Deep Inverse Reinforcement Learning

Matthew Alger

October 21, 2015

Inverse reinforcement learning

Given: Observations of an agent's behaviour in an environment

Inverse reinforcement learning

Given: Observations of an agent's behaviour in an environment
Find: The agent's motivations

Outline

Preliminaries — Markov decision processes

Markov decision process

$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}^a_{ss'}, R, \gamma)$

- $\mathcal{S}$, finite set of states
- $\mathcal{A}$, finites set of actions
- $\{\mathcal{P}^a_{ss'}\}$, set of transition probabilities
- $R : \mathcal{S} \to \mathbb{R}$, reward function
- $\gamma \in [0, 1)$, discount factor

Inverse reinforcement learning

Given: Observations of an agent's behaviour in an MDP $\mathcal{M}$
Find: The agent's motivations

Preliminaries — Policies

Policy

$\pi : \mathcal{S} \to \mathcal{A}$

Preliminaries — Policies

Policy

$\pi : \mathcal{S} \to \mathcal{A}$

Trajectory/path

$\zeta = [(s_1, a_1), (s_2, a_2), \ldots, (s_n, a_n)]$

Preliminaries — Policies

Policy

$\pi : \mathcal{S} \to \mathcal{A}$

Trajectory/path

$\zeta = [(s_1, a_1), (s_2, a_2), \ldots, (s_n, a_n)]$

Trajectory reward

$R(\zeta) = \sum_{s \in \zeta} R(s)$

Preliminaries — Value

Value function

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k+1}) \,\middle|\, s_t = s\right]$$

Preliminaries — Value

Value function

$$V^\pi(s) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k+1}) \,\middle|\, s_t = s\right]$$

Bellman equation for V^π

$$V^\pi(s) = \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^{\pi(s)} \left(R(s') + \gamma V^\pi(s')\right)$$

Preliminaries — Value

Value function

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k+1}) \,\middle|\, s_t = s \right]$$

Bellman equation for V^π

$$V^\pi(s) = \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^{\pi(s)} \left(R(s') + \gamma V^\pi(s')\right)$$

Action–value function

$$Q^\pi(s, a) = \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^{a} \left(R(s') + \gamma V^\pi(s')\right)$$

Preliminaries — Optimal policy

Optimal policy

π^* s. t. $V^{\pi^*}(s) \geq V^{\pi'}(s)$

Inverse reinforcement learning

Given: Ideally π and $\mathcal{P}^a_{ss'}$, or minimally $\{\zeta\}$
Find: $R(s)$ such that π is an optimal policy or $\{\zeta\}$ is generated by an optimal policy

Outline

LP — Notation

$$(\boldsymbol{V}^\pi)_i = V^\pi(s_i)$$
$$(\boldsymbol{R})_i = R(s_i)$$
$$(\boldsymbol{P}^a)_{ij} = \mathcal{P}^a_{s_i s_j}$$
$$a^\pi \equiv \pi(s)$$

LP — Notation

$$
\begin{aligned}
(\boldsymbol{V}^{\pi})_i &= V^{\pi}(s_i) \\
(\boldsymbol{R})_i &= R(s_i) \\
(\boldsymbol{P}^{a})_{ij} &= \mathcal{P}^{a}_{s_i s_j} \\
a^{\pi} &\equiv \pi(s)
\end{aligned}
$$

Can now write the value as a matrix equation:

$$\boldsymbol{V}^{\pi} = (\boldsymbol{R} + \gamma \boldsymbol{P}^{a^{\pi}} \boldsymbol{V}^{\pi})$$

LP — Reward constraint

$$\boldsymbol{V}^{\pi} = (\boldsymbol{R} + \gamma \boldsymbol{P}^{a^{\pi}} \boldsymbol{V}^{\pi})$$

LP — Reward constraint

$$\begin{aligned} \boldsymbol{V}^{\pi} &= (\boldsymbol{R} + \gamma \boldsymbol{P}^{a^{\pi}} \boldsymbol{V}^{\pi}) \\ &= (\boldsymbol{I} - \gamma \boldsymbol{P}^{a^{\pi}})^{-1} \boldsymbol{R} \end{aligned}$$

An explicit solution for $\boldsymbol{V}^{\pi}$!

