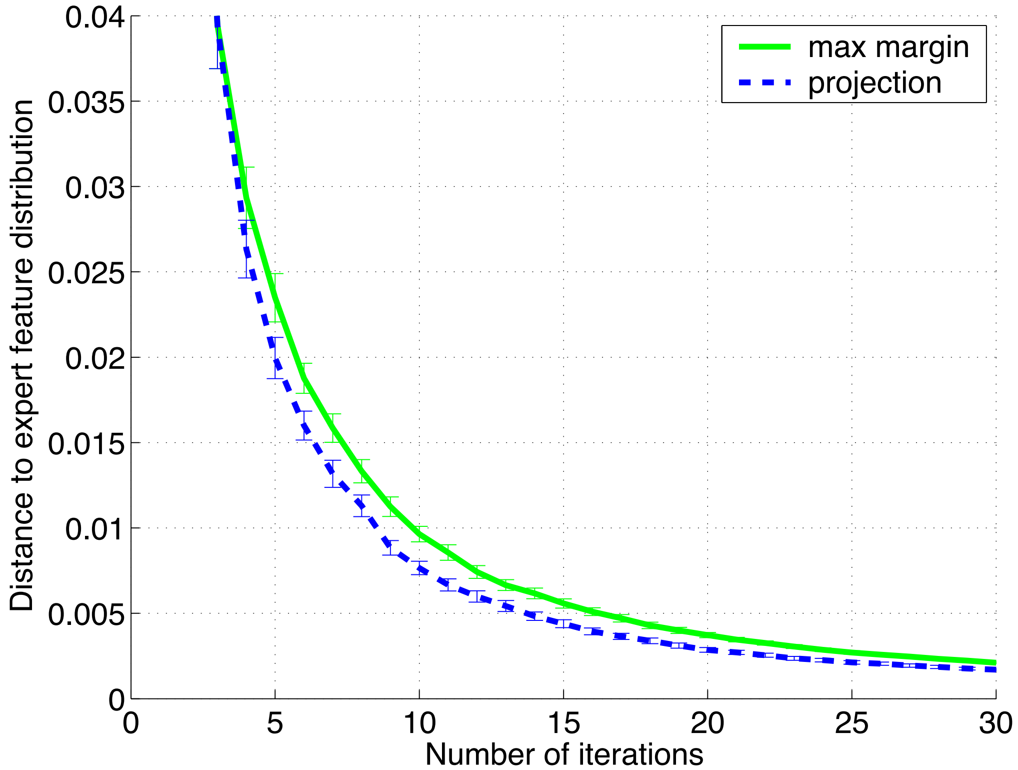
$$\begin{aligned} \max_{t,w} \quad & t \\ \text{s.t.} \quad & w^T \mu_E \ge w^T \mu^{(j)} + t, \; j = 0, \ldots, i-1 \\ & ||w||_2 \le 1 \end{aligned}$$

