Imitation Learning (part I)

Human-Interactive Robot Learning (HIRL)
Silvia Tulli - Kim Baraka - Mohamed Chetouani

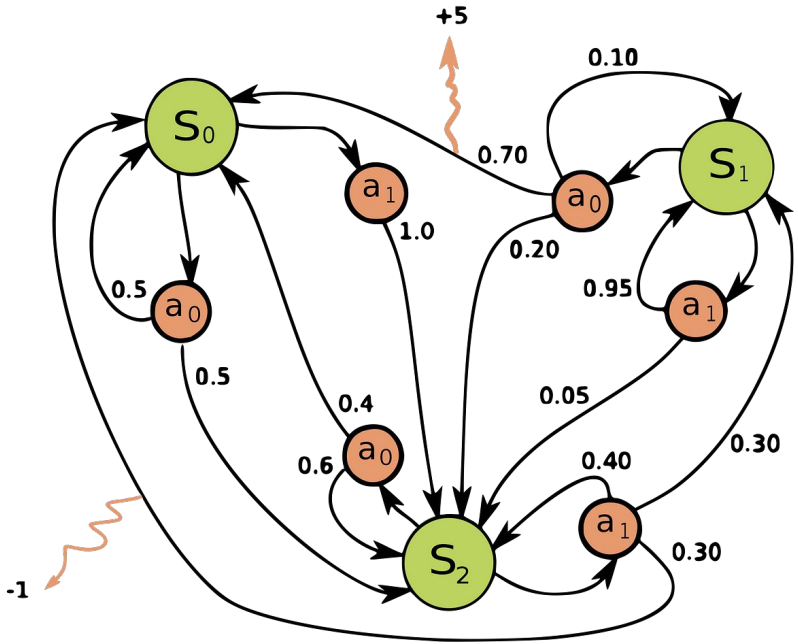
Learning goals

By the end of this lecture, you should be able to:

- Explain each of the key ingredients of imitation learning (demonstrations, environment, policy class, loss function, learning algorithm)
- Define different **imitation learning techniques**, their strengths and limitations:
    a. state visitation frequency, occupancy frequency
    b. distributional shift
- Explain the underlying differences between direct policy learning vs. Inverse Reinforcement Learning (IRL)
- Explain different reward function representations in IRL
- Define four main problems and solutions related to IRL:
    - reward function ambiguity (degeneracy)
    - expert suboptimality
    - computational intensiveness
- Master the following three learning algorithms:
    - Behavioral Cloning (BC)
    - DAgger
    - Apprenticeship Learning IRL

Prerequisites

- Familiarity with basic terminology of control problems such as policy, state, action, observation, time step, bayes net, markov property and common notation



Example of a **simple MDP** with three states (green circles) and two actions (orange circles), with two rewards (orange arrows)

Prerequisites

- the goal of reinforcement learning is to **determine closed-loop control policies** that result in the maximization of an accumulated reward
- the **reward function is known**, and both typically rely on collecting system data to either update a learned model (**model-based**), or directly update a learned value function or policy (**model-free**).

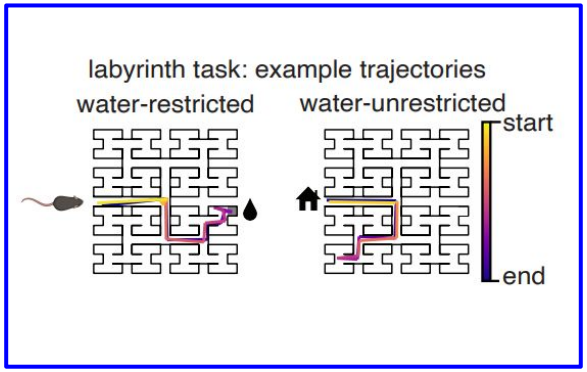
lilianweng.github.io

Children Learn Through Imitation

Can we reason about what the expert is trying to achieve?

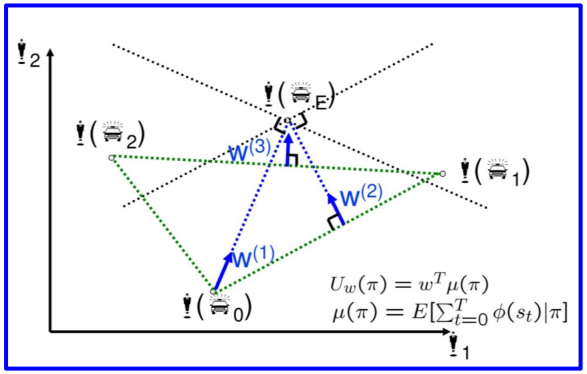
[Ilabs](), University of Washington

Motivations and Application domains

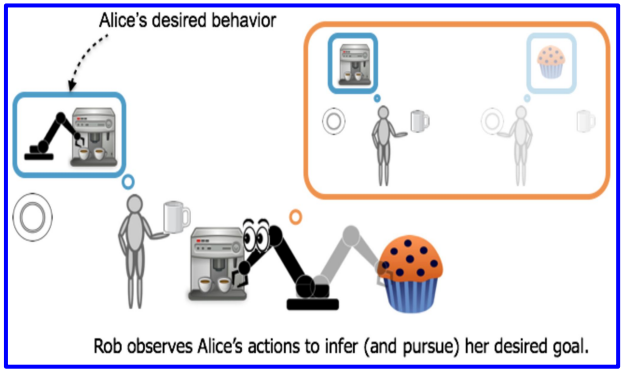


Scientific Inquiry:

- Model animal and human behavior [Ashwood et al. 2022]
- e.g., bee foraging, songbird vocalization, [Kálmán 1964], [Ng and Russell, 2000]



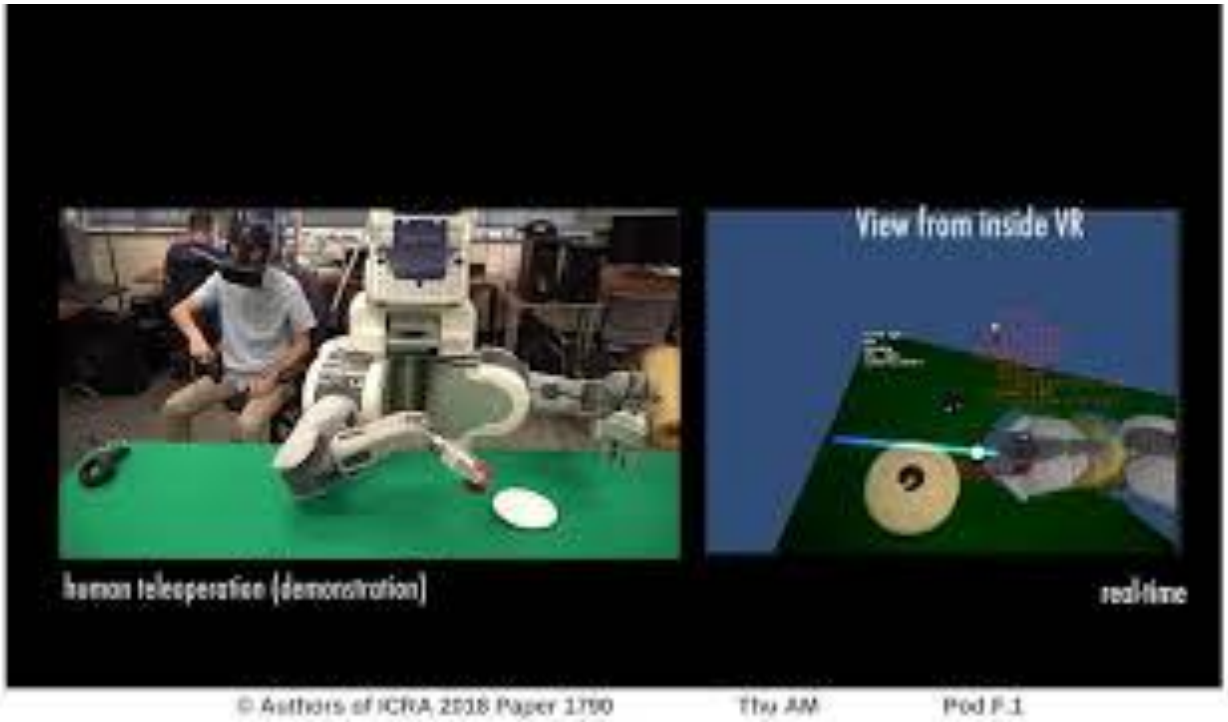
Advancing the state of the art in various robotic domains, [Abbeel et al. 2006]