LP — Reward constraint

$$\begin{aligned}\boldsymbol{V}^{\pi} &= (\boldsymbol{R} + \gamma \boldsymbol{P}^{a^{\pi}} \boldsymbol{V}^{\pi}) \\ &= (\boldsymbol{I} - \gamma \boldsymbol{P}^{a^{\pi}})^{-1} \boldsymbol{R}\end{aligned}$$

An explicit solution for $\boldsymbol{V}^{\pi}$!

Assuming π is optimal:

$$\forall a \in \mathcal{A} \setminus a^{\pi}. \quad \boldsymbol{P}^{a^{\pi}} \boldsymbol{V}^{\pi} \geq \boldsymbol{P}^{a} \boldsymbol{V}^{\pi}$$

LP — Reward constraint

$$
\begin{aligned}
\boldsymbol{V}^\pi &= (\boldsymbol{R} + \gamma \boldsymbol{P}^{a^\pi} \boldsymbol{V}^\pi) \\
&= (\boldsymbol{I} - \gamma \boldsymbol{P}^{a^\pi})^{-1} \boldsymbol{R}
\end{aligned}
$$

An explicit solution for $\boldsymbol{V}^\pi$!

Assuming π is optimal:

$$\forall a \in \mathcal{A} \setminus a^\pi. \quad \boldsymbol{P}^{a^\pi} \boldsymbol{V}^\pi \geq \boldsymbol{P}^a \boldsymbol{V}^\pi$$

Using our $\boldsymbol{V}^\pi$ solution:

$$\forall a \in \mathcal{A} \setminus a^\pi. \quad (\boldsymbol{P}^{a^\pi} - \boldsymbol{P}^a)(\boldsymbol{I} - \gamma \boldsymbol{P}^{a^\pi})^{-1} \boldsymbol{R} \geq \mathbf{0}$$

LP — Heuristic

Lots of solutions, so employ a heuristic:

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

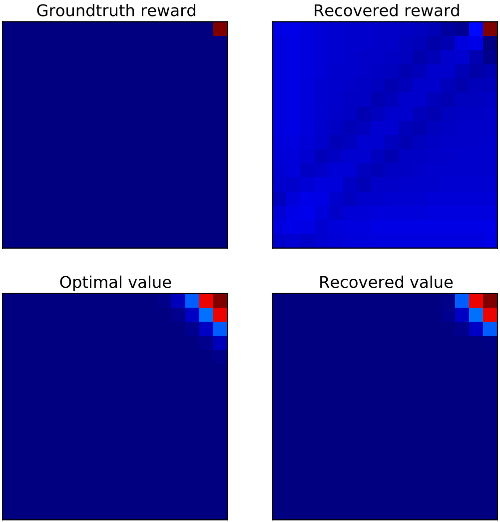
LP — Linear programming problem

$$\text{maximise} \sum_{i=1}^{|\mathcal{S}|} \min_{a\in(\mathcal{A}\setminus a^{\pi})}((\boldsymbol{P}^{a^{\pi}})_i - (\boldsymbol{P}^{a})_i)(\boldsymbol{I} - \gamma \boldsymbol{P}^{a^{\pi}})^{-1} \cdot \boldsymbol{R} - \lambda ||R||_1$$

$$\text{s.t. } \forall i \in 1, \ldots, |\mathcal{S}|. \forall a \in (\mathcal{A} \setminus a^{\pi}). -((\boldsymbol{P}^{a^{\pi}})_i - (\boldsymbol{P}^{a})_i)(\boldsymbol{I} - \gamma \boldsymbol{P}^{a^{\pi}})^{-1} \cdot \boldsymbol{R} \leq 0$$

$$\text{and } \forall i \in 1, \ldots, |\mathcal{S}|. |R_i| \leq R_{\text{max}}$$

