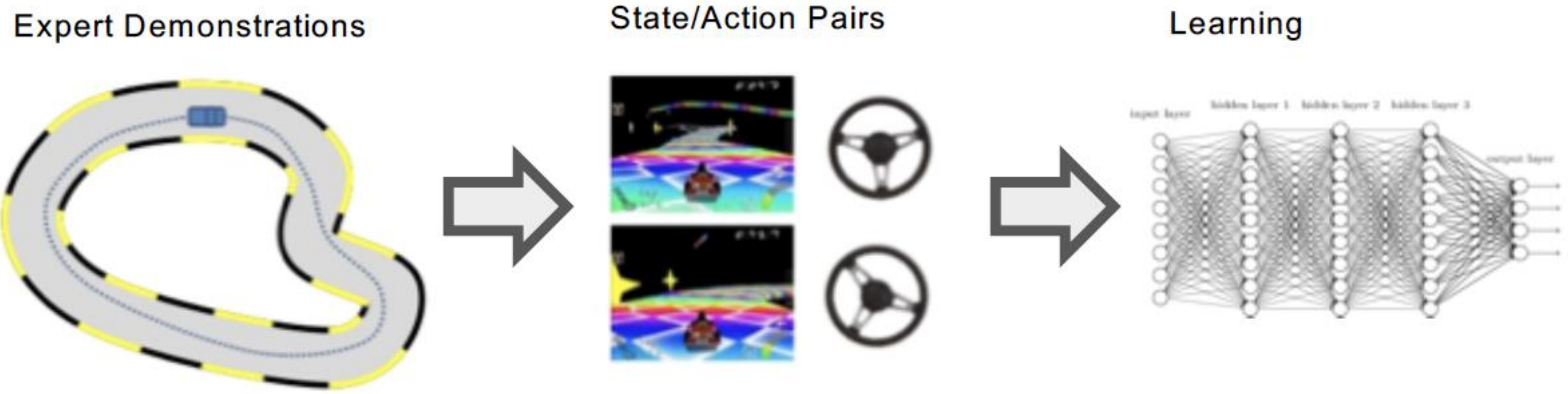
CSCE-642 Reinforcement Learning
Imitation Learning

Instructor: Guni Sharon

Based on slides by: Yisong Yue and Hoang M. Le

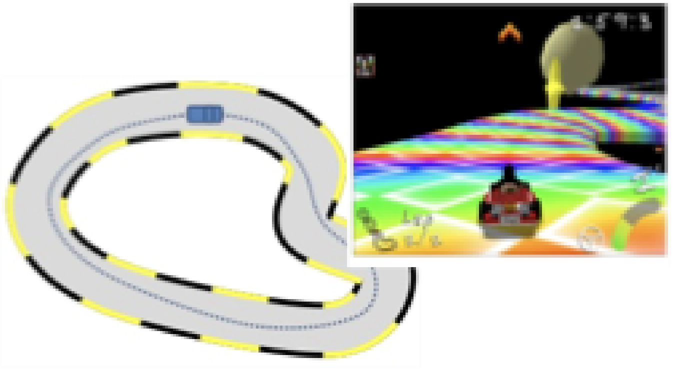
Imitation Learning in a Nutshell

- **Given**: demonstrations or a demonstrator (interactive)
- **Goal**: train a policy to imitate the demonstrator