Modeling of other agents, both adversarial and cooperative, [Hadfield-Menell et al. 2016]

Check out [Abbell, 2010]

Imitation Learning for Complex Manipulation Tasks

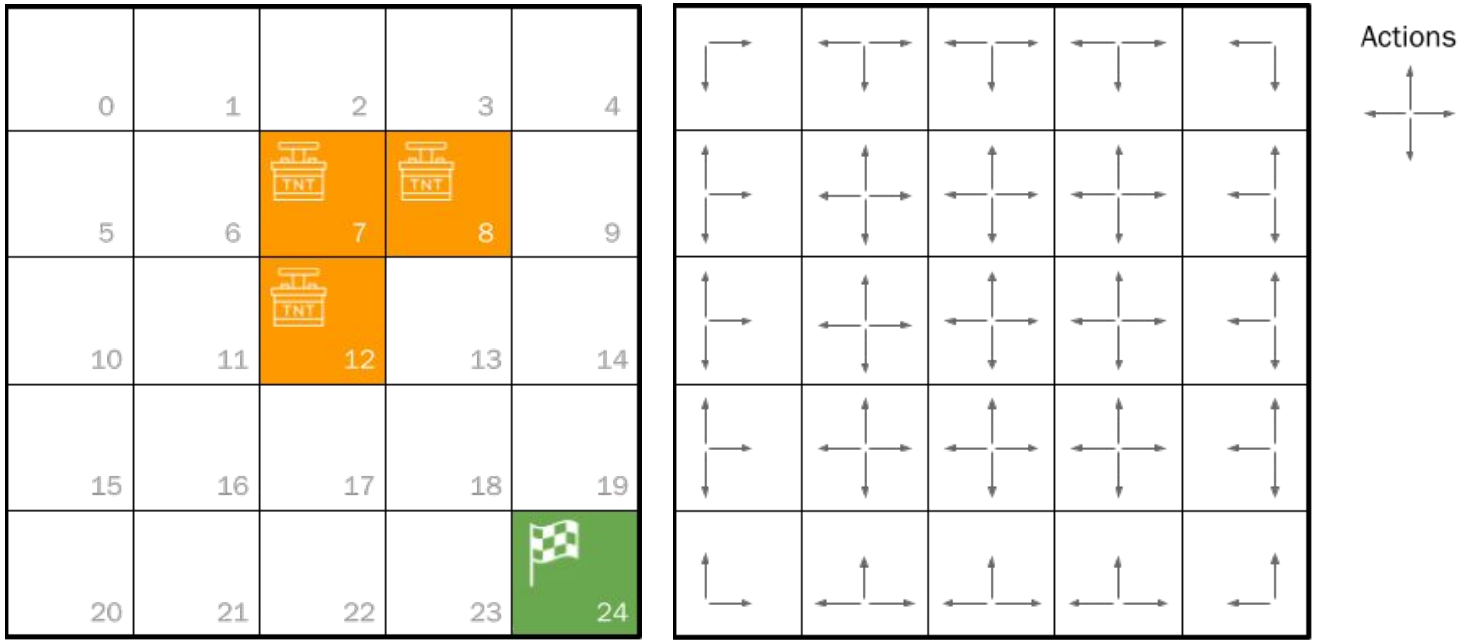


Zhang et al. 2018

Teaching a Robot to Grasp Real Fish by Imitation Learning

Dyrstad et al. 2018

Toy Example

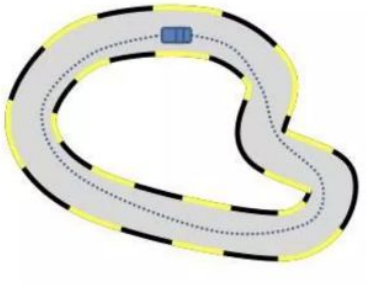


Imitation Learning Intuition

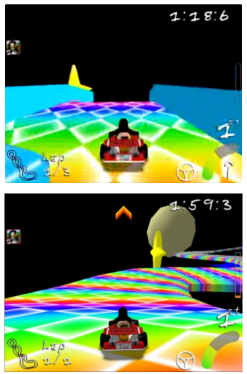
Given demonstrations or demonstrator

Goal train a policy to mimic demonstrations

Expert Demonstrations

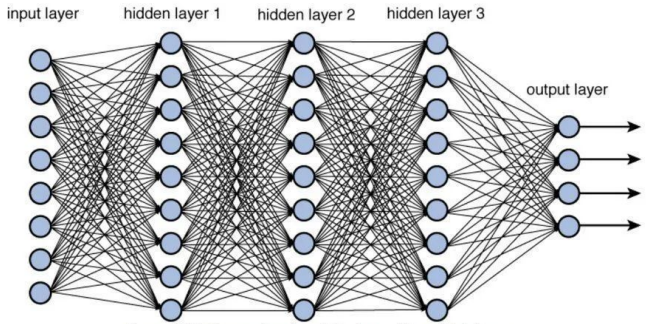


State/Action Pairs



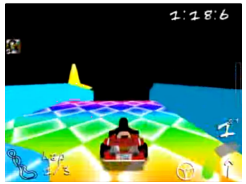
Steering Angle
[-1, 1]

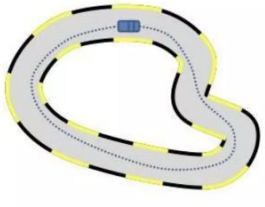
Supervised Learning



Imitation Learning Tutorial ICML 2018, Images Ross (2010)

