Lecture 5: Imitation Learning

1 Introduction

Logistics

1. HW1 due yesterday
2. HW2 released today. HW2 will primarily cover material covered in today's class.

Review: Markov Decision Process (MDP)

- Agent – Environment loop
- States s_t, actions a_t, rewards $r(s_t, a_t)$.
- Key quantity is the policy $\pi(a \mid s)$. This is what we're trying to learn.
- *Discounted* objective: $\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$.
- Dynamics $p(s_{t+1} \mid s_t, a_t)$ and reward function are unknown.

Special cases of the MDP Use the figure above for illustration.

1. Does time matter? If not, use a bandit method
2. Do you know the model of your system? If so, do CEM + MPC
3. Do you have expert data? [today] If so, try imitation learning.
4. If all else fails, try RL! [Thur]

Learning objectives for today At the end of this lecture, you will be able to

1. Apply imitation learning to learn policies from expert demonstrations.
2. Explain the challenges associated with imitation learning: outperforming the "expert," handling noise.
3. Be able to apply some simple techniques for addressing these challenges

2 Imitation Learning $\neq$ Reinforcement Learning

Examples from slides

- ALVINN
- TRI
- NVIDIA car
- Human imitation

Today's class will focus on mimicking previously-seen policies and strategies. It will not focus on finding better strategies. The imitation learning concepts that we discuss today will lay the groundwork for building (actual) RL methods in coming weeks. In fact, we'll see that one of the most common RL methods is a very small changes from the behavioral cloning method that we will introduce today.

- Goal: mimicking behaviors. Recall examples from slides. The goal is *not* to maximize rewards. The imitation learning problem is different from RL.
- Intuitively, we want to learn a policy that visits the same states and actions as an observed expert.

- There are many different types of imitation learning. E.g., do you observe *how* the expert does it (i.e., their actions) or just their states? Can you interact with the environment to try out possible mimicry strategies? Do you try to infer what reward function the expert is optimizing, or just learn their policy?

2.1 Behavioral Cloning [8]

Behavioral cloning (BC) is by far the most common imitation learning algorithm. It doesn't require interactions; it will learn purely from a dataset of states and actions.

- The inputs are a dataset $\{(s, a)\}$ of state-action pairs from an expert. This could be some learned policy. It could be a human expert. E.g., for ALVINN, this was the grad student driving the car. For TRI, it was the human teleoperator.
- The output is a policy $\pi(a \mid s)$.
- The objective is to maximize the likelihood of the observed actions.

$$\max_{\pi} \mathbb{E}_{(s,a)\sim\mathcal{D}}[\log \pi(a \mid s)] \tag{1}$$

$$\max_{\theta} \mathbb{E}_{(s,a)\sim\mathcal{D}}[\log \pi_\theta(a \mid s)]. \tag{2}$$

  Note that we can write this either as an optimization problem over a policy, or over the parameters of the policy. In practice, we just have samples from this distribution, so we write this objective as

$$\max_{\pi} \frac{1}{|\mathcal{D}|} \sum_{(s,a)\sim\mathcal{D}} [\log \pi(a \mid s)]. \tag{3}$$

- Intuition: Your model is some probability distribution over the actions. It's parameters are given by some function (e.g., a neural network) of the input observation. For example, think of a Gaussian distribution, so your neural network is predicting the mean and variance. In BC, your aim is to tune those neural network weights so that the predicted mean and variance result in assigning high likelihood to the observed data points.
- BC is an example of supervised learning. If the actions are discrete, this is just a classification problem: given an observation, predict whether to turn left or right. If actions are continuous, this is just a regression problem: predict the steering angle.
- So, is BC just supervised learning? Yes, but the evaluation will be different.

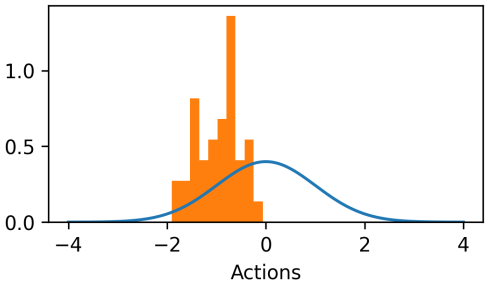

Figure 1: Maximum likelihood involves changing the parameters of a density function (blue) so that it assigns higher weight to the observed data (orange).

