Inverse Reinforcement Learning

Artem Sokolov

Institute for Computational Linguistics, Heidelberg University

15 October 2018

Previously we saw that IL can be done by:

1. converting structured prediction into a search problem with specified search space and actions;
2. defining structured features over each state to capture the inter-dependency between output variables;
3. constructing a reference policy based on training data;
4. learning a policy that imitates the reference policy.
5. 'imitates' could mean:
   - behavioral cloning
   - cost-sensitive improvements to the policy (Searn)
   - or correcting the student model with queries to the expert (DAgger)

[Ng&Russel'00]

[T]he entire field of reinforcement learning is founded on the presupposition that the reward function,... is the most succinct, robust, and transferable definition of the task.

In other words:

Reward Hypothesis

All goals can be described by the maximisation of expected cumulative reward.

- real-world applications follow complex dynamics (unknown or hard to specify exactly)
- often hard to specify what cost function should be minimized to obtain the desired behavior
- so, it is hard to apply traditional RL methods to obtain a good controller
- on the other hand, demonstrations of the desired behavior are easy

Idea of IRL

Let's recover the reward first from demonstrations, and then use RL for control/planning.

Implicit assumptions:

- it's easier to learn the reward function than the policy directly
- the reward function generalizes better over states or similar tasks

Tabular Rewards

- finite horizon MDP $(\mathcal{S}, \mathcal{A}, P, C, \rho_0, T)$
  - ➡ $\mathcal{S}$ – set of S states
  - ➡ $\mathcal{A}$ – set of A actions
  - ➡ $P_t : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ – transition distribution
  - ➡ $C_t : \mathcal{S} \times \mathcal{A} \to [0, 1]$ – cost distribution
  - ➡ R – unknown reward
- π^* – expert policy we wish to mimic
- $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ – some policy (here, deterministic)
- γ – discount factor
- $V^*(s_1) = \mathbb{E}[R(s_1) + \gamma R(s_2) + \gamma^2 R(s_3) + \ldots |\pi]$ – state-value function
- $Q^*(s, a) = R(s) + \gamma \mathbb{E}_{s' \sim P_{sa}[V^\pi(s')]}$ – action-value function
- $V^*(s) = \max_\pi V^\pi(s)$ – optimal state-value function
- $Q^*(s, a) = \max_\pi Q^\pi(s, a)$ – optimal action-value function

Bellman Equations:

$$V(s) = \mathbb{E}[R(s) + \gamma \sum_{s'} P_{s'\pi(s)}(s') V^{\pi}(s')]$$
$$Q(s,a) = \mathbb{E}[R(s) + \gamma \sum_{s'} P_{s'a}(s') V^{\pi}(s')]$$

Optimal Policy:

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

IRL task

Find a set of possible reward functions $R(s)$ such that the expert's policy π is the optimal policy in MDP (S, A, P, γ, R).

- assume that optimal π is $\pi(s) \equiv a_1, \forall s$
- can rename action on every state if necessary

Thm.

Policy $\pi(s) \equiv a_1, \forall s$ iif

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

Proof.
From Bellman equation: $V^\pi = (I - \gamma P_{a_1})^{-1} R$. From π optimality:

$$\pi(s) \equiv a_1 \Leftrightarrow$$
$$\sum_{s'} P_{s' a_1}(s) V^\pi(s') \geq \sum_{s'} P_{s' a}(s) V^\pi(s'), \forall s, a \Leftrightarrow$$
$$P_{a_1}(I - \gamma P_{a_1})^{-1} R \succeq P_a (I - \gamma P_a)^{-1} R, \forall a \in \mathcal{A} \setminus a_1$$

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

- $R = 0$ would be a solution
- need additional restrictions on the solution
- One way to avoid ambiguity:

$$\sum_{s \in \mathcal{S}} Q^\pi(s, a_1) - \max_{a \in \mathcal{A} \setminus a_1} Q^\pi(s, a) \rightarrow \max$$

