17-740 Spring 2025
Algorithmic Foundations of Interactive Learning

Lecture 9: DAgger & Covariate Shift in IL

Lecturer: J. Andrew (Drew) Bagnell
Scribe: Bardienus Duisterhof, Kimberly Truong, Khush Agrawal

9.1 Overview

We will cover the basics of imitation learning, *an invitation to imitation.* Among other things, we will cover DAgger (Dataset Aggregation) and how it can improve imitation learning (IL) performance in the presence of covariate shift.

9.2 Imitation Learning (IL)

Imitation learning is when we do not have access to a reward function, and instead aim to learn from expert demonstrations. One flavor of imitation learning is *offline* behavioral cloning, where we try to mimic the actions of the expert directly. In contrast, *interactive* imitation learning algorithms (as we discuss in this lecture) instead attempt to match the outcomes of expert actions / overall expert behavior.

A natural question one might have *"how is IL different from most other ML problems?"*

- **Decisions have consequences**: errors pass through a feedback loop and compound → distribution shift, test $\neq$ train distribution. For example, a small steering error in a self-driving car can cause the vehicle to drift toward the edge of the road. If the training data only contains examples from driving in the center of the lane, the policy won't know how to correct this drift, potentially causing the car to drive off the track entirely.
- **Sequential decisions**: Actions and decisions are purposeful and sequential. They shape future states and returns, building toward long-term goals rather than just responding to the current state.
- **Non-IID data**: The sequential nature of decision-making breaks the independent and identically distributed data assumption required by traditional supervised learning methods.

9.2.1 Learning to drive by imitation

Consider an RGB camera image (state s) → policy $\pi(a|s)$ → distribution over actions a, where actions represent steering angles. As proposed by [1], we can use behavioral cloning to learn from expert demonstrations. Given a dataset $\mathcal{D} = \{(s_i, a_i)\}_{i=1}^N$ of state-action pairs demonstrated by an expert policy π_E, we solve the maximum likelihood problem:

$$\max_{\pi\in\Pi} \sum_{\xi\in\mathcal{D}} \log \mathbb{P}_\pi(\xi) = \max_{\pi\in\Pi} \sum_{\xi_E\in\mathcal{D}} \log \left(\prod_h^H \pi(a_h^E|s_h^E) \right) = \max_{\pi\in\Pi} \sum_{\xi_E\in\mathcal{D}} \sum_h^H \log \pi(a_h^E|s_h^E) \tag{9.1}$$

This objective aims to maximize the probability of taking the same actions as the expert in the states encountered by the expert and is the standard, *supervised learning* to imitation.

Now let's say, at every step h, there is an ϵ probability the policy disagrees with the expert. Each mistake at time step h can lead the learner to deviate from the expert on all subsequent steps. This results in compounding errors. After H steps, the resulting gap on $J(\pi)$ can be expressed as:

$$J(\pi) - J(\pi_E) \leq \epsilon \sum_{h=1}^H (H-h) = \epsilon \frac{H(H-1)}{2} \in O(H^2\epsilon) \tag{9.2}$$

The quadratic dependence on horizon length H tells us that errors compound over time.

So then how do we mitigate this compounding error?

Core idea: use interaction. Intuitively, this is because interaction allows us to see states from the test distribution at training time, eliminating the covariate shift.

Algorithm 1: **Forward Training**. We train a sequence of policies $\pi_1, \pi_2, \ldots, \pi_H$, where π_h is trained to predict the action the expert would have taken at time step h, given that we followed policies $\pi_1, \ldots, \pi_{h-1}$ for the first $h-1$ steps. So, at $h = 1$, we perform vanilla behavioral cloning to learn π_1. Then, at $h = 2$, we roll out π_1 and ask the expert what they would have done, had they been in our situation, collecting a dataset of action labels. We then train π_2 via behavioral cloning on this new, on-policy dataset and repeat till $h = H$.

Assume that at each step, we make a mistake with probability ϵ and such a mistake can at most cost us u (we'll explore what u means in a bit). Then, simply by summing up over timesteps, we have

$$J(\pi_{\text{forward}}) - J(\pi_E) \leq \sum_h^H p(\text{mistake}) \cdot \text{max cost} = \epsilon u H \tag{9.3}$$

The bound is now linear in H rather than quadratic, because each policy π_h is trained on the distribution of states that the learner actually encounters at step h. However, this approach requires training H separate policies, which becomes impractical for large H, begging the question of how we train a stationary policy interactively.

Algorithm 2: **DAgger: dataset aggregation**. DAgger [2] performs a similar procedure to learn a single, stationary policy. Algorithm 1 describes the algorithm.

Algorithm 1 DAgger: Dataset Aggregation

```
Require: Initial dataset $\mathcal{D}_0$ collected from offline data
  Train initial policy $\pi_1$ on $\mathcal{D}_0$
  for $i = 1$ to $N$ do
    Execute $\pi_i$ in the environment to collect trajectories $\xi_i$
    Query expert for action labels at all states in $\xi_i$
    Aggregate action labels into dataset: $\mathcal{D}_i \leftarrow \mathcal{D}_{i-1} \cup$ action labels
    Train new policy $\pi_{i+1}$ on aggregated dataset $\mathcal{D}_i$ via behavioral cloning
  end for
  return Best of $N$ policies on validation data.
```

As we'll prove below, with large enough N, we get:

$$J(\pi) - J(\pi_E) \leq uH\epsilon. \tag{9.4}$$

With DAgger we reduce the problem of imitation learning to that of no-regret online learning. In particular, the "dataset aggregation" procedure is implementing the follow the (regularized) leader algorithm we discussed earlier. The environment is not an adversary per se but the no-regret framework is still useful because we're not in the statistical / iid setting.

