Imitation Learning: Behavior Cloning + IRL

Alina Vereshchaka

CSE4/510 Reinforcement Learning
Fall 2019

avereshc@buffalo.edu

October 15, 2019

*Slides are adopted from Berkley Deep RL course CS294-112 & Deep RL and Control, CMU 10703

The Imitation Learning problem

The agent needs to come up with a policy whose resulting state, action trajectory distribution matches the expert trajectory distribution.

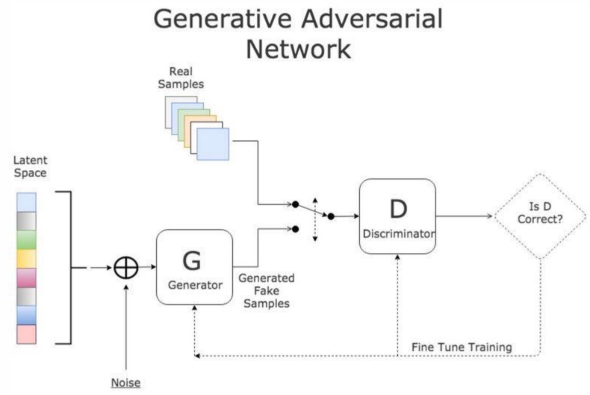
The Imitation Learning problem

The agent needs to come up with a policy whose resulting state, action trajectory distribution matches the expert trajectory distribution.

Does this remind us of something?

The agent needs to come up with a policy whose resulting state, action trajectory distribution matches the expert trajectory distribution.

Does this remind us of something?



Two broad approaches:

Imitation Learning

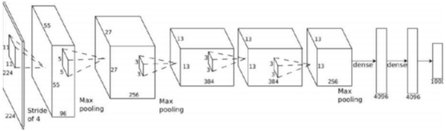
Two broad approaches:

- **Direct(Behavior cloning):** Supervised training of policy (mapping states to actions) using the demonstration trajectories as groundtruth
- **Indirect (Inverse Reinforcement Learning):** Learn the unknown reward function/goal of the teacher, and derive the policy from these

Table of Contents

1. Behavior Cloning
2. Inverse Reinforcement Learning

$\mathbf{o}_t$

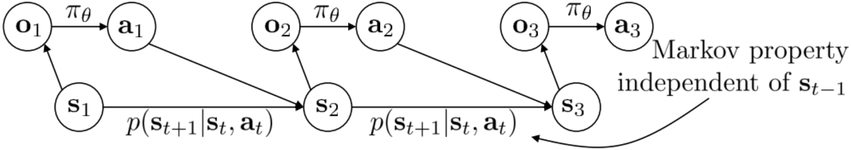


$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

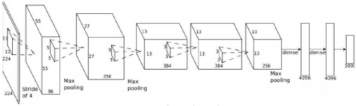
$\mathbf{a}_t$

$\mathbf{s}_t$ – state
$\mathbf{o}_t$ – observation
$\mathbf{a}_t$ – action

$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ – policy
$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ – policy (fully observed)



$\mathbf{o}_t$



$\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$

$\mathbf{u}_t$

$\mathbf{x}_t$ – state
$\mathbf{o}_t$ – observation
$\mathbf{u}_t$ – action

$\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ – policy

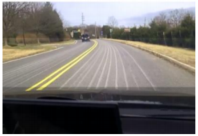
a bit of history...