  - ➡ maximize the differences between the optimal quality and next best one
  - ➡ similar in spirit to large-margin learning
- Another way - regularization:

$$-\lambda \, ||R||_1$$

  - ➡ produces sparser rewards

Linear programming task:

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

Can be solved with LP for small state-spaces. [Ng and Russell, 2000]

Linear Rewards

[Ng and Russell, 2000]

- $\mathcal{S} = \mathbb{R}^n$
- assume there is a subroutine for approximating V^π
- assume there is a finite set of fixed bounded basis functions $\phi_i(s)$ ($\simeq$features)
- we will look for rewards that are a linear function of features

$$R(s) = \alpha_1\phi_1(s) + \cdots + \alpha_d\phi_d(s)$$

$$V^\pi(s) = \alpha_1 V_1^\pi(s) + \cdots + \alpha_d V_d^\pi(s)$$

- where V_i^π is a value for π if the reward is ϕ_i
- from the requirements of optimality of π

$$\mathbb{E}_{s'\sim P_{sa_1}}[V^\pi(s')] \geq \mathbb{E}_{s'\sim P_{sa}}[V^\pi(s')]$$

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

- IRL can be understood as linear programming
- ambiguous solutions require additional assumptions
- LP can be solved for small sets of states
- for large spaces can be reduced to LP again via assuming a functional structure on R

$$\mu(\pi) = \mathbb{E}_\pi[\sum_{t=1}^{\infty} \gamma^t \phi(s_t)] \in \mathbb{R}^k$$

$$V^\pi(s) = w \cdot \mu(\pi)$$

Observation

If $||w|| \leq 1$, $\phi(\cdot) \in [0, 1]$, and $||\mu(\pi) - \mu(\pi^*)|| \leq \epsilon$ [Ng and Russell, 2000]

$$\begin{aligned} \left| V^\pi - V^{\pi^*} \right| &= \left| w^\top \mu(\pi) - w^\top \mu(\pi^*) \right| \\ &\leq ||w|| \, ||\mu(\pi) - \mu(\pi^*)|| \\ &\leq 1 \cdot \epsilon = \epsilon \end{aligned}$$

Meaning: if we match features, we'll get a policy not worse than the expert's one.

- start with some π_0
- algorithm works by iteratively improving a mixture of policies

$$\sum_{i=1}^{n} \lambda_i \mu(\pi_i), \lambda_i \geq 0, \sum_i \lambda_i = 1$$

- (randomization takes place once before the start)
- find the best weighting of features μ s.t.

$$\begin{aligned} \max_{t,w} \quad & t \\ \text{s.t.} \quad & w^\top \mu(\pi^*) \geq w^\top \mu_i + t, \quad j = 0, \ldots, i-1 \\ & ||w|| \leq 1 \end{aligned}$$

- after w is found, run an RL control algorithm to get a corresponding policy π_i
- add the π_i to the set and repeat

- if the algorithm terminates with $t_{n+1} \leq \xi$

$$\forall w, ||w|| \leq 1 \quad \exists i \text{ s.t. } w^\top \mu(\pi_i) \geq w^\top \mu(\pi^*) - \xi$$

- one needs $O(k \ln k)$ samples of expert's behavior in order to get $|V - V^*| < \epsilon$

IRL as Games

IRL as Games

- all we required is feature expectation match
- so the previous approach can be as good as the expert
- but also as bad as the expert

Assumptions: [Syed and Schapire, 2008]

- $||w|| = 1$ and $w \succeq 0$, $w \in \mathbb{R}^k$
- k-dim features $\phi(\cdot) \in [-1, 1]^k$
- assume that the set of all (mixed) policies is fixed: Ψ

Objective:

$$V^* = \max_{\psi \in \Psi} \min_{w \in \mathbb{R}^k} [w^\top \mu(\psi) - w^\top \mu(\pi^*)]$$

