

Last Time:

- Boltzmann Rationality
- Intent Inference & Expression

lecture 7

HR1, FALL '25

Andrea Bajcsy

This Time:

- Reward learning
- Policy learning

Reward learning vs. Policy learning?

It's often easier to demonstrate good behavior than to manually encode it into a (MDP) model.

ex. in autonomous driving it is hard to write down a reward function that models your personal driving "style."

Assume we are given some trajectory demonstrations:

$$\tau_D = (s_0, a_0, s_1, a_1, \dots, s_T, a_T)$$

Inverse Reinforcement Learning (IRL)

Both cases of
imitation learning

Behavior Cloning (BC)

→ learn a reward function which, when optimized, yields a policy that does the task in the desired way

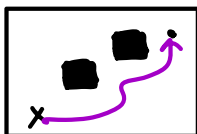
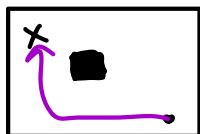
→ seeks to understand human demonstrator's objective, rather than just copying their actions

⊕ "interpretable" b/c you know "what matters" to the agent

⊕ can adapt/generalize to new environments (via RL)

ex.  learn: $R = \text{mind-the-}$ 

Generalizes!



⊖ Typically needs extra "RL steps"

→ Directly learn a policy (π) that imitates the human demonstrator, mapping states \rightarrow actions

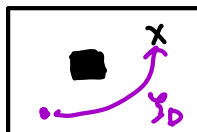
→ Does NOT explicitly infer the demonstrator's underlying objective; just mimics it

⊕ "simple" b/c you have the inputs (ex. states) and you have the "labels" (ex. actions); no need to infer reward.

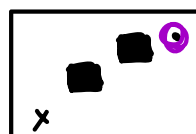
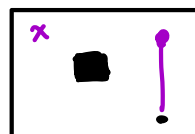
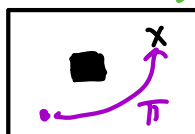
⊕ computationally cheaper (no extra RL step)

⊖ compounding errors

⊖ Struggles in unseen situations

ex.  learn: $\pi(a|s)$

only imitates! ✓



INVERSE REINFORCEMENT LEARNING - IRL (aka. Inverse optimal control)

Recall: "Forward" Reinforcement Learning

• Given: $s \in S, a \in A$, (sometimes) $P(s'|s, a)$, $r(s, a)$

• Goal: learn $\pi^*(a|s)$ s.t

$$\pi^* := \underset{\pi}{\operatorname{argmax}} \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t)) \right]$$

over $P(S_{t+1}|s_t, \pi)$

INVERSE RL:

• Given: $s \in S, a \in A$, (sometimes) $P(s'|s, a)$ AND $\mathcal{D} := \{ \tau_i \}_{i=1}^N$ demonstrations from π^* e.g. Humans!

• Goal: learn $r_{\theta}(s, a) \leftarrow (+ \text{ then use it to learn } \pi^*)$
↑ reward params.

Ⓢ Wait, but how do I search over all reward functions to find the "best" one given my dataset \mathcal{D} ?

A1 Linear rewards are where a lot of foundations started b/c easier to optimize! Assume:

$$r_{\theta}(s) = \theta_1 \phi_1(s) + \theta_2 \phi_2(s) + \dots + \theta_d \phi_d(s) \in \mathbb{R}$$

$\theta := \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \in \mathbb{R}^d$

$$= \theta^T \phi(s)$$

feature vector: ex. $\phi(s) := \begin{bmatrix} \text{distance to goal} \\ \text{distance to obstacle} \\ \vdots \end{bmatrix} \in \mathbb{R}^d$

$s = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix} \in \mathbb{R}^n$

$\phi_d(s)$ feature function

MAX ENTROPY IRL [Ziebart et. al, AAAI 2008]

Relaxes the assumption that τ_D is perfect, but it still allows the robot to learn H 's reward.

Key idea is to treat demonstrations as observations drawn from some distribution that models the demonstrator as approximately optimal

↓
SOUND FAMILIAR? ;)

$$(1) \quad P(\zeta | \theta) = \frac{e^{R_{\theta}(\zeta)}}{\int e^{R_{\theta}(\bar{\zeta})} d\bar{\zeta}}$$

⇒ let's plug in our linear reward model to get.

$$P(\zeta | \theta) = \frac{e^{\theta^T \phi(\zeta)}}{\int e^{\theta^T \phi(\bar{\zeta})} d\bar{\zeta}} \quad (\otimes)$$

Ⓚ How do we find θ (i.e. the reward parameter) given $\mathcal{D} = \{\zeta_D\}$?

