imitation learning
1. Learning from reinforcement alone is hard
2. Exploration is hard
3. Credit assignment is hard

2. Yet people are pretty good at many tasks
3. Perhaps we can use them to help
From Mario AI competition 2009

Input:

Interface

Output:

Jump in \{0,1\}
Right in \{0,1\}
Left in \{0,1\}
Speed in \{0,1\}

High level goal:
Watch an expert play and learn to mimic her behavior
Training (expert)

Video credit: Stéphane Ross, Geoff Gordon and Drew Bagnell
A policy maps observations to actions:

\[ \pi(\text{obs.}) = a \]
1. Collect trajectories from expert $\pi^{\text{ref}}$
2. Store dataset $D = \{ (o, \pi^{\text{ref}}(o)) \mid o \sim \pi^{\text{ref}} \}$
3. Train classifier $\pi$ on $D$

Let $\pi$ play the game!
Test - time execution (sup. learning)

Video credit: Stéphane Ross, Geoff Gordon and Drew Bagnell
What's the (biggest) failure mode?

The expert never gets stuck next to pipes \( \implies \) Classifier doesn't learn to recover!
From demonstrations → expert decisions

From expert decisions → expert costs

Whence the expert?

Combining experts and reward
What's the (biggest) failure mode?

Classifier doesn't learn to recover!

- We'd like to train the policy on all states
- Can't do that
- Let's train it where it visits
Learning from an expert: DAgger

1. Collect trajectories from expert $\pi^{ref}$
2. Dataset $D_0 = \{ (o, \pi^{ref}(o,y)) | o \sim \pi^{ref} \}$
3. Train $\pi_1$ on $D_0$
4. Collect new trajectories from $\pi_1$
   ➢ But let the expert steer!
5. Dataset $D_1 = \{ (o, \pi^{ref}(o,y)) | o \sim \pi_1 \}$
6. Train $\pi_2$ on $D_0 \cup D_1$

● In general:
   ● $D_n = \{ (o, \pi^{ref}(o,y)) | o \sim \pi_n \}$
   ● Train $\pi_{n+1}$ on $\bigcup_{i \leq n} D_i$

If $N = T \log T$, $L(\pi_n) < T \mathbb{E}_N + O(1)$ for some $n$
Test - time execution (DAGGER)

Video credit: Stéphane Ross, Geoff Gordon and Drew Bagnell
What's the biggest failure mode?

- Classifier only sees right versus not-right
  - No notion of better or worse
  - No partial credit
  - Must have a single target answer

\[ \pi \neq \pi_1 \neq \pi_2 \neq \pi^* \]
Classifier: $h : x \rightarrow [K]$

- $(x,y) \in X \times [K]$
  - $\min_h \Pr( h(x) \neq y )$

- $(x,c) \in X \times [0,\infty)^K$
  - $\min_h \mathbb{E}_{(x,c)}[ c_{h(x)} ]$
Classifier: $h : x \rightarrow [K]$

- $(x,c) \in X \times [0,\infty)^K$
- $\min_h E_{(x,c)} [ c_{h(x)} ]$

Solution learn a K-dimensional regressor on costs; pick minimal cost
1. Let learned policy \( \pi \) drive for \( t \) timesteps to obs. \( o \)

2. For each possible action \( a \):
   - Take action \( a \), and let expert \( \pi^{\text{ref}} \) drive the rest
   - Record the overall loss, \( c_a \)

3. Update \( \pi \) based on example:
   \[(o, \langle c_1, c_2, ..., c_K \rangle)\]

4. Goto (1)
● From demonstrations $\rightarrow$ expert decisions

● From expert decisions $\rightarrow$ expert costs
what is the expert's API?

- Behavioral cloning
  - Input:
  - Output:
- Dagger
  - Input:
  - Output:
- Aggrevate
  - Input:
  - Output:

observation
optimal(ish) action
where does an expert come from?

Option 1: An actual real life human being
Option 2: Simulation
The (expected) minimum achievable loss is:

$$\min_{(a_t, a_{t+1}, \ldots)} E \text{loss}(a)$$

optimal action

optimal Q values

$a_t$
Rollout to some depth → Intermediate loss → Return best start
e.g., Monte Carlo Tree Search

Image credit: Michele Sebag and DeepMind
also effective for structured prediction

Image credit: Klein et al., 2017
How well does this strategy work?

Ross + Bagnell, AIStats'10

Chang + D + He + Langford + Ross, NIPS'16

Captioning

Parsing

ASR

Bengio + Vinyals + Navdeep + Shazeer, NIPS'16
● From demonstrations → expert decisions
● From expert decisions → expert costs
● Whence the expert?
combining experts & rewards

- General solution... joint loss: optimize $E[reward] + z E[imitation loss]$

- Big question: how to set $z$?
1. Let learned policy $\pi^{in}$ drive for $t$ timesteps to obs. $o$
2. For each possible action $a$:
   - Take action $a$, and let something $\pi^{out}$ drive the rest
   - Record the overall loss, $c_a$
3. Update $\pi$ based on example: $(o, \langle c_1, c_2, ..., c_K \rangle)$
4. Goto (1)
<table>
<thead>
<tr>
<th></th>
<th>Reference</th>
<th>Mixture</th>
<th>Learned</th>
</tr>
</thead>
<tbody>
<tr>
<td>roll-out →</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>roll-in</td>
<td>Inconsistent</td>
<td>Good</td>
<td>RL</td>
</tr>
<tr>
<td>Reference</td>
<td>Not locally opt.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Learned</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In "Good" setting, can prove that:

- If ref is optimal, will compete with ref
- If ref is suboptimal, either:
  - Improve upon ref
  - Achieve approximate local optimality

Furthermore, whp:

\[ \text{Regret} = O \left( (KT)^{2/3} \sqrt{\frac{\log(N|\Pi|)}{N}} + T \delta_{\text{class}} \right) \]
Key insight: verifying if low-level trajectory is successful is cheaper than labeling low-level trajectory

- labeling effort = high-level horizon + low-level horizon only a fraction of the full horizon (as low as sqrt of the full horizon)

- subpolicies are only learnt in the relevant part of the state space
Dagger: A reduction of imitation and structured prediction to no-regret online learning; Ross, Gordon, Bagnell; ICML 2011

Aggrevate: Ross & Bagnell, Reinforcement and Imitation Learning via No-Regret Learning; Arxiv 2013

LOLS: Chang, Krishnamurthy, Agarwal, D, Langford; Learning to search better than your teacher; ICML 2015 (Follow-up by D, L, Sharaf, ICLR 2018)

MCTree: Browne et al., A Survey of MC Tree Search Methods, 2012

Hierarchies: Le, Jiang, …, Hierarchical Imitation and Reinforcement Learning; ICML 2018
imitation learning summary

successes:

- if all you have are demonstrations, life can be difficult
- if you have expert, iterate to get right state distribution
- experts come from people or planning
- several ways to combine experts and reinforcement

open problems:

- what is the right way to incorporate experts?
- can we learn from observations of experts?
- how to reduce # of expert samples needed?

Thank you! Queries!