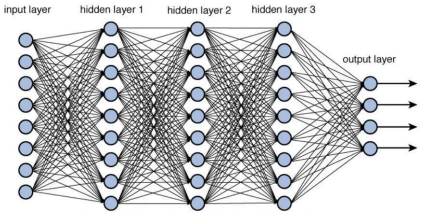
Imitation Learning Key Ingredients



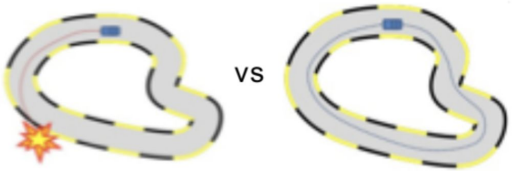


Demonstrator/Demonstrations

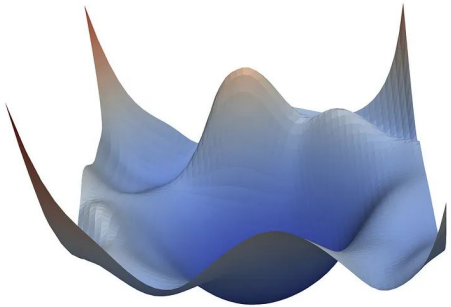
Environment/Simulator



Policy Class

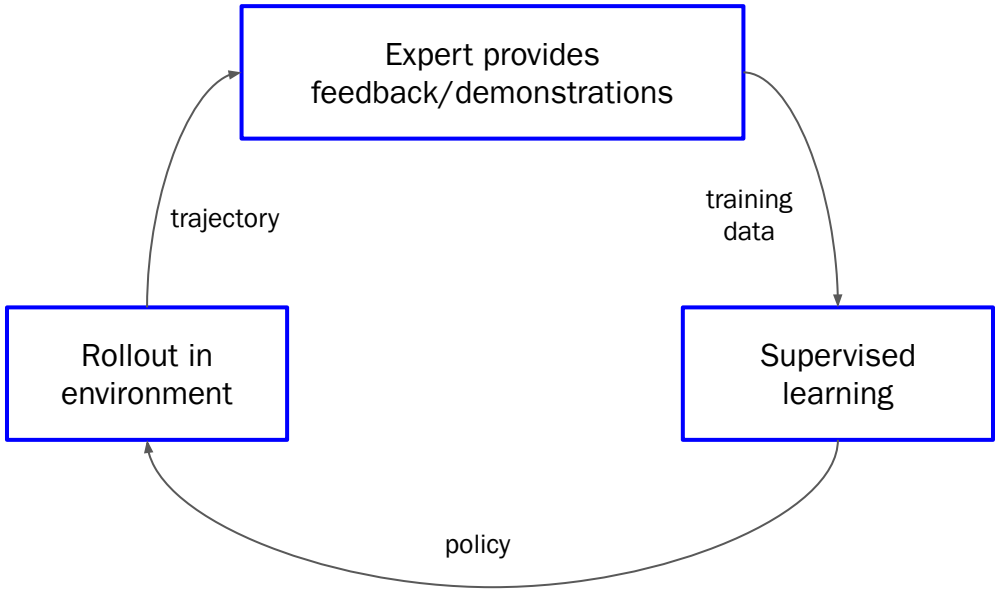


Loss Function



Learning Algorithm

Imitation Learning Loop



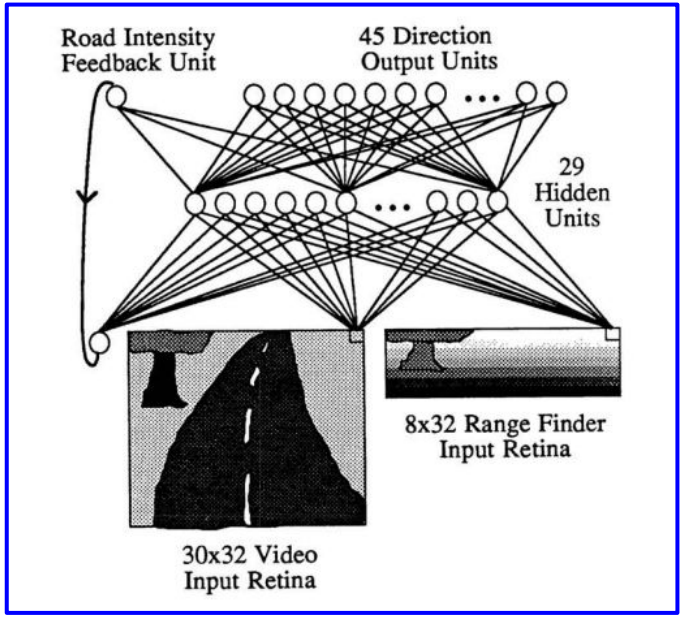
Algorithm 1 - Behavioral Cloning

Behavioral Cloning

IL formulated as **standard machine learning problem**

- Fix a policy class
  - e.g., support vector machine, neural network, decision tree, deep belief net, …
- Estimate a policy (= mapping from states to actions) from the training examples

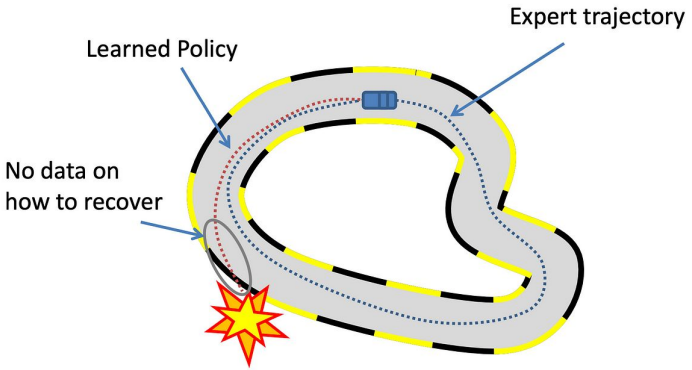
$$\mathcal{D} = \{(s_0, a_0), (s_1, a_1), (s_2, a_2), \cdots, (s_n, a_n)\}$$



Pomerleau, D.A. (1988). ALVINN, an autonomous land vehicle in a neural network.

Check out Abbell, 2010

Distributional Shift

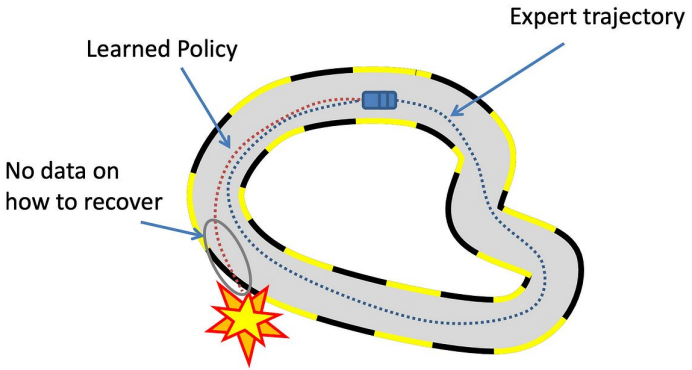


Common assumption is that train and test set are independent and identically distributed (i.i.d.)

However, $p_{\pi^*}(o_t) \neq p_{\pi_\theta}(o_t)$

Check out Abbell, 2010 and CS287-FA19 Advanced Robotics at UC Berkeley, 2019

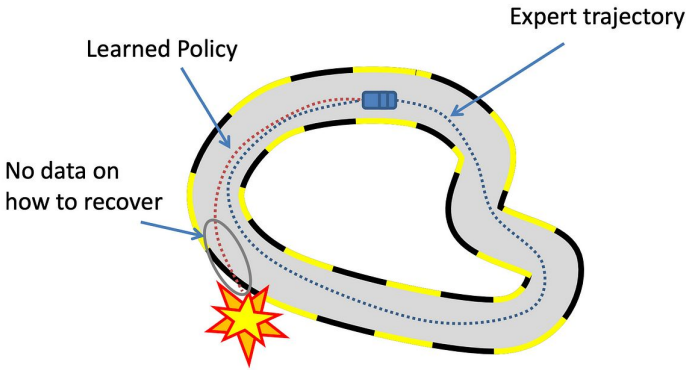
Distributional Shift - Why It Matters



- At each step, BC policy has error rate ε (e.g., 5% mistakes)
- After T steps, expected errors: $O(\varepsilon T^2)$
- Example: 1% error rate over 100 steps ≈ 63% chance of failure

