Computer Engineering Department

Imitation Learning

Mohammad Hossein Rohban, Ph.D.

Spring 2024

Most slides are adopted from CMU 10703, CS285 Berkeley, and Yue & Le's tutorial at ICML 2018

So far in the course …

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 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

Imitation Learning: Learning from demonstrations

Given: demonstrations or demonstrator

Goal: train a policy to mimic demonstrations

General Imitation Learning:

$$\underset{\theta}{\operatorname{argmin}} \; \mathbb{E}_{s \sim P(s|\pi_\theta)} \left[\mathcal{L}\left(\pi^*(s), \pi_\theta(s)\right) \right]$$

- State distribution $P(s|\pi_\theta)$ depends on rollout determined by current policy π_θ

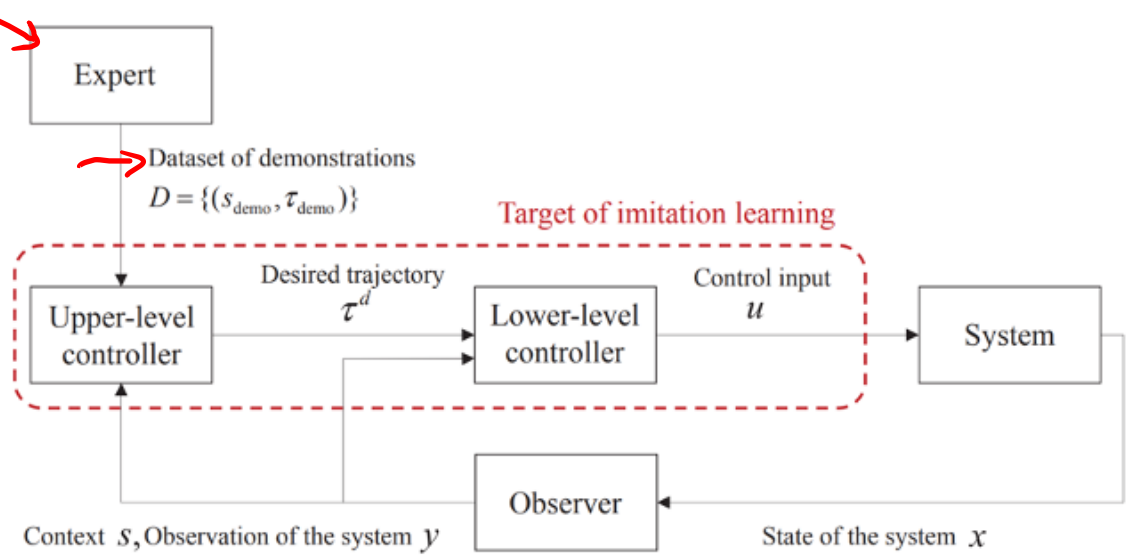


Figure 3.1: Control diagram of a robotic system with imitation learning. An expert demonstrates the desired behavior generating a dataset D. Based on D and observations about the current context and system state an upper-level controller generates the desired trajectory τ^d. A lower-level feedback controller tries to follow τ^d using observation feedback to generate a control u which causes a change to the system state x and a new observation. In imitation learning, the controllers are tuned to imitate the expert demonstrations.

Image: An Algorithmic Perspective on Imitation Learning, Osa et al. 2018

Use cases

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

- coming up with a reward function that would generate such behavior
- coding up with the desired policy directly, and the sample complexity is manageable

