Imitation learning

Imitation learning vs. planning vs. reinforcement learning

- Planning:
  - We have a goal state s_n
  - We know the model $T(s, a, s')$
  - We create a plan of actions $a_1, a_2, a_3, \ldots a_n$
- Reinforcement learning:
  - We can get samples of s, a, r, s' either from our own experience or observing somebody else
  - We are searching for a policy $\pi^*(s)$ that maximizes utility (roughly, rewards received)

Imitation learning vs. planning vs. reinforcement learning

- Imitation learning
  - We have **demonstrations** of the form:
    - $s_1, a_1, s_2, a_2, s_3, a_3 \ldots$
  - We don't have rewards. We don't necessarily know the goal state.
  - We vaguely assume that whomever did the demonstrations mostly knew what they were doing
  - But we do not assume that they were optimal.
  - We are searching for a policy $\pi^*(s)$ that has the same goals of the demonstrator.

NOTE: This is my (Lotzi's) definition.

What people believe about imitation learning

- Just replay $a_1, a_2, a_3 \ldots$
- That is not learning, that is replay.
- Might be useful in high precision industrial robotics
  - e.g. paint all the cars the same way.
  - it has nothing to do with AI
- The term "learning from demonstrations" might be more accurate, but it is used less often.
- **The challenge**: it is unlikely that you will see exactly the states in the demonstrations again. And even if you see them, the randomness in the transition function might land you in a different state afterwards!
  - Replay won't work!

Two approaches to imitation learning

- **Behavior cloning**
  - Assume that the demos were done by an agent using an unknown policy $\pi_{demo}(s)$
  - Learn a policy $\pi \approx \pi_{demo}$ using the demonstrations as training data.
- **Inverse reinforcement learning**
  - Assume that the demos were done by an agent pursuing a certain set of unknown rewards $r_{demo}(s, a, s')$
  - Reverse engineer a $r(s, a, s') \approx r_{demo}(s, a, s')$
  - Use RL to find a π that maximizes the rewards in r

Behavior cloning

Behavior cloning

- Assume an underlying MDP $M = \{S, A, T, R, \gamma\}$. Unknown T and R.
- Let us assume that the expert has a (nearly) optimal policy π^*
- We will denote with d^π the *state visitation frequency* implied by a policy π
- Demonstrations are samples drawn from the state visitation frequency of the optimal policy

$$\mathcal{D} = (s_i^*, a_i^*)_{i=1}^{M} \sim d^{\pi^*}$$

- Goal is to learn a policy π_{BC} that is as good as the expert π^*

Behavior cloning (cont'd)

- Let us assume that we are choosing our policies from a certain parameterized policy class $\pi_{BC} \in \Pi$
  - These days, this usually means that it is a neural network with weights $\mathbf{w}$
- Behavior cloning is essentially supervised learning

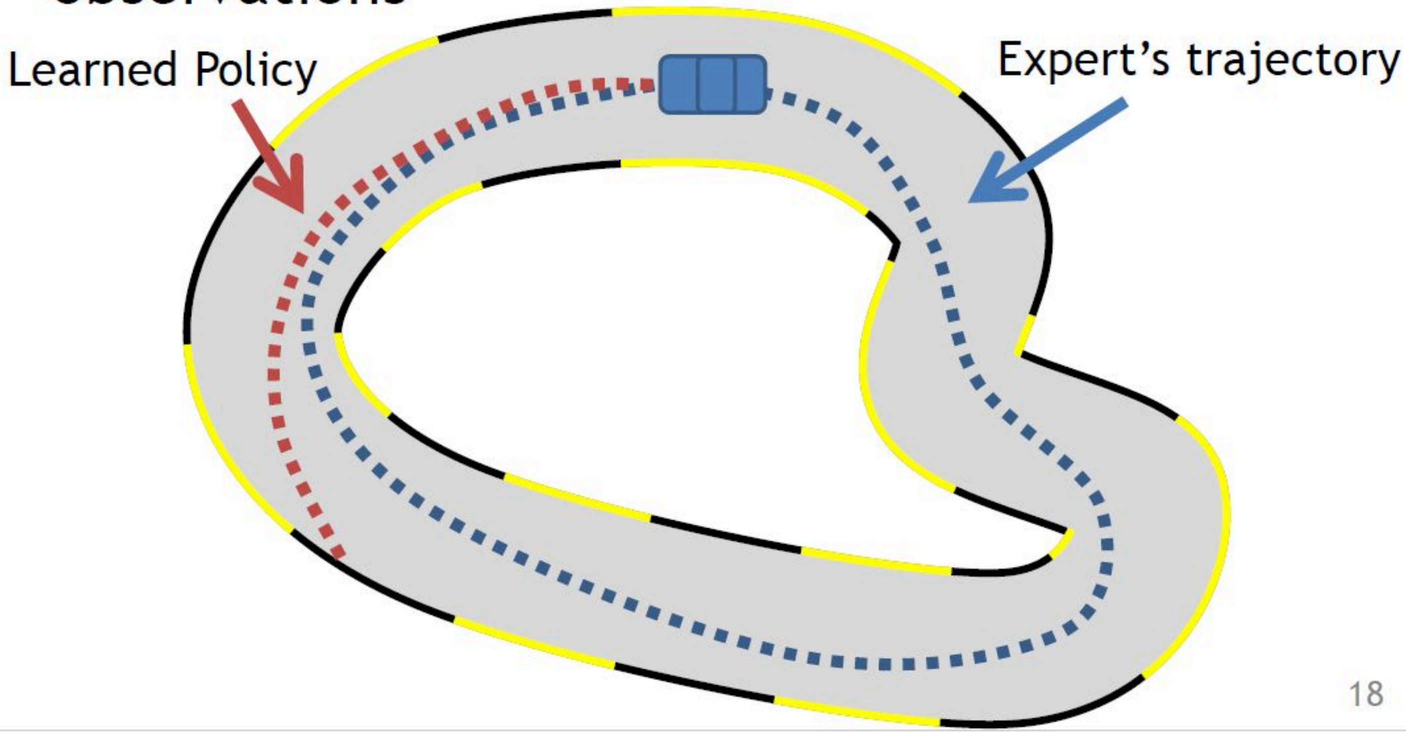
$$\hat{\pi} = arg \min_{\pi \in \Pi} \sum_{i=1}^{M} \mathcal{L}(\pi, s_i^*, a_i^*)$$

