BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Deep Reinforcement Learning

2022-23 Second Semester, M.Tech (AIML)

Session #15:
Imitation Learning

Instructors :
1. Prof. S. P. Vimal (vimalsp@wilp.bits-pilani.ac.in),
2. Prof. Sangeetha Viswanathan (sangeetha.viswanathan@pilani.bits-pilani.ac.in)

Agenda for the classes

- Imitation Learning
  - Behaviour Cloning
  - Inverse RL

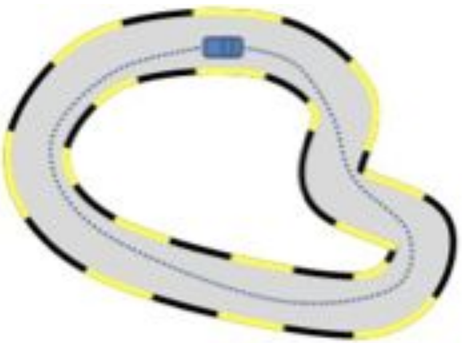
Acknowledgements: Some of the slides were adopted *with permission* from the course CSCE-689 (Texas A&M University) by Prof. Guni Sharon

Imitation Learning in a Nutshell

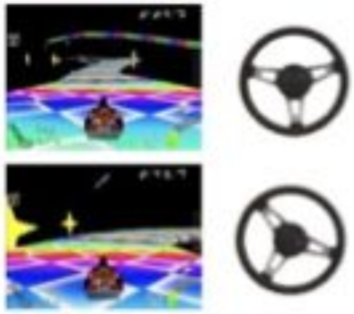
Given: demonstrations or demonstrator

Goal: train a policy to mimic demonstrations

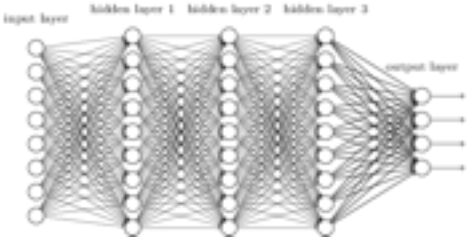
Expert Demonstrations



State/Action Pairs

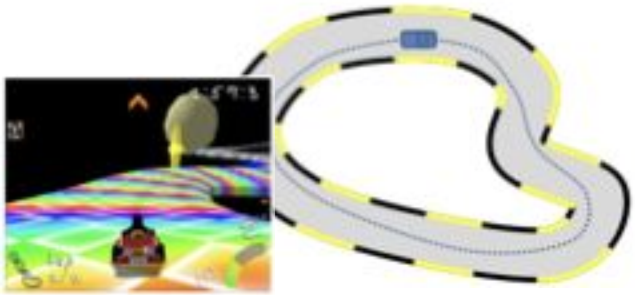


Learning



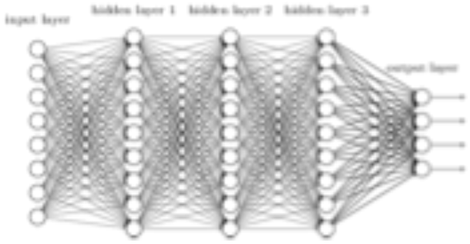
Images from Stephane Ross

Ingredients of Imitation Learning



Demonstrations or Demonstrator

Environment / Simulator



Policy Class

vs

Loss Function

Learning Algorithm

Some Interesting Examples

- ALVINN
  https://www.ri.cmu.edu/publications/alvinn-an-auton`omous-land-vehicle-in-a-neural-network/
  Dean Pomerleau et al., 1989-1999 https://www.youtube.com/watch?v=ilP4aPDTBPE
- Helicopter Acrobatics

Learning for Control from Multiple Demonstrations - Adam Coates, Pieter Abbeel, Andrew Ng, ICML 2008

An Application of Reinforcement Learning to Aerobatic Helicopter Flight - Pieter Abbeel, Adam Coates, Morgan Quigley, Andrew Y. Ng, NIPS 2006

https://www.youtube.com/watch?v=0JL04JJjocc

- Ghosting (Sports Analytics) - Next Slide.

Some Interesting Examples

Some Interesting Examples

What's Hidden in the Hidden Layers?

