Imitation Learning in the Era of Foundation Models

Wanli Qian

Overview

This handout introduces *imitation learning*, a framework where an agent learns to perform a task by copying examples from an expert rather than by being given explicit rules or a reward function. We will:

- Define imitation learning formally,
- Explain two key algorithms: Behavior Cloning and DAgger,
- Discuss common issues such as covariate shift and compounding errors,
- Show how these ideas appear in modern systems, including robotic foundation models, game agents, and large language models trained with human feedback.

1 What is Imitation Learning?

Imitation learning assumes that an expert (often a human) can perform a task, such as

- driving a car,
- controlling a robot arm,
- playing a video game,
- producing high-quality answers in a dialogue.

As the expert acts, at each time step they observe a *state* s_i and choose an *action* a_i. We record these interactions as a dataset of state–action pairs:

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

The goal of imitation learning is to learn a *policy* $\pi_\theta(s)$, parameterized by θ (for example, the weights of a neural network), that maps states to actions and behaves similarly to the expert:

$$\pi_\theta : s \mapsto a.$$

This setup is very general:

- s could be an image, a stack of sensor readings, or a dialogue history,
- a could be a motor command, a keyboard key, or the next token in a sentence.

2 Formal Setup

We consider a sequential decision-making process, such as a Markov decision process (MDP). At each time step t:

- The agent receives a state s_t,
- The policy π_θ outputs an action a_t,
- The environment transitions to a new state s_{t+1}.

In imitation learning, we do *not* focus on designing a reward function. Instead, we focus on matching the expert's behavior.

Given the dataset $\mathcal{D}$, a common objective is to minimize a loss measuring how different the policy's action is from the expert's action:

$$\theta^* = \arg\min_\theta \sum_{(s_i,a_i)\in\mathcal{D}} \ell\big(\pi_\theta(s_i), a_i\big),$$

where ℓ is a loss function, such as mean squared error (for continuous actions) or cross-entropy loss (for discrete actions).

Intuitively, we are asking the policy to answer the question:

> "If the expert were in this state, what would they do?"

3 Behavior Cloning (BC)

3.1 Basic Idea

Behavior Cloning (BC) is the simplest and most widely used form of imitation learning. It treats imitation as a standard supervised learning problem:

- Input: state s_i,
- Label: expert action a_i,
- Model: policy $\pi_\theta(s)$,
- Objective: minimize the prediction error between $\pi_\theta(s_i)$ and a_i.

Formally, Behavior Cloning solves:

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

If you are familiar with training image classifiers or regressors, the optimization procedure is essentially the same. The key difference is that the label is now an *action* rather than a class name or numeric value.

3.2 Examples

Driving. A neural network takes front-camera images as input and predicts steering angles. Training data comes from recorded human driving.

Robot manipulation. A policy takes in camera images plus a language instruction (for example, "put the apple in the bowl") and predicts low-level robot commands. It is trained on teleoperation demonstrations of humans controlling a robot arm.

Language models (supervised fine-tuning). After pretraining, a large language model is fine-tuned on prompt–response pairs, where responses are written by humans. The model is trained to imitate these high-quality responses.

In all of these cases, Behavior Cloning uses the same supervised learning machinery but applied to action outputs.

4 Covariate Shift and Compounding Errors

While Behavior Cloning often works surprisingly well, it has a key failure mode known as *covariate shift*.

4.1 Training vs. Test Distributions

- During training, all states s_i in $\mathcal{D}$ come from the expert's behavior.
- At test time, the agent acts according to the learned policy π_θ, which is imperfect.

Even small errors can push the agent into states that the expert rarely or never visited. In these states, there is little or no training data, so the model's predictions are less accurate. This can lead to:

- Larger errors,
- Visiting even more unfamiliar states,
- *Compounding errors* over time.

4.2 Lane-Keeping Example

Consider a lane-keeping controller for a car:

- The expert stays centered in the lane; the trained model sees only "good" central-lane images during training.

- At test time, the learned policy makes a small steering error and drifts slightly to one side. This off-center image is different from the training distribution.
- The policy makes a worse prediction, drifting further. Eventually, it may leave the lane entirely.