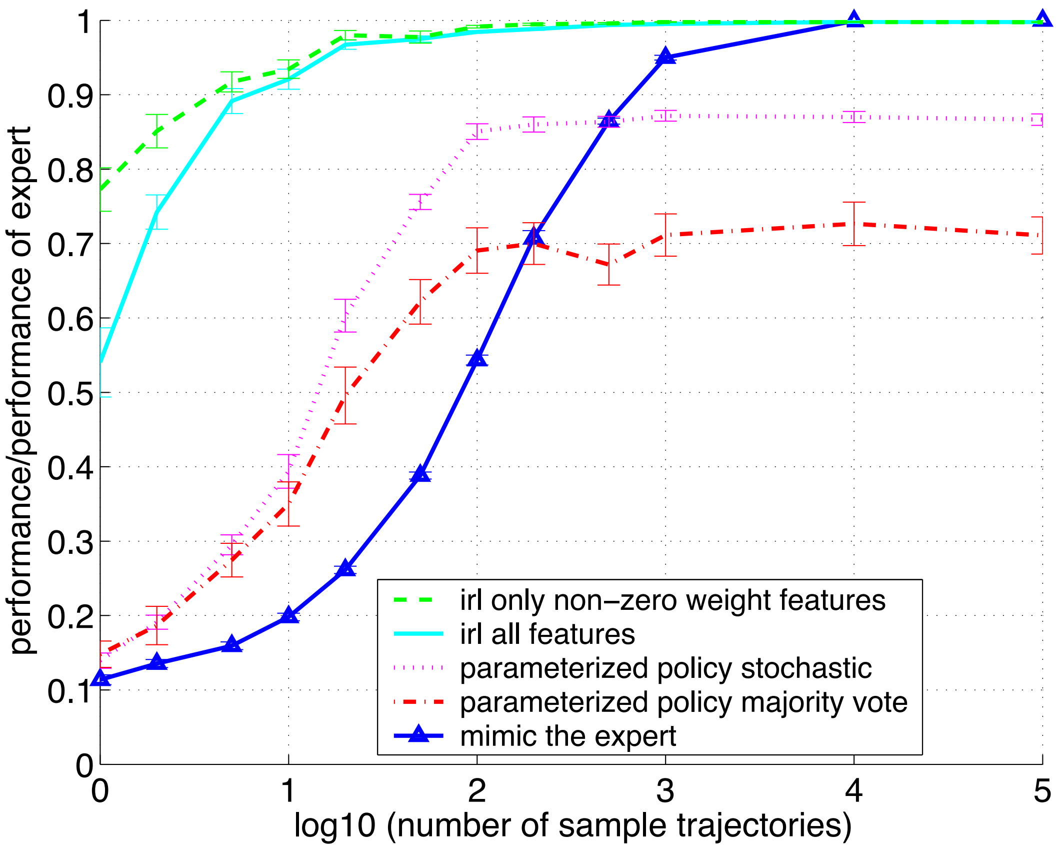
At termination,

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

Abbeel and Ng. ICML 2004. **Algorithms for Inverse RL.**

Apprenticeship Learning





Abbeel and Ng. ICML 2004. **Algorithms for Inverse RL.**

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 ζ, with features f.
- Reward weights are θ
- 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: $\max H(P(\zeta_i))$ 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 θ_0, w_0
2: **for** $i = 0, 1, 2, \ldots$ **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

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

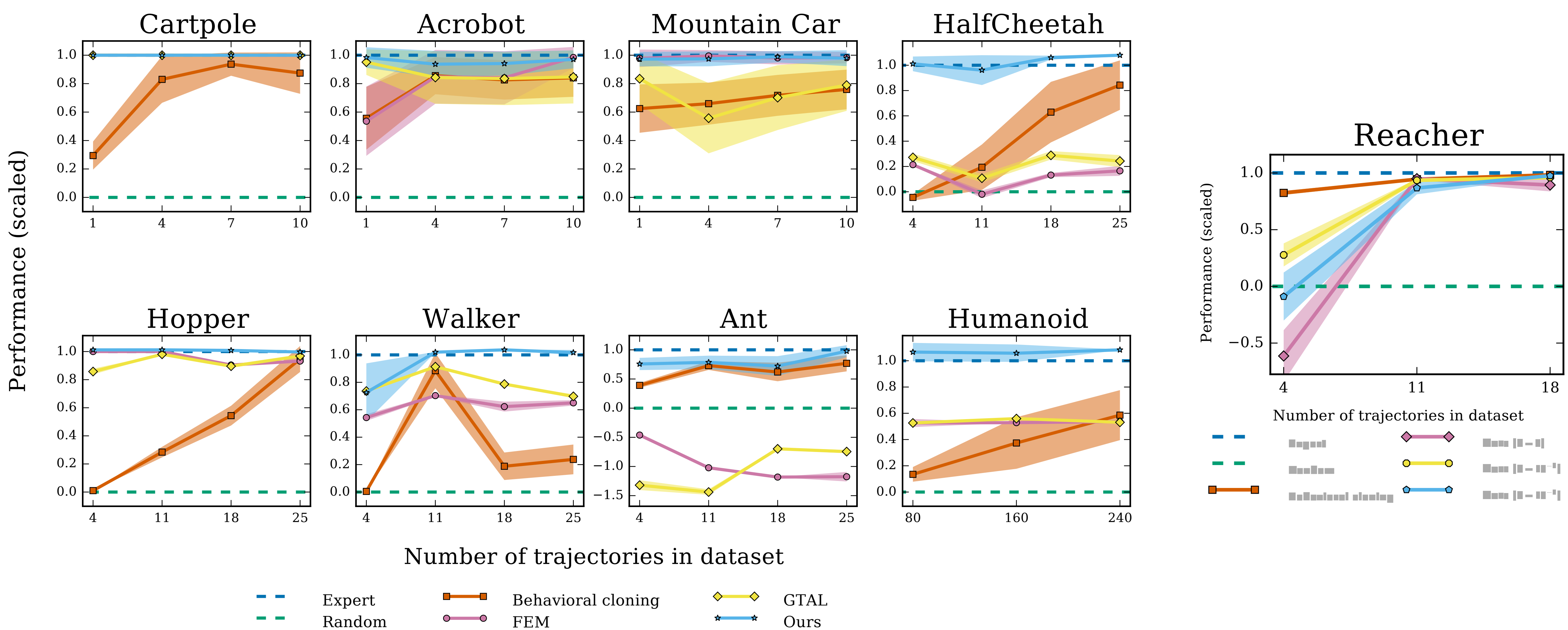
5: Take a policy step from θ_i to θ_{i+1}, using the TRPO rule with cost function $\log(D_{w_{i+1}}(s,a))$. Specifically, take a KL-constrained natural gradient step with