The contents can be easy to find with a geometrical problem, but the hidden layers have yet to give up all their secrets

David S. Touretzky and Dean A. Pomerleau

tions, we fed the network road images taken under a wide variety of viewing angles and lighting conditions. It would be impractical to try to collect thousands of real road images for such a data set. Instead, we developed a synthetic road-image generator that can create as many training examples as we need.

To train the network, 1200 simulated road images are presented 40 times each, while the weights are adjusted using the back-propagation learning algorithm. This takes about 30 minutes on Carnegie Mellon's Warp systolic-array supercomputer. (This machine was designed at Carnegie Mellon and is built by General Electric. It has a peak rate of 100 million floating-point operations per second and can compute weight adjustments for back-propagation networks at a rate of 20 million connections per second.)

Once it is trained, ALVINN can accurately drive the NAVLAB vehicle at about 3½ miles per hour along a path through a wooded area adjoining the Carnegie Mellon campus, under a variety of weather and lighting conditions. This speed is nearly twice as fast as that achieved by non-neural-network algorithms running on the same vehicle. Part of the reason for this is that the forward pass of a back-propagation network can be computed quickly. It takes about 200 milliseconds on the Sun-3/160 workstation installed on the NAVLAB.

The hidden-layer representations ALVINN develops are interesting. When trained on roads of a fixed width, the network chooses a representation in which hidden units act as detectors for complete roads at various positions and orientations. When trained on roads of variable

continued

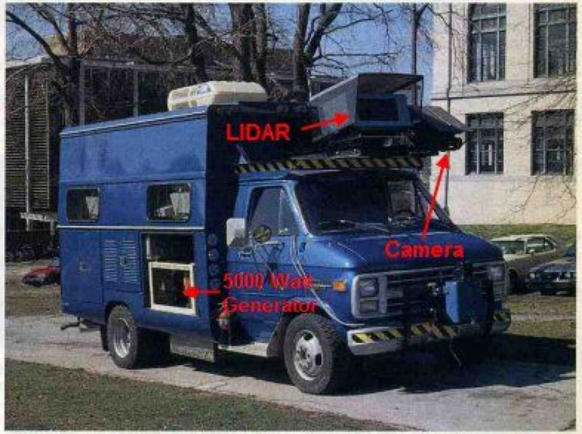


Photo 1: *The NAVLAB autonomous navigation test-bed vehicle and the road used for trial runs.*

Some Interesting Examples

Learning for Control from Multiple Demonstrations - Adam Coates, Pieter Abbeel, Andrew Ng, ICML 2008

An Application of Reinforcement Learning to Aerobatic Helicopter Flight - Pieter Abbeel, Adam Coates, Morgan Quigley, Andrew Y. Ng, NIPS 2006

Ghosting

https://www.youtube.com/watch?v=WI-WL2cj0CA

Data Driven Ghosting using Deep Imitation Learning

Hoang M. Le et al., SSAC 2017

Some Interesting Examples

Notation & Set-up

State: s (sometimes x) (**state may only be partially observed)

Action: a (sometimes y)

Policy: π_θ (sometimes h)

- Policy maps states to actions: $\pi_\theta(s) \rightarrow a$
- ...or distributions over actions: $\pi_\theta(s) \rightarrow P(a)$

State Dynamics: $P(s'|s,a)$

- Typically not known to policy.
- Essentially the simulator/environment

Notation & Set-up

Rollout: sequentially execute $\pi(s_0)$ on an initial state

- Produce trajectory $\boldsymbol{\tau}=(s_0,a_0,s_1,a_1,\ldots)$

$P(\boldsymbol{\tau}|\pi)$: distribution of trajectories induced by a policy

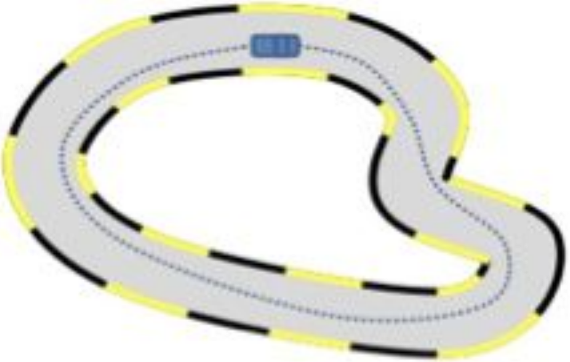
1. Sample s_0 from P_0 (distribution over initial states), initialize $t = 1$.
2. Sample action a_i from $\pi(s_{t-1})$
3. Sample next state s_t from applying a_t to s_{t-1} (requires access to environment)
4. Repeat from Step 2 with $t=t+1$