vs. Supervised Learning

vs. Offline Reinforcement Learning

$\pi \neq \pi_E$

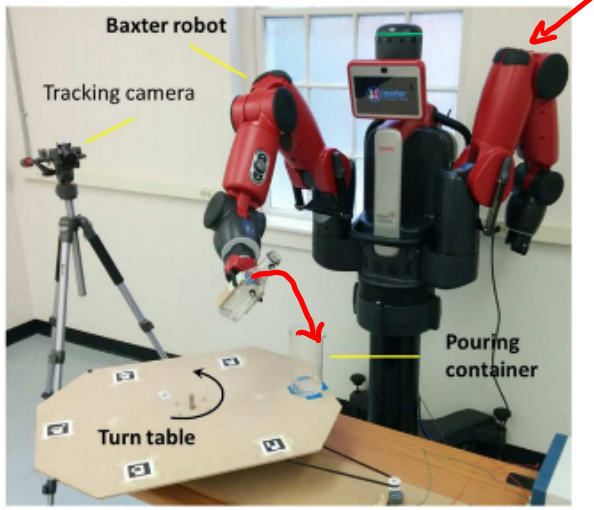
Examples & demos

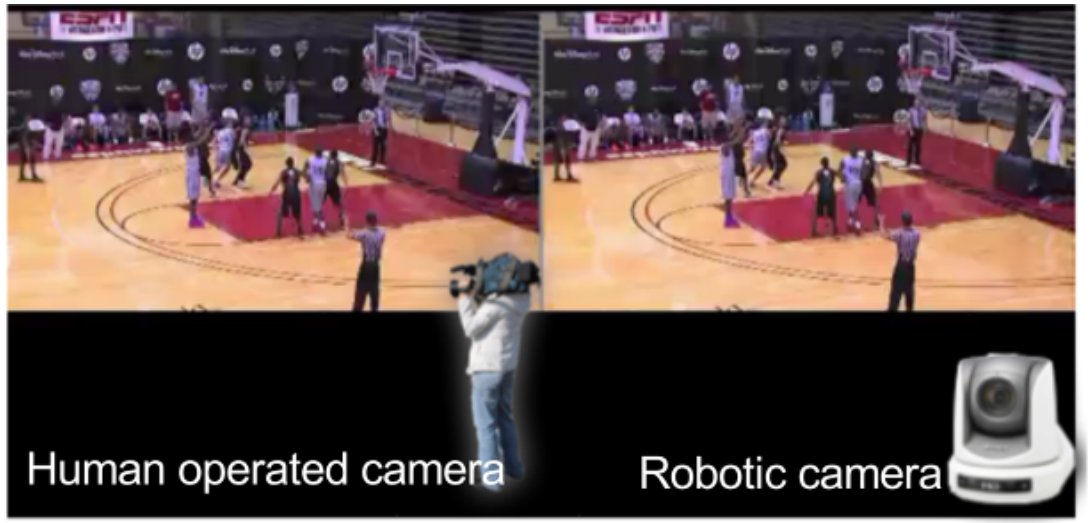
$D = \{ \tau_i = \}$

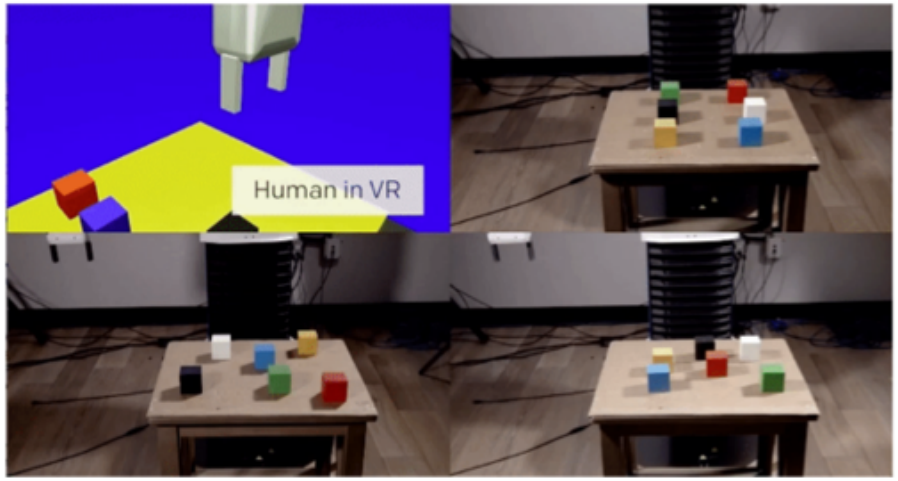
(s_j, a_j)

