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 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?
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.
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
**Terminology & Notations**

- $o_t$ – state
- $o_t$ – observation
- $a_t$ – action

$$\pi_\theta(a_t|o_t)$$ – policy

$$\pi_\theta(a_t|s_t)$$ – policy (fully observed)

Markov property independent of $s_{t-1}$
Terminology & Notations

\[ \pi_\theta(u_t | o_t) \] is the policy.

- \( o_t \) – observation
- \( u_t \) – action
- \( s_t \) – state
- \( a_t \) – action

A bit of history...

- State: \( x_t \)
- Action: \( u_t \)
- State: \( s_t \)
- Action: \( a_t \)

Lev Pontryagin

Richard Bellman
Imitation Learning

\( o_t \)  \[ \pi_\theta(a_t|o_t) \]  \( a_t \)

Behavior cloning

Images: Bojarski et al. ‘16, NVIDIA
Does it work?
Data Distribution Mismatch

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

- Learned Policy
- Expert trajectory
- No data on how to recover
Data Distribution Mismatch

<table>
<thead>
<tr>
<th></th>
<th>Supervised Learning</th>
<th>Supervised learning + Control</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Train</strong></td>
<td>$(x, y) \sim D$</td>
<td>$s \sim d_{\pi^*}$</td>
</tr>
<tr>
<td><strong>Test</strong></td>
<td>$(x, y) \sim D$</td>
<td>$s \sim d_{\pi}$</td>
</tr>
</tbody>
</table>

Supervised learning succeeds when training and test data distributions match. But state distribution under learned $\pi$ differs from those generated by $\pi^*$. 
Data Distribution Mismatch

<table>
<thead>
<tr>
<th></th>
<th>Supervised Learning</th>
<th>Supervised learning + Control</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Train</strong></td>
<td>$(x, y) \sim D$</td>
<td>$s \sim d_{\pi^*}$</td>
</tr>
<tr>
<td><strong>Test</strong></td>
<td>$(x, y) \sim D$</td>
<td>$s \sim d_{\pi}$</td>
</tr>
</tbody>
</table>

- Supervised Learning succeeds when training and test data distributions match.
- But state distribution under learned $\pi$ differs from those generated by $\pi^*$. 
Does it work?

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 is 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.

\[^{1}\text{https://bair.berkeley.edu/blog/2019/06/07/data_aug/}\]
Demonstration augmentation: DAVE-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

---

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

- Left camera
- Center camera
- Right camera

Steering wheel angle (via CAN bus)

External solid-state drive for data storage

NVIDIA DRIVE™ PX
DAVE-2: Training the neural network

Recorded steering wheel angle → Adjust for shift and rotation

Left camera
Center camera
Right camera → Random shift and rotation

CNN

Back propagation weight adjustment

Desired steering command

Network computed steering command

Error
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?

Can we make \( p_{\text{data}}(o_t) = p_{\pi_\theta}(o_t) \)?
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 \( D = \{o_1, a_1, \ldots, o_N, a_N\} \)
2. Run \( \pi\theta(a_t|o_t) \) to get dataset \( D = \{o_1, \ldots, o_M\} \)
3. Ask human to label \( D_{\pi} \) 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 $D_\pi$ 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 $D_\pi$ 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

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

\[ \pi_\theta (a_t \mid o_t) \]

behavior depends only on current observation

\[ \pi_\theta (a_t \mid o_1, \ldots, o_t) \]

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

Typically, LSTM cells work better here.
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.

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
Learning From Showing and Telling

a. User demonstrates the action directly on unmodified GUIs of third party apps

b. APPINITE asks the user to describe intentions for actions

c. Multi-turn conversations help users refine ambiguous descriptions

d. User can view the result for the current query and the originally clicked UI object

e. APPINITE generates formal executable data description queries to be used in automation scripts

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.
Inverse Reinforcement Learning (IRL)

Computer Games
reward

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 Optimal Control / Inverse Reinforcement Learning:
infer reward function from demonstrations

(IOC/IRL)  (Kalman ’64, Ng & Russell ’00)

given:
- state & action space
- samples from $\pi^*$
- 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:
    
    $s_0, a_0, s_1, a_1, s_2, a_2, ...$
    (= trace of the teacher’s policy $\pi^*$)

• **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)

Reinforcement Learning

Environment → Rewards → RL → Behavior

Inverse Reinforcement Learning

Environment → Rewards → IRL → Behavior
Inverse Reinforcement Learning (IRL)

"forward" reinforcement learning

given:
- states $s \in S$, actions $a \in A$
- (sometimes) transitions $p(s'|s, a)$
- reward function $r(s, a)$

learn $\pi^*(a|s)$

inverse reinforcement learning

given:
- states $s \in S$, actions $a \in A$
- (sometimes) transitions $p(s'|s, a)$
- samples $\{\tau_i\}$ sampled from $\pi^*(\tau)$

learn $r_\psi(s, a)$

...and then use it to learn $\pi^*(a|s)$

linear reward function:

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

neural net reward function:

$$r_\psi(s, a)$$

parameters $\psi$
Inverse Reinforcement Learning (IRL)

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