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

- Let $P_t(s|\pi)$ denote distribution over t-th state
- $P(s|\pi) = (1/T)\sum_t P_t(s|\pi)$

Example #1: Racing Game

(Super Tux Kart)



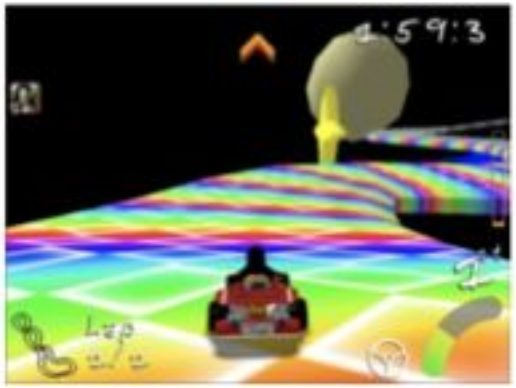
s = game screen

a = turning angle

Training set: D={r:=(**s**,**a**)} from π^*

- **s** = sequence of s
- **a** = sequence of a

Goal: learn π_θ(s)→a



Images from Stephane Ross

Example #2: Basketball Trajectories

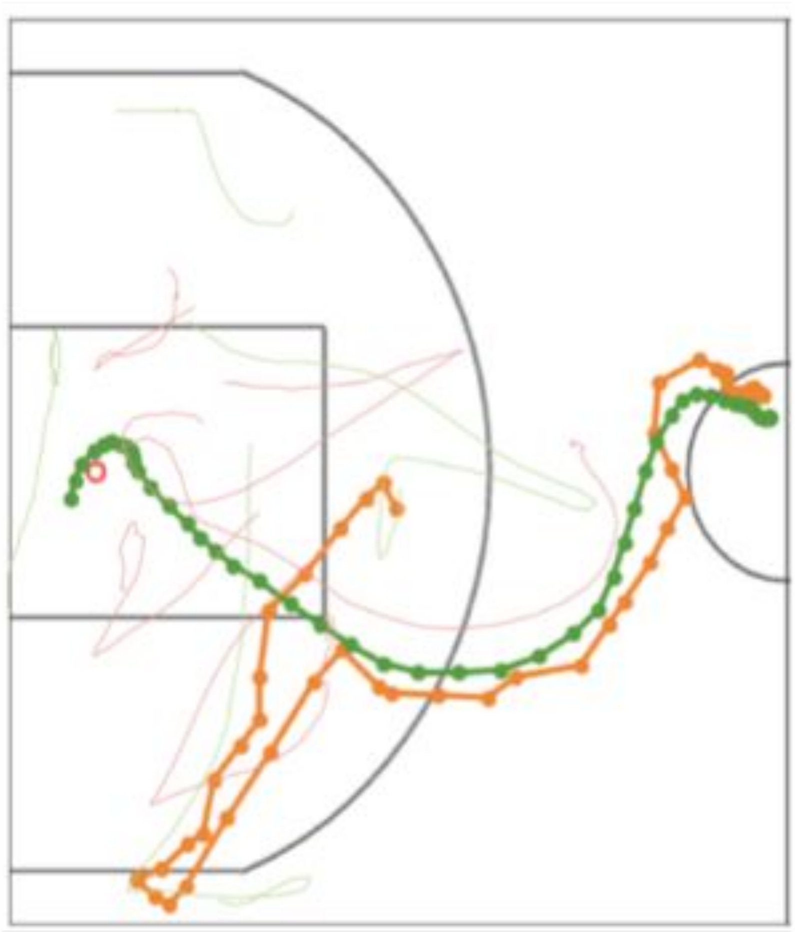
s = location of players & ball

a = next location of player

Training set: D={r:=(**s**,**a**)} from π^*

- **s** = sequence of s
- **a** = sequence of a

Goal: learn π_θ(s)→a

