Imitation Learning: Inverse Reinforcement Learning

Alina Vereshchaka

University at Buffalo
avereshc@buffalo.edu

June 13, 2019

*Slides are adopted from Berkley Deep RL course CS294-112 & Deep RL and Control, CMU 10703
Inverse Reinforcement Learning (IRL)

Computer Games

Real World Scenarios

- robotics
- dialog
- autonomous driving

Mnih et al. ‘15

what is the **reward**?
often use a proxy

**frequently easier to provide expert data**

Inverse reinforcement learning: infer reward function from roll-outs of expert policy
**Inverse Optimal Control / Inverse Reinforcement Learning:**
infer reward function from demonstrations

(IOC/IRL)  
(Kalman ’64, Ng & Russell ’00)

given:
- state & action space
- samples from $\pi^*$
- dynamics model (sometimes)

goal:
- recover reward function
- then use reward to get policy

**Challenges**
underdefined problem
difficult to evaluate a learned reward
demonstrations may not be precisely optimal
Problem Setup

- **Given:**
  - State space, action space
  - No reward function
  - Dynamics (sometimes) $T_{s,a}(s_{t+1}|s_t, a_t)$
  - Teacher’s demonstration:
    $s_0, a_0, s_1, a_1, s_2, a_2, ...$
    ($= \text{trace of the teacher’s policy } \pi^*$)

- **Inverse RL**
  - Can we recover $R$?

- **Apprenticeship learning via inverse RL**
  - Can we then use this $R$ to find a good policy?

- **Behavioral cloning (previous)**
  - Can we directly learn the teacher’s policy using supervised learning?
Inverse Reinforcement Learning (IRL)

Reinforcement Learning

Environment  \(\downarrow\)  Rewards  \(\rightarrow\)  RL  \(\rightarrow\)  Behavior

Inverse Reinforcement Learning

Environment  \(\downarrow\)  Rewards  \(\leftarrow\)  IRL  \(\leftarrow\)  Behavior
Inverse Reinforcement Learning (IRL)

"forward" reinforcement learning

given:
states $s \in S$, actions $a \in A$
(sometimes) transitions $p(s' | s, a)$
reward function $r(s, a)$

learn $\pi^*(a | s)$

--

inverse reinforcement learning

given:
states $s \in S$, actions $a \in A$
(sometimes) transitions $p(s' | s, a)$
samples $\{\tau_i\}$ sampled from $\pi^*(\tau)$

learn $r_\psi(s, a)$

...and then use it to learn $\pi^*(a | s)$

linear reward function:

$r_\psi(s, a) = \sum_i \psi_i f_i(s, a) = \psi^T f(s, a)$

neural net reward function:

$r_\psi(s, a)$ parameters $\psi$
Inverse Reinforcement Learning (IRL)

IRL problem is to find a reward function that can explain the expert behavior.
Inverse Reinforcement Learning (IRL)

Environment Model (MDP)

Cost Function $c(s)$

Inverse Reinforcement Learning (IRL)

$c$ rationalizes expert trajectories

Expert's Trajectories $s_0, s_1, s_2, \ldots$

Optimal Policy $\pi_E$
Inverse RL with linear reward/cost function

\[ \pi^*: x \rightarrow a \]

**Expert**

Interacts

**Demonstration**

\[ y^* = (x_1, a_1) \rightarrow (x_2, a_2) \rightarrow (x_3, a_3) \rightarrow \cdots \rightarrow (x_n, a_n) \]

\[ w^T f(y^*) = w^T + w^T + w^T + \cdots + w^T \]

Expert trajectory reward/cost

Jain, Hu
Principle: Expert is optimal

- Find a reward function $R^*$ which explains the expert behavior
- i.e., assume expert follows optimal policy, given her $R^*$
- Find $R^*$ such that

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) | \pi^*\right] \geq \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) | \pi\right] \quad \forall \pi$$
(We assume reward is linear over features)

Let \( R(s) = w^T \phi(s) \) where \( w \in \mathbb{R}^n \), and \( \phi : S \to \mathbb{R}^n \)

\[
\begin{align*}
\mathbb{E}\left[ \sum_{t=0}^{\infty} \gamma^t R(s_t) | \pi \right] &= \mathbb{E}\left[ \sum_{t=0}^{\infty} \gamma^t w^T \phi(s_t) | \pi \right] \\
&= w^T \mathbb{E}\left[ \sum_{t=0}^{\infty} \gamma^t \phi(s_t) | \pi \right] \\
&= w^T \mu(\pi)
\end{align*}
\]
Feature Based Reward Function

(We assume reward is linear over features)

Let \( R(s) = w^T \phi(s) \) where \( w \in \mathbb{R}^n \), and \( \phi : S \to \mathbb{R}^n \)

\[
\mathbb{E}\left[ \sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi \right] = \mathbb{E}\left[ \sum_{t=0}^{\infty} \gamma^t w^T \phi(s_t) \mid \pi \right] = w^T \mathbb{E}\left[ \sum_{t=0}^{\infty} \gamma^t \phi(s_t) \mid \pi \right] = w^T \mu(\pi)
\]

expected discounted sum of feature values or feature expectations—dependent on state visitation distributions

Substituting\
\[
\mathbb{E}\left[ \sum_{t=0}^{\infty} \gamma^t R^*(s_t) \mid \pi^* \right] \geq \mathbb{E}\left[ \sum_{t=0}^{\infty} \gamma^t R^*(s_t) \mid \pi \right] \quad \forall \pi
\]

gives us: \( \text{Find } w^* \text{ such that } w^{*T} \mu(\pi^*) \geq w^{*T} \mu(\pi) \quad \forall \pi \)
1. Guess an initial reward function \( R(s) \)