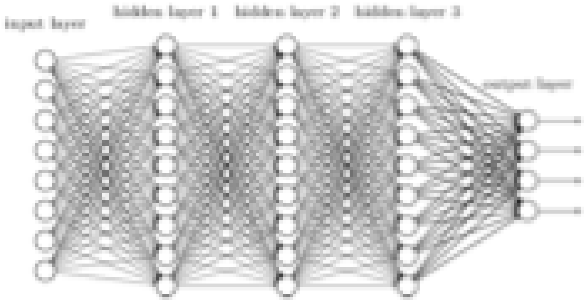
Images from Stephane Ross

Ingredients of Imitation Learning



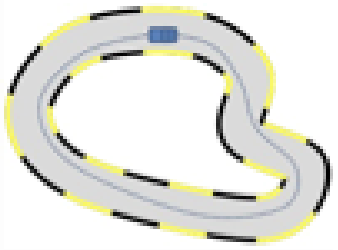
Demonstration or demonstrator

Environment / Simulator

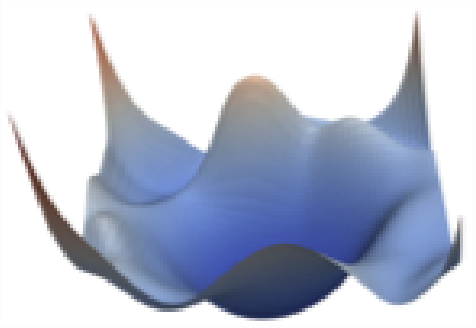


Policy Class

vs



Loss function



Learning algorithm

Notation

- $P(\tau|\pi)$: distribution of trajectories induced by a policy
  - $s_0 \sim \rho_0(s_0),\ \ a_t \sim \pi(a_t|s_t),\ \ s_{t+1} \sim P(s_{t+1}|s_t, a_t)$
  - $\rho_0(s_0)$ is the distribution of the initial state s_0
- $P(S|\pi)$: distribution of states induced by a policy
- D (set of demonstrations) $\sim P\left(S|\pi^d\right)$
  - Rollouts using π^d

Imitation Learning Algorithms

- Behavioral Cloning
  - Learn to do exactly what the demonstrator showed you
- Direct (interactive) Imitation Learning
  - Solicit the demonstrator for new (relevant) examples as you learn
- Inverse RL
  - Learn the demonstrator’s objective (its reward function)
  - Train a policy that optimizes the same objective
- Generative adversarial
  - Learn to act indistinguishably from the demonstrator

Behavioral Cloning

- Reduction to Supervised Learning
- Learning the policy is defined as an optimization problem
  - $\theta = \arg\min_{\theta} E_{s,a\sim D}\left[loss\left(a, \pi_{\theta}(s)\right)\right]$
- Results in minimizing 1-step deviation error along the expert trajectories
  - What's the problem here?
- Doesn't generalize
- The learned policy is clueless when a state outside of the demonstration set is encountered
- "As soon as the learner makes a mistake, it may encounter completely different observations than those under expert demonstration, leading to a compounding of errors." [Ross et al. 2011]

Video from Stephane Ross

Worst case analysis

- Assume 0,1 loss (negative of reward)

$$loss(s,a) = \begin{cases} 0 & \text{if } a = \pi^d(s) \\ 1 & \text{else} \end{cases}$$

- Assume $\forall s \in D,\ \pi_\theta\big(a \neq \pi^d(s)|s\big) \leq \epsilon$
  - Low training error
- $E[\sum_t l(s_t, a_t)] \leq \underbrace{\epsilon T}_{\text{Made a mistake on step 1 -> unknown terrain -> continue to make mistakes until the end}} + \underbrace{(1-\epsilon)\epsilon(T-1)}_{\text{Made a mistake on step 2 -> unknown terrain -> continue to make mistakes until the end}} + (1-\epsilon)^2\epsilon(T-2) + \cdots$

- $= O(\epsilon T^2)$ Vary bad! (even for small epsilon)

Worst case analysis

- Let's consider a stronger (yet reasonable) assumption: the policy can generalize well to the demonstration distribution
- The weak assumption, $\forall s \in D,\ \pi_\theta\big(a \neq \pi^d(s)|s\big) \leq \epsilon$
- Becomes: $\forall s \sim P\big(s|\pi^d\big),\ E\big[\pi_\theta\big(a \neq \pi^d(s)|s\big)\big] \leq \epsilon$
- Do we get a better worst case?
  - No, still $O(\epsilon T^2)$
  - See [Ross et al. 2011]

When to use Behavioral Cloning?