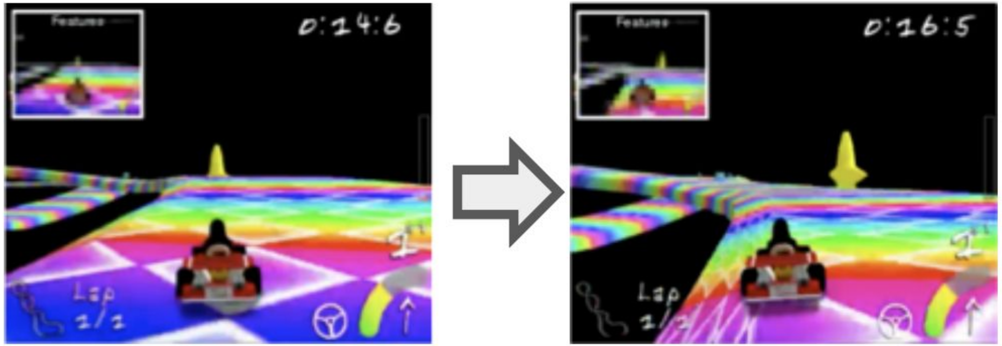


Behavioral Cloning = Reduction to Supervised Learning (Ignoring regularization for brevity.)

Write:

Behavioral Cloning vs. Imitation Learning

Limitations of Behavioral Cloning

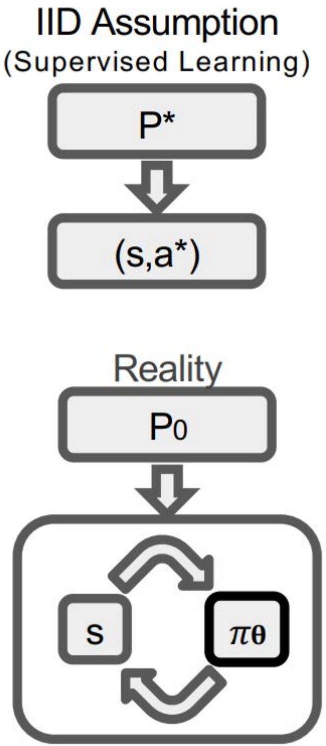


π_θ makes a mistake

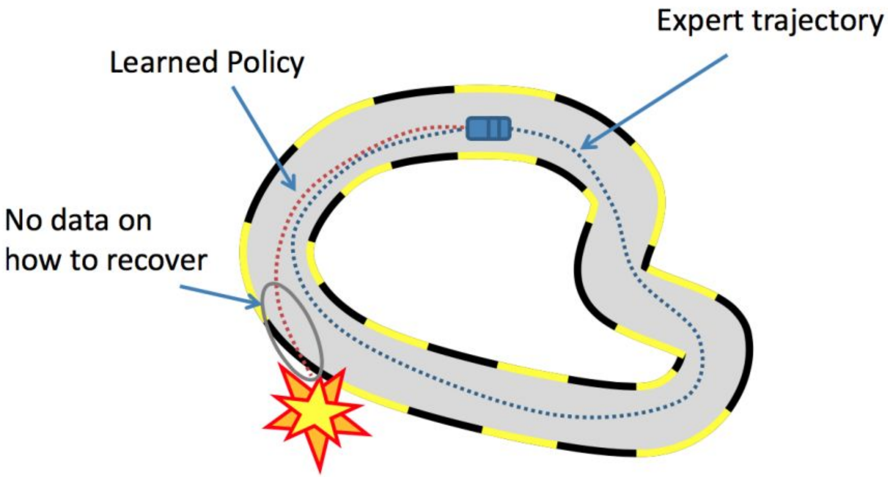
New state sampled not from P*!

Worst case is catastrophic!

Images from Stephane Ross



Limitations of Behavioral Cloning



Compounding Errors

Data distribution mismatch!

In supervised learning, $(x, y) \sim D$ during train **and** test. In MDPs:

- Train: $s_t \sim D_{\pi^*}$
- Test: $s_t \sim D_{\pi_\theta}$

When to use Behavioral Cloning?

