Policy Learning

Imitation Learning

collecting demonstrations

1. teleoperation
   - also suffers from real-real gap
2. Remote Teleoperation
   - human operate virtual environment
   - can leverage scaling up ability of synthetic environment
3. synthetic data

behavior cloning

1. method
   - collect expert demonstrations
   - imitate the expert's behavior by training a neural network to predict the expert's actions given the expert's observations
2. does it work?
   - no, because of distribution drift
3. how to make it work?
   i. make training data more diverse, covering more status that might be seen during test
   ii. make the network better fit the experts' behavior by using more complex models

DAgger

1. target: we wish

$$p_{data}(o_t) = p_{\pi_\theta}(o_t)$$

   or we want to see more states in training that may be occured during test.
2. methods:
   i. train $\pi_\theta(a_t|o_t)$ on expert demonstrations $D = \{(o_i, a_i)_{i=1}^N\}$
   ii. run policy $\pi_\theta(a_t|o_t)$ in the real environment and collect data $D_\pi = \{o_1, \ldots, o_M\}$
   iii. ask human to label D_π with actions a_t
   iv. Aggregate $D \leftarrow D \cup D_\pi$
   v. go back to step 1

how to provide human label in step 3?

1. from optimal solution: for example, searching algorithm or motion planning.

NOTE: why do we need learning if we already have existing algorithms? Because these algorithms usually requires oracle knowledge, which is not available for a single observation.

2. from a teacher policy: we have a teacher policy to provide the optimal action under a state.

NOTE: what's the difference between teacher policy and expert demonstration? Teacher policy is a online policy, it usually uses more information than the real policy to decide on the optimal action.

Better Fitting

1. motivation: we don't want to collect and label so many data like DAgger, we want out model to mimic the expert very accurately without overfitting.
2. why will we fail to fit the expert?

Non-Markovian behavior

1. human decisions usually depends on the current state and history information.
2. can we simply provide the history information to the model, and use LSTM to address this problem?
   - this might work poorly. The more history you provide, the more possible that the model will find some special clues that helps it jumps to the decision, which leads to causal confusion.

Multimodal behavior

This problem derives from the fact that we can have multiple valid actions for a state, like turn right or turn left to avoid an obstacle. Output a single action for a unimodal distribution might not be enough to capture the expert's behavior.

There are several ways to mitigate this problem:

1. Ouput mixture of Guassians

then the problem will be how many Guassians should we predict?

2. Latent variable model

- Conditional variational autoencoder (CVAE)
- Normalizing flow/realNVP
- Stein Variational Gradient Descent (SVGD)

3. Diffusion models

4. Autoregressive discretization

For hign dimension problems, we discretize one dimension at a time. Then we use LSTM or transformer, taking current state and decisions on previous action dimensions as input to predict the next action dimension.

For an action of 3-dimention:

$$\begin{aligned} firststep &: \pi(a_{t,1}|o_t) && (1) \\ secondstep &: \pi(a_{t,2}|o_t, a_{t,1}) && (2) \\ thirdstep &: \pi(a_{t,3}|o_t, a_{t,1}, a_{t,2}) && (3) \end{aligned}$$

Multi-task learning

Also known as Goal-conditioned behavioral cloning. We are trying to learn $\pi_\theta(a_t|s_t, g)$, where s_t is the state and g is the goal.

- motivations:
  - more states can be seen because demostrations have different goals
  - some tracks are similar between different goals, we can learn to imitate them better
- problems:
  - we see two distribution shift here:
    a. distribution shift between the goal in training and the goal in test
    b. distribution shift between the state in training and the state in test

Summary: Imitation learning

- Often insufficient by itself
  - due to distribution mismatch problem
- Sometimes works well
  - using some hacks: auto-driving probably use three cameras to capture different views of the environment, if the front camera sees the image that should be seen by the left camera, it is heuristically decided that the car should turn right.
  - sample from a stable trajectory distribution
  - add more on-policy data, like DAgger
  - Better models that fit more accurately

Reinforcement Learning

- features of RL:
  i. Data is not i.i.d.: previous outputs influence future inputs
  ii. Ground truth answer is not known, not known is we succeed or fail
    - more general, we know the reward