Check out Abbell, 2010 and CS287-FA19 Advanced Robotics at UC Berkeley, 2019

Distributional Shift - Why It Matters

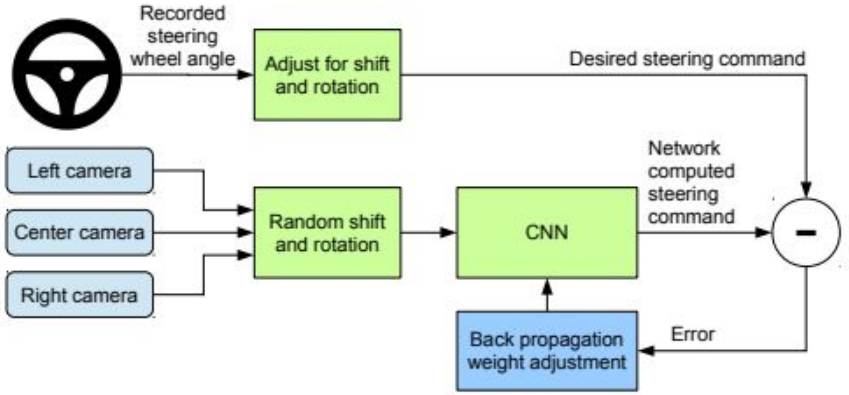


Why i.i.d. breaks:

- Supervised learning:
  test data ~ training data
- Sequential decisions:
  $p(s_t|\pi) \neq p(s_t|\pi^*)$ after errors

Check out Abbell, 2010 and CS287-FA19 Advanced Robotics at UC Berkeley, 2019

DAVE-2 System



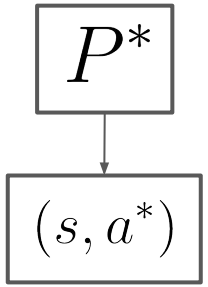
Bojarski et al. 2016

(General) Imitation Learning vs Behavioral Cloning

- Behavioral Cloning (supervised learning):

$$\arg\min_{\theta} \mathbb{E}_{(s,a^*)\sim P^*} L\left(a^*, \pi_\theta(s)\right)$$

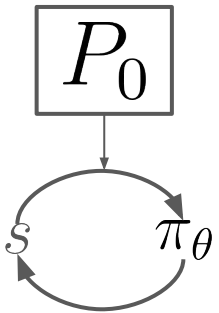
Distribution provided exogenously



- (General) Imitation Learning:

$$\arg\min_{\theta} \mathbb{E}_{s\sim P(s|\theta)} L\left(\pi^*(s), \pi_\theta(s)\right)$$

Distribution depends on the rollout
$P(s|\theta)$ = state distribution of $\pi\theta$



Principles of Robot Autonomy II, Stanford University, Iliad

Types of Imitation Learning

	Direct Policy Learning	Reward Learning	Access to Environment	Interactive Demonstrator	Pre-collected Demonstrations
Behavioral Cloning	Yes	No	No	No	Yes
Direct Policy Learning (interactive IL)	Yes	No	Yes	Yes	Optional
Inverse Reinforcement Learning	No	Yes	Yes	No	Yes

Principles of Robot Autonomy II, Stanford University, Iliad

Algorithm 2 - DAgger

DAgger

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.

DAgger Example

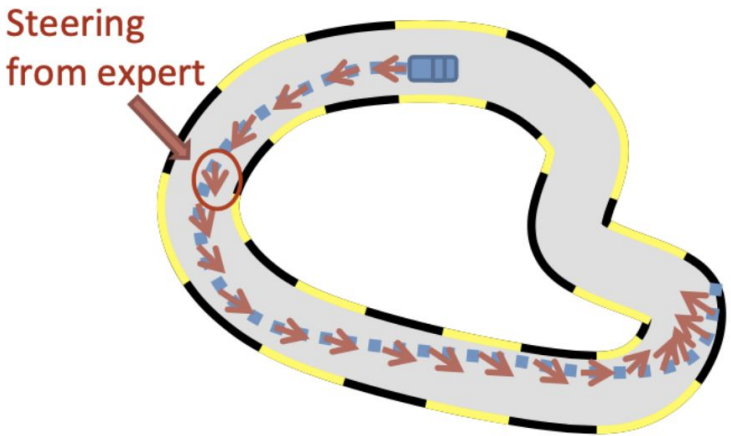
Ross et al. 2018, and RAIL, 2021

Interactive Expert

Can query expert at any state
Construct loss function: $L\left(\pi^*(s), \pi(s)\right)$

Typically applied to rollout trajectories of policies we are training: $s \sim P(s \mid \pi)$

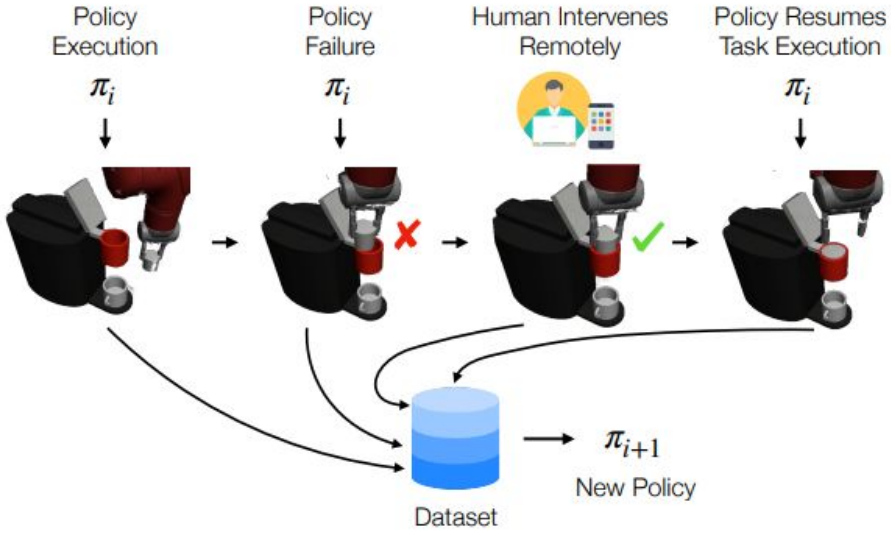
Driving example: $L(\pi^*(s), \pi(s)) = \left(\pi^*(s) - \pi(s)\right)^2$

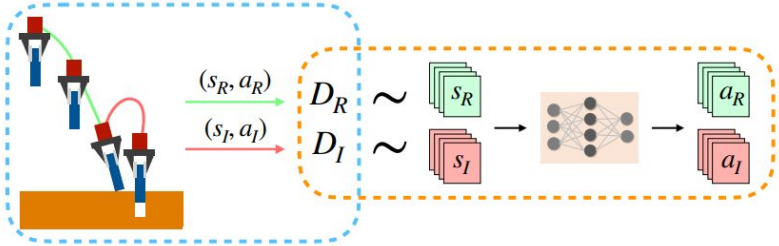


BC vs Interactive IL: Concrete Example

Method	Training Data	What Happens
Behavioral Cloning	Only center-of-lane images → "steer straight"	Slight drift → edge of lane → NO DATA for recovery → crash
DAgger	Center images + near-edge images (from rollout) → expert corrections	Slight drift → have data for "steer back to center" → recovers

Human-in-the-Loop Imitation Learning using Remote Teleoperation





Intervention Weighted Regression

[Mandlekar et al. 2020]

When to Use Which Method

Criterion	Behavioral Cloning	DAgger	Apprenticeship IRL
Expert availability	Offline dataset only	Must be interactive	Offline dataset
Environment access	Not needed	Required	Required
Best for	Large datasets, deterministic tasks	Critical applications	Transfer, understanding intent
Computational cost	Low (one-time supervised learning)	Medium (iterative)	High (RL in inner loop)
Sample efficiency	Poor (needs many demos)	Good (targeted queries)	Moderate
Handles suboptimality	No	No	Yes (with modifications)
Transfer to new tasks	Poor	Poor	Good

