COS 435/ECE 433 Week 3 Precept Notes

February 20, 2024

1 Introduction to Imitation Learning

Imitation Learning (IL) is a framework where an agent learns to mimic expert behavior, bypassing the need for explicit reward functions typical in Reinforcement Learning (RL). This approach is especially valuable in complex environments where defining a comprehensive reward structure is impractical.

1.1 Formal Framework

In formal terms, IL can be viewed as a supervised learning problem where the objective is to learn a mapping from states $s \in \mathcal{S}$ to actions $a \in \mathcal{A}$, given a dataset of expert demonstrations $\mathcal{D} = \{(s_i, a_i)\}_{i=1}^{N}$. The goal is to minimize the difference between the actions taken by the policy π and the actions demonstrated by the expert, typically measured by a loss function $\mathcal{L}$.

1.2 Advantages and Limitations

While IL simplifies the learning process by using supervised learning techniques, it inherits the limitations of supervised learning, including the need for large, representative datasets of expert behavior and susceptibility to overfitting. Moreover, IL approaches often struggle with the distribution shift problem, where the policy encounters states not present in the training data, leading to compounding errors.

1.3 Detailed Examples

1.3.1 Autonomous Driving Example

Consider an autonomous driving system being trained via behavioral cloning. The expert demonstrations consist of human driving sessions, capturing a wide range of driving decisions in various traffic conditions, weather, and road types. The challenge here is the high-dimensional input space (images or sensor data) and the need to predict steering angles and acceleration commands accurately. Distribution shift becomes apparent when the autonomous vehicle encounters situations not well-represented in the training data, such as road construction or unusual weather conditions, potentially leading to unsafe driving behavior.

1.3.2 Robotics Arm Manipulation Example

Using imitation learning, a robotic arm is trained to perform precise manipulation tasks, such as assembling small components. The expert demonstrations might come from a human operator controlling the arm via a joystick or by manually moving the arm through the desired motions (kinesthetic teaching). Challenges include the fine-grained control required for manipulation and the variability in object sizes, shapes, and

orientations. Techniques such as DAGGER can be particularly useful for refining the robot's control policies by iteratively correcting the arm's actions based on expert feedback.

1.3.3 Game Playing AI Example

An AI trained to play complex video games, such as real-time strategy games, uses imitation learning to mimic the strategies and tactics of expert players. The AI observes game states (e.g., unit positions, resources available) and the corresponding actions taken by the expert (e.g., unit commands, resource allocations). Challenges include the vast array of possible game states and the strategic depth of expert decisions. Behavioral cloning helps the AI learn basic strategies, while methods like DAGGER allow for adjustments based on the AI's unique encounters during gameplay.

2 Behavioral Cloning (BC)

Behavioral Cloning is a form of IL where the agent learns a deterministic policy $\pi_\theta : \mathcal{S} \to \mathcal{A}$, mapping states S to actions A, from expert demonstrations $\mathcal{D} = \{(s_i, a_i)\}_{i=1}^N$.

2.1 Mathematical Formulation

Given a dataset of expert demonstrations $\mathcal{D}$, BC seeks to find the parameters θ of the policy π_θ that minimize the empirical loss:

$$\min_\theta \frac{1}{|\mathcal{D}|} \sum_{(s,a)\in\mathcal{D}} \mathcal{L}(\pi_\theta(s), a),$$

where $\mathcal{L}$ is a loss function, such as mean squared error for continuous actions or cross-entropy for discrete actions.

Variants of Behavioral Cloning Behavioral Cloning (BC) has evolved to include several variants, each addressing specific challenges and applications:

1. Single-shot Behavioral Cloning. The simplest form is where the agent is trained once on a dataset of expert demonstrations. This method is quick but prone to the distribution shift problem.

2. Iterative Behavioral Cloning. An extension that involves retraining the model on a mix of expert demonstrations and the agent's experiences, aiming to reduce the distribution shift.

3. DAGGER (Dataset Aggregation). A sophisticated iterative BC approach where the expert corrects the agent's trajectory by providing the correct actions for states encountered by the policy, effectively mitigating the distribution shift.

4. Filtered Behavioral Cloning. Focuses on selecting the most relevant or challenging expert demonstrations for training based on specific criteria such as state rarity or action diversity.

5. Reweighted Behavioral Cloning. Modify the loss function to emphasize more challenging or informative state-action pairs:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} w_i \cdot l(\pi_\theta(s_i), a_i),$$

where w_i is a weight reflecting the importance of the i-th sample, which could be based on the variance in expert actions or the rarity of the state.

2.2 Implications of Behavioral Cloning

Behavioral Cloning's simplicity and direct approach come with several implications:

- Efficiency and rapid prototyping capabilities are beneficial in the early development stages.
- Challenges with distribution shift, requiring strategies for continuous learning.
- Dependence on the quality and quantity of expert demonstrations.
- Potential struggles with generalization to new tasks or environments.

2.3 Detailed Examples

2.3.1 Autonomous Flying Drones

Drones can be trained via BC to navigate by mimicking expert pilots. Iterative BC refines their ability to handle unexpected scenarios, like avoiding moving obstacles or compensating for wind conditions.