$\mathbf{x}_t$ – state
$\mathbf{u}_t$ – action
управление

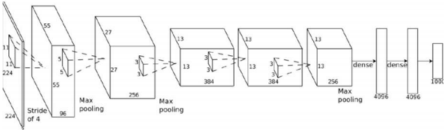
Lev Pontryagin

$\mathbf{s}_t$ – state
$\mathbf{a}_t$ – action

Richard Bellman



$\mathbf{o}_t$



$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

$\mathbf{a}_t$

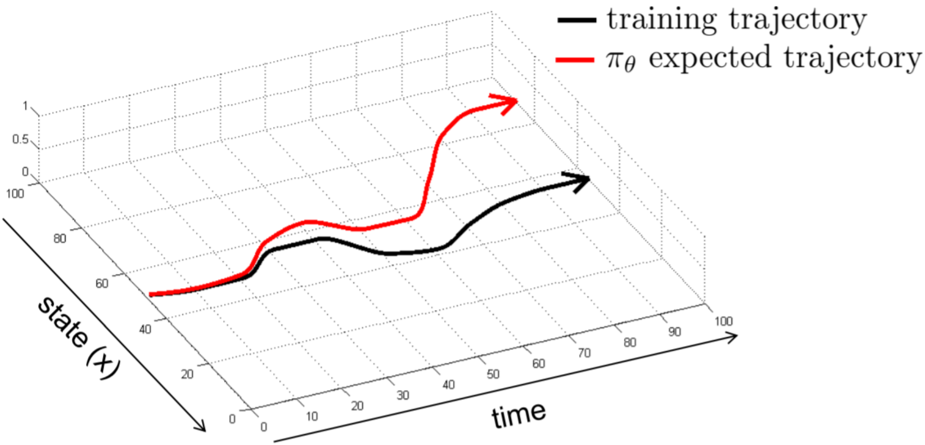
$\mathbf{o}_t$
$\mathbf{a}_t$

training data → supervised learning → $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

behavior cloning

Images: Bojarski et al. '16, NVIDIA

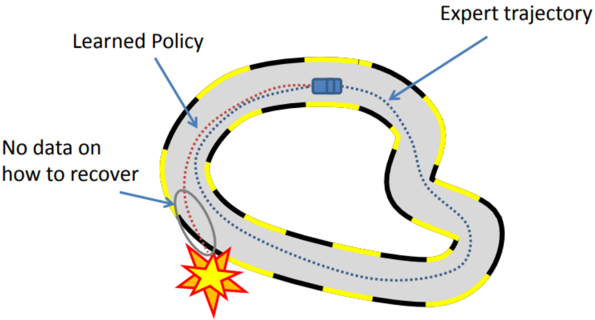
Does it work?



training trajectory
π_θ expected trajectory
1
0.5
0
100
80
60
40
20
0
state (x)
0
10
20
30
40
50
60
70
80
90
100
time

Data Distribution Mismatch

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



Data Distribution Mismatch

	Supervised Learning	Supervised learning + Control
Train	$(x, y) \sim D$	$s \sim d_{\pi*}$
Test	$(x, y) \sim D$	$s \sim d_{\pi}$

Data Distribution Mismatch

	Supervised Learning	Supervised learning + Control
Train	$(x, y) \sim D$	$s \sim d_{\pi*}$
Test	$(x, y) \sim D$	$s \sim d_{\pi}$

- Supervised Learning succeeds when training and test data distributions match
- But state distribution under learned π differs from those generated by $\pi*$

Does it work? Yes!

Video: Bojarski et al. '16, NVIDIA

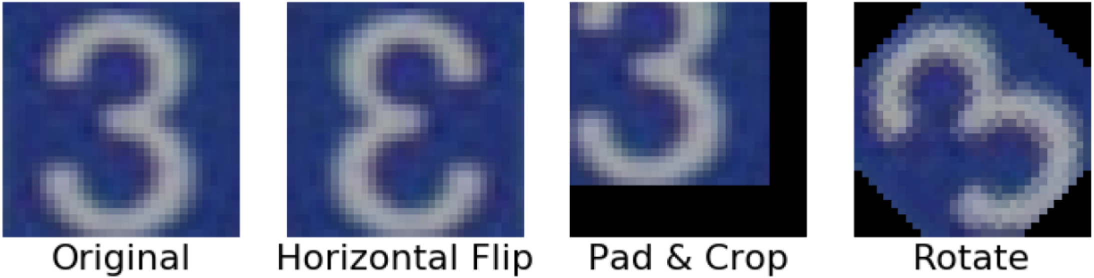
Solution: Demonstration augmentation

Change $p_{data}(o_t)$ using demonstration augmentation.
Label additional examples generated by the learned policy, drawn from $p_{\pi^{learned}}(o_t)$.
How?