2. Learn policy \( \pi(s) \) that optimizes \( R(s) \)

3. Whenever \( \pi(s) \) chooses action different from expert \( \pi^*(s) \)
   
   - Update estimate of \( R(s) \) to assure value of \( \pi^*(s) \) > value of \( \pi(s) \)

4. Go to 2
Feature Matching

- **Inverse RL starting point**: find a reward function such that the expert outperforms other policies

Let \( R(s) = w^T \phi(s) \), where \( w \in \mathbb{R}^n \), and \( \phi : S \rightarrow \mathbb{R}^n \)

Find \( w^* \) such that \( w^*^T \mu(\pi^*) \geq w^T \mu(\pi) \quad \forall \pi \)

Here we define \( \mu(\pi^*) \) as the expected discounted sum of feature values obtained by following this policy.

Given \( m \) trajectories generated by following the policy, we estimate it as

\[
\hat{\mu}_E = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{\infty} \gamma^t \phi(s_t^{(i)})
\]

Abbeel and Ng 2004
Apprenticeship Learning [Abbeel & Ng, 2004]

- Assume $R_w(s) = w^T \phi(s)$ for a feature map $\phi : S \to \mathbb{R}^n$

- Initialize: pick some policy $\pi_0$

- Iterate for $i = 1, 2, \ldots$:
  - **"Guess" the reward function:**
    Find a reward function such that the teacher maximally outperforms all previously found policies
    
    $$\max_{\gamma, w : \|w\|_2 \leq 1} \gamma$$
    
    s.t. $w^T \mu(\pi^*) \geq w^T \mu(\pi) + \gamma$  $\forall \pi \in \{\pi_0, \pi_1, \ldots, \pi_{i-1}\}$

  - **Find optimal control policy** $\pi$ for the current guess of the reward function $R_w$
  
  - $\gamma \leq \varepsilon/2$ exit the algorithm
Learn a cost function $c$, such that

$$
\pi_E = \arg \min_{\pi} \mathbb{E}_{\pi}[c(s, a)] - H(\pi)
$$

(1)

$H(\pi)$ is a regularizer

**Solution:** Maximum casual entropy IRL:

$$
\max_{c \in C} \left( \min_{\pi \in \Pi} -H(\pi) + \mathbb{E}_{\pi}[c(s, a)] \right) - \mathbb{E}_{\pi_E}[c(s, a)]
$$

(2)

$\mathbb{E}_{\pi_E}[c(s, a)]$ - expert has small cost

$\min_{\pi \in \Pi} -H(\pi) + \mathbb{E}_{\pi}[c(s, a)]$ - everything else has high cost
Characterizing the policy

- Demonstration → Cost
  \( \psi \)-regularized IRL, finds a cost function, such that
  \[
  IRL_\psi(\pi_E) = \arg \max_{c \in \mathbb{R}^{S \times A}} -\psi(c) + \left( \min_{\pi \in \Pi} \mathbb{E}_\pi[c(s, a)] - H(\pi) \right) - \mathbb{E}_{\pi_E}[c(s, a)] \tag{3}
  \]

- Cost → Policy
  Reinforcement Learning
  \[
  RL(c) = \arg \min_{\pi \in \Pi} \mathbb{E}_\pi[c(s, a)] - H(\pi) \tag{4}
  \]

- Directly characterize \( RL \circ IRL_\psi(\pi_E) \)
ψ-regularized IRL seeks a policy whose occupancy measure (ψ) is close to the expert:

$$RL \circ IRL_\psi(\pi_E) = \arg \min_{\pi \in \Pi} \psi^*(\rho_\pi - \rho_{\pi_E}) - H(\pi)$$ (5)

ψ* is a convex conjugate of ψ ρ - occupancy measure is a distribution of state-action pairs that an agent encounters when navigating the environment with policy π

$$\rho_\pi(s, a) = \pi(a|s) \sum_{t=0}^{\infty} \gamma^t P(s_t = s|\pi)$$ (6)
Goal: find $\pi$ performing better than $\pi_E$ over costs linear in the features

$$\min_{\pi} \max_{c \in C} \mathbb{E}_{\pi} [c(s, a)] - \mathbb{E}_{\pi_E} [c(s, a)]$$

(7)
GAN Recap

Generative Adversarial Network

\[
\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]
\]
Generative Adversarial Network (GAN)

**Algorithm 1** Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, $k$, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations do
  for $k$ steps do
    • Sample minibatch of $m$ noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
    • Sample minibatch of $m$ examples $\{x^{(1)}, \ldots, x^{(m)}\}$ from data generating distribution $p_{data}(x)$.
    • Update the discriminator by ascending its stochastic gradient:
      \[
      \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D\left(x^{(i)}\right) + \log \left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \right].
      \]
  end for
  • Sample minibatch of $m$ noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
  • Update the generator by descending its stochastic gradient:
    \[
    \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log \left(1 - D\left(G\left(z^{(i)}\right)\right)\right).
    \]
end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.
Solution: use a more complex class of cost functions

\[
\min_{\pi} \sup_{D \in (0,1)^{S \times A}} \mathbb{E}_\pi[\log(D(s,a))] + \mathbb{E}_{\pi_E}[\log(1 - D(s,a))] \quad (8)
\]
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** $i = 0, 1, 2, \ldots$ **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

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

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

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

6: **end for**