Imitation Learning: Behavior Cloning

Alina Vereshchaka

University at Buffalo
avershc@buffalo.edu

June 13, 2019

*Slides are adopted from Berkley Deep RL course CS294-112 & Deep RL and Control, CMU 10703
Reinforcement Learning: Learning policies guided by sparse rewards, e.g., win the game.

- **Good**: simple, cheap form of supervision
- **Bad**: High sample complexity

Where is it successful so far?

- In simulation, where we can afford a lot of trials, easy to parallelize
- Not in robotic systems:
  - action execution takes long
  - we cannot afford to fail
  - safety concerns

Learning from Demonstration for Autonomous Navigation in Complex Unstructured Terrain, Silver et al. 2010
Reward Shaping

Ideally we want dense in time rewards to closely guide the agent closely along the way.

Who will supply those shaped rewards?

1. We will manually design them: “cost function design by hand remains one of the ‘black arts’ of mobile robotics, and has been applied to untold numbers of robotic systems”

2. We will learn them from demonstrations: “rather than having a human expert tune a system to achieve desired behavior, the expert can demonstrate desired behavior and the robot can tune itself to match the demonstration”

Learning from Demonstration for Autonomous Navigation in Complex Unstructured Terrain, Silver et al. 2010
Learning from demonstration

Learning from demonstrations a.k.a. **Imitation Learning**: Supervision through an expert (teacher) that provides a set of **demonstration trajectories**: sequences of states and actions.

**Imitation learning** is useful when it is easier for the expert to demonstrate the desired behavior rather than:

a) coming up with a reward function that would generate such behavior,

b) coding up with the desired policy directly.

and the sample complexity is manageable
Two broad approaches:

- **Direct**: Supervised training of policy (mapping states to actions) using the demonstration trajectories as ground-truth (a.k.a. behavior cloning)

- **Indirect**: Learn the unknown reward function/goal of the teacher, and derive the policy from these, a.k.a. Inverse Reinforcement Learning
**Supervised training**

- Behavior Cloning: Imitation learning as supervised learning
- Compounding errors
- Demonstration augmentation techniques
- DAGGER

**Inverse reinforcement learning**

- Feature matching
- Max margin planning
- Maximum entropy IRL
Terminology & Notations

- $o_t$ - observation
- $a_t$ - action
- $s_t$ - state

$\pi(\mathbf{a}_t | o_t)$ - policy
$\pi(\mathbf{a}_t | s_t)$ - policy (fully observed)

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

- $x_t$ - state
- $o_t$ - observation
- $u_t$ - action
- $\pi_\theta(u_t|o_t)$ - policy

A bit of history...

- $x_t$ - state
- $u_t$ - action
- $s_t$ - state
- $a_t$ - action

Left: Lev Pontryagin

Right: Richard Bellman
Imitation Learning

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

behavior cloning

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

![Graph showing training trajectory and \( \pi_\theta \) expected trajectory over time and state.](image)
Data Distribution Mismatch

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

- No matter how good it, the policy will make a mistake
- Small errors compound over time
- New states will be completely new to the agent, that wasn’t in the training set
- Eventually it may fail
- Decisions are purposeful, in supervised learning we don’t have a goal or planning problem
Does it work?

Yes!

Video: Bojarski et al. ’16, NVIDIA
Can we make it work more often?

stability
Can we make it work more often?

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

\[ p_{\pi_\theta} (o_t) \]

\[ p_{\text{data}} (o_t) \]

can we make \( p_{\text{data}} (o_t) = p_{\pi_\theta} (o_t) \)?
Solution: Demonstration Augmentation

Change $p_{\pi^*}(o_t)$ using demonstration augmentation!

Have expert label additional examples generated by the learned policy (e.g., drawn from $p_{\pi^{\text{learned}}}(o_t)$).
Solution: Demonstration Augmentation

Change $p_{\pi^*}(o_t)$ using demonstration augmentation!

Have expert label additional examples generated by the learned policy (e.g., drawn from $p_{\pi^{\text{learned}}}(o_t)$)

How?

1. use human expert

2. synthetically change observed $o_t$ and corresponding $u_t$
What is the problem?

can we make $p_{\text{data}}(o_t) = p_{\pi_\theta}(o_t)$?
idea: instead of being clever about $p_{\pi_\theta}(o_t)$, be clever about $p_{\text{data}}(o_t)$!

**DAgger: Dataset Aggregation**
goal: collect training data from $p_{\pi_\theta}(o_t)$ instead of $p_{\text{data}}(o_t)$
how? just run $\pi_\theta(a_t|o_t)$
but need labels $a_t$!