sensors

Recorded actions of expert







Examples & demos

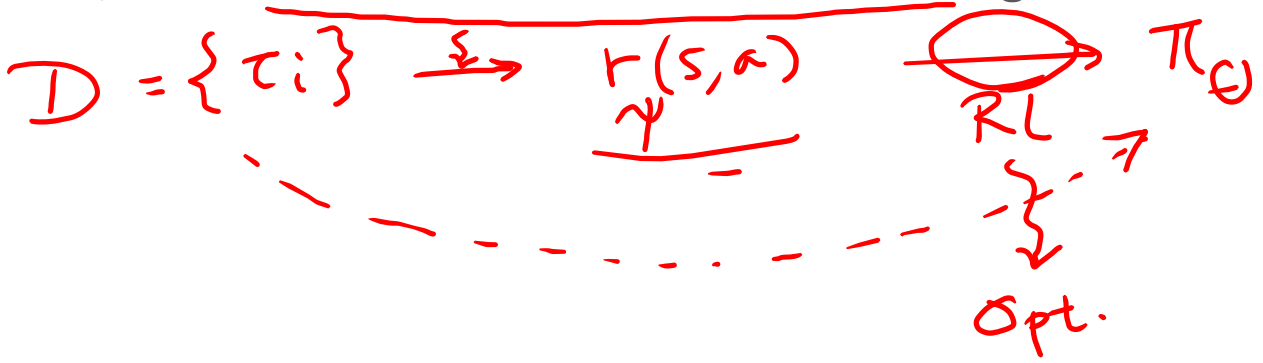
A Deep Learning Approach for Generalized Speech Animation, Taylor et al., SIGGRAPH 2017

Imitation Learning methods

Two broad approaches:

$\pi_\theta \approx \pi^*$ or π_E

- ***Direct:*** Supervised training of policy (mapping states to actions) using the demonstration trajectories as groundtruth (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

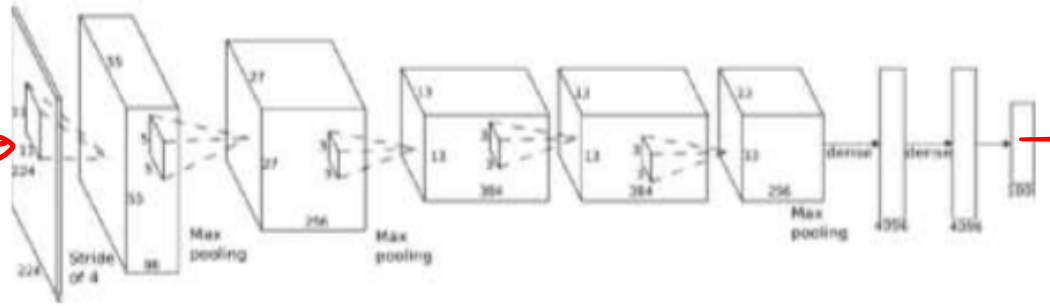


A Deep Learning Approach for Generalized Speech Animation, Taylor et al., SIGGRAPH 2017

Direct Imitation Learning

Behavioral Cloning

$\mathbf{o}_t$

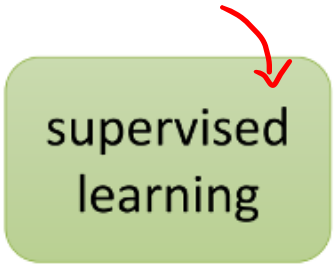


$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

$\mathbf{a}_t$

$\mathbf{o}_t$

$\mathbf{a}_t$



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

Images: Bojarski et al. '16, NVIDIA

Behavioral Cloning

$$\underset{\theta}{\operatorname{argmin}} \underbrace{\mathbb{E}_{(s^*,a^*)\sim P^*}\left[\mathcal{L}\left(a^*, \pi_\theta(s^*)\right)\right]}_{=\sum_{i=1}^{N} \mathcal{L}\left(a_i^*, \pi_\theta(s_i^*)\right)}$$