LP — Gridworld



LP — Large state space approximation

$$R(s) = \boldsymbol{\alpha} \cdot \phi(s)$$

LP — Large state space approximation

$$R(s) = \boldsymbol{\alpha} \cdot \phi(s)$$

$$V^{\pi}(s) = \sum_{i} \alpha_i V_i^{\pi}(s)$$

LP — Large state space approximation

$$R(s) = \boldsymbol{\alpha} \cdot \phi(s)$$

$$V^\pi(s) = \sum_i \alpha_i V_i^\pi(s)$$

$$\forall a \in \mathcal{A} \setminus a^\pi. \quad \boldsymbol{P}_s^{a^\pi} \cdot \boldsymbol{V}^\pi \geq \boldsymbol{P}_s^a \cdot \boldsymbol{V}^\pi$$

LP — Large state space approximation

$$R(s) = \boldsymbol{\alpha} \cdot \phi(s)$$

$$V^\pi(s) = \sum_i \alpha_i V_i^\pi(s)$$

$$\forall a \in \mathcal{A} \setminus a^\pi. \quad \boldsymbol{P}_s^{a^\pi} \cdot \boldsymbol{V}^\pi \geq \boldsymbol{P}_s^a \cdot \boldsymbol{V}^\pi$$

$$\text{maximise} \sum_{s \in S_0} \min_{a \in (\mathcal{A} \setminus a^*)} p\left(\boldsymbol{P}_s^{a^\pi} \cdot \boldsymbol{V}^\pi - \boldsymbol{P}_s^a \cdot \boldsymbol{V}^\pi\right)$$

$$\text{s.t. } \forall i \in 1, \ldots, D. \quad |\alpha_i| \leq 1$$

$$p(x) = \begin{cases} x, & x \geq 0 \\ 2x, & x < 0 \end{cases}$$

LP — Summary

Given: π and $\mathcal{P}^a_{ss'}$
Maximise: $\sum_{s \in S_0} \min\limits_{a \in (\mathcal{A} \setminus a^*)} p\left(\boldsymbol{P}_s^{a^\pi} \cdot \boldsymbol{V}^\pi - \boldsymbol{P}_s^a \cdot \boldsymbol{V}^\pi\right)$
Such that: $\forall i \in 1, \ldots, D. \quad |\alpha_i| \leq 1$
Result: $R(s) = \boldsymbol{\alpha} \cdot \phi(s)$

Outline

MaxEnt — Feature expectations

Feature counts

$\phi_\zeta = \sum_{s \in \zeta} \phi(s)$

Feature expectations

$\tilde{\phi} = \frac{1}{N} \sum_{i=1}^{N} \phi_\zeta$

Good reward functions have optimal policies that generate matching feature expectations.

MaxEnt — Distribution

$$P(\zeta) = \frac{1}{Z} \exp(R(\zeta)) \quad \text{(deterministic)}$$

MaxEnt — Distribution

$$P(\zeta) = \frac{1}{Z} \exp(R(\zeta)) \quad \text{(deterministic)}$$

$$P(\zeta \mid \boldsymbol{\alpha}) \approx \frac{\exp(\boldsymbol{\alpha} \cdot \boldsymbol{\phi}_\zeta)}{Z(\boldsymbol{\alpha})} \prod_{s_{t+1}, a_t, s_t \in \zeta} \mathcal{P}^{a_t}_{s_t s_{t+1}} \quad \text{(non-deterministic)}$$

MaxEnt — Optimisation

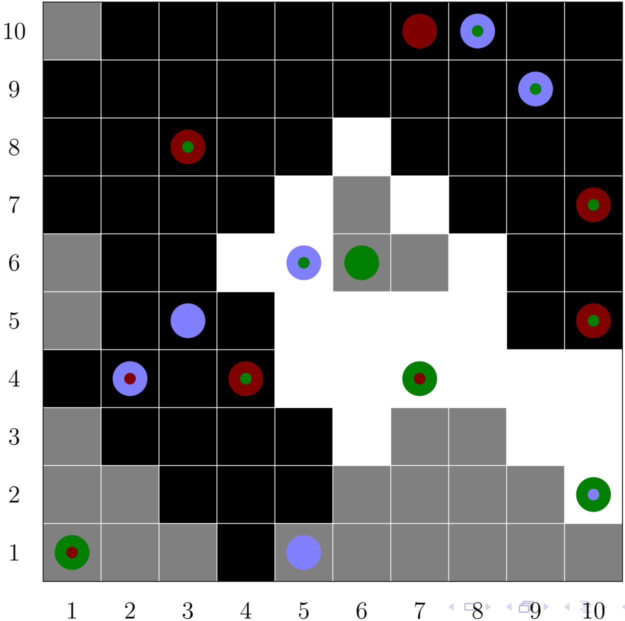
$$\boldsymbol{\alpha}^* = \operatorname*{argmax}_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \log P(\zeta_i \mid \boldsymbol{\alpha})$$