If we denote the game matrix $G(i, j) = \mu_j(i) - \mu^*(i)$, where μ_j is the vector of feature expectations for deterministic policy π_j then

$$v^* = \max_{\psi \in \Psi} \min_{w \in \mathbb{R}^k} [w^\top G\psi] = \min_{w \in \mathbb{R}^k} \max_{\psi \in \Psi} [w^\top G\psi]$$

Two observations:

- $v^* \geq 0$ (for any w the optimal policy has a non-negative v^*: G is defined w.r.t the the π^*)
- could be even $v^* > 0$ if $\mu(\phi) \succ \mu(\pi^*)$, because $w \succeq 0$
- $\Rightarrow$ we can improve over the expert (provided a sufficiently large Ψ)

Sketch of the algorithm

1: init: $w_0(i) = 1$
2: $G(i, \mu) = ((1 - \gamma)(\mu(i) - \mu^*(i)) + 2)/4$
3: **for** $t = 0, \ldots$ **do**
4: $\rho(i) = \frac{w_t(i)}{\sum_i w_t(i)}$
5: compute the optimal policy π_t w.r.t. $R(s) = w^\top \phi(s)$
6: compute feature expectations $\mu_t = \mu(\pi_t)$
7: $w_{t+1}(i) = w_t \cdot e^{\ln \beta G(i, \mu_t)}$
8: return: mixed policy ψ that assign prob. $\frac{1}{T}$ to all π^t

- similar in spirit to expert advice
- adversarial losses are the game values relative to the expert
- can be solved with online convex optimization
- sample complexity $O(\ln k)$ (for feature matching it was $O(k \ln k)$)
- can also be applied to the case of no expert (set $\mu^* = 0$)
- potentially can produce policies that are better than the expert

Several ways to find ambiguity in reward recovery:

- maximizing the difference to the next-best action-values [Ng and Russell, 2000]
- matching feature expectations with a max-margin on rewards [Abbeel and Ng, 2004]
- formulating an adversarial game [Syed and Schapire, 2008]
- global decisions (a departure from a local, state-action, decision making)
  - ➡ minimizing trajectory disagreement with a task-dependent margin [Ratliff et al., 2006]
  - ➡ another way: maximize the entropy of trajectory distribution [Ziebart et al., 2008]

Max-Margin Reward Learning

The structured SVM model from a previous lecture

StructSVM Objective

$$R(w) = \frac{1}{N}\sum_{i=1}^{N}\Big(\max_{y\in\mathcal{Y}}(w^\top\phi(x_t,y)+\ell(y_t,y)) - w^\top\phi(x_t,y_t)\Big) + \frac{\lambda}{2}\,||w||^2$$

- driving the trajectories to be similar
- deviations are penalized using the task loss
- convex loss $\Rightarrow$ FTL (SGD) applies

[Ratliff et al., 2006]

Maximum Entropy

- again, match the feature expectations (this way the state-values are close to the expert's)
- maximizing the entropy of a distribution under constraints of feature expectations = maximizing the likelihood of demonstrations under the exponential distribution over trajectories $P(\tau) = \frac{e^{w^\top \sum_{s \in \tau} \phi(s)}}{Z(w)}$

$$w^* = \arg\max_w L(w) = \arg\max_w \sum_{\mathcal{D}} \log P(\tau | w)$$

- gradient has simple form

$$\nabla_w L(w) = \sum_{s \in \tau} \phi(s) - \mathbb{E}[\sum_{s \in \tau} \phi(s) | w]$$

- calculating the expectations in practice
  - ➡ small finite spaces:
    - value-iteration (backward/outside algorithm) for chains/trees
  - ➡ continuous spaces:
    - MC sampling
    - beam search

Abbeel, P. and Ng, A. Y. (2004).
Apprenticeship learning via inverse reinforcement learning.
In ICML.

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

Ratliff, N. D., Bagnell, J. A., and Zinkevich, M. A. (2006).
Maximum margin planning.
In ICML.

Syed, U. and Schapire, R. E. (2008).
A game-theoretic approach to apprenticeship learning.
In NIPS.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. (2008).
Maximum entropy inverse reinforcement learning.
In AAAI.