Apprenticeship Learning using Inverse Reinforcement Learning and Gradient Methods

Gergely Neu*†
neu.gergely@gmail.com

Csaba Szepesvári*‡
szepesva@cs.ualberta.ca

† Budapest University of Technology and Economics, Hungary
‡ Department of Computing Science, University of Alberta, Canada
* Machine Learning Research Group, Computer and Automation Research Institute, Hungarian Academy of Sciences, Hungary

Apprenticeship learning in MDPs

- Assumption: $\exists$ expert following an optimal policy π_E
- Samples from the expert: $(X_t, A_t)_{0 \le t \le T}$
- Empirical estimate of π_E:

$$\hat{\pi}_{E,T}(a|x) = \sum_{t=0}^{T} \mathbb{I}_{\{X_t = x, A_t = a\}} / \sum_{t=0}^{T} \mathbb{I}_{\{X_t = x\}}$$

- Goal: find π best maching π_E
- Question: how to generalize to unknown states?
- Answer: find the reward function explaining $\pi_{E,T}$!

Markovian Decision Problems

MDPs: Making decisions in a stochastic world with a long-term goal.

- States: $\mathcal{X}$
- Actions: $\mathcal{A}$
- Transitions: $P(x'|x,a)$
- Rewards: $r(x,a)$

Policies and value functions

- Policy: $\pi : \mathcal{A} \times \mathcal{X} \to [0,1]$,
  $\sum_{a \in \mathcal{A}} \pi(a|x) = 1, \forall x \in \mathcal{X}$
- Set of all policies: Π
- Action-values under policy π:

$$Q^{\pi}(x,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(X_t, A_t) \,\middle|\, X_0 = x, A_0 = a \right],$$

  where the sequence $(X_t, A_t)_{t \ge 0}$ is generated by π
- Optimal action values: $Q^*(x,a) = \sup_{\pi} Q^{\pi}(x,a)$
- Bellman optimality equation for action values:

$$Q^*(x,a) = r(x,a) + \gamma \sum_{y \in \mathcal{X}} P(y|x,a) \max_{b \in \mathcal{A}} Q^*(y,b)$$

- Optimal policy: π^* maximizing Q for all state-action pairs

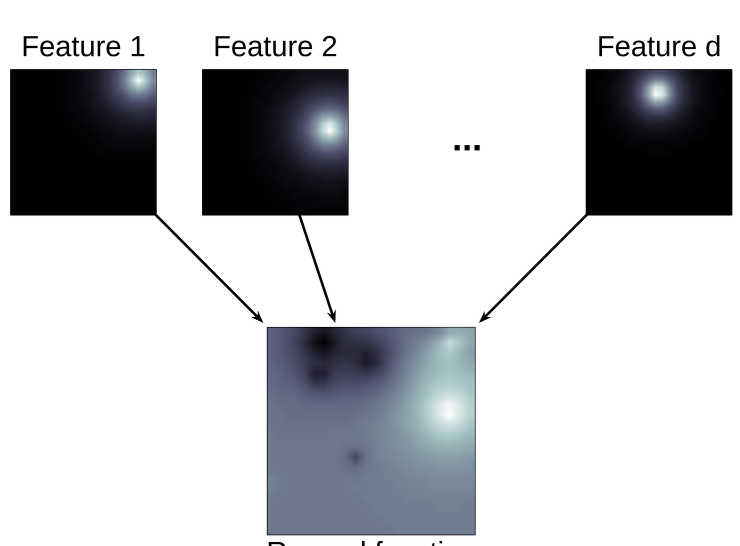
Inverse reinforcement learning

IRL problem: find r that π_E is optimal for! (Ng and Russell 2000)

- Advantage: better generalization
- Difficulty: ill-posed problem

Previous work on IRL: "maximum margin" algorithm of Abbeel and Ng (2004)

- Linearly parametrized rewards: $r_{\theta}(x,a) = \sum_{i=1}^{d} \theta_i \phi_i(x,a)$



Reward function

- Aims to match the *feature expectations* of the expert:

$$\psi^{\pi} = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \phi(X_t, A_t) \,\middle|\, X_0 \sim D \right],$$

  where the trajectory is generated by π and D is some distribution over $\mathcal{X}$
- Problem: Solution is highly variant to the scaling of the features!