2.3.2 Video Game Non-Player Characters (NPCs)

NPCs in complex video games can be made to exhibit human-like behaviors using BC, enhancing realism. Filtered BC focuses on critical interactions such as combat or decision-making.

2.3.3 Assistive Robotics in Healthcare

Robots in healthcare learn from professionals to assist with daily activities. Reweighted BC prioritizes tasks requiring high precision or direct patient interaction to ensure safety and comfort.

3 BC Hardnesses and Algorithms

3.1 Hardnesses in Behavioral Cloning

Main challenge: Distribution shift The core difficulty in BC arises from the distribution shift problem, as well as from the inherent limitations of supervised learning. Specifically, in BC, the trained policy may perform well on the training distribution but poorly on unseen states. This issue arises because the agent, deviating even slightly from the expert's trajectory, can encounter states that were not covered by the training data, leading to errors that compound over time.

Another Challenge: Multi-modality BC's tendency to average over actions in multi-modal distributions can dilute the effectiveness of the learned policy. This is particularly problematic in environments where different actions can lead to vastly different outcomes. For example, consider a driving simulator where an expert might take sharp turns in certain situations (e.g., avoiding obstacles) and gentle turns in others. In all cases, a BC model might learn to perform moderate turns, failing to react appropriately to critical situations.

3.2 Addressing the Challenges

Several algorithms and techniques have been proposed to address the challenges in BC, including:

- **Dataset Aggregation (DAGGER)**: An iterative algorithm that addresses the distribution shift by aggregating training data across a mixture of the distributions induced by the expert's policy and the learner's policy.

$$\pi_{\theta_{i+1}} = \arg\min_{\theta} \frac{1}{|\mathcal{D}_i|} \sum_{(s,a)\in\mathcal{D}_i} \mathcal{L}(\pi_\theta(s), a),$$

  where $\mathcal{D}_i$ is augmented at each iteration i with labeled data from the learner's policy.

- **Inverse Reinforcement Learning (IRL)**: While not a BC method per se, IRL seeks to infer the underlying reward function that the expert is assumed to be optimized, which can then be used to train a policy via reinforcement learning. This approach indirectly addresses the distribution shift by focusing on the reward structure rather than the policy itself.

3.3 Advanced Techniques

Recent advances in IL have also explored incorporating model-based approaches, where a model of the environment is used to simulate outcomes of actions, and techniques from deep learning, such as utilizing generative adversarial networks (GANs) to model the distribution of expert actions.

4 Conclusion

Imitation Learning, and specifically Behavioral Cloning, offers a powerful framework for learning complex behaviors from expert demonstrations. Despite challenges such as distribution shift and the smoothing of multi-modal distributions, advances like DAGGER, DART, and extensions like Filtered and Reweighted BC contribute to overcoming these obstacles, making IL a viable approach in scenarios where traditional RL struggles.

5 Value Function

Given an (infinite horizon) MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, p, \gamma)$, a state value function $V^\pi : \mathcal{S} \to \mathbb{R}$ is defined by

$$V^\pi(s) = \mathbb{E}_\pi\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid s_0 = s\right],$$

a state-action value function $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is defined by

$$Q^\pi(s, a) = \mathbb{E}_\pi\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid s_0 = s, a_0 = a\right],$$

by the rule of total expectation, it is easy to show that (Bellman equation for fixed policy π)

$$V^\pi(s) = \langle Q^\pi(s, a), \pi(a|s)\rangle_{\mathcal{A}},$$

and

$$Q(s, a) = r(s, a) + \gamma \mathbb{E}_{s'\sim p(s'|s,a)}[V^\pi(s')].$$

Estimation We consider the estimation for V^π and Q^π without distribution shift, i.e. the dataset $\mathcal{D} = \{\big(s_h^i, a_h^i, r(s_h^i, a_h^i)\big)_{i\geq 0}\}_{i\in[n]}$, where $(s_h^i, a_h^i, r(s_h^i, a_h^i)) \sim \pi$. We simply do the following regression:

$$Q^\pi = \text{argmin}_{Q\in\mathcal{F}} \frac{1}{n} \sum_{(s_0^i, a_0^i), i\in[n]} \big(Q(s_0^i, a_0^i) - \sum_{t\geq 0} \gamma^t r(s_h^i, a_h^i)\big)^2.$$

Better than vanilla Monte Carlo: doesn't need to see all $(s, a) \in \mathcal{S} \times \mathcal{A}$. Just run regression based on function class $\mathcal{F}$, and then extrapolate. $\mathcal{F}$ could be anything from a linear function class or a random forest to a deep neural network. If the policy π is known to us, we can further use the *fixed-Q-estimation:*

$$Q^\pi = \text{argmin}_{Q\in\mathcal{F}} \sum_{(s,a)\in\mathcal{D}} \big(Q(s, a) - (r(s, a) + \gamma\langle\pi(a'|s'), Q(s', a')\rangle_{\mathcal{A}})\big)^2,$$

where (s', a') is successor of (s, a) in the dataset. Such a method further reduces the variance of estimation.