$$\begin{aligned} &\hat{\mathbb{E}}_{\tau_i}\left[\nabla_\theta \log \pi_\theta(a|s) Q(s,a)\right] - \lambda \nabla_\theta H(\pi_\theta), \\ &\text{where } Q(\bar{s},\bar{a}) = \hat{\mathbb{E}}_{\tau_i}[\log(D_{w_{i+1}}(s,a)) \mid s_0 = \bar{s}, a_0 = \bar{a}] \end{aligned} \tag{18}$$

6: **end for**

Abbeel and Ng. ICML 2004. **Algorithms for Inverse RL.**

GAIL



(a) (b)

Figure 1: (a) Performance of learned policies. The y-axis is negative cost, scaled so that the expert achieves 1 and a random policy achieves 0. (b) Causal entropy regularization λ on Reacher.

Abbeel and Ng. ICML 2004. **Algorithms for Inverse RL.**

Max Margin Planning

- $$\min_{w,\zeta} \frac{1}{2} ||w||_2 + \zeta$$
  $$s.t. \; w^T \mu(\pi^*) + \zeta \geq \max_{\pi} w^T \mu(\pi) + D(\pi, \pi^*)$$
- Maximize the margin between the return for the expert demonstrations and the returns of any other policy
- Hard to solve, as we have a lot of constraints
- Trick:
  - Iteratively solve for w by collecting π's that maximally violate the constraint (Remember Structural SVMs?)
  - Find π that maximizes $w^T \mu(\pi) + D(\pi, \pi^*)$ (an RL problem)

Ratliff et al. ICML 2006. **Maximum Margin Planning.**

MaxEnt IRL

- $\max H(P(\zeta_i))$ s.t. $\tilde{f} = \sum_{\zeta_i} P(\zeta_i) f_{\zeta_i}$
- it can be shown — by calculus of variations in the general continuous case, and by ordinary calculus in the discrete case — that the optimal solution p^* takes the form

$$P(\zeta_i) = \frac{e^{\theta^T f_{\zeta_i}}}{Z(\theta)}$$

- θ can be estimated by maximum likelihood of expert demonstrations under $P_\theta(\zeta_i)$.
- Derivative comes out to be:

$$\nabla L(\theta) = \tilde{\mathbf{f}} - \sum_{\zeta} P(\zeta|\theta, T)\mathbf{f}_\zeta = \tilde{\mathbf{f}} - \sum_{s_i} D_{s_i}\mathbf{f}_{s_i}$$

Wainwright and Jordan. **Graphical Models, Exponential Families, and Variational Inference**

MaxEnt IRL

- Even when matching expectation of state features, there is still ambiguity among policies
- Using the maximum entropy principle to decide what policy
- $$\text{reward}(\mathbf{f}_\zeta) = \theta^\top \mathbf{f}_\zeta = \sum_{s_j \in \zeta} \theta^\top \mathbf{f}_{s_j}$$
- $$\sum_{\text{Path } \zeta_i} P(\zeta_i)\mathbf{f}_{\zeta_i} = \tilde{\mathbf{f}}$$
- $$P(\zeta_i|\theta) = \frac{1}{Z(\theta)} e^{\theta^\top \mathbf{f}_{\zeta_i}} = \frac{1}{Z(\theta)} e^{\sum_{s_j \in \zeta_i} \theta^\top \mathbf{f}_{s_j}}$$
- $$\theta^* = \operatorname*{argmax}_\theta L(\theta) = \operatorname*{argmax}_\theta \sum_{\text{examples}} \log P(\tilde{\zeta}|\theta, T)$$
- $$\nabla L(\theta) = \tilde{\mathbf{f}} - \sum_\zeta P(\zeta|\theta, T)\mathbf{f}_\zeta = \tilde{\mathbf{f}} - \sum_{s_i} D_{s_i} \mathbf{f}_{s_i}$$