When to Use Which Method

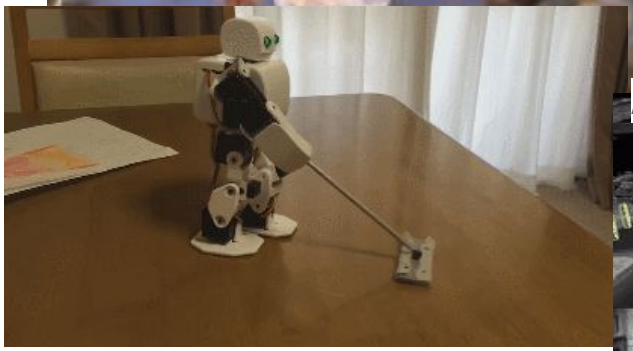
- BC: Large expert dataset, deterministic environments, offline learning
- DAgger: Access to expert during training, expensive but effective
- IRL: Transfer to new environments, expert suboptimality, understanding intent matters

Algorithm 3 - Apprenticeship Learning via IRL

Where Does the Reward Function Come From?

Computer games

Robotics



Dialog

Autonomous driving

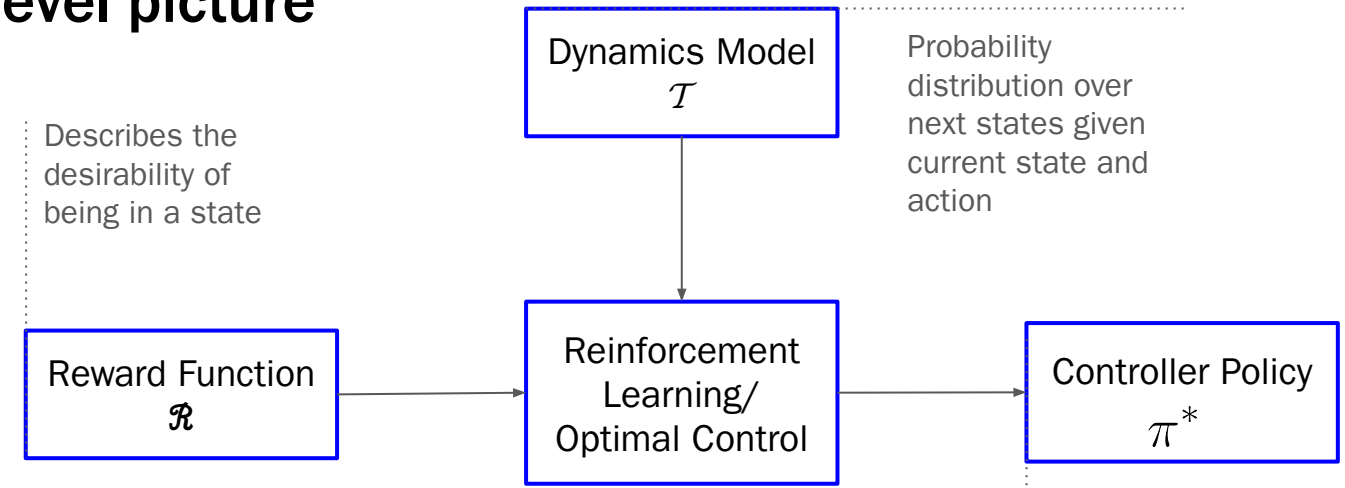
[Mnih et al. 2015](), [Chelsea Finn Deep RL Bootcamp]()

Motivations and Application domains

Aerobatic Helicopter Flight,
Abbeel et al. (2006)

End-to-End Autonomous Driving,
Mero et al. 2022

High-level picture

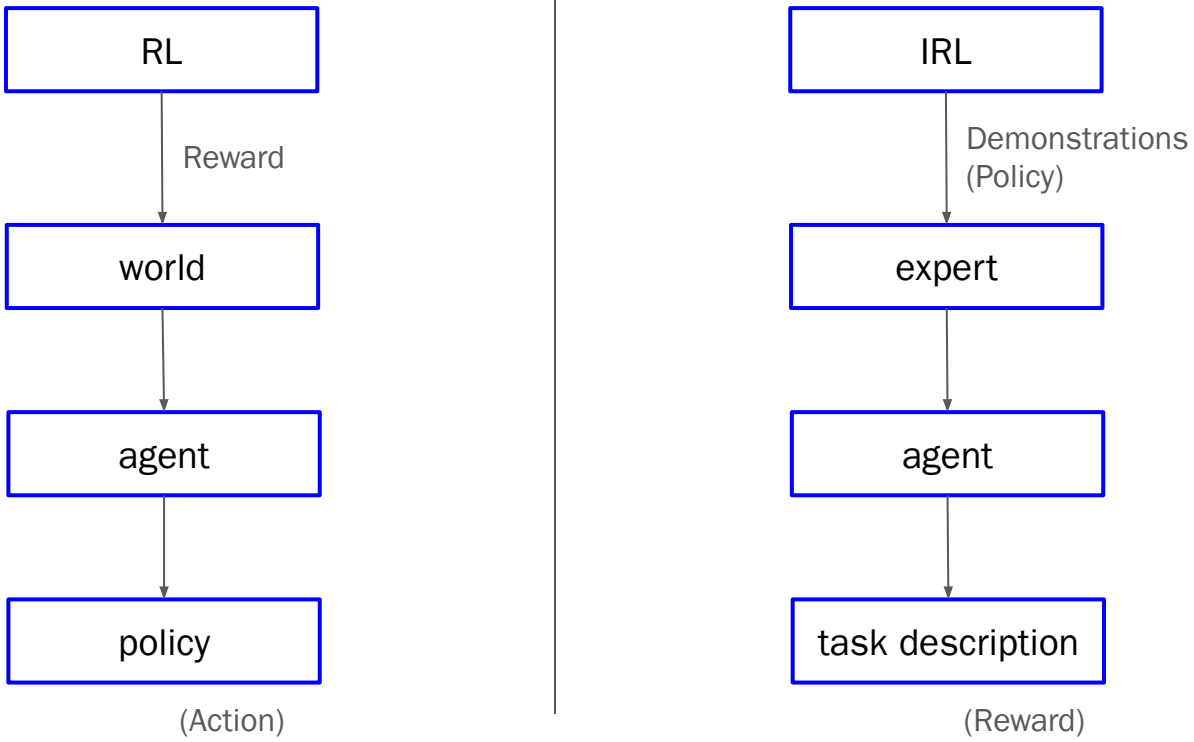


Inverse RL:

- Given π^* and $\mathcal{T}$, can we recover $\mathcal{R}$?
  More generally, given execution traces, can we recover $\mathcal{R}$?

Check out Abbell, 2010

RL vs IRL



Check out Melo, 2009

Problem setup

Input:
State space $\mathcal{S}$ action space $\mathcal{A}$

(sometimes partially observable)

Transition model $P_{sa}\left(s_{t+1} \mid s_t, a_t\right)$

MDP with no reward function

Teacher's demonstration:

$$\mathcal{D} = \{(s_0, a_0), (s_1, a_1), (s_2, a_2), \cdots, (s_n, a_n)\}$$

trace of the teacher's policy π^*
that maps states to distributions
over actions $\pi^*(s) \rightarrow P(s)$

- **Behavioral Cloning or Direct Imitation:**

  Can we directly learn the teacher's policy using supervised learning?