Advantages

- Simple
- Simple
- Efficient

Use When:

- 1-step deviations not too bad
- Learning reactive behaviors
- Expert trajectories “cover” state space

Disadvantages

- Distribution mismatch between training and testing
- No long-term planning

Don’t Use When:

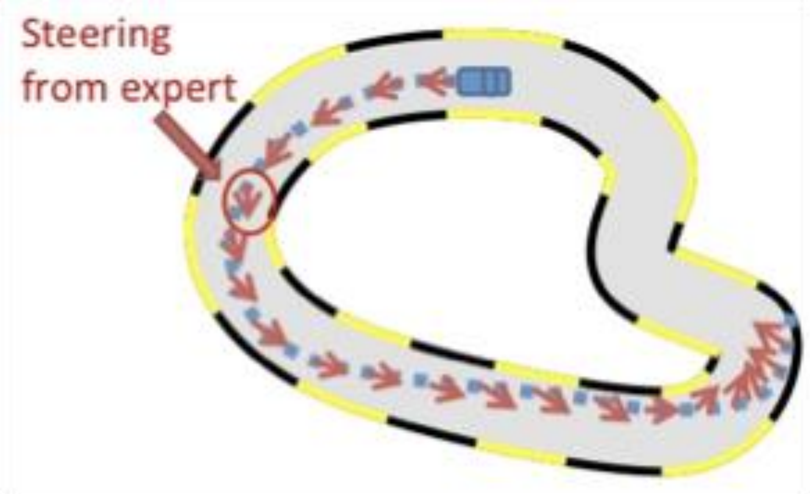
- 1-step deviations can lead to catastrophic error
- Optimizing long-term objective (at least not without a stronger model)

(robust) Imitation Learning

- The learned policy might encounter states that weren't visited by the demonstrator (they are not in the demonstration trajectories)
- Instead of minimizing the (chosen action) loss for states drawn from $D \sim P\left(s|\pi^d\right)$, minimize for states drawn from the learned policy: $P(s|\pi_\theta)$
- $\theta = \arg\min_\theta E_{s,a\sim D}\left[loss\left(a, \pi_\theta(s)\right)\right] \rightarrow \arg\min_\theta E_{s\sim P(S|\pi_\theta)}\left[loss\left(\pi^d(s), \pi_\theta(s)\right)\right]$

Interactive IL

- Use an interactive demonstrator to shift from $\arg\min_{\theta} E_{s,a\sim D}\left[loss\left(a, \pi_{\theta}(s)\right)\right]$ to
  $\arg\min_{\theta} E_{s\sim P(S|\pi_{\theta})}\left[loss\left(\pi^{d}(s), \pi_{\theta}(s)\right)\right]$
- **Assumption**: Can query expert at any state
  - $\forall s, \pi^{d}(s)$ is available
- **Key**: sample trajectories using $\pi_{\theta}(s)$
  - Evaluate the loss on the sampled trajectories



Alternating Optimization (Naïve Attempt)

Init θ_0

Repeat:

1. $P_i = P\left(S|\pi_{\theta_i}\right)$
2. $\theta_{i+1} = \arg\min_{\theta} E_{s\sim P_i}\left[loss\left(\pi^d(s), \pi_\theta(s)\right)\right]$

Might oscillate – Not Guaranteed to Converge

Data Aggregation (DAgger) [Ross et al. 2011]

- “DAGGER proceeds by collecting a dataset at each iteration under the current policy and trains the next policy under the aggregate of all collected datasets”
- “trains a deterministic policy that achieves good performance guarantees under its induced distribution of states”
- “collecting a dataset at each iteration under the current policy and trains the next policy under the aggregate of all collected datasets”

Data Aggregation (DAgger) [Ross et al. 2011]

- "To better leverage the presence of the expert in our imitation learning setting, we optionally allow the algorithm to use a modified policy π_i that queries the expert to choose controls a fraction of the time while collecting the next dataset"
- "the first few policies, with relatively few datapoints, may make many more mistakes and visit states that are irrelevant as the policy improves"

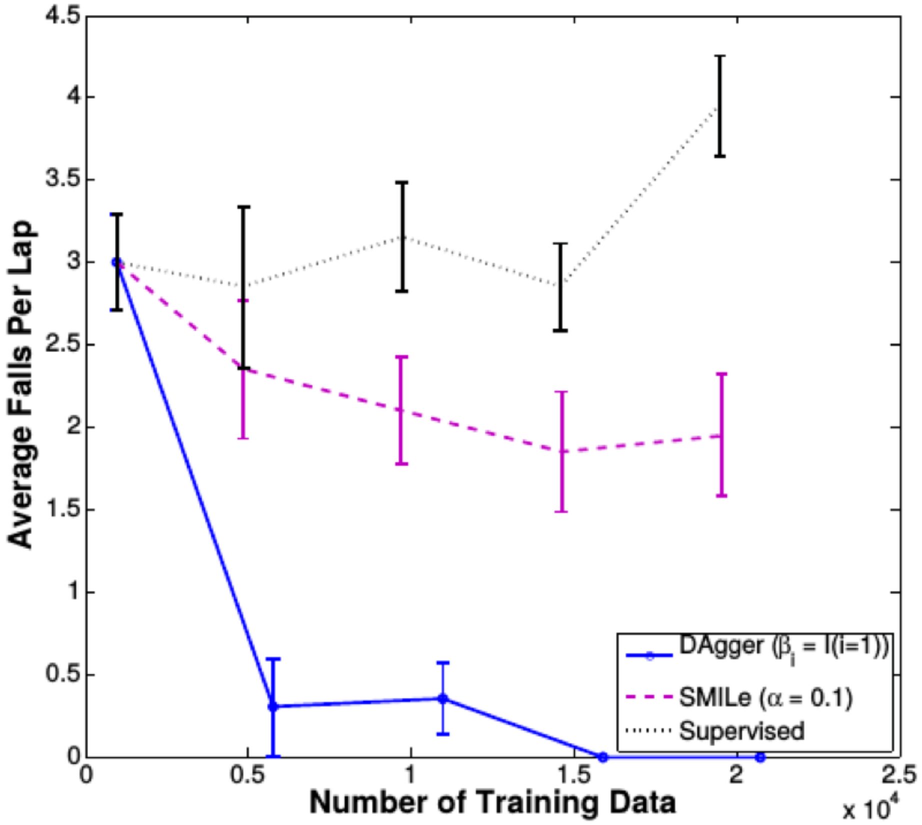
Initialize $\mathcal{D} \leftarrow \emptyset$.
Initialize $\hat{\pi}_1$ to any policy in Π.
for $i = 1$ **to** N **do**
  Let $\pi_i = \beta_i \pi^* + (1 - \beta_i)\hat{\pi}_i$.
  Sample T-step trajectories using π_i.
  Get dataset $\mathcal{D}_i = \{(s, \pi^*(s))\}$ of visited states by π_i and actions given by expert.
  Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{D}_i$.
  Train classifier $\hat{\pi}_{i+1}$ on $\mathcal{D}$.
end for
Return best $\hat{\pi}_i$ on validation.

No-regret algorithm

- Online learning provides a policy, π_i, at every iteration
- Applying π_i result in a loss (-reward), $l_i(\pi_i)$
- The loss (gradient) is used to train π_{i+1} which, when applied, incurs $l_{i+1}(\pi_{i+1})$
- $Regret_n = \frac{1}{n}\sum_{i=1}^{n} l_i(\pi_i) - \min_{\pi \in \Pi} \frac{1}{n}\sum_{i=1}^{n} l_i(\pi)$
- No-regret algorithm: $\lim_{n \to \infty} Regret_n = 0$
- DAgger is proven to be no-regret* = converges relatively quickly
  - See assumptions regarding convexity of the loss function and learning steps in [Ross et al. 2011]