ⓐ If $\theta \in \{\theta^A, \theta^B, \theta^C, \dots, \theta^Z\}$ in (small) discrete, we could use Bayes' Rule to obtain $\hat{\theta}^* = \underset{\theta}{\operatorname{argmax}} P(\theta | \mathcal{D})$ via lecture 5/6

ⓐ If $\theta \in \mathbb{R}^d$ is continuous, then it's not tractable to have an exact posterior, then you compute instead the maximum likelihood estimate via gradient descent

MLE: $\hat{\theta} = \underset{\theta}{\operatorname{argmax}} P(\zeta_D | \theta)$

⇒ $= \underset{\theta}{\operatorname{argmax}} \log P(\zeta_D | \theta) \Rightarrow \underset{\theta}{\operatorname{max}} \log \frac{e^{\theta^T \phi(\zeta_D)}}{\int e^{\theta^T \phi(\bar{\zeta})} d\bar{\zeta}}$

$$= \max_{\theta} \left[\theta^T \phi(\xi_D) - \log \left[\int e^{\theta^T \phi(\xi)} d\bar{\xi} \right] \right]$$

Know from opt. that we can find $\hat{\theta}$ via gradient ascent!

$$\begin{aligned} \nabla_{\theta} \boxed{} &= (\text{algebra} + \text{calculus} \dots) \\ &= \phi(\xi_D) - \mathbb{E}_{\xi \sim P(\xi|\theta)} [\phi(\xi)] \end{aligned}$$

Max Ent IRL Gradient Ascent Rule:

$$\theta_{i+1} \leftarrow \theta_i + \alpha \left(\phi(\xi_D) - \mathbb{E}_{\xi \sim P(\xi|\theta_i)} [\phi(\xi)] \right)$$

feature values
from expert demos.

expected feature values
induced by reward param.
guess θ_i

Behavior Cloning (BC)

The "simplest" algorithm to run on your dataset $\mathcal{D} := \{ \tau_D \}$

Train a model (e.g. NN) to predict expert actions given observed states, treating it as if it was supervised learning:

$$\min_{\theta} \sum_{(s_i, a_i) \in \mathcal{D}} \ell(a_i, \pi_{\theta}(s_i))$$

some loss penalizing predicted vs. real expert actions:
(ex. $\|a - \pi(s)\|_2^2$)

looks like supervised learning, trying to match expert @ every state in demonstrations \mathcal{D}

⇒ Issues w/ BC:

In standard SL, we assume that the data pts. are

independent & identically distributed (IID)

imitation learning

$$P_{\mathcal{I}_1}(x) = P_{\mathcal{I}_k}(x) \forall x \forall k$$

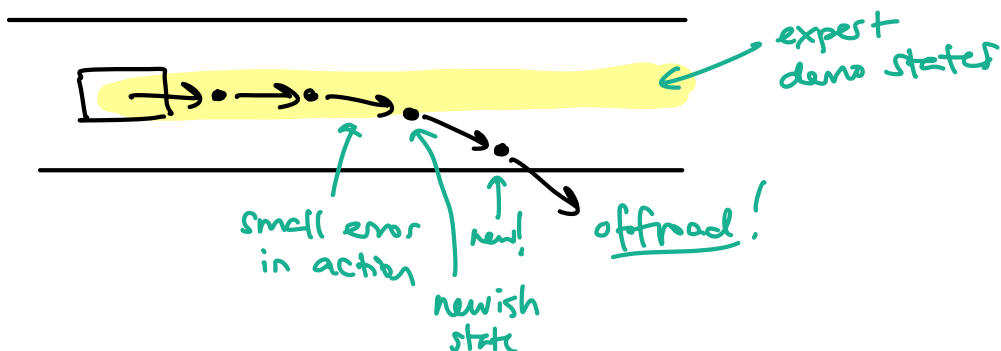
indep. $P_{\mathcal{I}_1 \dots \mathcal{I}_N}(x_1 \dots x_n) = \prod_{i=1}^N P_{\mathcal{I}_i}(x_i)$

However, IL is in a sequential domain where

current actions influence future states ⇒ assumption is wrong!

⇒ this means BC is prone to error accumulation & can't recover as it drifts away from demonstration distribution

ex. ALVINN (CMU '88)



Solutions to BC Error Accumulation:

↳ DAgger (Dataset Aggregation, 2011), DART (2017),
GAIL (2016) + more!