New approach – IRL as an optimization task:

- Choose a parametric family of rewards $(r_{\theta})_{\theta \in \Theta}$ – not necessarily linear!
- Define a loss function, e.g.

$$J_T(\pi) = \sum_{x \in \mathcal{X}, a \in \mathcal{A}} \mu_T(x) (\pi(a|x) - \hat{\pi}_{E,T}(a|x))^2,$$

  μ_T: empirical estimate of stationary distribution
- Find θ minimizing $J_T(\pi_{\theta})$! (π_{θ} optimal for r_{θ})

The gradient can be computed $\to$ natural gradient search (Amari 1998) can be used to eliminate dependence on the scaling of the features!

Calculating the Gradients

Need the gradient $\nabla J(\pi_{\theta}) = J'(\pi_{\theta}) \pi'_{\theta}$

- Calculating $J'(\pi_{\theta})$ is trivial
- Main question: $\pi'_{\theta} = ?$

Problem: greedy policies are not differentiable! (because a "max" function is involved)
Solution: use a smooth mapping $G : Q \mapsto \pi$

- Boltzmann-policies:

$$G(Q)(a|x) = \frac{\exp[\beta Q(x,a)]}{\sum_{b \in \mathcal{A}} \exp[\beta Q(x,b)]}$$

- ...and their derivatives:

$$\frac{\partial \pi_{\theta}}{\partial \theta_k}(a|x) = \pi_{\theta}(a|x) \frac{\partial \ln[\pi_{\theta}(a|x)]}{\partial \theta_k} = \pi_{\theta}(a|x) \beta \left(\frac{\partial Q^*_{\theta}(x,a)}{\partial \theta_k} - \sum_{b \in \mathcal{A}} \pi_{\theta}(b|x) \frac{\partial Q^*_{\theta}(x,b)}{\partial \theta_k} \right)$$

Main result about calculating the subgradient of Q^*_{θ}:

Proposition. *Assume that the reward function r_{θ} is differentiable w.r.t. θ with uniformly bounded derivatives: $\sup_{(\theta,x,a) \in \mathbb{R}^d \times \mathcal{X} \times \mathcal{A}} \|r'_{\theta}(x,a)\| < +\infty$. The following statements hold:*

(1) *Q^*_{θ} is uniformly Lipschitz-continuous as a function of θ in the sense that for any (x,a) pair, $\theta, \theta' \in \mathbb{R}^d$, $|Q^*_{\theta}(x,a) - Q^*_{\theta'}(x,a)| \le L' \|\theta - \theta'\|$ with some $L' > 0$;*

(2) *Except on a set of measure zero, the gradient, $\nabla_{\theta} Q^*_{\theta}$, is given by the solution of the following fixed-point equation:*

$$\psi_{\theta}(x,a) = (r'_{\theta}(x,a))^T + \gamma \textstyle\sum_{y \in \mathcal{X}} P(y|x,a) \sum_{b \in \mathcal{A}} \pi(b|y) \psi_{\theta}(y,b),$$

where π is any policy that is greedy with respect to Q_{θ}.

Fixed-point equation analogous to the Bellman-equation:

$$r_{\theta}(x,a) \leftrightarrow (r'_{\theta}(x,a))^T, Q_{\theta}(x,a) \leftrightarrow \psi_{\theta}(x,a)$$

$\to$ known RL methods can be used to find the derivatives!

Algorithm

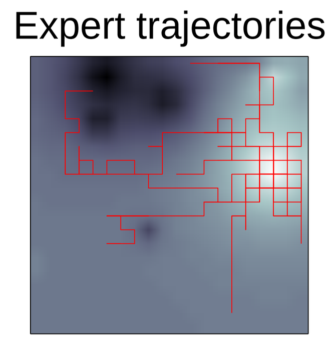
1. function **IRL**$(MDP \backslash r, \phi, \pi_{E,T}, \mu_T)$
2. $\theta_0 \leftarrow 0$
3. **for** $s = 0$ to *iteration-limit* **do**
4. $\quad r \leftarrow r_{\theta_s}$
5. $\quad MDP \leftarrow (MDP \backslash r) \cup r$
6. $\quad Q^* \leftarrow$ **find-optimal-Q**(MDP)
7. $\quad \pi_{\theta_s} \leftarrow G(Q^*)$
8. $\quad$ **for** $j = 1$ to d **do**
9. $\quad\quad MDP \leftarrow (MDP \backslash r) \cup \phi_j$
10. $\quad\quad (\psi_{\theta_s})_j \leftarrow$ **find-Q**(MDP, π_{θ_s})
11. $\quad$ **end for**
12. $\quad \nabla J \leftarrow$ **calculate-grad-J**$(\psi_{\theta}, \pi_{\theta}, \pi_E, \mu_T)$
13. $\quad \Delta\theta \leftarrow$ **compute-step**$(\nabla J, \nabla \pi_{\theta_s})$
14. $\quad \theta_{s+1} \leftarrow \theta_s - \Delta\theta$
15. **end for**
16. **return** r

The function **compute-step** may involve natural gradient computation or RPROP step size selection (Igel and Hüsken 2000).

