Imitation Learning

Lain Mustafaoglu

Problem

How can we find optimal policies for sequential decision-making?

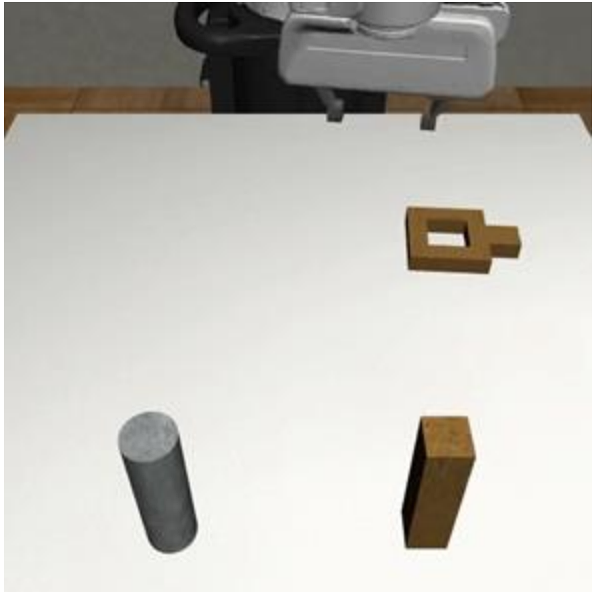
One class of methods: RL **(policy gradient methods)**

- Parameterize policies with policy network Θ and maximize the expected return – **policy objective function J(Θ)** – through direct policy optimization to find **π***
- **Training policy networks:** Sample multi-hop trajectories for policy π, optimize policy, recollect samples with new policy and repeat
- **Challenge:** Reward modeling in RL

*What if we mimic **expert behavior** instead of modeling the reward explicitly?*

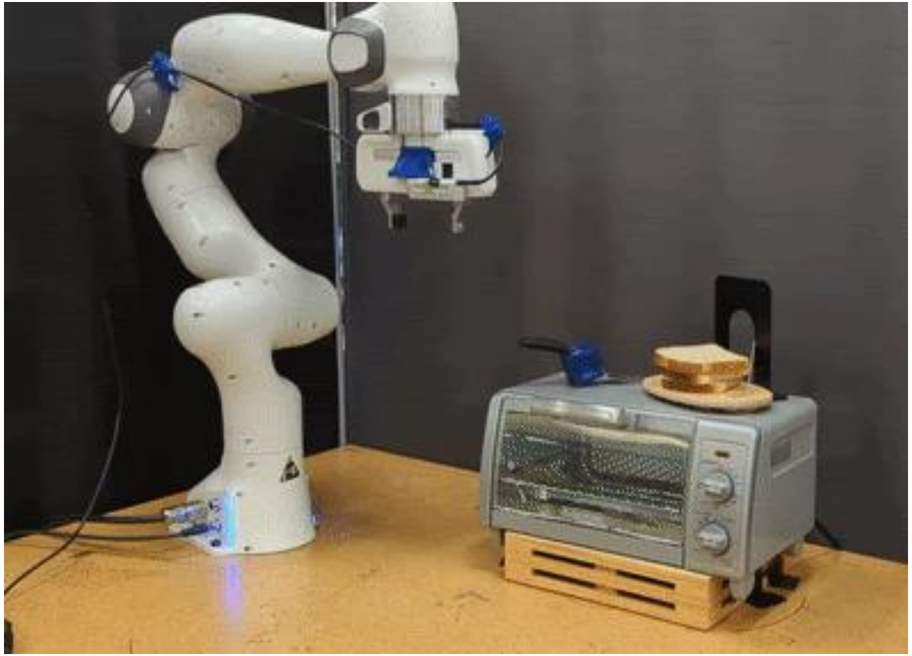
→ **Need expert demonstrations!**

IL Demonstrations in Simulation


What Matters in Learning from Offline Human Demonstrations for Robot Manipulation

IL Demonstrations on Real-World Robots



HYDRA: Hybrid Robot Actions for Imitation Learning. Belkhale et al. 2023.