- **Inverse RL:**

  Can we recover R ?

- **Apprenticeship learning via inverse RL:**

  Can we then use this R to find a good policy?

Check out Abbell, 2010

Problem setup

Rollout: Sequentially execute $\pi(s0)$ on an initial state, produce trajectory: $\tau = s0, a0, s1, a1, \ldots$.

$P(\tau|\pi)$: Distribution of trajectories induced by a policy:

1. Sample $s0$ from $P0$ (distribution over initial states)
2. Initialize $t = 1$. Sample action at from $\pi(st-1)$.
3. Sample next state st from applying at to $st-1$ (requires access to environment).
4. Repeat form step 2 with $t = t + 1$.

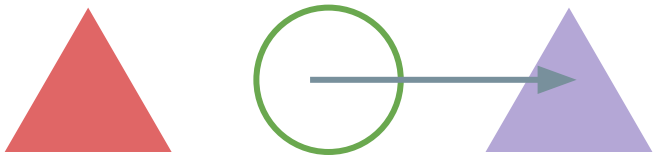
$P(s|\pi)$: Distribution of states induced by a policy:

1. Let $Pt(s|\pi)$ denote distribution over t-th state.
2. $P(s \mid \pi) = \frac{1}{T} \sum_{t} P_t(s \mid \pi)$

Principles of Robot Autonomy II, Stanford University, Iliad

Challenges

- IRL is a undefined problem
- It is difficult to evaluate a learned reward
- Demonstrations might not be optimal



Check out Chelsea Finn, 2017

Behavioral Cloning vs Inverse Control

Behavioral Cloning has no notion of *intention:*

- Expert suboptimality
- Embodiment (correspondence problem)
- Robustness

Which has the most succinct description of a task

$$\pi^* \text{ vs. } R \text{ ?}$$

Especially in planning oriented tasks, the reward function is often much more succinct than the optimal policy.

Feature based reward function

Let $R(s) = w^\top \phi(s)$, where $w \in \Re^n$ and $\phi : S \to \Re^n$

$$\begin{aligned} \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t R\left(s_t\right) \mid \pi\right] &= \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t w^\top \phi\left(s_t\right) \mid \pi\right] \\ &= w^\top \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t \phi\left(s_t\right) \mid \pi\right] \\ &= w^\top \mu(\pi) \end{aligned}$$

Snipping into: $\mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t R^*\left(s_t\right) \mid \pi^*\right] \geq \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t R^*\left(s_t\right) \mid \pi\right] \quad \forall \pi$

Gives us:

Find w^* such that $w^{*\top} \mu\left(\pi^*\right) \geq w^{*\top} \mu(\pi) \quad \forall \pi$

Expected cumulative discounted sum of feature values or "feature expectations"

Check out Abbell, 2010

Feature based reward function

Grid Navigation Example:

φ(s) =
[distance_to_goal(s),
// Lower is better near_obstacle(s),
// Avoid velocity(s),
// Efficiency smoothness(s)
// Comfort]

Feature vector (observable quantities)

w = [-1.0, -5.0, +0.5, +0.2] // Learned weights

Weight vector (what matters?)

Feature based reward function

Why linear?
Tractable: Convex optimization
Interpretable: See what expert values
Transferable: Same features, new weights for new tasks
Limited: Can't capture complex preferences

Alternative representations:

- Deep networks: R(s) = NN(s)
- State-action: R(s,a) = w^T φ(s,a)
- Trajectory: R(τ) = f(s_0,…,s_T)

Feature based reward function

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

Let $R(s) = w^\top \phi(s)$, where $w \in \Re^n$ and $\phi : S \rightarrow \Re^n$

Find w^* such that $w^{*\top}\mu\left(\pi^*\right) \geq w^{*\top}\mu(\pi) \quad \forall \pi$

For a policy to be guaranteed to perform as well as the expert policy, it suffices that the feature expectations “match”

Check out Abbell, 2010

Apprenticeship Learning via IRL

Let $R(s) = w^\top \phi(s)$, where $w \in \Re^n$ and $\phi : S \to \Re^n$

- Initialize some policy π_0
- Iterate for *i = 1,2,3, …*
  - Guess the reward: find a reward function such that the demonstrator policy maximally outperforms all previously found policy
  - Find an optimal control policy π_i for the current reward function
    - If the expert is suboptimal, pick the best policy in a mixture
  - Exit if $\gamma \geq \epsilon/2$

Degeneracy:
we have a lot of reward functions that could possibly be optimal, how do we pick one?

Check out Abbell & Ng, 2004

Apprenticeship Learning via IRL

Grid Example:
Expert always goes: S → → ↓ → G

All these rewards make this optimal:

1. $R_1(s)$ = 0 everywhere (degenerate!)
2. $R_2(s)$ = +10 at goal, 0 elsewhere
3. $R_3(s)$ = -distance to goal
4. $R_4(s)$ = +100 at goal, -1 per step

Why this matters: in supervised learning - one correct f(x)=y, in IRL - infinitely many correct R(s)

Degeneracy:
we have a lot of reward functions that could possibly be optimal, how do we pick one?

Check out Abbell & Ng, 2004

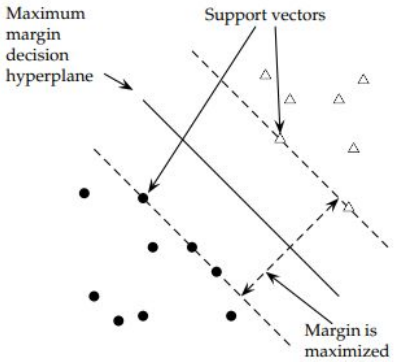
Apprenticeship Learning via IRL: Solutions to Degeneracy

- **Max Margin**: Find R that makes expert better than alternatives by maximum margin
- **Maximum Entropy**: Find R that makes expert have maximum entropy distribution
- **Bayesian**: Posterior distribution over R given demos
- **Pragmatic**: Just match feature expectations

Degeneracy:
we have a lot of reward functions that could possibly be optimal, how do we pick one?

Check out Abbell & Ng, 2004

Resolving ambiguity in the reward function



"Structured prediction" Max Margin

$$\begin{aligned} \min_w \quad & \|w\|_2^2 \\ \text{s.t.} \quad & w^\top \mu(\pi^*) \geq w^\top \mu(\pi) + m(\pi^*, \pi) \quad \forall \pi \end{aligned}$$

number of states in which π^* was observed and in which π and π^* disagree

Challenges:

- Assumes we know the entire expert policy, assumes we can estimate expert feature expectations
- $R = 0$ is a solution (now:), more generally: **reward function ambiguity**
- Assumes the expert is indeed optimal — became even more of an issue with the more limited reward function expressiveness!
- Computationally: assumes we can enumerate all policies

Check out Abbell, 2010

Resolving expert suboptimality

Challenges:

- Assumes we know the entire expert policy assumes we can estimate expert feature expectations
- $R = 0$ is a solution (now: $w = 0$), more generally: reward function ambiguity
- Assumes the **expert is indeed optimal** — became even more of an issue with the more limited reward function expressiveness!
- Computationally: assumes we can enumerate all policies

"Structured prediction" Max Margin with slack variables

$$\begin{aligned} \min_{w,\xi^{(i)}} \quad & \|w\|_2^2 + C \sum_i \xi^{(i)} \\ \text{s.t.} \quad & w^\top \mu\left(\pi^{(i)*}\right) \geq w^\top \mu\left(\pi^{(i)}\right) + m\left(\pi^{(i)*}, \pi^{(i)}\right) - \xi^{(i)} \quad \forall i, \pi^{(i)} \end{aligned}$$