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 $D_\pi = \{o_1, \ldots, o_M\}$
3. Ask human to label $D_\pi$ with actions $a_t$
4.Aggregate: $D \leftarrow D \cup D_\pi$

Ross et al. ‘11
What is the problem?

1. train $\pi_\theta(a_t|o_t)$ from human data $\mathcal{D} = \{o_1, a_1, \ldots, o_N, a_N\}$
2. run $\pi_\theta(a_t|o_t)$ to get dataset $\mathcal{D}_\pi = \{o_1, \ldots, o_M\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions $a_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

Ross et al. ‘11
Can we make it work without more data?

• DAgger addresses the problem of distributional “drift”
• What if our model is so good that it doesn’t drift?
• Need to mimic expert behavior very accurately
• But don’t overfit!
Why might we fail to fit the expert?

1. Non-Markovian behavior
2. Multimodal behavior

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

behavior depends only on current observation
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?

Typically, LSTM cells work better here.
Why might we fail to fit the expert?

1. Non-Markovian behavior
2. Multimodal behavior

1. Output mixture of Gaussians
2. Latent variable models
3. Autoregressive discretization
Imitation learning: recap

- 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
A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots

Alessandro Giusti¹, Jérôme Guazzi¹, Dan C. Cireșan¹, Fang-Lin He¹, Juan P. Rodríguez¹
Flavio Fontana², Matthias Faessler², Christian Forster²
Jürgen Schmidhuber¹, Gianni Di Caro¹, Davide Scaramuzza², Luca M. Gambardella¹
Case study 2: Imitation with LSTMs

Learning real manipulation tasks from virtual demonstrations using LSTM

Rouhollah Rahmatizadeh¹, Pooya Abolghasemi¹, Aman Behal² and Ladislau Bölöni¹
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
Aside: notation

\[ s_t \] – state
\[ a_t \] – action
\[ r(s, a) \] – reward function

\[ x_t \] – state
\[ u_t \] – action
\[ c(x, u) \] – cost function

\[ r(s, a) = -c(x, u) \]
Notations

\[ r(s, a) = \log p(a = \pi^*(s) | s) \]

\[ c(s, a) = \begin{cases} 
0 & \text{if } a = \pi^*(s) \\
1 & \text{otherwise} 
\end{cases} \]

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 \( D_\pi = \{o_1, \ldots, o_M\} \)
3. Ask human to label \( D_\pi \) with actions \( a_t \)
4. Aggregate: \( D \leftarrow D \cup D_\pi \)

Ross et al. ‘11
End to End Learning for Self-Driving Cars

Abstract

We trained a convolutional neural network (CNN) to map raw pixels from a single front-facing camera directly to steering commands. This end-to-end approach proved surprisingly powerful. With minimum training data from humans the system learns to drive on local roads with or without lane markings and on highways. It also operates in areas with unclear visual guidance such as in parking lots and on unpaved roads.

Figure 4: CNN architecture. The network has about 27 million connections and 250 thousand parameters.
“DAVE-2 was inspired by the pioneering work of Pomerleau [6] who in 1989 built the Autonomous Land Vehicle in a Neural Network (ALVINN) system. Training with data from only the human driver is not sufficient. The network must learn how to recover from mistakes. …”

End to End Learning for Self-Driving Cars, Bojarski et al. 2016
DAVE 2 Driving a Lincoln

- A convolutional neural network
- Trained by human drivers
- Learns perception, path planning, and control
  "pixel in, action out"
- Front-facing camera is the only sensor

Synthesizes new state-action pairs by rotating and translating input image, and calculating compensating steering command
Dataset AGGregation: bring learner’s and expert’s trajectory distributions closer by iteratively labelling expert action for states generated by the current policy

1. train $\pi_\theta(u_t|o_t)$ from human data $D_{\pi^*} = \{o_1, u_1, ..., o_N, u_N\}$
2. run $\pi_\theta(u_t|o_t)$ to get dataset $D_\pi = \{o_1, ..., o_M\}$
3. Ask human to label $D_\pi$ with actions $u_t$
4. Aggregate: $D_{\pi^*} \leftarrow D_{\pi^*} \cup D_\pi$
5. GOTO step 1.

Problems:
- execute an unsafe/partially trained policy
- repeatedly query the expert

A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning, Ross et al. 2011
Caveats:

1. Is hard for the expert to provide the right magnitude for the turn without feedback of his own actions!  
   Solution: provide visual feedback to expert

2. The expert’s reaction time to the drone’s behavior is large, this causes imperfect actions to be commanded.  
   Solution: play-back in slow motion offline and record their actions.

3. Executing an imperfect policy causes accidents, crashes into obstacles.  
   Solution: safety measures which again make the data distribution matching imperfect between train and test, but good enough.

Learning monocular reactive uav control in cluttered natural environments, Ross et al. 2013
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.