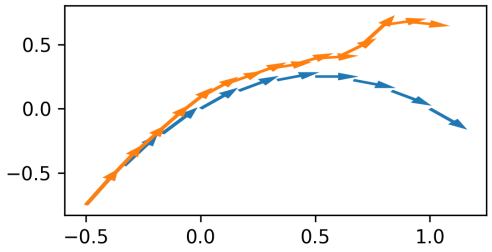

Figure 2: Distribution shift: A policy that achieves low loss on the training might have a much higher loss on different states, including those states that it would actually visit if evaluated in the environment.

Worked Example To build more intuition for this objective, let's consider the special case where your policy is a Gaussian over 1-dimensional actions:

$$\pi_\theta(a \mid s) = \mathcal{N}(a; \mu_\theta(s), \sigma = 1) \tag{4}$$

$$= \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(\mu_\theta(s)-a)^2}. \tag{5}$$

We can then write the maximum likelihood objective as

$$\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\log \pi(a \mid s)\right] = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[-\frac{1}{2}(\mu_\theta(s) - a)^2 - \log 2\right]. \tag{6}$$

Note that this is just minimizing the MSE between the predicted mean and the actual action.

2.2 3 Challenges with BC

1. Distribution Shift
2. Implicit Smoothing
3. Outperforming the expert.

3 Challenge 1: Distribution Shift / Compounding Errors

The main challenge with behavioral cloning is distribution shift: while the policy is trained to select actions on states from a certain training set, using that policy to interact with the world may result in visiting states that were unseen during training.

The root cause is a difference between training and evaluation. During training, states are sampled from the dataset (one distribution); during evaluation, we let the policy select actions, which can lead to visiting states (second distribution). While the policy may achieve a low loss on one of these state distributions, it's not guaranteed to achieve a low loss on the other distribution.

3.1 A Few Solutions

1. Data augmentation [1]. Requires domain knowledge, but can be quite effective. See Fig. 3 for an example. For every state-action pair, we augment the state (e.g., shift it up) and (often) augment the action to compensate (e.g., shift it down). This is often done in autonomous driving applications. The car might drive off the road very infrequently, so it's hard to collect data where the car takes *corrective* actions. This data augmentation exactly serves that purpose.
2. DART [6]. Add noise while collecting the data. This *broadens* the data distribution, increasing the likelihood that the learned policy will continue to visit states that it has seen before in the dataset.
3. DAGGER [9]. This is the most principled approach, it has the strongest theoretical guarantees, but it's pretty hard to use in practice. As visualized in Fig. 5, in entails three steps:

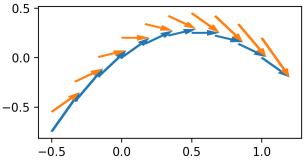

Figure 3: Data augmentation for behavioral cloning [1]

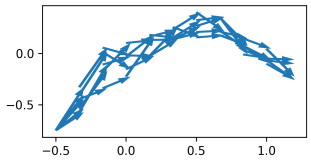

Figure 4: Injecting noise during data collection [6]

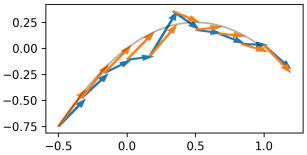

Figure 5: DAGGER [9]

(a) Run BC on an initial dataset
(b) Collect data with the learned policy
(c) Ask an expert for action labels: $(s, a) \to (s, a^*)$. Go back to step 1.

4 Challenge 2: Implicit Smoothing

A second, subtle challenge with behavioral cloning is that the true distributions over actions are often multi-modal. See, e.g., Fig. 6. In this example, what would happen if you attempted to fit a unimodal policy? (Your learned policy would frequently select actions that are never see in the training data.