- Use human expert
- Synthetically change observed o_t and corresponding u_t

What it data augmentation?

Data augmentation [1] significantly increase the diversity of data available for training models, without actually collecting new data. Data augmentation techniques such as cropping, padding, and horizontal flipping.



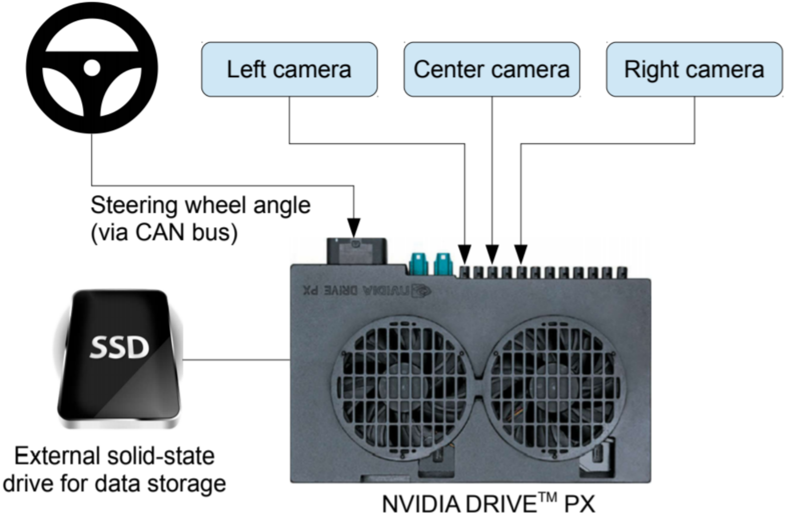
[1] https : //bair.berkeley.edu/blog/2019/06/07/data_aug/

Demonstration augmentation: DAVE-2 [2]

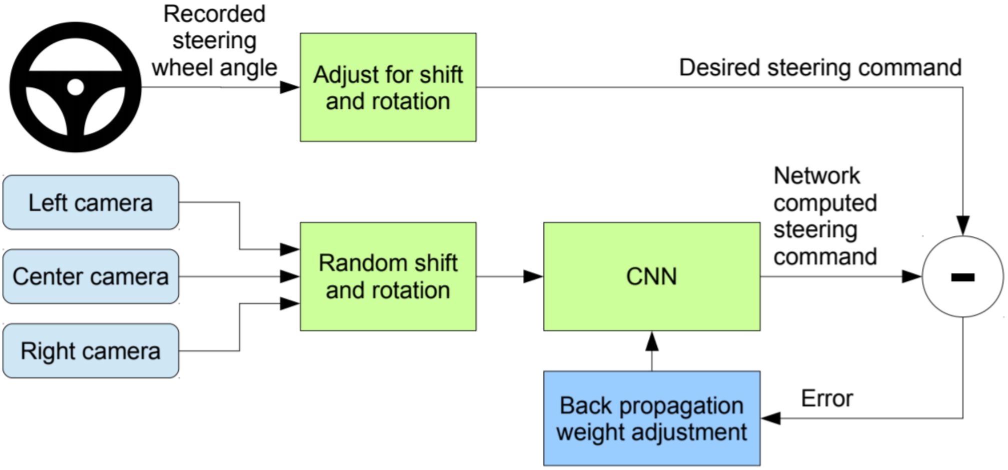
- Trained CNN to map raw pixels from a single front-facing camera directly to steering commands.
- With minimum training data the system learns to drive in traffic on local roads and operates in areas with unclear visual guidance such as in parking lots and on unpaved roads.
- The system learns detecting useful road features with only the human steering angle as the training signal.
- DAVE-2 driving Lincoln YouTube video