MaxEnt IRL

Algorithm 1 Expected Edge Frequency Calculation

Backward pass

1. Set $Z_{s_i,0} = 1$
2. Recursively compute for N iterations

$$Z_{a_{i,j}} = \sum_k P(s_k|s_i, a_{i,j}) e^{\text{reward}(s_i|\theta)} Z_{s_k}$$

$$Z_{s_i} = \sum_{a_{i,j}} Z_{a_{i,j}}$$

Local action probability computation

3. $P(a_{i,j}|s_i) = \dfrac{Z_{a_{i,j}}}{Z_{s_i}}$

Forward pass

4. Set $D_{s_i,t} = P(s_i = s_{\text{initial}})$
5. Recursively compute for $t = 1$ to N

$$D_{s_i,t+1} = \sum_{a_{i,j}} \sum_k D_{s_k,t} P(a_{i,j}|s_i) P(s_k|a_{i,j}, s_i)$$

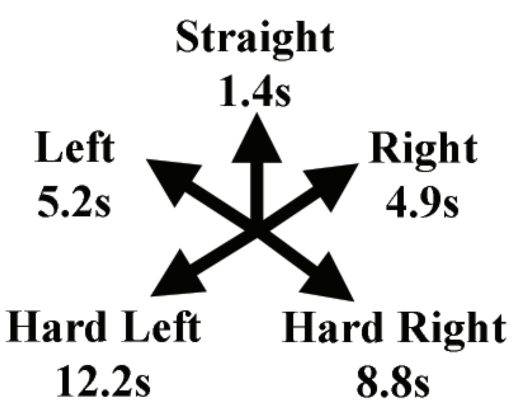
Summing frequencies

6. $D_{s_i} = \sum_t D_{s_i,t}$

See Brian Ziebart's PhD Thesis

MaxEnt IRL (Results)

	Matching	90% Match	Log Prob
Time-based	72.38%	43.12%	N/A
Max Margin	75.29%	46.56%	N/A
Action	77.30%	50.37%	-7.91
Action (costs)	77.74%	50.75%	N/A
MaxEnt paths	**78.79%**	**52.98%**	**-6.85**



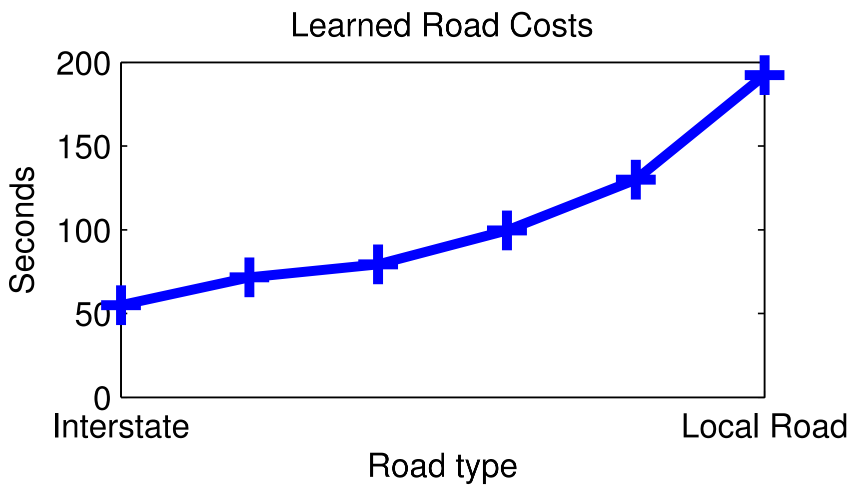


Figure 4: Destination distribution (from 5 destinations) and remaining path distribution given partially traveled path. The partially traveled path is heading westward, which is a very inefficient (i.e., improbable) partial route to any of the eastern destinations (3, 4, 5). The posterior destination probability is split between destinations 1 and 2 primarily based on the prior distribution on destinations.

Thank you