- State distribution P^* provided by expert
- Reduces to supervised learning problem

Algorithm 1 Abstract of behavioral cloning

Collect a set of trajectories demonstrated by the expert $\mathcal{D}$
Select a policy representation $\pi_{\boldsymbol{\theta}}$
Select an objective function $\mathcal{L}$
Optimize $\mathcal{L}$ w.r.t. the policy parameter $\boldsymbol{\theta}$ using $\mathcal{D}$
return optimized policy parameters $\boldsymbol{\theta}$

Images: Bojarski et al. '16, NVIDIA

ALVINN: An Autonomous Land Vehicle In a Neural Network

Real Road Image Simulated Road Image

Figure 2: Real and simulated road images

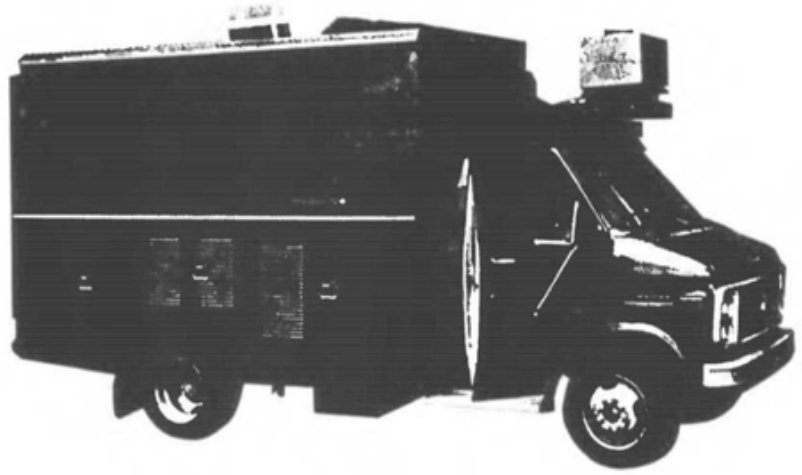


Figure 3: NAVLAB, the CMU autonomous navigation test vehicle.