[2] $https://arxiv.org/pdf/1604.07316.pdf$

DAVE-2: High-level view of the data collection system

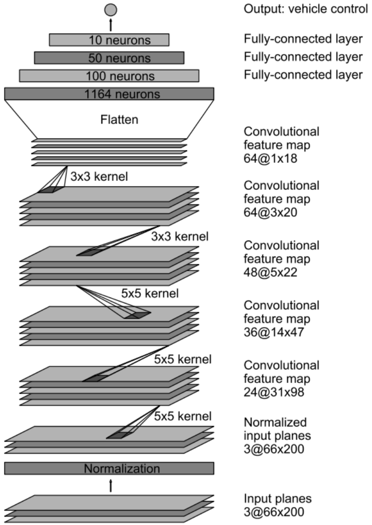


DAVE-2: Training the neural network

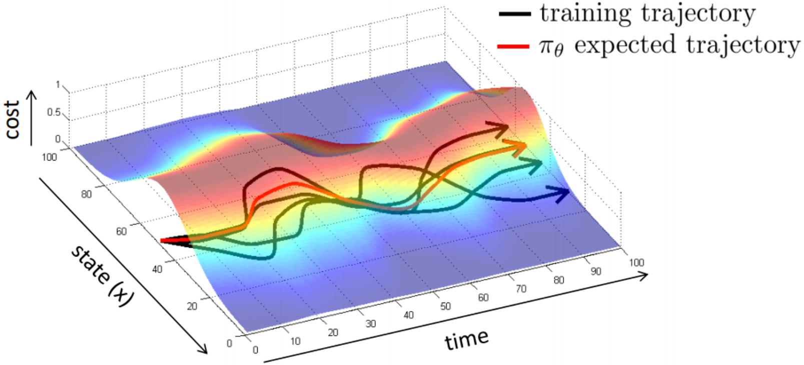


DAVE-2: CNN architecture

- The network has about 27 million connections and 250 thousand parameters.

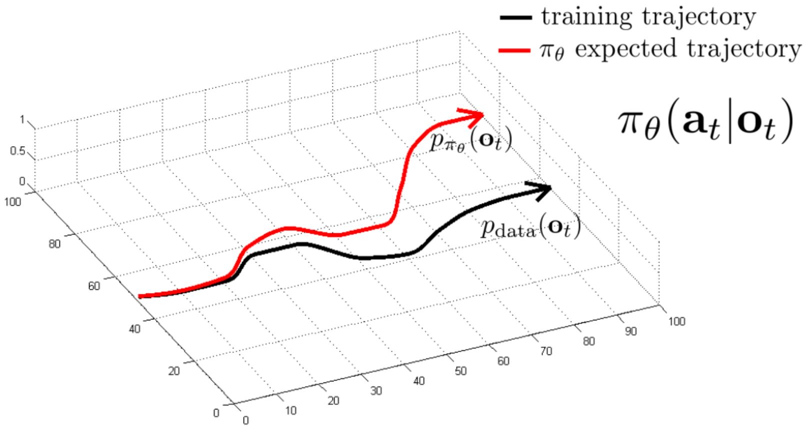


Can we make it work more often?



stability

Can we make it work more often?



$$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$$

can we make $p_{\text{data}}(\mathbf{o}_t) = p_{\pi_\theta}(\mathbf{o}_t)$?

Can we make it more often?

Can we make $p_{data}(o_t) = p_{\pi_\theta}(o_t)$?

DAgger: Dataset Aggregation

Goal: collect training data from $p_{\pi_\theta}(o_t)$ instead of $p_{data}(o_t)$

How?

Can we make it more often?

Can we make $p_{data}(o_t) = p_{\pi_\theta}(o_t)$?