Advantages

- Simple
- Simple
- Efficient

Disadvantages

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

Use When:

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

Don’t Use When:

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

Types of Imitation Learning

Behavioral Cloning

$$\operatorname{argmin}_{\theta} E_{(s,a^*)\sim P^*} L(a^*, \pi_{\theta}(s))$$

Works well when P^* close to P_{θ}

Inverse RL

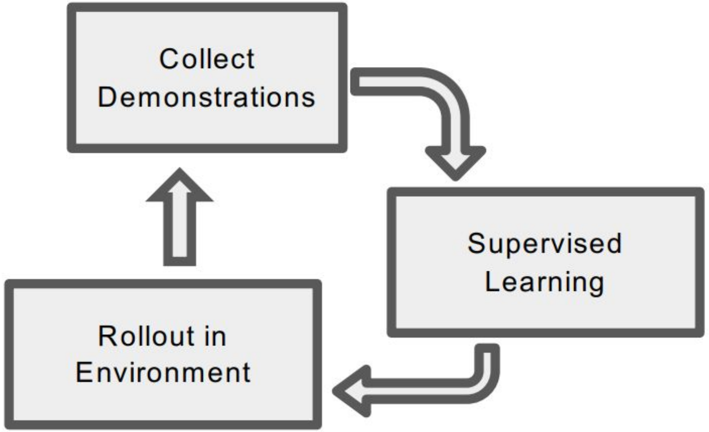
Learn r such that:

$$\pi^* = \operatorname{argmax}_{\theta} \underbrace{E_{s\sim P(s|\theta)} r(s, \pi_{\theta}(s))}_{\text{RL problem}}$$

Assumes learning r is statistically easier than directly learning π^*

Direct Policy Learning

via Interactive Demonstrator



Requires Interactive Demonstrator
(BC is 1-step special case)

Interactive Expert

Can query expert at any state

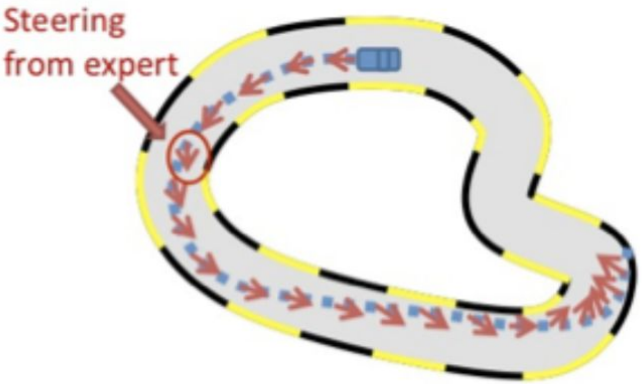
Construct loss function

- $L(\pi^*(s), \pi(s))$

Typically applied to rollout trajectories

- $s \sim P(s|\pi)$

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



Example from Super Tux Kart
(Image courtesy of Stephane Ross)

Expert provides feedback on
state visited by policy

Images from Stephane Ross

DAGGER: Dataset Aggregation

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.

β_i: a decreasing coefficient s.t. $\frac{1}{N}\sum_{i=1}^{N} \beta_i \rightarrow 0$ as $N \rightarrow \infty$

- Idea: Get more labels of the expert action along the path taken by the policy computed by behavior cloning
- Obtains a stationary deterministic policy with good performance under its induced state distribution

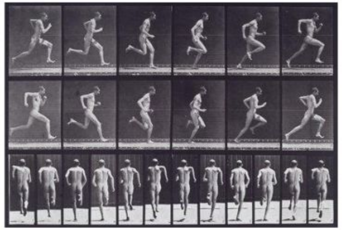
Inverse RL

- What if we don’t have an online 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 why?
    - Learn the demonstrator’s objective (reward) function
    - Apply RL

Inverse RL

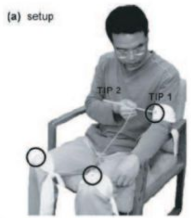
- What if we don’t have an online demonstrator?
  - We only have access to an offline set of demonstrated trajectories
- Behavioral cloning is not robust
  - Suffers from overfitting
  - We know what to do in observed states but can’t generalize well to other states
- How can we learn to mimic the demonstrator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

Inverse RL



Muybridge (c. 1870)

Mombaur et al. '09



Li & Todorov '06

Ziebart '08

$$\mathbf{a}_1, \ldots, \mathbf{a}_T = \arg\max_{\mathbf{a}_1, \ldots, \mathbf{a}_T} \sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t)$$