MaxEnt — Optimisation

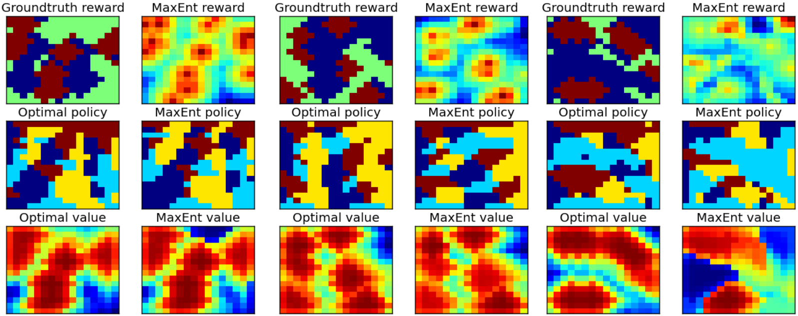
$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \sum_{i=1}^{N} \log P(\zeta_i \mid \boldsymbol{\alpha})$$

$$\frac{\partial}{\partial \boldsymbol{\alpha}} \sum_{i=1}^{N} \log P(\zeta_i \mid \boldsymbol{\alpha}) = \tilde{\phi}_{\mathrm{obs}} - \sum_{i=1}^{N} P(\zeta_i \mid \boldsymbol{\alpha}) \phi_{\zeta_i} = \tilde{\phi}_{\mathrm{obs}} - \sum_{s \in \mathcal{S}} D(s) \phi(s)$$

Objectworld



MaxEnt — Objectworld



Outline

DeepMaxEnt — Removing linearity

$$R(s) = \boldsymbol{\alpha} \cdot \phi(s)$$

DeepMaxEnt — Removing linearity

$$R(s) = \boldsymbol{\alpha} \cdot \phi(s)$$

$$R(s) = \boldsymbol{\alpha} \cdot \varphi(s)$$

$$\varphi(s) = \sigma(\boldsymbol{W} \cdot \phi(s))$$

DeepMaxEnt — Removing linearity

$$R(s) = \boldsymbol{\alpha} \cdot \phi(s)$$

$$R(s) = \boldsymbol{\alpha} \cdot \varphi(s)$$
$$\varphi(s) = \sigma(\boldsymbol{W} \cdot \phi(s))$$

$$R(s) = \boldsymbol{\alpha} \cdot \varphi_n(s)$$
$$\varphi_n(s) = \sigma(\boldsymbol{W}_n \cdot \varphi_{n-1}(s))$$
$$\vdots$$
$$\varphi_1(s) = \sigma(\boldsymbol{W}_1 \cdot \phi(s))$$

DeepMaxEnt — Gradient

$$\partial_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \log P(\zeta_i \mid \boldsymbol{\alpha}) = \tilde{\boldsymbol{\varphi}}_{\text{obs},n} - \sum_{s \in \mathcal{S}} D(s) \boldsymbol{\varphi}_n(s)$$

DeepMaxEnt — Gradient

$$\partial_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \log P(\zeta_i \mid \boldsymbol{\alpha}) = \tilde{\boldsymbol{\varphi}}_{\text{obs},n} - \sum_{s \in \mathcal{S}} D(s) \boldsymbol{\varphi}_n(s)$$

$$\partial_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \log P(\zeta_i \mid \boldsymbol{\alpha}) = \sum_{s \in \mathcal{S}} (D_{\text{obs}}(s) - D(s)) \boldsymbol{\varphi}_n(s)$$

