Deep Reinforcement Learning (Sp25)

Instructor: Dr. Mohammad Hossein Rohban

Summary of Lecture 21: Imitation Learning

Summarized By: Amirhossein Asadi

Imitation Learning

- in RL, defining a clear and effective reward function is often complex and non-trivial. This complexity is particularly evident in robotics applications, where specifying a reward that accurately reflects desired behaviors can be challenging.
- Such scenarios often necessitate extensive **reward engineering**, a process that involves designing reward functions to guide the agent's learning process effectively. This aspect can constitute a significant portion of the overall effort in developing RL systems.
- IL enables agents to acquire effective policies by observing **expert demonstrations**, effectively bypassing the need for explicit reward functions. However, designing a suitable reward function remains a significant challenge.
- While IL allows agents to replicate expert behavior, integrating reward shaping can enable agents to surpass the expert's performance. This advancement is achieved by learning the reward function from demonstrations, facilitating the discovery of more optimal policies beyond mere imitation.
- IL setup :

$$\arg\min_{\theta} \mathbb{E}_{s\sim P(s|\pi_\theta)} \left[\mathcal{L}\left(\pi^*(s), \pi_\theta(s)\right)\right] \tag{1}$$

  the objective is to learn a policy π_θ that closely approximates the **expert policy** π^* . The setup involves observing expert demonstrations and using them to guide the learning process. The goal is to minimize the discrepancy between the learned policy and the expert's policy, thereby enabling the agent to perform tasks effectively without explicit reward signals.
- A key point is that we aim to perform this learning in the **state space of the learned policy** π_θ. However, the data available comes from the expert policy π^*, which means we must modify the setup slightly to accommodate this discrepancy.
- $$\begin{cases} \text{Samples are collected from the expert policy } \pi^*, \text{ so IL is not a form of SL.} \\ \text{We have a simulator available, so IL is not offline RL.} \end{cases}$$
- IL encompasses two primary methodologies:
  - **Direct Approach (Behavioral Cloning)**: This method involves supervised training to directly map states to actions by learning from expert demonstrations. It is akin to SL, where the model learns to predict the actions taken by the expert in corresponding states.
  - **Inverse RL (IRL)**: In contrast, IRL focuses on inferring the underlying reward function that the expert is presumed to be optimizing. Once the reward function is learned, RL techniques are employed to derive the optimal policy that maximizes this inferred reward function.
- The general setup for **Behavioral Cloning**:

$$\arg\min_{\theta} \mathbb{E}_{(s^*,a^*)\sim P^*} \left[\mathcal{L}\left(a^*, \pi_\theta(s^*)\right)\right] \tag{2}$$

- In the behavioral cloning setup, the following inequality shows that the data distribution and the learned policy distribution may not match:

$$P_{\text{data}}(s) \neq P_{\pi_\theta}(s)$$

This mismatch causes the model to encounter **Out Of Distribution** data, making the setup fail.

- The first solution is to align the distributions of the learned policy π_θ and the expert policy π^*. This can be done by adjusting the learning process so that the state distribution of the learned policy matches the expert's distribution. The goal is:

$$\pi_\theta \to \pi^*$$

  This approach ensures that the agent learns a policy that behaves similarly to the expert policy by focusing on the same state distribution.

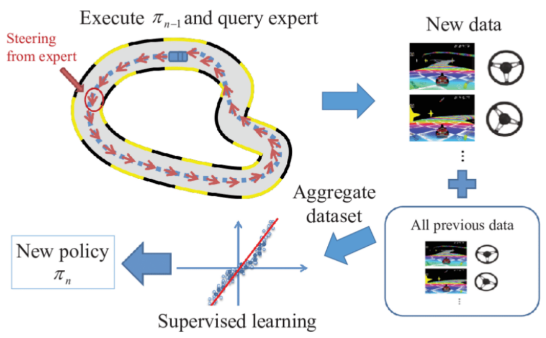
- This alignment is challenging due to several reasons:
  - Humans do not behave in a **Markovian** manner, as their decisions are influenced by past experiences and reasoning. In contrast, expert policies are Markovian, relying solely on the current state for decision-making.
    Solution: consider a sequence of states for π_θ, using models like LSTM, RNN, or Transformer. This approach is why BC involves complex networks, as these models capture temporal dependencies between states.
  - In Gaussian Mixture Models (GMMs), when applying MLE to estimate the mean parameter to estimated mean ends up at the center of the data, the resulting model may exhibit **mean-seeking behavior**.
    Solution: discretizing the actions and use cross-entropy loss. However, this requires the action space dimensionality to be low. If the action space is high, we can apply the chain rule to handle the complexity.

- Despite various approaches, IL may still fail to capture human-like behavior due to the complexity of human decision-making. In such cases, we aim to align the data distribution with the learned policy π_θ, i.e.,

$$P(data) \approx \pi_\theta(s)$$

  This alignment ensures that the agent's behavior closely matches the observed data distribution, addressing the limitations of previous methods.

- In Dataset Aggregation (DAgger), the expert re-labels the states encountered by the current policy. This process involves collecting a set of sample trajectories and having the expert provide the correct labels for the states where the policy exhibits weaknesses. By iteratively adding these corrected labels to the dataset, the agent gradually encounters the state distributions it will experience during deployment, thereby mitigating the distributional shift problem.


Execute π_{n-1} and query expert
Steering from expert
New data
Aggregate dataset
All previous data
New policy π_n
Supervised learning