Imitation Learning

- Reward function not known a priori but described implicitly through **expert demonstrations**
  - **Goal:** use demonstrations to mimic expert behavior
  - **Key difference from RL:** IL copies the expert, RL learns actions with high rewards

- **Demonstrations:** sequence of state-control (action) pairs

$$D = \{(s_i, a_i)\}_{i=1}^{N}$$

- **Objective:**

$$\theta^* = \arg\min_{\theta} \sum_{i} \ell(\pi_\theta(a_i \mid s_i), a_i)$$

Online vs. Offline Training in IL and RL

Online data –> environment interactions during training

Offline data –> static dataset of demonstrations collected prior to training

- There are online vs offline methods for both IL and RL

IL Approaches

Imitation Learning

- **Online IL**: Expert demonstrations are augmented with real-time expert interactions for dynamic error correction (e.g. **DAgger**)
- **Offline IL**: The agent only has access to a fixed dataset of demonstrations, learning passively (e.g. **Behavior Cloning**)
- **Inverse RL**: Infer the reward function from expert demonstrations instead of manually designing rewards

Limitations of IL methods:

- Only as good as experts
- Distribution shift issue

Behavior Cloning (BC)

Goal: Learn a policy $\pi_\theta(a \mid s)$ that copies an expert's behavior

- Given expert demonstrations:

$$D = \{(s_i, a_i)\}_{i=1}^{N}$$

- Train supervised learning model:

$$\theta^* = \arg\min_\theta \sum_i \ell(\pi_\theta(a_i \mid s_i), a_i)$$

- BC reduces IL to supervised learning
  - BC policies do not learn a performance measure

Issue:
Distribution shift → agent encounters unseen states → errors compound!

Compounding Errors in BC

BC only learns from expert data, but errors compound:

- **Training:** BC trains on expert data: it sees only the “good” trajectories
  - Assumption: The agent will always stay in these states
- **Inference:** No expert data for unseen states → agent guesses randomly → mistakes compound → agent drifts further from expert’s behavior → catastrophic failure

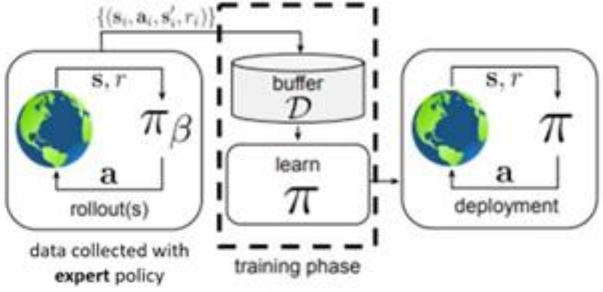
One solution: Online IL (e.g., DAgger) solves this by letting the expert correct mistakes – instead of just mimicking past demonstrations, the agent interacts with the expert during training; expert labels new states the agent visits, so it learns how to recover from mistakes

- **BC:** Trains once on expert data -> no recovery from drift
- **Online IL (e.g. DAgger):** Keeps improving by gathering online expert corrections

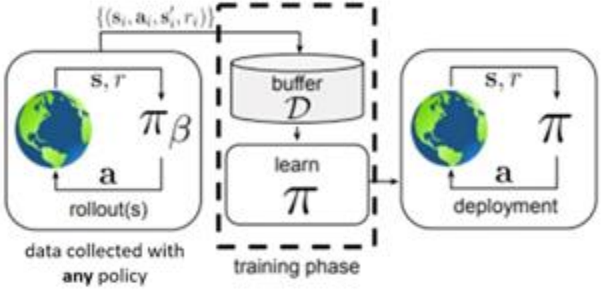
Comparison of IL, RLHF and Offline RL

- **IL:** Mimic expert actions based on demonstrations
- **RLHF:** Obtain reward signal from expert FEEDBACK, not demonstrations
- **Batch (Offline) Reinforcement Learning:** Learn policies directly using a static dataset of past experiences without online interaction
  - Unlike IL, which tries to mimic expert actions, offline RL optimizes rewards from past data (e.g., conservative Q-learning)

Imitation learning:



Offline RL:



Takeaways

- Bootstrapping with demonstrations → sample efficiency (30x sample efficiency with 20 demonstrations in DAPG)

- This is the current paradigm in LLMs as well: **IL + RL**
  - Supervised learning on text data → imitation learning
  - RLHF fine-tuning after SFT → RL