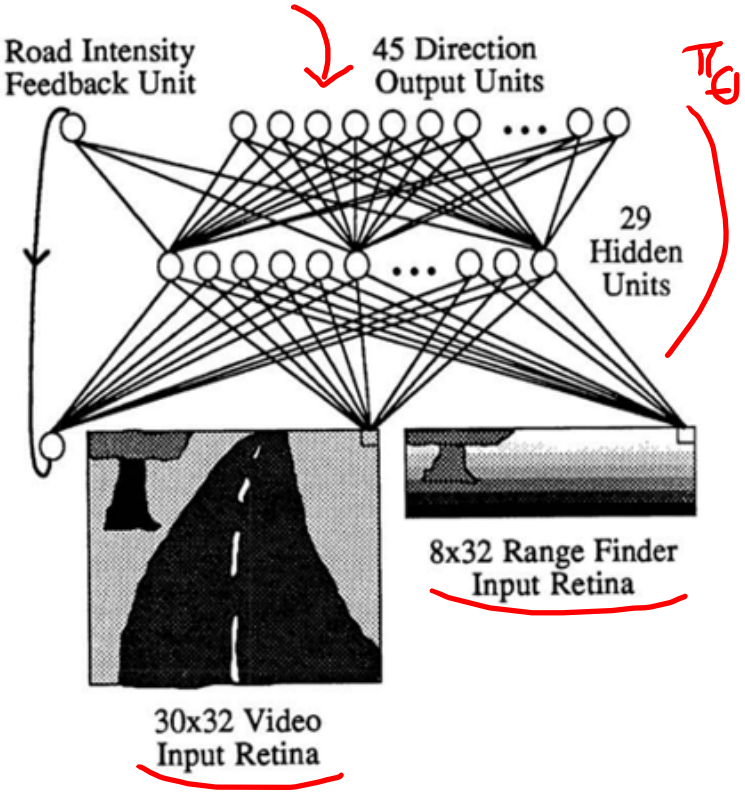


Figure 1: ALVINN Architecture

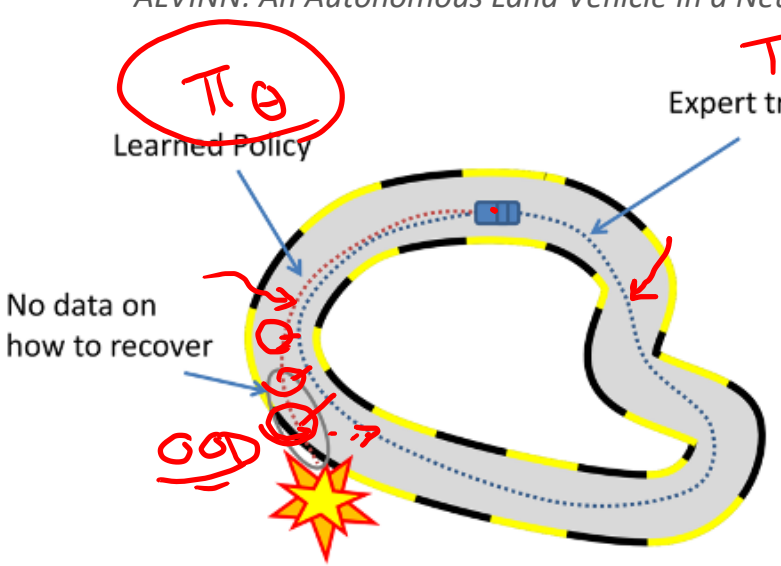
After 40 epochs of training on the 1200 simulated road snapshots, the network correctly dictates a turn curvature within two units of the correct answer approximately 90% of the time on novel simulated road images. The primary testing of the ALVINN's performance has been conducted on the NAVLAB (See Figure 3). The NAVLAB is a modified Chevy van equipped with 3 Sun computers, a Warp, a video camera, and a laser range finder, which serves as a testbed for the CMU autonomous land vehicle project [Thorpe et al., 1987]. Performance of the network to date is comparable to that achieved by the best traditional vision-based autonomous navigation algorithm at CMU under the limited conditions tested. Specifically, the network can accurately drive the NAVLAB at a speed of 1/2 meter per second along a 400 meter path through a wooded

ALVINN: An autonomous land vehicle in a neural network, Pomerleau, 1988

Policy Mismatch Problem

"In addition, the network must not solely be shown examples of accurate driving, but also how to recover (i.e. return to the road center) once a mistake has been made. Partial initial training on a variety of simulated road images should help eliminate these difficulties and facilitate better performance"

ALVINN: An Autonomous Land Vehicle In a Neural Network, [Pomerleau 1989]