DeepMaxEnt — Gradient

$$\partial_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \log P(\zeta_i \mid \boldsymbol{\alpha}) = \tilde{\boldsymbol{\varphi}}_{\text{obs},n} - \sum_{s \in \mathcal{S}} D(s) \boldsymbol{\varphi}_n(s)$$

$$\partial_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \log P(\zeta_i \mid \boldsymbol{\alpha}) = \sum_{s \in \mathcal{S}} (D_{\text{obs}}(s) - D(s)) \boldsymbol{\varphi}_n(s)$$

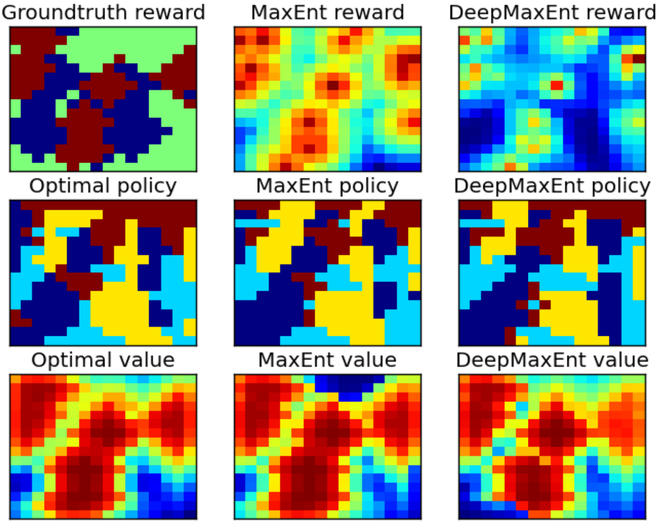
$$R(s) = \boldsymbol{\alpha} \cdot \boldsymbol{\varphi}_n(s)$$

DeepMaxEnt — Gradient

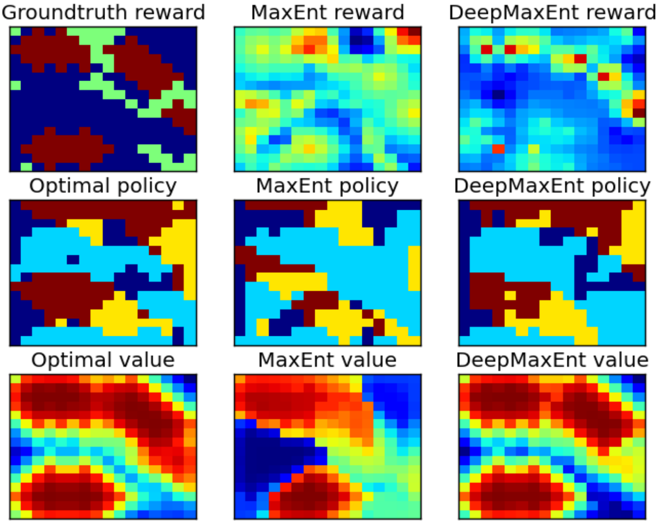
$$\partial_{\boldsymbol{\alpha}} L = \sum_{s \in \mathcal{S}} (D_{\text{obs}}(s) - D(s)) \varphi_n(s)$$

$$\begin{aligned} \partial_{\boldsymbol{W}_i} L &= \sum_{s \in \mathcal{S}} \frac{\partial L}{\partial R(s)} \frac{\partial R(s)}{\partial \boldsymbol{W}_i} \\ &= \sum_{s \in \mathcal{S}} (D_{\text{obs}}(s) - D(s)) \partial_{\boldsymbol{W}_i} R(s) \end{aligned}$$