Can be generalised to multiple MDPs

Check out Abbeel, 2010

Resolving computational intensiveness

Constraint generation

$$\max_{\pi^{(i)}} w^\top \mu\left(\pi^{(i)}\right) + m\left(\pi^{(i)*}, \pi^{(i)}\right)$$

Find the optimal policy for the current estimate of the reward function (+ loss augmentation *m*)

Challenges:

- Assumes we know the entire expert policy, assumes we can estimate expert feature expectations
- $R = 0$ is a solution (now: $w = 0$), more generally: reward function ambiguity
- Assumes the expert is indeed optimal — became even more of an issue with the more limited reward function expressiveness!
- Computationally: assumes we can **enumerate all policies**

Imitation Learning (part II)

Human-Interactive Robot Learning (HIRL)

Silvia Tulli - Kim Baraka - Mohamed Chetouani

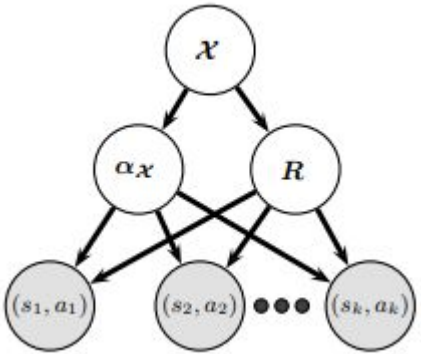
Learning goals

By the end of this lecture, you should be able to:

- Recap of last course
- Describe different probabilistic Inverse Reinforcement Learning approaches
- Master three main algorithms:
  - Bayesian Inverse Reinforcement Learning
  - Guided Cost Learning
  - Generative Adversarial Imitation Learning (GAIL)

(Maximum Entropy Inverse Reinforcement Learning)

Bayesian Inverse Reinforcement Learning



Combine prior knowledge and evidence from the expert's actions to derive a **probability distribution over the space of reward functions.**

- Learn the rewards by itself (as in preference elicitation)
- Learn the policy through apprenticeship learning (learning policies from an expert)

They derive a posterior distribution for the rewards from a prior distribution and a probabilistic model of the expert's actions given the reward function.

(Ramachandran & Amir, 2007)

Challenges of IRL

Many different costs can explain a given set of demonstrations:

- **Solution 1.** Maximum margin formulations (Abbeel & Ng, 2004, Ratliff et al., 2006)
- **Solution 2.** Probabilistic models that explain suboptimal behavior as noise (Ramachandran & Amir, 2007, Ziebart et al., 2008).

Limitations of these solutions:

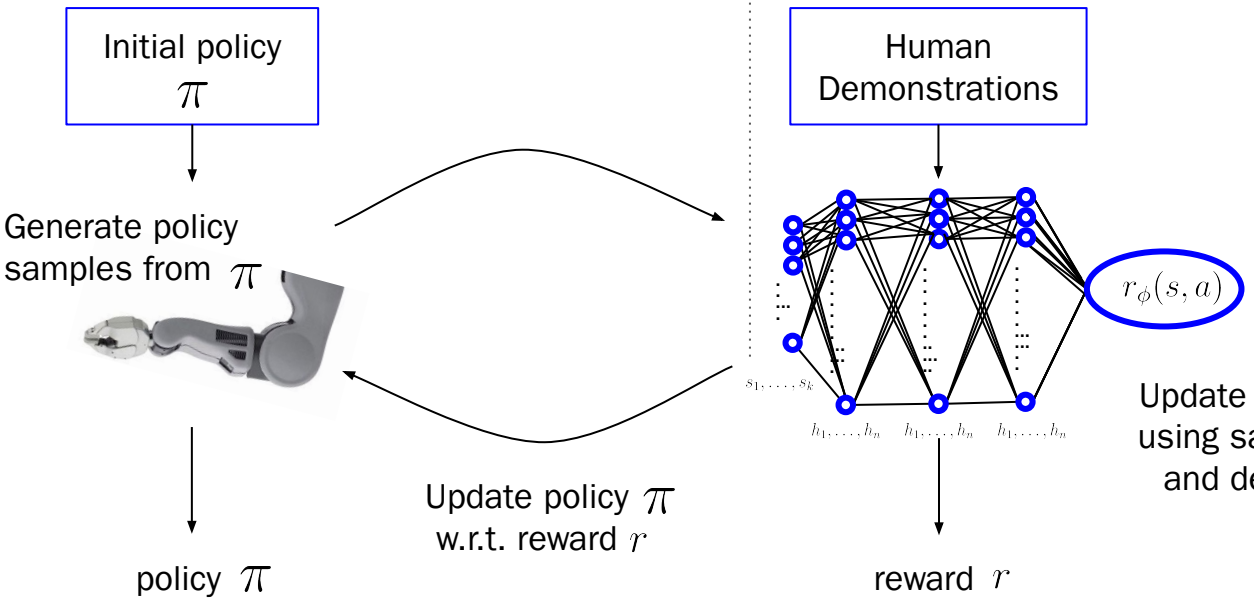
- Detailed features created using domain knowledge (Abbeel & Ng, 2004, Boularias et al. 2011, Muelling et al. 2014, Doerr et al. 2015) require engineering burden.

Guided Cost Learning

Learns **nonlinear cost functions from demos**, at the same time as **learning a policy** to perform the task

=

Policy optimization "guides" the cost toward good regions of the space



raw state representations rather than features

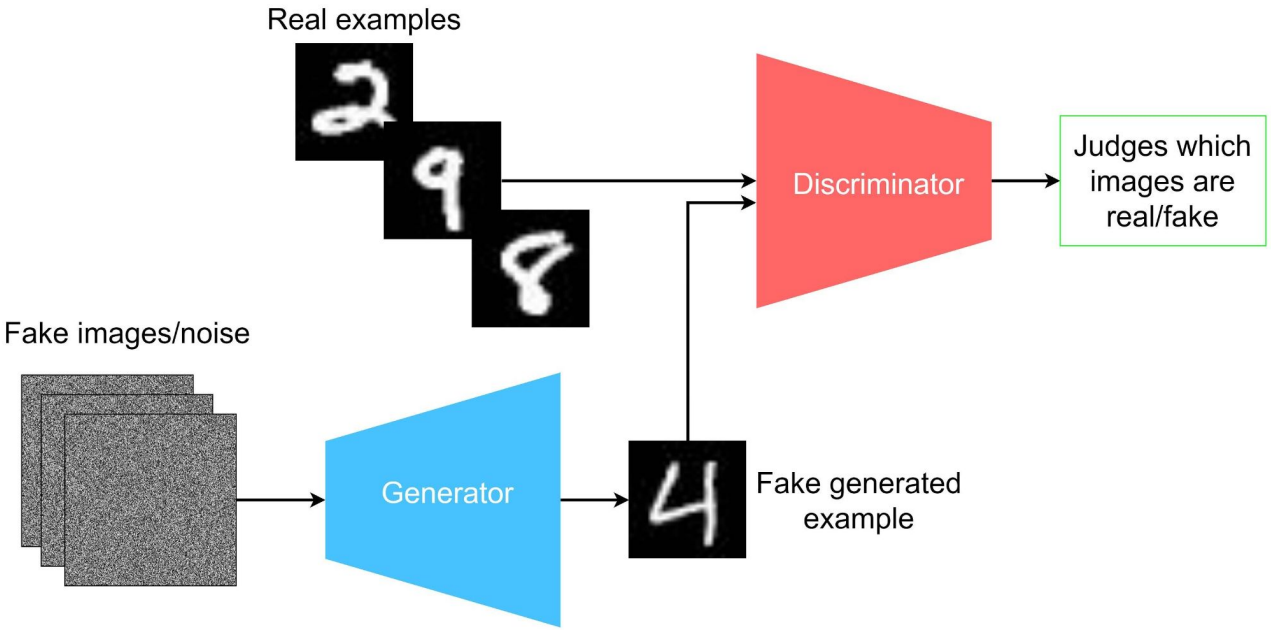
Check out Finn et al. 2016

IRL algorithms are extremely expensive

- **Problem 1.** IOC is fundamentally underdefined in that many costs induce the same behavior
    - Difficulty of learning the cost function under unknown dynamics for high-dimensional continuous systems
- **Problem 2.** Learned cost function lack the structure of hand-designed features.
    - Need for informative features and effective regularization to impose structure on the cost

- **Solution 1.** Use expressive, non linear function approximators, such as neural networks to represent the cost function.
- **Solution 2.** Use regularization techniques able to handle unknown dynamics and high-dimensional systems.