DAgger: Dataset Aggregation

Goal: collect training data from $p_{\pi_\theta}(o_t)$ instead of $p_{data}(o_t)$

How? Just run $\pi_\theta(a_t|o_t)$, but need label a_t

1. Train $\pi_\theta(a_t|o_t)$ from human data $D = \{o_1, a_1, \dots, o_N, a_N\}$
2. Run $\pi_\theta(a_t|o_t)$ to get dataset $D = \{o_1, \dots, o_M\}$
3. Ask human to label D_π with actions a_t
4. Aggregare: $D \leftarrow D \cup D_\pi$
5. Go to Step 1

What is the problem?

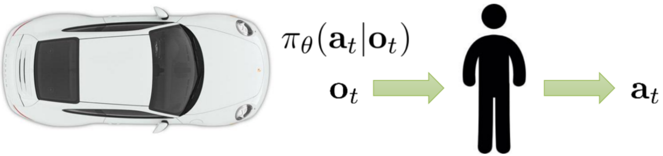
DAgger: Dataset Aggregation

1. Train $\pi_\theta(a_t|o_t)$ from human data $D = \{o_1, a_1, \dots, o_N, a_N\}$
2. Run $\pi_\theta(a_t|o_t)$ to get dataset $D = \{o_1, \dots, o_M\}$
3. Ask human to label D_π with actions a_t
4. Aggregate: $D \leftarrow D \cup D_\pi$
5. Go to Step 1

What is the problem?

DAgger: Dataset Aggregation

1. Train $\pi_\theta(a_t|o_t)$ from human data $D = \{o_1, a_1, \ldots, o_N, a_N\}$
2. Run $\pi_\theta(a_t|o_t)$ to get dataset $D = \{o_1, \ldots, o_M\}$
3. Ask human to label D_π with actions a_t
4. Aggregate: $D \leftarrow D \cup D_\pi$
5. Go to Step 1



Why might we fail to fit the expert?

- Non-Markovian behavior
- Multimodal behavior

Non-Markovian behavior

$$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$$

behavior depends only
on current observation

$$\pi_\theta(\mathbf{a}_t|\mathbf{o}_1, ..., \mathbf{o}_t)$$

behavior depends on
all past observations

Non-Markovian behavior

$$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$$

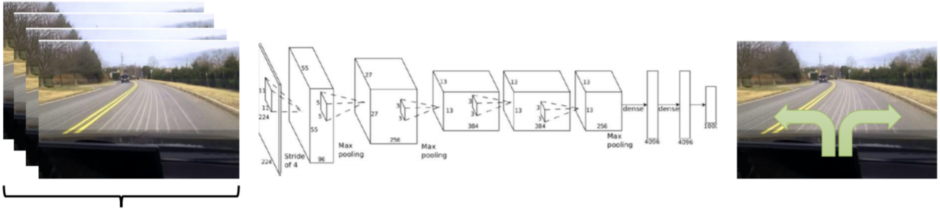
behavior depends only
on current observation

$$\pi_\theta(\mathbf{a}_t|\mathbf{o}_1, ..., \mathbf{o}_t)$$

behavior depends on
all past observations

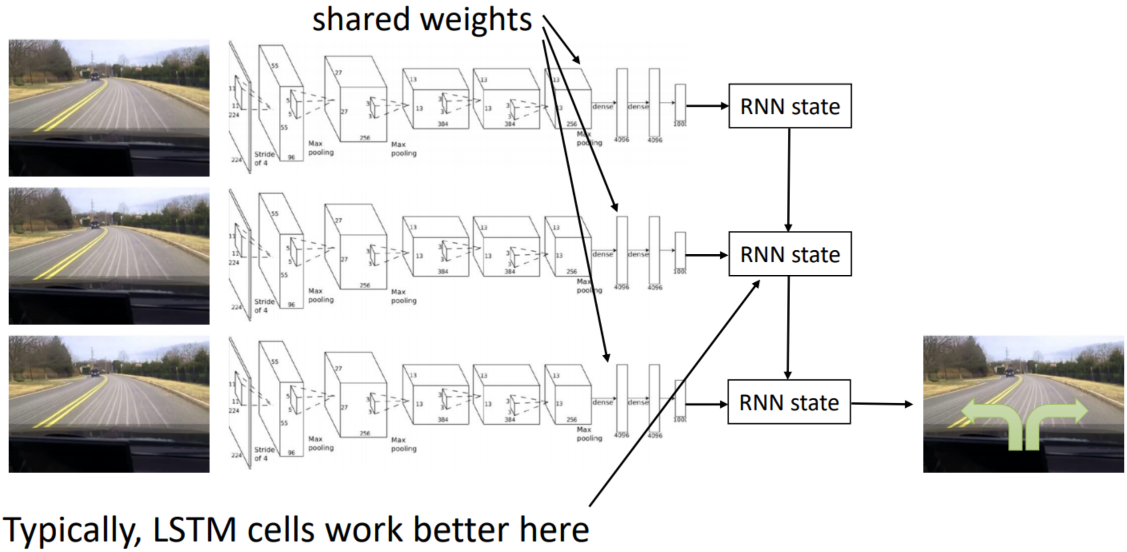
- If we see the same thing twice, we do the same thing twice, regardless of what happened before
- Often very unnatural for human demonstrators

