Lecture 13: Imitation Learning

Akshay Krishnamurthy
akshay@cs.umass.edu

April 25, 2022

The last few lectures have covered most of what we know about provably efficient reinforcement with function approximation. The picture is still developing, but it is quite clear that fairly strong assumption are necessary to obtain efficient algorithms. We have mostly focused on *modeling* assumptions that place restrictions on the function class or MDP dynamics, but another very fruitful idea is to make completely different kinds of assumptions, for example on what type of access you have to the environment. These assumptions can help us avoid some of the most challenging aspects of reinforcement learning but can also lead to very practical methods in some settings.

One of the most popular and practically useful approaches in this direction is to assume some access to an *expert demonstrator* who can help guide the search for a good policy. Broadly, these approaches are referred to as *imitation learning*, where the goal is typically to imitate or match the behavior of the expert demonstrator. Many different formulations exist and they typically differ in the kind of access to the expert that is available. Depending on the precise setting, a number of real-world problems (in robotics, autonomous driving, etc.) naturally can be cast in the imitation learning framework. Some examples include:

- When training a self-driving car, we might instrument cars to collect data about steering angles and acceleration when humans drive. This could be viewed as an offline dataset of (s, a) pairs collected by an expert.
- Later in the training cycle, we might run the self-driving system with a human supervisor and allow the supervisor to take over control from the system. This provides *interactive access* to the expert.
- In robotics, we might have a human manually move a robotic arm through a sequence of desired configurations. Here we would see only the states visited by an expert, but not the actions chosen.

More generally, you could argue that humans learn quite a lot by imitating others! This setting perhaps could be modeled through some change in reference frame or perspective.

1 Offline imitation learning

Offline imitation learning refers to settings where the expert information is collected in advance and provided in bulk to the learner, but the learner does not have interactive or query access to the expert. In many formulations here, we are also unable to interact with the environment at all.

The most basic formulation is as follows. There is a discounted MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$ that we do not get to interact with. Instead we see a dataset $\{(s_i, a_i)\}_{i=1}^n$ where $(s_i, a_i) \sim d^{\pi^\star}$ for some optimal policy $\pi^\star$ (Recall that $d^\pi(s, a) = (1-\gamma)\sum_{t=0}^\infty \gamma^t \mathbb{P}^\pi[s_t = s, a_t = a]$ is the discounted occupancy measure for π). Using this data only, we'd like learn a policy $\hat{\pi}$ that is near optimal for the MDP, that is, we'd like $J(\pi^\star) - J(\hat{\pi}) \leq \varepsilon$.

Remark 1 (Reward function)**.** *Note that in this formulation, the learner never observes the rewards, although implicitly we are assuming that one exists and that the expert optimizes it. Since there is no access to the reward at all, we cannot just default to existing RL techniques (even if they require strong assumptions). On the other hand, specifying a reward function can be quite challenging, and capturing it implicitly through expert behavior may be more practically feasible.*

Behavior cloning. This setting is quite constrained in terms of what we can do algorithmically. We cannot interact with the MDP, we have no access to the reward function, etc. Essentially all we can do is try to predict the states and actions chosen by the expert. This is known as *behavior cloning*. Given a policy class Π we optimize

$$\hat{\pi} \leftarrow \operatorname*{argmax}_{\pi \in \Pi} \frac{1}{n} \sum_{i=1}^{n} \log \pi(a_i \mid s_i),$$

which is a form of maximum likelihood estimation. For this method, we can prove the following theorem, using mostly tools you have already seen in the course.

Theorem 2 (Behavior Cloning guarantee)**.** *Let Π be a finite policy class ($|\Pi| < \infty$) that contains $\pi^\star$ (realizability). Then with probability $1-\delta$ the policy $\hat{\pi}$ output by behavior cloning satisfies*

$$J(\pi^\star) - J(\hat{\pi}) \lesssim \frac{1}{(1-\gamma)^2} \sqrt{\frac{\log(|\Pi|/\delta)}{n}}$$

While algorithmically we are just performing supervised learning, this guarantee also address the distribution shift issue, since the mistakes that $\hat{\pi}$ makes will result in us visiting states that are not visited by $\pi^\star$. This is how the horizon factors arise in the analysis. Later we will see when and how these horizon factors can be sharpened.