Demonstration Augmentation: NVIDIA 2016

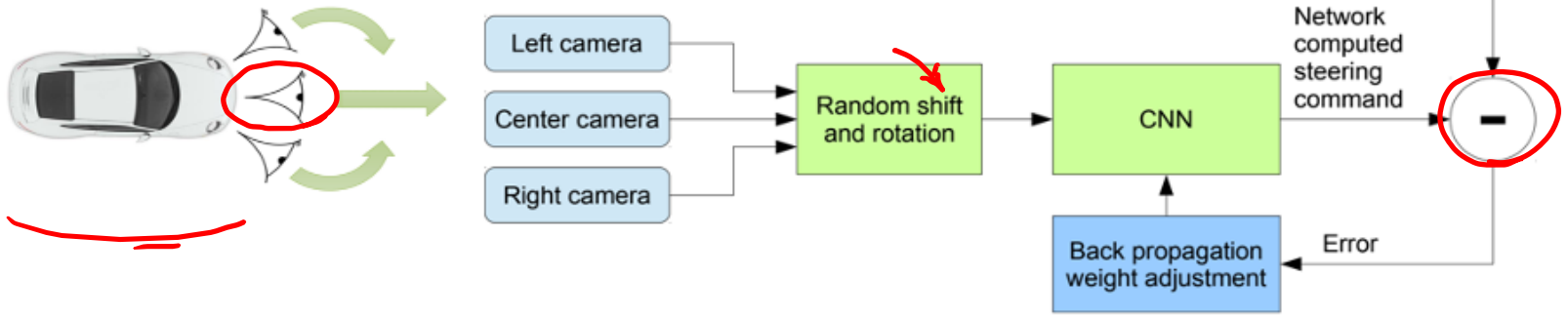
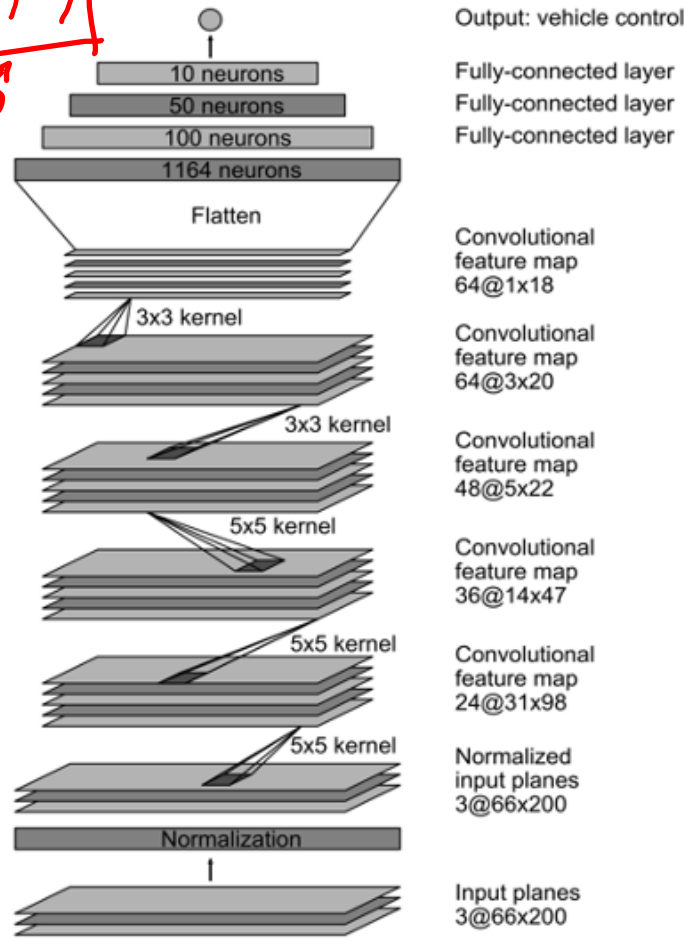


Figure 2: Training the neural network.



"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

Challenges of Behavior Cloning

- Behavior cloning makes i.i.d. assumptions
  - Next state is sampled from states observed during expert demonstration
  - Thus, next state is sampled independently from action predicted by current policy
- What if π_θ makes a mistake?
  - Enters new states that haven't been observed before
  - New states not sampled from same (expert) distribution anymore
  - Cannot recover, catastrophic failure in the worst case
- What can we do to overcome this train/test distribution mismatch?

When to use Behavioral Cloning?

Advantages

- Simple
- Simple
- Efficient

Use When:

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

Disadvantages

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

Use DAgger instead!

Don't Use When:

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

DAgger

Algorithm 3 DAGGER [Ross et al., 2011]

Input: initial dataset of demonstrations $\mathcal{D} = \{(\boldsymbol{x}, \boldsymbol{u})\}$, $\{\beta_i\}$ such that $\frac{1}{N}\sum_{i=1}^{N} \beta_i \to 0$

Initialize: π_1^{L}

for $i = 1$ **to** N **do**

Let $\pi_i = \beta_i \pi^{\mathrm{E}} + (1 - \beta_i)\pi_i^{\mathrm{L}}$.

Sample trajectories $\boldsymbol{\tau} = [\boldsymbol{x}_0, \boldsymbol{u}_0, ..., \boldsymbol{x}_T, \boldsymbol{u}_T]$ using π_i

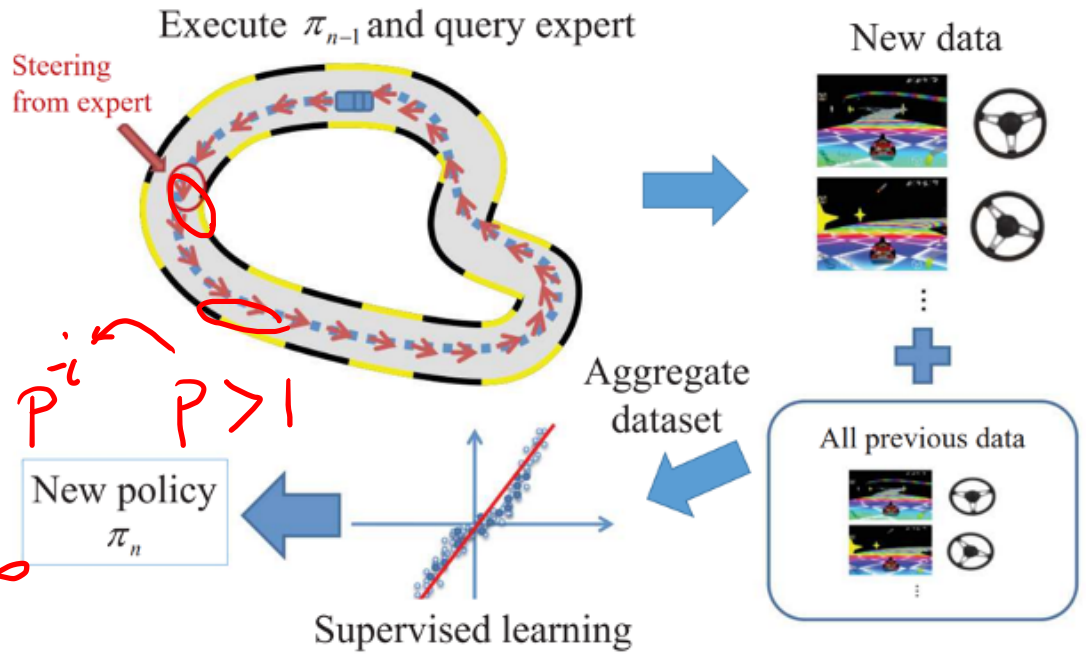
Get dataset $\mathcal{D}_i$ of visited states by π_i and actions given by expert.

Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$

Train the policy π_{i+1}^{L} on $\mathcal{D}$.

end for

return best π_i^{L} on validation.



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

Image: An invitation to imitation, Bagnell, 2015

DAgger

Algorithm 3 DAGGER [Ross et al., 2011]

Input: initial dataset of demonstrations $\mathcal{D} = \{(\boldsymbol{x}, \boldsymbol{u})\}$, $\{\beta_i\}$ such that $\frac{1}{N}\sum_{i=1}^{N}\beta_i \to 0$

Initialize: π_1^{L}

for $i = 1$ **to** N **do**

Let $\pi_i = \beta_i \pi^{\mathrm{E}} + (1-\beta_i)\pi_i^{\mathrm{L}}$.