Analysis. To analyze the performance of DAgger, consider the following "zero-one" loss: [1]

$$\mathcal{L}_{\pi_E}(\pi, s) = \mathbb{E}_{a\sim\pi}\left[\mathbf{1}(a \neq \pi_E(s))\right]. \tag{9.5}$$

This is the probability that our action doesn't match the expert's. We can take the expectation over states generated by π_i to get the sequence of loss functions we will feed to our no-regret online learner (FTRL / dataset aggregation in this case):

$$\ell_i(\pi) = \mathbb{E}_{s\sim\rho_\pi}[\mathcal{L}_{\pi_E}(\pi, s)]. \tag{9.6}$$

Critically, notice that this expectation is taken over states from the learner's state distribution rather than the expert's – this is what distinguishes this loss from behavioral cloning.

[1] We're assuming the expert is deterministic here for simplicity, see [2] for the more general proof.

We can then express the optimization problem we're solving at each round of DAgger as

$$\pi_{i+1} = \min_{\pi \in \Pi} \sum_{j}^{i} \ell_j(\pi). \tag{9.7}$$

A natural question when reading the above equation might be "*why are we optimizing over the history of past visitation distributions when we care about our current policy's visitation distribution*?" Observe that if the policy we're choosing doesn't change too much from what we've seen in the past, minimizing the above equation will give us good guarantees. Intuitively, this is what the no-regret property of FT(R)L means. Perhaps the easiest way to see this is to notice that each new dataset is a vanishing fraction of the overall dataset as $N \to \infty$, which means our learning process should eventually stabilize.

Let's write out the cumulative loss accumulated by the no-regret online learner:

$$\frac{1}{N}\sum_{i=1}^{N}[l_i(\pi_i)] = \underbrace{\frac{1}{N}\sum_{i=1}^{N}[l_i(\pi_i) - l_i(\pi^\star)]}_{\text{average regret}} + \underbrace{\frac{1}{N}\sum_{i=1}^{N} l_i(\pi^\star)}_{\text{expert loss}}, \tag{9.8}$$

where $\pi^\star = \min_{\pi \in \Pi} \sum_i^N \ell_i(\pi)$. Note: $\pi^\star$ need not necessarily be π_E if we're in the *misspecified* setting (i.e. $\pi_E \notin \Pi$). If we're in the *realizable* setting (i.e. $\pi_E \in \Pi$), the second term on the RHS must be 0. We'll assume this for simplicity for the rest of the note.

Due to the no-regret property of FT(R)L / dataset aggregation, we know that the first term goes to zero on average. Via the Performance Difference Lemma, we can link the regret of our online learner to the performance of the learned policy. To do so, we'll first need to discuss the concept of *recoverability* more formally – the u from above.

Recoverability. A key challenge in imitation learning is ensuring that mistakes made by the learned policy do not cause large deviations in performance from the expert. The notion of recoverability helps formalize this. Recoverability in this context means that the maximum cost a single-step deviation from the expert could inflict is bounded. In math,

$$Q_h^{\pi_E}(s,a) - Q_h^{\pi_E}(s, \pi_E(s)) \leq u, \, \forall (s,a) \in \mathcal{S} \times \mathcal{A}. \tag{9.9}$$

Consider the following two cases: (1) Flat Terrain: You're walking on a wide, flat field. If you take a slightly wrong step, you can easily recover — the cost u is small. (2) Walking Along a Cliff: Now suppose you're walking along a narrow mountain ridge. A small misstep could send you tumbling down the cliff — the cost u is very high. In both cases, ϵ could be the same (say, you make a mistake 10% of the time). But the overall performance degradation depends heavily on u. On the flat field, $T\epsilon u$ might still be small. But on the cliff, even small ϵ leads to catastrophic performance — hence, the bound becomes loose unless you can ensure very low ϵ. This highlights why recoverability matters. If the expert can't recover from mistakes (i.e., the environment is unforgiving), even DAgger can't guarantee good performance unless the learner gets very accurate.

Completing the Proof. Assume that the expected 0-1 classification loss (probability of disagreement with expert) at each time step is bounded by ϵ. This corresponds to the average regret is bounded by ϵ in the realizable setting. If the average regret is bounded by ϵ, there must exist some $i \in [N]$ that has relative loss bounded by ϵ. Furthermore, assume that the recoverability condition 9.9 holds for some constant u. Then, the performance gap between the learner and the expert can be bounded as

$$J(\pi) - J(\pi_e) \leq H\epsilon u. \tag{9.10}$$

The PDL tells us that for any policy π,

$$J(\pi) - J(\pi_E) = \sum_{h}^{H} E_{s_h \sim \pi}[Q_h^{\pi_E}(s_h, \pi(s_h)) - Q_h^{\pi_E}(s_h, \pi_E(s_h))]. \tag{9.11}$$

By Hölder's inequality, each term in the sum is bounded by ϵu for some $i \in [N]$, giving us:

$$\min_{i \in N} J(\pi_i) - J(\pi_E) \leq \sum_{h}^{H} \epsilon u = H\epsilon u. \tag{9.12}$$

QED!

9.3 *

References

[1] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

[2] Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning, 2011.