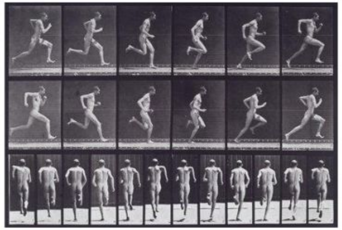
$$\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t)$$

optimize this to explain the data

$$\pi = \arg\max_{\pi} E_{\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t), \mathbf{a}_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t)} [r(\mathbf{s}_t, \mathbf{a}_t)]$$

$$\mathbf{a}_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t)$$

Inverse RL


Muybridge (c. 1870)


Mombaur et al. '09

Li & Todorov '06


Ziebart '08

$$\mathbf{a}_1, \dots, \mathbf{a}_T = \arg \max_{\mathbf{a}_1, \dots, \mathbf{a}_T} \sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t)$$

$$\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t)$$

optimize this to explain the data

$$\pi = \arg \max_{\pi} E_{\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t), \mathbf{a}_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t)} [r(\mathbf{s}_t, \mathbf{a}_t)]$$

$$\mathbf{a}_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t)$$

Inverse RL

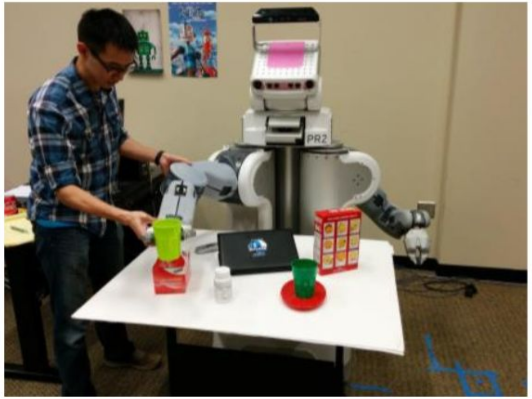
The imitation learning perspective

Standard imitation learning:

- copy the *actions* performed by the expert
- no reasoning about outcomes of actions

Human imitation learning:

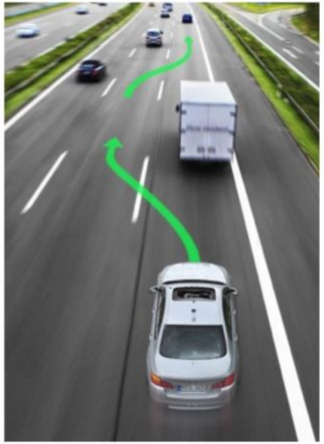
- copy the *intent* of the expert
- might take very different actions!



Inverse RL

The reinforcement learning perspective

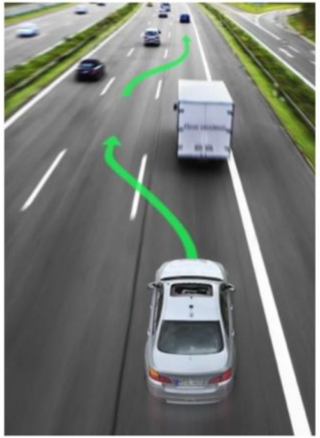
reward



what is the reward?

Inverse RL

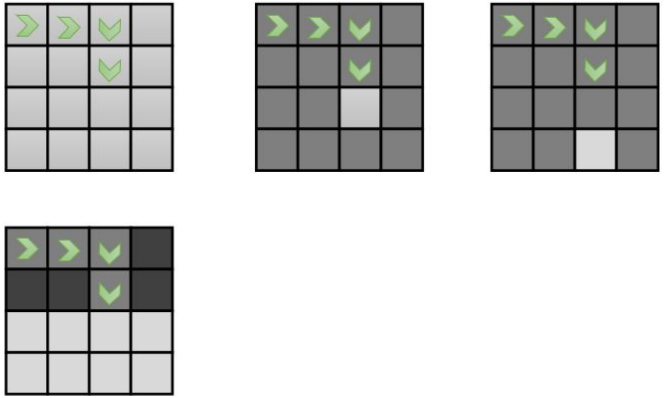
Infer **reward functions** from **demonstrations**