- There are many choices the loss function L can take:
  - Negative log-likelihood $\mathcal{L}(\pi, s_i^*, a_i^*) = -ln\, \pi(a^*|s^*)$
  - Square loss (if a is a continuous signal like steering angle) $\mathcal{L}(\pi, s_i^*, a_i^*) = ||\, \pi(s) - a^*||_2^2$

What could go wrong?

[Pomerleau89,Daume09]

- Predictions affect future inputs/ observations



18

Distribution shift

- Here is the general argument
    - Assume perfect demonstrations which tell you what to do in a set of states s
        - eg. keeping the car in the middle of the road
    - Some unexpected thing will happen, which gets you into a state s' from which you don't have information in demonstrations
        - eg. drifted from the middle of the road
    - The farther you are from the demonstrations, the worse your policy
        - so, you will wear more and more off the road

Making behavior cloning work

- Many of these arguments turned out to have too many simplifying assumptions.
  - The problem of distribution shift was presented as a fundamental problem that dooms BC in general cases
- But people learn from demonstrations!
- Couple of ways forward:
  - Maybe the imitation happens in a favorable latent space
  - Maybe you also have imperfect demonstrations and demonstrations of self-correction
  - Maybe there is an underlying policy of getting back to known states

Inverse reinforcement learning

Inverse reinforcement learning

- Assume an underlying MDP $M = \{S, A, T, R, \gamma\}$. Unknown R. Usually, known T.
- Let us assume that the expert has a (nearly) optimal policy π^*
- Demonstrations are samples drawn from the state visitation frequency of the optimal policy

$$\mathcal{D} = (s_i^*, a_i^*)_{i=1}^{M} \sim d^{\pi^*}$$

- The setting is almost the same as behavior cloning.

Inverse RL

- We assume that the expert is optimizing some kind of reward R
- Our goal is to reverse engineer the reward
- Then we can create a policy π_{IRL} by solving the MDP
- Possible benefits (over behavior cloning)
  - The rewards seem to better capture the meaning of the action (compared to cloning the behavior)
  - We can, possibly, transfer the reward structure to a completely different MDP, with different transitions etc.
  - We can perform better than the expert if we manage to optimize better for the same reward!

Inverse RL

- Challenge: the actions of an expert do not uniquely define the rewards it follows
    - Eg. scaling...
    - But those different reward functions might generate different policies...
- One possible solution: **maximum entropy IRL**
    - From all the reward functions that explain the observed behaviors, choose the one that maximizes the entropy over the distribution of possible behaviors
    - Assume an expert that is as random as possible given the observed data
    - The goal is to avoid introducing additional biases, or to overfit to the training data.

Video time:

- Autonomous Helicopters Teach Themselves to Fly Stunts
  - https://www.youtube.com/watch?v=M-QUkgk3HyE
- Learning real manipulation tasks from virtual demonstrations using LSTM
  - https://www.youtube.com/watch?v=9vYlIG2ozaM