Results

1:59:3
Lap
1/1
1st



Average Falls Per Lap
Number of Training Data
x 10^4
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
DAgger (β_i = I(i=1))
SMILe (α = 0.1)
Supervised

Inverse RL

- What if we don’t have an interactive demonstrator?
  - We only have access to an offline set of demonstrated trajectories
- Behavioral cloning is not robust
  - Suffers from overfitting
  - We know what to do in observed states but can’t generalize well to other states
- How can we learn to mimic the demonstrator in a general way?
  - Learn the demonstrator’s objective (reward) function
  - Apply RL

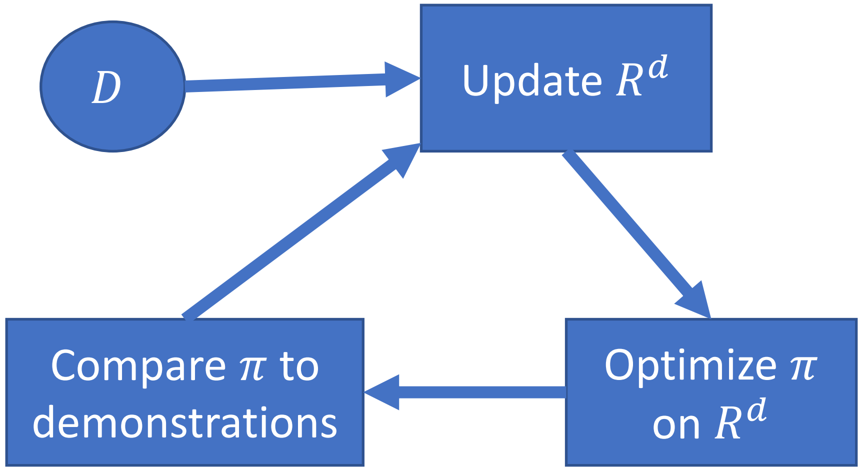
Inverse RL

- **Input**: $D = \{\tau_1, \ldots, \tau_m\} \sim \pi^d$
- **Learn**:

reward function R^d such that $\forall \pi$, $\mathbb{E}[\sum_t \gamma^t R^d(s_t) \,|\pi^d] \geq \mathbb{E}[\sum_t \gamma^t R^d(s_t) \,|\pi]$

  - Must assume that such an R^d exists (the demonstrator is optimal in some sense)

- **Output**: $\widehat{\pi^d}$ trained using RL with reward function R^d



Fitted reward is ambiguous

- With no knowledge of chess rules
- What reward function results in the white player policy?
- For most observations of behavior there are many fitting reward functions. The set of solutions often contains many degenerate solutions, e.g., assigning zero reward to all states

$\widehat{R^d}$ as a linear approximator

- Define $R(s) = w^\top f(s)$, where w is a set of weights and $f(s)$ is a vector of state features
- $\mathbb{E}[\sum_t \gamma^t R^d(s_t)\,|\pi] = \mathbb{E}[\sum_t \gamma^t w^\top f(s_t)\,|\pi] = w^\top \mathbb{E}[\sum_t \gamma^t f(s_t)\,|\pi] = w^\top \mu(\pi)$
- Where $\mu(\pi)$ is the expected cumulative discounted sum of feature values
- Now we can rewrite: $\mathbb{E}[\sum_t \gamma^t R^d(s_t)\,|\pi^d] \geq \mathbb{E}[\sum_t \gamma^t R^d(s_t)\,|\pi]\ \forall\pi$
  - As: ${w^d}^\top \mu(\pi^d) \geq {w^d}^\top \mu(\pi)\ \forall\pi$
- Solve w^d as an optimization problem

Solve w^d as a constraint optimization

Abbeel & Ng, ICML '04

$\text{Max}_w\ \delta$

In order to discourage ambiguity, maximize the difference in accumulated reward, δ, achieved by π^d compared to any other observed policies.