$r(\mathbf{s}, \mathbf{a})$

by itself, this is an **underspecified** problem

many reward functions can explain the **same** behavior



Inverse RL

"forward" reinforcement learning

given:

states $\mathbf{s} \in \mathcal{S}$, actions $\mathbf{a} \in \mathcal{A}$

(sometimes) transitions $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$

reward function $r(\mathbf{s}, \mathbf{a})$

learn $\pi^\star(\mathbf{a}|\mathbf{s})$

inverse reinforcement learning

given:

states $\mathbf{s} \in \mathcal{S}$, actions $\mathbf{a} \in \mathcal{A}$

(sometimes) transitions $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$

samples $\{\tau_i\}$ sampled from $\pi^\star(\tau)$

learn $r_\psi(\mathbf{s}, \mathbf{a})$

reward parameters

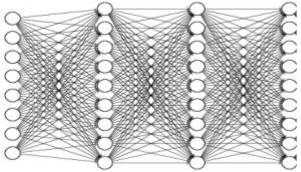
...and then use it to learn $\pi^\star(\mathbf{a}|\mathbf{s})$

linear reward function:

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

neural net reward function:

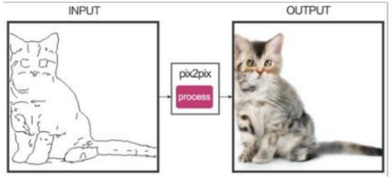
$\mathbf{s}$
$\mathbf{a}$



$r_\psi(\mathbf{s}, \mathbf{a})$

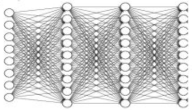
parameters ψ

GAN



Zhu et al. '17 Arjovsky et al. '17 Isola et al. '17

"generator"

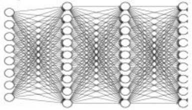
$\mathbf{z}$  $p_\theta(\mathbf{x}|\mathbf{z})$

samples from $p_\theta(\mathbf{x})$

data ("demonstrations") 

samples from $p^\star(\mathbf{x})$

$D(\mathbf{x}) = p_\psi(\text{real image}|\mathbf{x})$

$\mathbf{x}$ 

$$\psi = \arg\max_\psi \frac{1}{N} \sum_{\mathbf{x}\sim p^\star} \log D_\psi(\mathbf{x}) + \frac{1}{M} \sum_{\mathbf{x}\sim p_\theta} \log(1 - D_\psi(\mathbf{x}))$$

$$\theta \leftarrow \arg\max_\theta E_{\mathbf{x}\sim p_\theta} \log D_\psi(\mathbf{x})$$

Goodfellow et al. '14

Inverse RL as GAN

generator/policy

$\pi_\theta(\tau)$

samples from $\pi_\theta(\tau)$

data/demonstrations

samples from $p^\star(\tau)$

$$\psi \leftarrow \arg\max_\psi E_{\tau\sim p^\star}[\log D_\psi(\tau)] + E_{\tau\sim\pi_\theta}[\log(1 - D_\psi(\tau))]$$

$$D_\psi(\tau) = \frac{\frac{1}{Z}\exp(r(\tau))}{\prod_t \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) + \frac{1}{Z}\exp(r(\tau))}$$

$$\nabla_\theta \mathcal{L} \approx \frac{1}{M}\sum_{j=1}^{M} \nabla_\theta \log \pi_\theta(\tau_j) r_\psi(\tau_j)$$

policy changed to make it *harder* to distinguish from demos

Finn*, Christiano* et al. "A Connection Between Generative Adversarial Networks, Inverse Reinforcement Learning, and Energy-Based Models."

Inverse RL as GAN

initial
policy π

samples from $\pi_\theta(\tau)$

human
demonstrations

samples from $\pi^\star(\tau)$

$$\nabla_\psi \mathcal{L} \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\psi r_\psi(\tau_i) - \frac{1}{\sum_j w_j}\sum_{j=1}^{M} w_j \nabla_\psi r_\psi(\tau_j)$$

demos are made more likely, samples less likely

$$\nabla_\theta \mathcal{L} \approx \frac{1}{M}\sum_{j=1}^{M} \nabla_\theta \log \pi_\theta(\tau_j) r_\psi(\tau_j)$$

policy changed to make it *harder* to distinguish from demos

Required Readings and references

1. Human-in-the-Loop Deep Reinforcement Learning with Application to Autonomous Driving, Jingda Wu, Zhiyu Huang, Chao Huang, Zhongxu Hu, Peng Hang, Yang Xing, Chen Lv*

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Thank you