How can we use the whole history?



variable number of frames,
too many weights

Using Recurrent Neural Networks



Aside: why might this work poorly? [3]

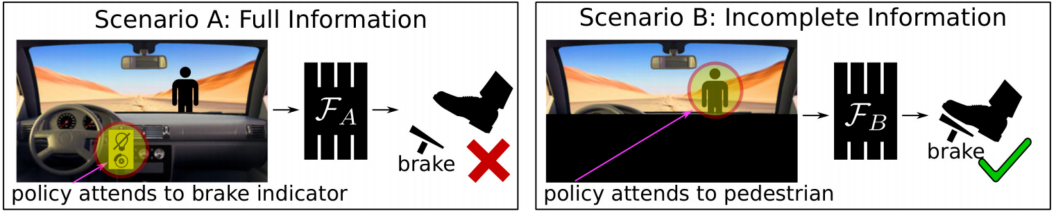
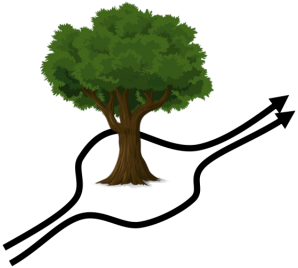


Figure 1: Causal confusion: *more* information yields worse imitation learning performance. Model A relies on the braking indicator to decide whether to brake. Model B instead correctly attends to the pedestrian.

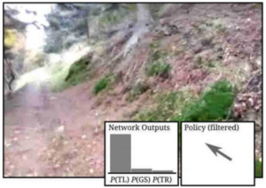
[3] de Haan, Pim, Dinesh Jayaraman, and Sergey Levine. "Causal Confusion in Imitation Learning." (2019)

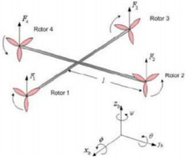
Multimodal Behavior



Imitation learning: what's the problem?

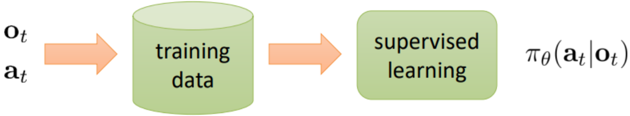
- Humans need to provide data, which is typically finite
  - Deep learning works best when data is plentiful
- Humans are not good at providing some kinds of actions



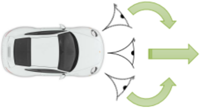


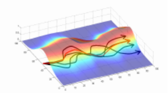
- Humans can learn autonomously; can our machines do the same?
  - Unlimited data from own experience
  - Continuous self-improvement

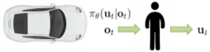
Behavior Cloning: Summary



- Often (but not always) insufficient by itself
  - Distribution mismatch problem
- Sometimes works well
  - Hacks (e.g. left/right images)
  - Samples from a stable trajectory distribution
  - Add more **on-policy** data, e.g. using Dagger
  - Better models that fit more accurately