S.T.

$$w^\top \mu(\pi^d) \geq [w^\top \mu(\pi_j) + \delta] \quad \forall j = \{0, \ldots, i-1\}$$
$$\|w\|_2 \leq 1$$

When assuming that all state features are valued from the range [0,1], bounding $\|w\|_2 \leq 1$ ensures that the rewards are bounded by 1 (Important for theoretical guarantees).

Non-linear constraint

- Can't solve as a linear program, solve as a quadratic program
- For each iteration i:
  - sample trajectory $i-1$ from π_{i-1}, $\widehat{R^d}(s; w)$ = solve the quadratic program
  - π_i = train RL on new $\widehat{R^d}$
  - stop once $\delta \leq \epsilon$ for some predefined ϵ

Solve w^d as a constraint optimization

Abbeel & Ng, ICML '04

$\mathrm{Max}_w\ \delta$

S.T.

$$w^\top \mu(\pi^d) \geq \min_{j=\{0,\ldots,i-1\}} \left[w^\top \mu(\pi_j) + \delta\right]$$

$$\|w\|_2 \leq 1$$

- For each iteration i:
  - sample trajectory $i-1$ from π_{i-1}, $\widehat{R^d}(s;w)$ = solve the quadratic program, π_i = train RL on $\widehat{R^d}$, stop once $\delta \leq \epsilon$ for some predefined ϵ
- Guaranteed to terminate after $O\left(\frac{k}{(1-\gamma)^2\epsilon^2}\log\frac{k}{(1-\gamma)\epsilon}\right)$ iterations
  - Where k is the number of state features
  - Only for linear reward approximation

MaxEnt approach

- A policy π induces a distribution over trajectories $P(\tau|\pi)$
  - $\sum_i p(\tau_i|\pi) = 1$
- We require that π is set such that it follows the same expected trajectory as π^d
  - $\mathbb{E}\left[\mu\left(\pi^d\right)\right] = \sum_i p(t_i|\pi)\,\mu(\tau)$
- **Maximum entropy principle:** The probability distribution which best represents the current state of knowledge is the one with largest entropy [E.T. Jaynes, 1957]

MaxEnt approach [Ziebart et al., AAAI ’08]

- Find a policy π which results in a trajectory distribution that is similar in expectancy to that induced by π^d without over-committing

$$\max_{\pi} \; -\sum_{\tau} p(\tau|\pi) \log p(\tau|\pi)$$

s.t.