This example illustrates how slight deviations can snowball when the agent's behavior changes the distribution of states it encounters.

5 DAgger: Dataset Aggregation

5.1 Motivation

To address covariate shift, we need a way to ensure that the policy is trained on the states it is likely to encounter when it is running on its own. *DAgger* (Dataset Aggregation) is a classic algorithm that does this by iteratively collecting more data from the learned policy's trajectories and asking the expert for actions in those states.

5.2 Algorithm

At a high level, DAgger proceeds as follows:

1. **Initial training:**
   (a) Collect expert demonstrations.
   (b) Train an initial policy π_0 on this data using Behavior Cloning.
2. **Rollout with the current policy:**
   (a) Let π_k act in the environment.
   (b) Record the sequence of states visited.
3. **Expert relabeling:**
   (a) For the visited states, ask the expert: "What action should we have taken here?"
   (b) Produce labeled pairs (s, a^*).
4. **Aggregate data:**
$$\mathcal{D} \leftarrow \mathcal{D} \cup \{(s, a^*)\}.$$
5. **Retrain:**
   (a) Train a new policy π_{k+1} on the updated dataset.
   (b) Repeat from Step 2 for several iterations.

Over time, the dataset $\mathcal{D}$ grows to include states from the *learned policy's own trajectories*, not just the expert's ideal trajectories. As a result, the policy learns both:

- How to behave in typical expert states,
- How to recover from states produced by its own mistakes.

This greatly reduces compounding errors in many sequential tasks.

6 Modern Applications of Imitation Learning

Imitation learning is a core building block in many modern AI systems.

6.1 Robotic Foundation Models

Recent *robotic foundation models* are large, often transformer-based policies trained on diverse robot demonstrations for many tasks. Typically:

- They take images and language instructions (e.g., "open the drawer") as input.
- They output low-level robot actions.
- They are trained by Behavior Cloning on large datasets of teleoperation trajectories, sometimes combined with web-scale pretraining for vision and language.

Although the architectures are sophisticated, the core training principle is still imitation learning: learning to act by copying expert demonstrations at scale.

6.2 Game Agents and Video PreTraining

In game settings such as Minecraft, agents can be trained from human gameplay videos:

- A small labeled dataset (video frames + recorded actions) is used to learn how to infer actions from short video clips.
- This model is then applied to large unlabeled video datasets to infer pseudo-actions.
- Finally, Behavior Cloning is run on this large pseudo-labeled dataset.

This allows agents to execute long, coherent sequences of actions by imitating human play.

6.3 Language Models and Human Feedback

Large language models also rely heavily on imitation:

- **Pretraining:** The model imitates Internet text via next-token prediction.
- **Supervised fine-tuning:** The model imitates curated human-written answers to prompts.
- **RLHF (Reinforcement Learning from Human Feedback):** Human preferences over model outputs are used to define a reward model, and reinforcement learning adjusts the policy toward human-preferred behavior.

Many of these stages can be interpreted as imitation learning with different types of "expert" signals (raw text, reference answers, preference labels).

7 Strengths and Limitations

7.1 Strengths

- Conceptually simple: uses standard supervised learning tools.
- No need to design a reward function, which can be difficult in complex tasks.
- Efficient when high-quality demonstrations are available.
- Directly encodes human expertise and preferences into the policy.

7.2 Limitations

- The policy is often limited by the quality of the expert; it is difficult to exceed expert performance using pure imitation.
- Performance depends strongly on the coverage and diversity of the demonstration dataset.
- Behavior Cloning alone is sensitive to covariate shift and compounding errors, especially for long-horizon tasks, unless combined with methods such as DAgger or reinforcement learning.

8 Summary

- **Imitation learning** trains policies to act by copying expert demonstrations.
- **Behavior Cloning** is the fundamental supervised-learning approach on state–action pairs.
- **Covariate shift** and compounding errors arise when the learned policy visits states unseen in the expert data.
- **DAgger** addresses this by aggregating data from states visited by the learned policy and relabeling them with expert actions.
- Modern systems in **robotics**, **games**, and **language modelling** rely heavily on imitation learning, often at very large scales.