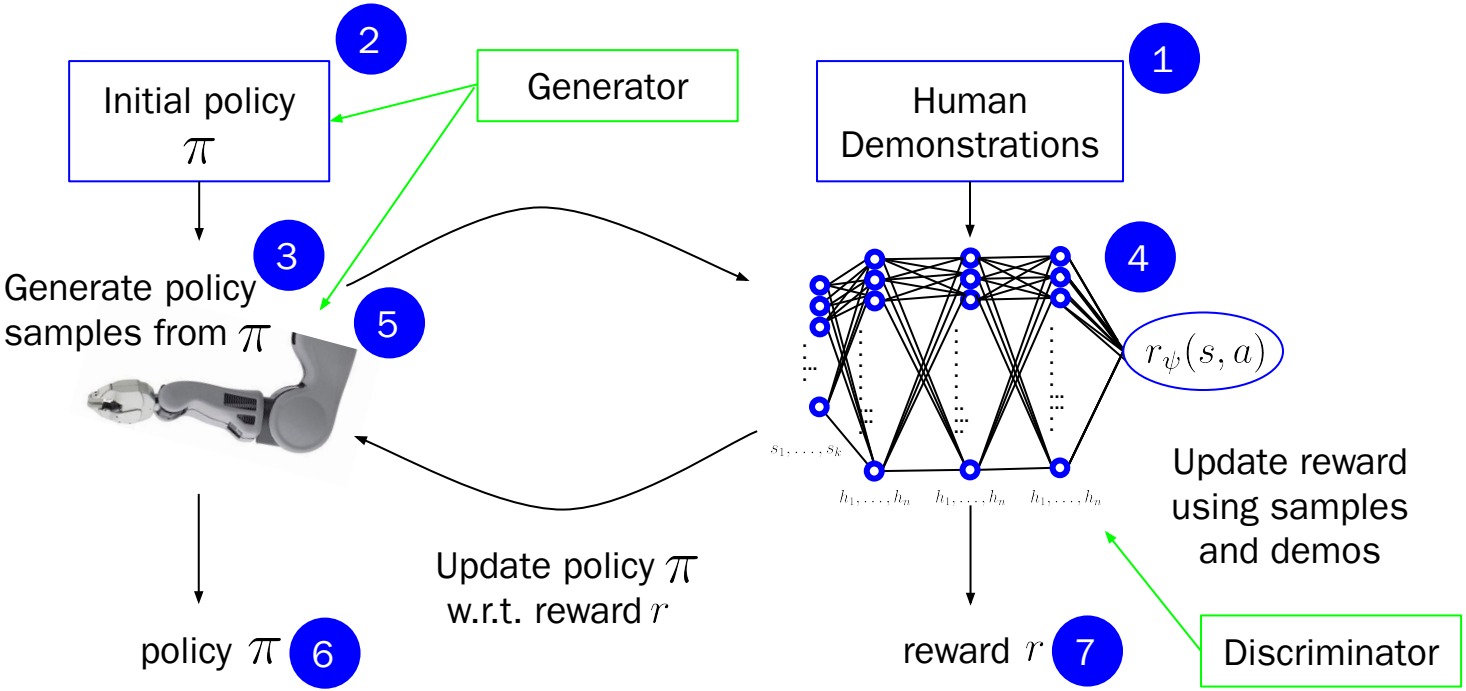
Generative Adversarial Networks (GANs)



Generative Adversarial Networks (GANs), IBM

Check out Poupart, 2020, imitation.readthedocs.io, wikidocs.net/213996

Generative Adversarial Imitation Learning (GAIL)



Check out Ho & Ermon 2016

Maximum Entropy Inverse Reinforcement Learning

Notation

$\tau = \{s_1, a_1, \ldots, s_t, a_t, \ldots, s_T\}$

trajectory

$R_\psi(\tau) = \sum_t r_\psi(s_t, a_t)$

learned reward

$\mathcal{D} : \{\tau_i\} \sim \pi^*$

expert demonstrations

Maximum Entropy Formulation

$$p(\tau) = \frac{1}{Z} \exp\left(R_\psi(\tau)\right)$$

$$Z = \int \exp\left(R_\psi(\tau)\right) d\tau$$

$$\max_\psi \sum_{\tau \in \mathcal{D}} \log p_{r_\psi}(\tau)$$

Check out Finn et al. 2016, Ziebart et al. 2008

Maximum Entropy IRL Optimization

$$\begin{aligned}\max_{\psi} \mathcal{L}(\psi) &= \sum_{\tau \in \mathcal{D}} \log p_{r_\psi}(\tau) \\ &= \sum_{\tau \in \mathcal{D}} \log \frac{1}{Z} \exp\left(R_\psi(\tau)\right) \\ &= \sum_{\tau \in \mathcal{D}} R_\psi(\tau) - M \log Z \\ &= \sum_{\tau \in \mathcal{D}} R_\psi(\tau) - M \log \sum_{\tau} \exp\left(R_\psi(\tau)\right)\end{aligned}$$

$$\nabla_\psi \mathcal{L}(\psi) = \sum_{\tau \in \mathcal{D}} \frac{dR_\psi(\tau)}{d\psi} - M \frac{1}{\sum_\tau \exp\left(R_\psi(\tau)\right)} \sum_{\tau} \exp\left(R_\psi(\tau)\right) \frac{dR_\psi(\tau)}{d\psi}$$

Policy Optimization under Unknown Dynamics

The policy optimization procedure aim to optimize the Gaussian trajectory distributions

$$q(\tau) = q(\mathbf{x}_1) \prod_t q(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t)\, q(\mathbf{u}_t \mid \mathbf{x}_t)$$

With respect to the expected cost $E_{q(\tau)}[c_\theta(\tau)]$

Solution 1. Iteratively optimize $E_{q(\tau)}[c_\theta(\tau)]$
With respect to the linear-Gaussian conditionals $q(\mathbf{u}_t \mid \mathbf{x}_t)$
under a linear-Gaussian estimate of the dynamics $q(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t)$
using linear-quadratic regulator.

Solution 2. Iteratively generating samples from $q(\mathbf{u}_t \mid \mathbf{x}_t)$
fitting $q(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t)$ to these samples, and
updating $q(\mathbf{u}_t \mid \mathbf{x}_t)$ under these fitted dynamics
by using linear-quadratic regulator (LQR).

Check out Levine & Abbeel, 2014


HUMANE AI NET


SORBONNE
UNIVERSITÉ


VU
UNIVERSITY
AMSTERDAM