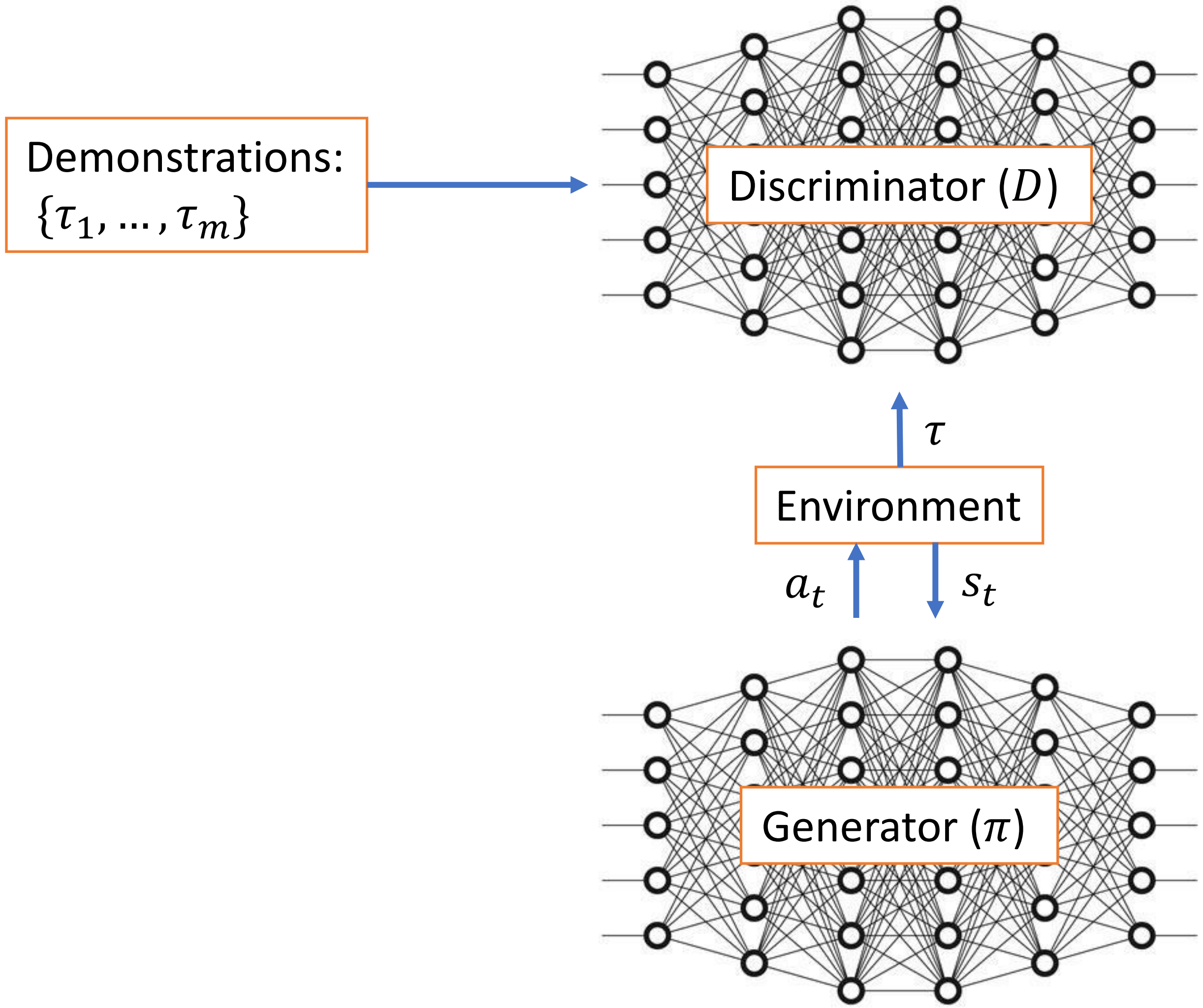
$$\sum_{\tau \in \pi} p(\tau|\pi)\mu(\tau) = \frac{1}{m} \sum_{\tau^d \in D} \mu(\tau^d)$$

Visited states distribution in demonstrations

- [Ziebart et al., AAAI ’08] considers linear approximation
- For generalization of MaxEnt to Deep IRL see: “Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization”, Finn et al., ICML ‘16

Generative Adversarial Imitation Learning

[Ho & Ermon, NeurIPS 2016]



Discriminator is trained to output 0 if input state-action is by the demonstrator else 1

Loss = log loss

The policy (generator) is trained using the policy gradient theorem:

$$\nabla j(\pi) = q(s,a)\nabla \log \pi(s,a)$$

In IL there is no reward function and consequently, no $q(s,a)$ value. Instead of attempting to maximize q attempt to fool the discriminator (maximize Loss of D for (s_t, a_t)).

$$\nabla j(\pi) = \sum_t \log D(s_t, a_t)\,\nabla \log \pi(s_t, a_t)$$

Generative Adversarial Imitation Learning

[Ho & Ermon, NeurIPS 2016]

Algorithm 1 Generative adversarial imitation learning

1: **Input:** Expert trajectories $\tau_E \sim \pi_E$, initial policy and discriminator parameters θ_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

$$\hat{\mathbb{E}}_{\tau_i}[\nabla_w \log(D_w(s,a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_w \log(1 - D_w(s,a))] \tag{17}$$