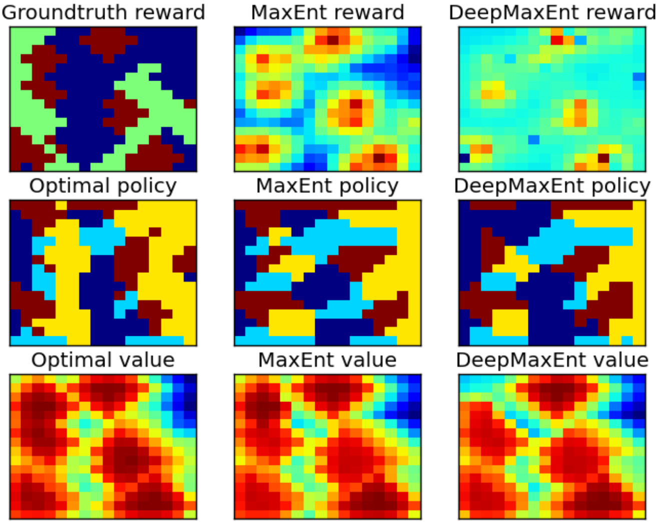
DeepMaxEnt vs MaxEnt



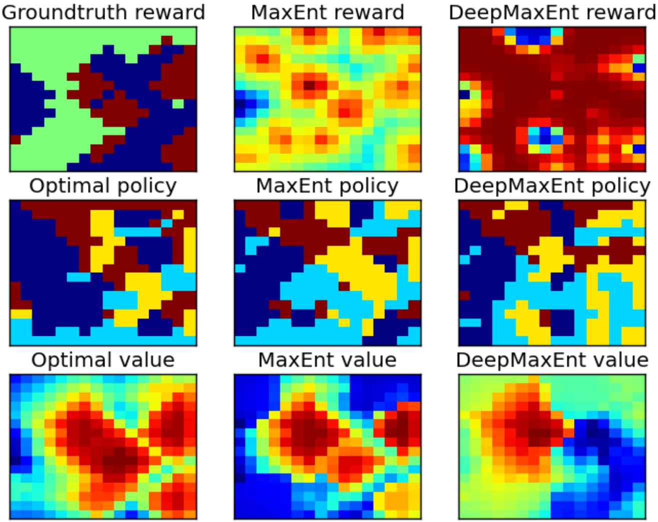
DeepMaxEnt vs MaxEnt



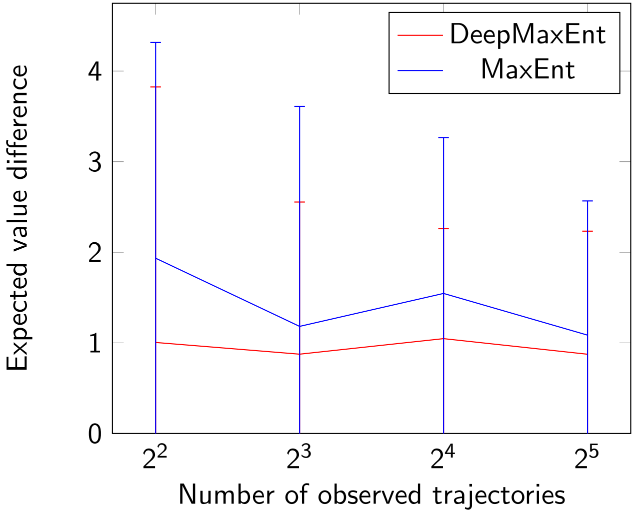
DeepMaxEnt vs MaxEnt



DeepMaxEnt vs MaxEnt

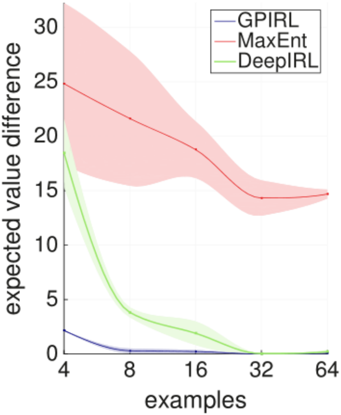


Results



DeepMaxEnt
MaxEnt
Expected value difference
0
1
2
3
4
2^2
2^3
2^4
2^5
Number of observed trajectories

Results — Wulfmeier



A. Y. Ng and S. Russell.
Algorithms for inverse reinforcement learning.
In *ICML*, pages 663–670, 2000.

R. S. Sutton and A. G. Barto.
Reinforcement learning: An introduction, 1998.

M. Wulfmeier, P. Ondruska, and I. Posner.
Deep inverse reinforcement learning.
arXiv preprint arXiv:1507.04888, 2015.

B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey.
Maximum entropy inverse reinforcement learning.
In *AAAI*, pages 1433–1438, 2008.