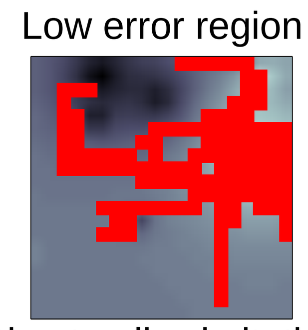
Experimental results

Improved generalization capabilities, policy matched on a great portion of $\mathcal{X}$:

Expert trajectories

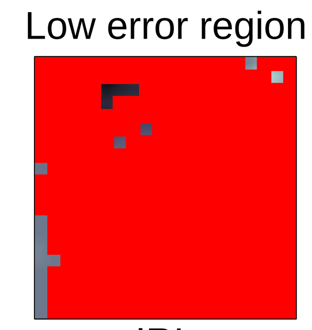


Low error region



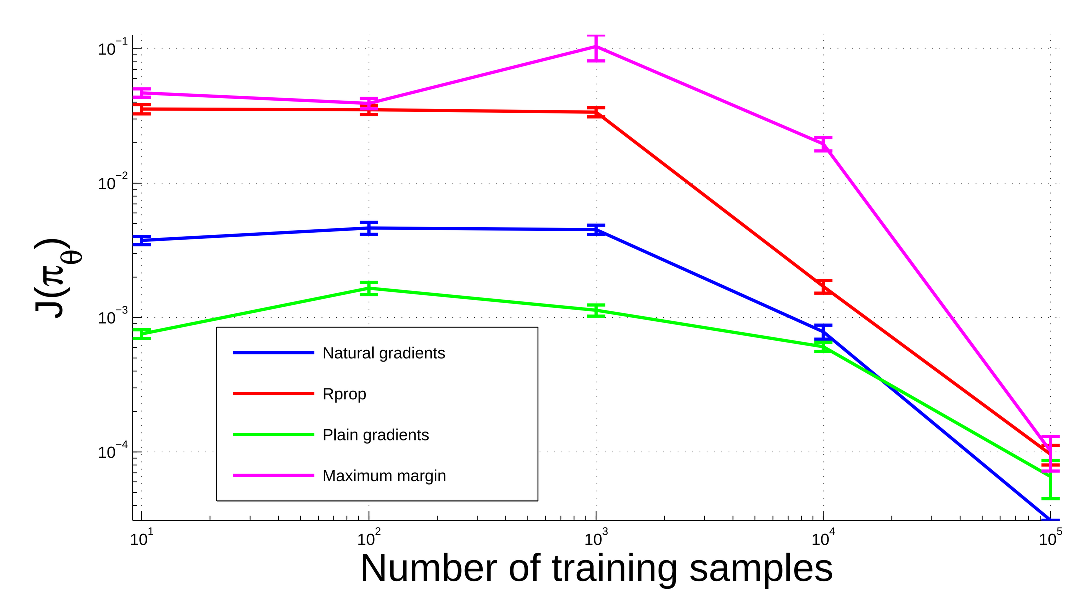
Direct policy imitation

Low error region



IRL

Performance vs. number of samples (results for the algorithm of Abbeel and Ng (2004) also shown):



	Natural gradients		RPROP		Plain gradients		Max margin	
	Mean	Deviation	Mean	Deviation	Mean	Deviation	Mean	Deviation
Original	0.0051	0.0010	0.0130	0.0134	0.0011	0.0068	0.0473	0.1476
Transformed	0	0	0.0110	0.0076	0.0256	0.0237	0.0702	0.0228
Perturbed	0.0163	0.0165	0.0197	0.0179	0.1377	0.3428	0.2473	0.3007

The results are shown for using original features, linearly transformed features and noise-perturbed features, respectively.

References

Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML'04*, pages 1–8, 2004. ISBN 1-58113-828-5.

S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2): 251–276, 1998.

Christian Igel and Michael Hüsken. Improving the Rprop learning algorithm. In *ICSC/NC 2000*, pages 115–121, 2000.

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

N.D. Ratliff, J.A. Bagnell, and M. Zinkevich. Maximum margin planning. In *ICML'06*, pages 729–736, 2006.