Training the discriminator

5: Take a policy step from θ_i to θ_{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

$$\hat{\mathbb{E}}_{\tau_i}\left[\nabla_\theta \log \pi_\theta(a|s) Q(s,a)\right] - \lambda \nabla_\theta H(\pi_\theta), \quad \text{where } Q(\bar{s},\bar{a}) = \hat{\mathbb{E}}_{\tau_i}[\log(D_{w_{i+1}}(s,a)) \mid s_0 = \bar{s}, a_0 = \bar{a}] \tag{18}$$

Training the policy

6: **end for**

Note: the Q value is **not** the action value but the discriminator's loss

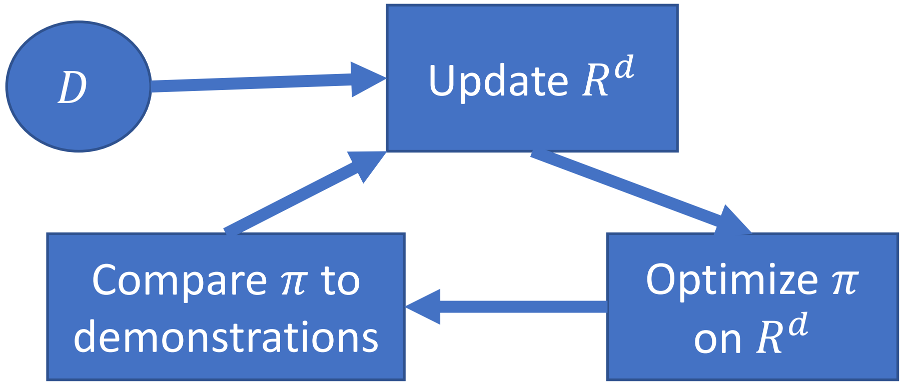
Note: H here is not for entropy. It is the Hessian matrix -- the original GAIL paper used TRPO (natural gradient).

Summary: Types of Imitation Learning

- Behavioral Cloning
  - $\pi = \arg\min_{\pi' \in \Pi} E_{s,a \sim D}\left[loss\left(a, \pi'(s)\right)\right]$
  - Works well when $P(\tau|\pi) \approx D$
- Direct Policy Learning
  - Use an interactive demonstrator to shift from $\arg\min_{\theta} E_{s,a \sim D}\left[loss\left(a, \pi_\theta(s)\right)\right]$ to $\arg\min_{\theta} E_{s \sim P(S|\pi_\theta)}\left[loss\left(\pi^d(s), \pi_\theta(s)\right)\right]$
    - Requires an online demonstrator

Summary: Types of Imitation Learning

- Behavioral Cloning
  - Works well when $P(\tau|\pi) \approx D$
- Direct Policy Learning
  - Requires an online demonstrator
- Inverse RL
  - Assumes learning $\widehat{R^d}$ is statistically easier than directly learning π^d
- MaxEnt
- Generative Adversarial
  - Learn to resemble the demonstrated patterns



Summary: Types of Imitation Learning

	Direct policy learning	Learn the reward function	Access to environment	Interactive demonstrator	Pre-collected demonstration
Behavioral cloning	✓	✗	✗	✗	✓
Interactive IL	✓	✗	✓	✓	Optional
Inverse RL	✗	✓	✓	✗	✓
Generative Adversarial	✗	✗	✓	✗	✓

Extra reading

- https://sites.google.com/view/icml2018-imitation-learning/
- https://arxiv.org/pdf/1011.0686.pdf
- https://ai.stanford.edu/~ang/papers/icml04-apprentice.pdf
- https://people.eecs.berkeley.edu/~pabbeel/cs287-fa12/slides/inverseRL.pdf
- https://arxiv.org/pdf/1606.03476.pdf
- https://arxiv.org/pdf/1710.11248.pdf

What next?

- **Lecture**: Transfer learning
- **Assignments:**
  - DDPG, by
  - A2C, by
- **Quiz (on Canvas)**:
  - Imitation Learning, by
  - Soft Actor-Critic, by
- **Project**:
  - Final Report, by