Proof. As we saw briefly in last lecture, we can get a generalization guarantee for maximum likelihood estimation with realizability. In this case, the guarantee is that with probability $1-\delta$ we have

$$\mathbb{E}_{s \sim d^{\pi^\star}} \|\hat{\pi}(\cdot \mid s) - \pi^\star(\cdot \mid s)\|_{\mathrm{TV}} \le \sqrt{\frac{\log(|\Pi|/\delta)}{n}}.$$

You should think of this as a standard supervised learning guarantee. In particular note that the states are drawn from $d^{\pi^\star}$, which matches the training distribution, so there is no distribution shift yet. Turning to the policy performance, by the PD lemma, Holder's inequality, and the fact that Q functions are bounded in $[0, \frac{1}{1-\gamma})$, we have

$$\begin{aligned} J(\pi^\star) - J(\hat{\pi}) &= \frac{1}{1-\gamma} \mathbb{E}_{(s,a) \sim d^{\pi^\star}}[A^{\hat{\pi}}(s,a)] = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^{\pi^\star}} \left[\mathbb{E}_{a \sim \pi^\star(\cdot|s)}[Q^{\hat{\pi}}(s,a)] - \mathbb{E}_{a \sim \hat{\pi}(\cdot|s)}[Q^{\hat{\pi}}(s,a)] \right] \\ &\le \frac{1}{(1-\gamma)^2} \mathbb{E}_{s \sim d^{\pi^\star}} \|\pi^\star(\cdot \mid s) - \hat{\pi}(\cdot \mid s)\|_{\mathrm{TV}}. \end{aligned}$$

Putting this together with the supervised learning guarantee, we conclude the proof. □

To understand the quadratic scaling in the horizon, it's easiest to think about the finite horizon case. Intuitively, $\hat{\pi}$ will disagree with $\pi^\star$ at every time step with some probability ϵ; however when we make a mistake earlier in the episode, the cost could be $\Omega(H)$ since this is the magnitude of the Q-function. Roughly speaking, in the first half of the episode, we expect to make a mistake with probability $\epsilon H/2$, but when we make such a mistake we incur at least $H/2$ performance loss. The issue here is that when we make a mistake, we may completely leave the distribution of states visited under $d^{\pi^\star}$ so we no longer have any idea how to make good decisions. Thus the best we can say is that when we make a mistake, the performance loss is the magnitude of the Q function.

Distribution matching. While quadratic horizon dependence is sharp for BC, it can be avoided in some settings. One setting is if we have access to the transition operator P, which may be feasible if you are doing imitation learning in simulation. This is still an interesting setting because we don't have access to the reward, so we can't use classical planning techniques. Instead, we need to use the expert data to extract information about the reward function.

The advantage of having the transition operator P is that we can correct the mistakes that we make earlier in the trajectory because we can use P to understand how to quickly get back onto the distribution $d^{\pi^\star}$. At a high level, this helps us break ties among all the policies that make mistakes an ϵ-fraction of the time. Among these, some of these will make mistakes and completely wander off, incurring maximal performance loss per mistake. On the other hand, some will make mistakes but then "self-correct" and return to $d^{\pi^\star}$. These policies should only incur a constant performance loss per mistake. Knowing the dynamics P helps us differentiate between these two types of policies by *distribution matching*.

The idea is to find a policy $\hat{\pi}$ that approximately minimizes $\|d^{\hat{\pi}} - d^{\pi^\star}\|_{\mathrm{TV}}$, so that $\hat{\pi}$ matches the distribution of $\pi^\star$. This requires some new techniques, but observe that this distribution matching objective captures some of the issues we have been discussing. Thinking about the episodic version of this objective, if $\hat{\pi}$ deviates from $\pi^\star$ early in the trajectory (and does not correct itself) then we'll also disagree on the remaining $\Omega(H)$ steps, so the TV distance will be very large. On the other hand deviating late in the episode, or correcting early deviations, will keep the TV distance relatively small.

The algorithm uses an estimation technique called *Scheffe sets* to estimate and minimize this TV distance. The idea is that the TV distance $\|P - Q\|_{\mathrm{TV}}$ can be written in a variational form as

$$\|P - Q\|_{\mathrm{TV}} = \sup_{f:\|f\|_\infty \leq 1} \mathbb{E}_{x\sim P} f(x) - \mathbb{E}_{x\sim Q} f(x).$$

The right hand side here is particularly nice for estimation from samples, since for a fixed function f, we can appeal to standard concentration arguments about empirical means. In addition, if P and Q are known, we can compute the argmax function $f_{P,Q}$ that witnesses the TV distance in advance and then we can estimate the TV distance given only samples from P and Q.

We want to use this technique to estimate $\|d^{\pi} - d^{\pi^\star}\|_{\mathrm{TV}}$ using just samples from $\pi^\star$, but since we don't know $d^{\pi^\star}$, we cannot compute the witness functions in advance. Instead, we'll compute a slightly larger class of functions $\mathcal{F}$ that contains the witness functions, and this will suffice for our purposes. We do this as follows: (1) Given the policy class Π and the transition operator P we can compute d^π for each policy π (although this would require enumeration of Π). (2) For each π, π' we can compute the witness function $f_{\pi,\pi'}$ that achieves the maximum in the variational form for $\|d^{\pi} - d^{\pi'}\|_{\mathrm{TV}}$. Letting $\mathcal{F} = \{f_{\pi,\pi'} : \pi, \pi' \in \Pi\}$ we finally compute

$$\hat{\pi} \leftarrow \operatorname*{argmin}_{\pi\in\Pi} \operatorname*{argmax}_{f\in\mathcal{F}} \left[\mathbb{E}_{(s,a)\sim d^\pi} f(s,a) - \frac{1}{n}\sum_{i=1}^{n} f(s_i, a_i) \right].$$

Theorem 3 (Distribution matching)**.** *Given a finite policy class* Π *containing* $\pi^\star$*, the distribution matching algorithm guarantees that with probability at least* $1-\delta$*:*

$$J(\pi^\star) - J(\hat{\pi}) \lesssim \frac{1}{1-\gamma}\sqrt{\frac{\log(|\Pi|/\delta)}{n}}.$$

Of course, the downside is that the algorithm requires access to the dynamics P and is not computationally efficient, since we must enumerate over Π.

Proof. The bulk of the proof involves showing that $\|d^{\hat{\pi}} - d^{\pi^\star}\|_{\mathrm{TV}} \leq \epsilon$ with high probability. The result follows immediately from this fact, since

$$J(\pi^\star) - J(\hat{\pi}) = \frac{1}{1-\gamma}\left[\mathbb{E}_{(s,a)\sim d^{\pi^\star}} r(s,a) - \mathbb{E}_{(s,a)\sim d^{\hat{\pi}}} r(s,a)\right] \leq \frac{\epsilon}{1-\gamma}.$$

To prove the TV bound, let $\mathcal{W}(\pi, \pi'; \mathcal{F}) = \max_{f\in\mathcal{F}} \mathbb{E}_{d^\pi} f(s,a) - \mathbb{E}_{d^{\pi'}} f(s,a)$ be the *integral probability metric* induced by function class $\mathcal{F}$. Our algorithm estimates $\mathcal{W}(\pi, \pi^\star; \mathcal{F})$ from samples of $d^{\pi^\star}$, so let us call the estimate $\widehat{\mathcal{W}}(\pi, \pi^\star; \mathcal{F})$. By a uniform convergence argument, using the fact that $|\mathcal{F}| \leq |\Pi|^2$ by construction, w.p. $1-\delta$:

$$\forall \pi : \left|\mathcal{W}(\pi, \pi^\star; \mathcal{F}) - \widehat{\mathcal{W}}(\pi, \pi^\star; \mathcal{F})\right| \lesssim \sqrt{\frac{\log(|\Pi|/\delta)}{n}}.$$

By the ERM-type analysis and using that $\pi^\star \in \Pi$:

$$\begin{aligned}\mathcal{W}(\hat{\pi}, \pi^\star; \mathcal{F}) &\lesssim \widehat{\mathcal{W}}(\hat{\pi}, \pi^\star; \mathcal{F}) + \sqrt{\frac{\log(|\Pi|/\delta)}{n}} \lesssim \widehat{\mathcal{W}}(\pi^\star, \pi^\star; \mathcal{F}) + \sqrt{\frac{\log(|\Pi|/\delta)}{n}} \\ &\lesssim \mathcal{W}(\pi^\star, \pi^\star; \mathcal{F}) + \sqrt{\frac{\log(|\Pi|/\delta)}{n}} = C\sqrt{\frac{\log(|\Pi|/\delta)}{n}},\end{aligned}$$

since $\mathcal{W}(\pi^\star, \pi^\star; \cdot) = 0$ for any function class, and where C is a universal constant. Finally, by construction of $\mathcal{F}$ and realizability, we know that $f_{\hat{\pi},\pi^\star} \in \mathcal{F}$ and that $f_{\hat{\pi},\pi^\star}$ is such that $\mathbb{E}_{d^{\hat{\pi}}} f_{\hat{\pi},\pi^\star}(s,a) - \mathbb{E}_{d^{\pi^\star}} f_{\hat{\pi},\pi^\star}(s,a) = \|d^{\hat{\pi}} - d^{\pi^\star}\|_{\mathrm{TV}}$. This directly implies that $\|d^{\hat{\pi}} - d^{\pi^\star}\|_{\mathrm{TV}} \leq \mathcal{W}(\hat{\pi}, \pi^\star; \mathcal{F})$, which concludes the proof. □

2 Interactive imitation learning

There is another way to correct the distribution shift issue and potentially avoid the quadratic horizon factor that may be more natural in some applications. This is to assume *interactive access* to the expert policy, which means that we can interact with the MDP dynamics and then query the expert for $\pi^\star(s)$ on any state s that we visit. Here we want to minimize the number of queries that we obtain (a fair comparison with previous methods would mean that we only get n queries).

Intuitively, being able to interact with the dynamics and query the expert should allow us to better correct for distribution shift, since once we have $\hat{\pi}$ we can query for the experts actions on $d^{\hat{\pi}}$ to learn how to recover. However, this creates a small issue, since the new information about the expert actions may result in an update to $\hat{\pi}$, which requires us to iterate the process. Indeed, in some sense we want to approximately minimize a sort of fixed point problem of the form

$$\text{minimize} \;\; \mathbb{E}_{(s,a)\sim d^\pi}[\mathbf{1}\{a \neq \pi^\star(s)\}].$$

This is not a traditional supervised learning problem since the "classifier" π influences both the state distribution (the features) and the predictions (the actions). We would like to find a policy $\hat{\pi}$ that makes good predictions on its own distribution $d^{\hat{\pi}}$.

One way to resolve the fixed point issue is through the use of online learning techniques. This is the basic idea of the DAgger algorithm. At a high level, we use an online learning algorithm to select a policy π^t, then we collect a sample (s^t, a^t) where $s^t \sim d^{\pi^t}$ and $a^t = \pi^\star(s_t)$ and we use the sample to create a loss function $\ell_t : \Pi \to \mathbb{R}$ that can be used to score policies. Typically the loss function takes the form $\ell(\pi(s); s, a)$ and checks whether $\pi(s)$ agrees with the expert action a in some manner. Feeding this loss function into the online learning algorithm results in a new policy π^{t+1}. At the end of the day, the online learning algorithm ensures that

$$\sum_{t=1}^{T} \ell_t(\pi^t) \leq \min_{\pi} \sum_{t=1}^{T} \ell_t(\pi) + \mathrm{Reg}(T)$$

Using the way that we created the loss functions, in particular that at round t we sample the state $s \sim d^{\pi^t}$, we find that there exists some $\tau \in [T]$ such that

$$\mathbb{E}_{s\sim d^{\pi^\tau}} \ell(\pi^\tau(s); s, \pi^\star(s)) \leq \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{s\sim d^{\pi^t}} \ell(\pi^t(s); s, \pi^\star(s)) \lesssim \frac{1}{T}\sum_{t=1}^{T} \ell_t(\pi^t) + \sqrt{\frac{1}{T}}.$$

And so we find that one policy approximately solves the fixed point problem defined in terms of our loss function.

An instantiation. This is quite a generic approach and we have flexibility in the online learning algorithm and in how we design the loss function. Perhaps the most basic instantiation is to take the loss function $\ell_t(\hat{a}; s, a) = \mathbf{1}\{\hat{a} \neq a\}$ to be the zero-one loss and to instantiate the online learning algorithm as exponential weights over the policy class. Then we have that one of the policies we find satisfies

$$\mathbb{E}_{s\sim d^{\pi^\tau}} \mathbf{1}\{\pi^\tau(s) \neq \pi^\star(s)\} \leq \min_{\pi\in\Pi} \sum_{t=1}^{T} \mathbf{1}\{\pi(s^t) \neq \pi^\star(s^t)\} + O\left(\sqrt{\frac{\log|\Pi|}{T}}\right), \tag{1}$$

where the first term on the right hand side is 0 under realizability. The next theorem converts this to a policy performance bound.

Theorem 4 (DAgger guarantee)**.** *Given a finite policy class Π containing $\pi^\star$, DAgger finds a policy $\hat{\pi}$ such that, with probability at least $1-\delta$:*

$$J(\pi^\star) - J(\hat{\pi}) \leq \frac{1}{1-\gamma} \cdot \sup_{s,a} |A^\star(s,a)| \cdot \sqrt{\frac{\log(|\Pi|/\delta)}{n}}$$

using n queries from the expert. (Recall that $A^\star = A^{\pi^\star}$ is the advantage function.)

Proof. Compared with the analysis for BC, here we use the PD lemma in the other direction

$$J(\pi^\star) - J(\hat{\pi}) = \frac{1}{1-\gamma}\mathbb{E}_{(s,a)\sim d^{\hat{\pi}}}\left[A^\star(s,a)\right] \leq \frac{1}{1-\gamma} \cdot \sup_{s,a} |A^\star(s,a)| \cdot \mathbb{E}_{(s,a)\sim d^{\hat{\pi}}}\left[\mathbf{1}\{a \neq \pi^\star(s)\}\right].$$

We conclude the proof by taking $\hat{\pi} = \pi^\tau$ and applying (1). □

In the worst case, this is no better than BC, since $|A^\star(s,a)|$ could be as large as $\frac{1}{1-\gamma}$. However, $A^\star(s,a)$ provides a better instance-dependent measure of how easy it is to recover from mistakes. For example it could also be that $A^\star(s,a) = O(1)$ for all s, a. (While you can do the same thing in the BC analysis, you will end up with $\sup_{s,a} |A^{\hat{\pi}}(s,a)|$ and there is no reason to believe that this would be small, even if you can recover from mistakes.)

As a couple final remarks, the standard DAgger is implemented using a "follow-the-leader" (FTL) strategy for the online learner rather than exponential weights. This is conceptually much simpler, since we simply choose the policy $\pi^t = \operatorname{argmin} \sum_{\tau=1}^{t-1} \ell_\tau(\pi)$, which is typically much more computationally efficient. Further, it can be justified mathematically when the losses are strongly convex (in the policy parametrization), since in this case FTL does have sublinear regret. However, this clearly doesn't work with the 0/1 loss we used above.

There is also a variant calls AggreVaTe that collects full roll-outs from the expert policy to get estimates for the advantage function directly. Essentially this algorithm uses the loss function $\ell_t(a) = A^\star(s^t, a)$ where $s^t \sim d^{\pi^t}$ in a scheme that is quite similar to DAgger. In the realizable setting, the algorithm has a very similar guarantee. But if you run this algorithm with tabular parameterization and exponential weights as the online learner, it looks extremely similar to natural policy gradient, except that we use $A^\star$ rather that A^{π^t} in our update rule. As we saw, the analysis for both of these methods goes through the performance difference lemma, so perhaps the similarities are not so surprising.

3 Imitation learning in practice

Imitation learning is one of the more successful RL subfields in practice. The methods have been used in demonstrations of autonomous driving (in a game called "super tux cart"), to imitating human demonstrators playing video games, to robot off-road driving, and even in problems that look nothing like reinforcement learning at all.

An example of the latter is a problem like machine translation. Here we might have a dataset consisting of sentences in a source and target language and we want to learn a model that can translate unseen sentences (which is standard generalization). This is a standard supervised learning problem although the output/label space is combinatorially large. To deal with this, we often consider models that produce the output in a sequential fashion, such as one word at a time. When we do this, we can think of the problem as a sequential decision making problem. For imitation learning, we can use the labeled dataset to define an expert that can access the labels to make decisions. This means we only have access to the expert at training time, but the goal in imitation learning is that we should not require the expert at test time. So these "structured prediction" problems can be naturally cast as imitation learning, although it can be tricky to set up the state/action space and the expert policy.

The off-road driving example is also an interesting case study in how we can define an expert policy. Here, due to power constraints, we want the car to use weak sensors at test time and we need it to make decisions at high frequency, so computation is a real constraint. But at training time we can instrument the car with better sensors and use more computation to select better actions. So here the expert is just a computationally expensive approach that we hope to approximate using a much faster parametric policy.

These two examples highlight the flexibility in the IL formulation, as well as some issues that have been captured by more advanced theory. In the driving example, it's quite likely that the "expert" policy is not $\pi^\star$, since it is just some computationally expensive heuristic. Moreover, in both settings, its quite likely that the policy we are trying to imitate is not in our policy class Π. For structured prediction this is true whenever there is uncertainty about the label given the features, since the expert policy will have access to this extra randomness. Lack of realizability is covered by robust versions of the existing results, but perhaps there is more work to do in this direction.