Sample trajectories $\boldsymbol{\tau} = [\boldsymbol{x}_0, \boldsymbol{u}_0, ..., \boldsymbol{x}_T, \boldsymbol{u}_T]$ using π_i

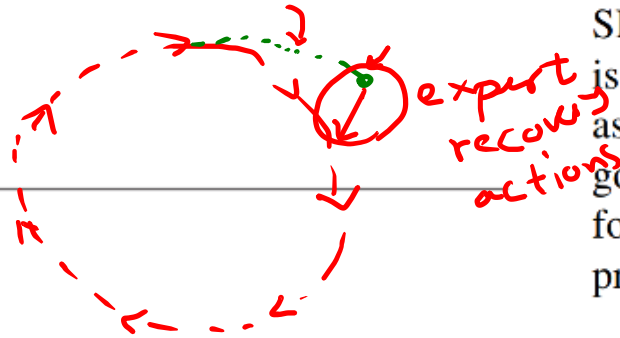
Get dataset $\mathcal{D}_i$ of visited states by π_i and actions given by expert.

Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$

Train the policy π_{i+1}^{L} on $\mathcal{D}$.

end for

return best π_i^{L} on validation.



To better leverage the presence of the expert in our imitation learning setting, we optionally allow the algorithm to use a modified policy $\pi_i = \beta_i \pi^* + (1-\beta_i)\hat{\pi}_i$ at iteration i that queries the expert to choose controls a fraction of the time while collecting the next dataset. This is often desirable in practice as the first few policies, with relatively few datapoints, may make many more mistakes and visit states that are irrelevant as the policy improves.

We will typically use $\beta_1 = 1$ so that we do not have to specify an initial policy $\hat{\pi}_1$ before getting data from the expert's behavior. Then we could choose $\beta_i = p^{i-1}$ to have a probability of using the expert that decays exponentially as in SMILe and SEARN. We show below the only requirement is that $\{\beta_i\}$ be a sequence such that $\overline{\beta}_N = \frac{1}{N}\sum_{i=1}^{N}\beta_i \to 0$ as $N \to \infty$. The simple, parameter-free version of the algorithm described above is the special case $\beta_i = I(i=1)$ for I the indicator function, which often performs best in practice (see Section 5). The general DAGGER algorithm is

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

DAgger caveats

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

π_L or π_θ

$\pi_\theta \neq \pi_E$ or π^*

Indirect Imitation Learning

Indirect Imitation Learning

Approaches that learn policies to imitate expert actions can be limited by several factors:

- Behavior cloning provides no way to understand the underlying reasons for the expert behavior (no reasoning about outcomes or intentions).
- The “expert” may actually be suboptimal.
- A policy that is optimal for the expert may not be optimal for the agent if they have different dynamics, morphologies, or capabilities.

An alternative approach: Learn expert’s intentions!

- **Inverse Reinforcement Learning:** Can we discover the reward function?
- **Apprenticeship Learning:** Can we then use the discovered reward to learn the optimal policy?

Example 10.4.1 (Apprenticeship Learning vs. Behavioral Cloning)**.** Consider a problem where the goal is to drive a car across a city in as short of time as possible. In the imitation learning formulation it is assumed that the reward function is not known, but that there is an expert who shows how to drive across the city (i.e. what routes to take). A behavioral cloning approach would simply try to mimic the actions taken by the expert, such as memorizing that whenever the agent is at a particular intersection it should turn right. Of course this approach is not robust when at intersections that the expert never visited!

The apprenticeship learning approach tries to avoid the inefficiency of behavioral cloning by instead identifying features of the expert's trajectories that are more generalizable, and developing a policy that experiences the same feature expectations as the expert. For example it could be more efficient to notice that the expert takes routes without stop signs, or routes with higher speed limits, and then try to find policies that also seek out those features!