One natural solution to this is to use more expressive policy classes. This is *not* saying that we're adding more layers to our neural network. Rather, we're saying that the output distribution has to be more expressive. Examples include Gaussian mixture models, discretization + softmax (Google's robots use this), diffusion models (TRI's robots use this).

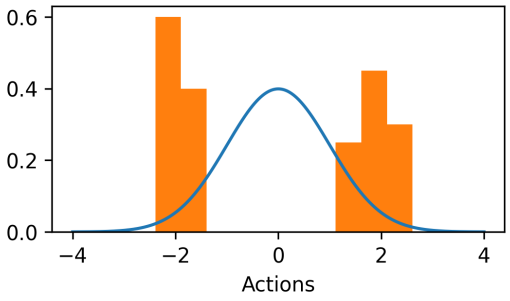

Figure 6: Fitting a multi-modal distribution over actions with a uni-modal policy can result in selecting actions that rarely occur in the data.

Historical note and open question. BC methods went out of style for several years, but are not back in vogue. We can even interpret today's LLMs as doing BC. Is the poor performance typical ascribed to BC caused by *(1)* compounding errors (the first problem we talked about) or *(2)* this implicit smoothing? The recent success of discretized and diffusion models hints at the latter.

5 Challenge 3: Outperforming the "Expert"

The third challenge with imitation learning methods is that they imitate the data; if your data were collected by a novice user, then the resulting policy may not be that good. We cannot expect that imitation learning methods will outperform the data they were trained on. If the data told you to run every red light, your learned policy will run every red light.

One way to address this problem is by filtering your dataset. This is perhaps the easiest approach, and can be done by simply removing some of your data. For example, if you were teleoperating a robot and made a mistake, you could simply discard that trial.

There are also automated ways of discarding data.

- **Filtered imitation learning** [5] In a game like chess, you could discard every trial where you lost the game. More generally, in a setting where you also have rewards, you could sort the trials by the rewards and just imitate the top 10% of trials. This strategy works surprisingly well!
- **Reward weighted regression** [7]. Another strategy is to reweight your data by the rewards, resulting in an objective that looks like the following:

$$\max_{\pi} \sum_{\tau} R(\tau) \sum_{t} \log \pi_\theta(a_t \mid s_t).$$

  We'll see this objective again on Thursday...
- **Goal-conditioned imitation learning** [2–4]. In settings where you have multiple tasks / rewards, you can shuffle around your data. Even if a trajectory wasn't optimal for one task, it may have been optimal for another task. For example, consider the problem of reaching goals. In this setting, we want to learn a policy $\pi(a \mid s, g)$. You have data $\{\tau\}$, which may not be optimal. But if you look at

the states that were actually reached in that data, then your data may look more sensible. This results in an objective that looks like:

$$\max_{\pi} \sum_{\tau} \sum_{t} \log \pi_\theta(a_t \mid s_t, s = s_T).$$

6 Summary

- Imitation requires demonstrations
- Imitation is just supervised learning, but evaluated differently.
- 3 challenges: compounding errors, implicit smoothing, outperforming the expert.

References

[1] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

[2] Ding, Y., Florensa, C., Abbeel, P., and Phielipp, M. (2019). Goal-conditioned imitation learning. *Advances in neural information processing systems*, 32.

[3] Emmons, S., Eysenbach, B., Kostrikov, I., and Levine, S. (2021). Rvs: What is essential for offline rl via supervised learning? *arXiv preprint arXiv:2112.10751*.

[4] Ghosh, D., Gupta, A., Reddy, A., Fu, J., Devin, C., Eysenbach, B., and Levine, S. (2019). Learning to reach goals via iterated supervised learning. *arXiv preprint arXiv:1912.06088*.

[5] Kostrikov, I., Nair, A., and Levine, S. (2021). Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*.

[6] Laskey, M., Lee, J., Fox, R., Dragan, A., and Goldberg, K. (2017). Dart: Noise injection for robust imitation learning. In *Conference on robot learning*, pages 143–156. PMLR.

[7] Peters, J. and Schaal, S. (2007). Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pages 745–750.

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

[9] Ross, S., Gordon, G., and Bagnell, D. (2011). A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings.