CS 181 LECTURE 4/18/24

Scribe Notes

Contents

1 Last Time: Value-based RL

We learn the Q-value in value-based RL. The core construct that is updated is the Q-function and then we derive a policy.

There are two other categories, which were only briefly mentioned before, are: 1) model-based RL and 2) direct policy learning.

We will begin with high level algorithm. In real settings, there have been a lot of work that has been done to combine these approaches. There are ways that you can put these together but you can only do that after you understand the basic building blocks, first.

2 Model-based RL

Given our history h, we will learn $\hat{T}$ and $\hat{R}$, then we apply planning - which was covered last week. This works because once we have T and R, then we can apply MDP. Thus, this method is called model-based.

A couple of issues come up with this method.

1. How do we model uncertainty?

   (a) "certainty-equivalent" RL = we do not bother modeling the uncertainty.

   (b) keep a distribution over $p(T, R|h)$

   But then how do we plan if we do not know the distribution?

   One approach is Thompson Sampling.

   We sample $\hat{T}, \hat{R} \sim p(T, R|h)$. Then we solve MDP. We update the posterior $p(T, R|h)$.

   Intuitively, we have some uncertainty over possible models. Ex. maybe I should take an Uber or maybe I should take the bus or maybe I should walk to work. Let us say that we have a sample model that says I should take an Uber. When I sample a model, it was not clear what was going to come out. But then say we gather data and we realize that Ubers are too expensive. We have data planned under a bad model, so after we update the posterior, we will not be picking Ubers anymore.

   If you had an entire posterior distribution, this would be quite tricky to do. So the approach is taking a single sample. We know how to write an MDP for a single sample at a time. Then, we update and use that MDP only for a little bit. Thus, this approach is computationally easier than finding the posterior distribution over the entire history at once.

2. How do we incorporate optimism?

   There are algorithms that state that unobserved s, a should get a "boost" in R.

   Assume that anything you have never tried is amazing and should get infinity reward. The idea is that the action you have never tried will either be better or you learn something. You don't want to be pessimistic because then you will never discover something good. These models are called deep exploration.

There are some issues with this idea because if your world of actions and states is large then you may have too many options to try. But in general, in some way or another, we do want to boost unobserved s, a.

3 Policy-based RL

Let us introduce a little bit of new notation.

$\pi_\theta(a|s) =$ Pr of action a in state s

$r(h) = \sum r_t$(note that there is no γ because we are working with a finite trajectory)

$J(\theta) = \mathbb{E}_{\pi,T}[r(h)]$

The idea is that we set θ to $\theta + \alpha\Delta_\theta J$.

So, how do we compute $\Delta_\theta J$?

$$\begin{aligned}\Delta_\theta J &= \Delta_\theta \mathbb{E}_{\pi,T}[r(h)] \\ &= \Delta_\theta \int_h p_{\pi,T}(h) r(h) \qquad \text{use sum instead of integral if discrete} \\ &= \int_h \Delta_\theta p_{\pi,T}(h) r(h)\end{aligned}$$

We can use this trick: $p\Delta \log p = p \cdot \frac{1}{p}\Delta p = \Delta p$.

$$\Delta_\theta J = \int_h r(h) p_{\pi,T}(h) \Delta_\theta \log p_{\pi,T}(h)$$

Since $p_{\pi,T}(h) = \prod_t \pi_\theta(a_t|s_t) \cdot T(s_{t+1}|a_t, s_t)$, then taking the log, we have the following:

$$\begin{aligned}\Delta_\theta J &= \int_h r(h) p_{\pi,T}(h) \Delta_\theta [\sum_t \log \pi_\theta(a_t|s_t) + \sum_t \log T(s_{t+1}|a_t, s_t)] \\ &= \int_h [r(h) \Delta_\theta [\sum_t \log \pi_\theta(a_t|s_t)]] p_{\pi,T}(h) \\ &= \mathbb{E}_{p_{\pi,T}(h)}[r(h) \Delta_\theta [\sum_t \log \pi_\theta(a_t|s_t)]]\end{aligned}$$

Now, we can take trajectories from $h_i \sim p_{\pi,T}(h)$ and compute $\Delta_\theta J$. In reinforcement learning, we don't have access to T but we do have access to simulations. So, given a policy π, we can run some simulations.

So, this is a high-level look on policy-based RL.

4 Thinking About the Three Approaches in RL

4.1 Data Efficiency

Direct-policy requires the most amount of data because every step you require simulations to recompute the gradient. The other two approaches you can do a lot more work with the data you have. So, in terms of data efficiency, the best is **model-based approach, then value-based, then direct-policy**. With value-based, we only use the data once to make an update to the Q-function.

In a setting like healthcare, data is limited so we value data-efficiency over computational labor.

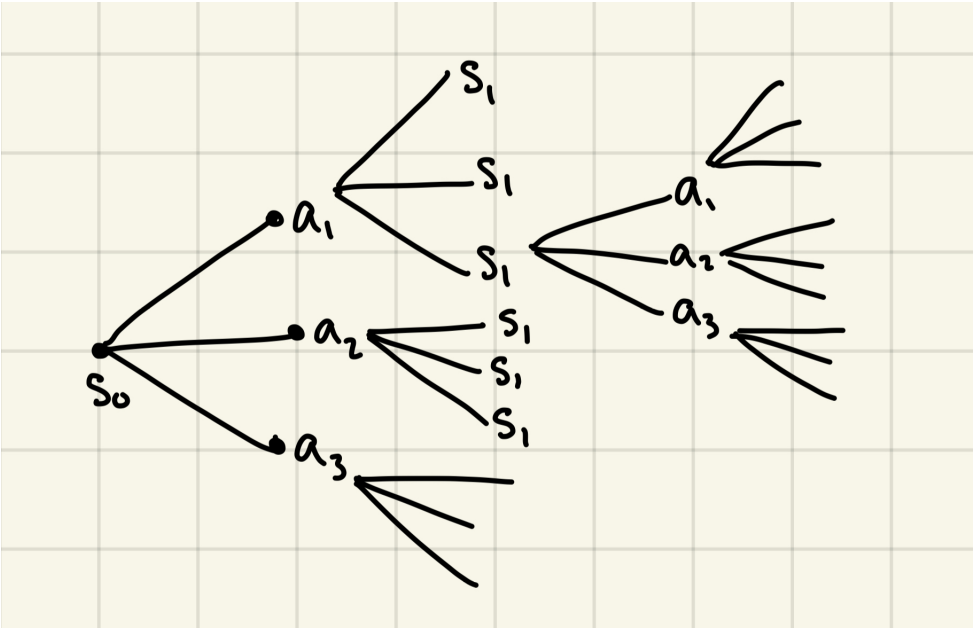
4.2 Computational Labor

Direct-policy is simple to implement. On the other hand, model-based RL takes a lot of work. So, the trade-off gives us an order of **direct-policy, value-based, model-based** for computational labor.

5 Games/Trees

In many cases, we have multiple agents! For example, two people playing each other or a team of robots on a search and rescue team. So, the multiple agents may be competitive or cooperative.

We have this action-state tree.

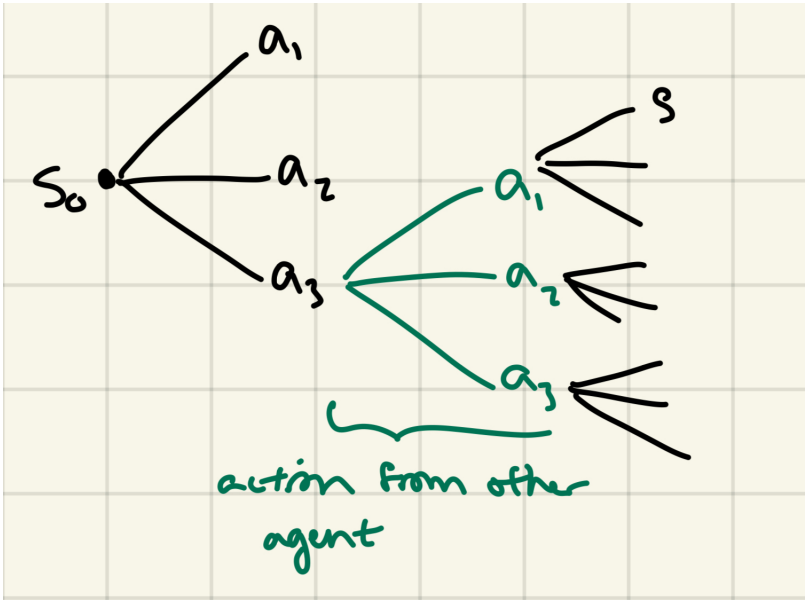


The problem is that this tree expands out exponentially. There are a couple of approaches that we can take.

At some point, we can stop expanding the tree and we estimate. We say that we have a way of playing and we stick to it. Or we use a $q(s, a)$ estimate to fill in.

If you think about the game of Go, there are so many options of actions that you can take. But a lot of them are quite bad - so you prune out those options and you only expand out the actions that you think could lead to a winning outcome.

In a multi-agent setting, then we add another layer.



In the famous game of Go, there was a moment where the computer had pruned for an action that the human did take. Thus, the computer did not prepare for the human's action. The human was able to put the AI in a board position that it had not seen before and was therefore not trained to compete in.

In the setting where you know what actions the other agent is playing, then we can treat the actions that the agents take as being influenced by all the other actions before it.

$$s \to a^1 \to a^2 \to a^3 \to s'$$

$$q(s, \underbrace{a^1}_{\text{known}}, \underbrace{a^2}_{\text{optimize}}, \underbrace{a^3}_{\text{base-policy}})$$

In other words, we know the actions that happened before a_2 and then we use a base-policy for actions that happen after.

6 Wrapping Up the Course

In this course, you received a sample of different techniques with ideas like overfitting and model selection that span all the different techniques. There were elements of this that were push-ups - the coding and taking derivatives. But, Professor Finale hopes that you take away the zen - learning core concepts like the bias-variance tradeoff. Another core idea is that there exists models of convenience and models that better reflect the real-world. Some models of convenience allow us to compute something that would would otherwise not been able to compute.

We learned the professional skills and ethics of doing ML. But there are other parts to consider such as: where does data come from? How is the AI/ML being used? So outside of the choices that you make while building the model, we need to think about societal ethics and responsibility. We need to think about how the ML model interacts with the world.