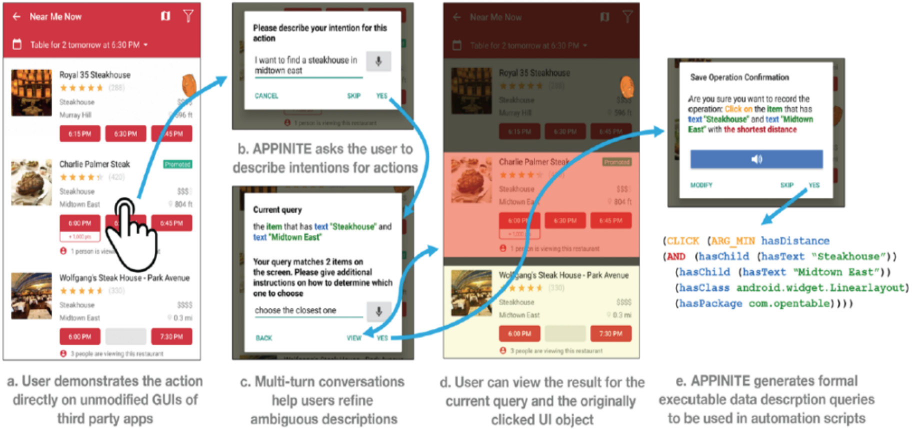


Fig. 1. Specifying data description in programming by demonstration using APPINITE: (a, b) enables users to naturally express their intentions for demonstrated actions verbally; (c) guides users to formulate data descriptions to uniquely identify target GUI objects; (d) shows users real-time updated results of current queries on an interaction overlay; and (e) formulates executable queries from natural language instructions.

Table of Contents

1 Behavior Cloning

2 Inverse Reinforcement Learning

Inverse Reinforcement Learning (IRL)

Computer Games

reward

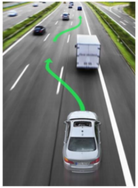
Mnih et al. '15

Real World Scenarios

robotics

dialog

autonomous driving



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 Reinforcement Learning (IRL)

Inverse Optimal Control / Inverse Reinforcement Learning:
infer reward function from demonstrations
(IOC/IRL) (Kalman '64, Ng & Russell '00)

given:
- state & action space
- samples from π*
- dynamics model (sometimes)

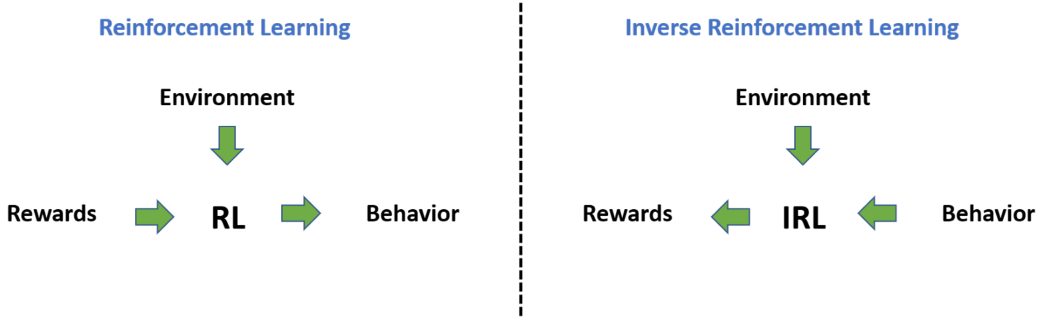
goal:
- recover reward function
- then use reward to get policy

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:
    $s_0, a_0, s_1, a_1, s_2, a_2, \ldots$
    (= trace of the teacher's policy π^*)
- **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)



Inverse Reinforcement Learning (IRL)

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

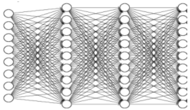
linear reward function:

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

neural net reward function:

$\mathbf{s}$

$\mathbf{a}$



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

parameters ψ

IRL problem is to find a reward function that can explain the expert behavior