Lecture 6: The Policy Gradient

Ungraded Calculus Quiz

1. $\nabla_x \log x =$ ______________________

2. $\nabla_x \log f(x) =$ ______________________

3. $\nabla_\theta \log f_\theta(x) =$ ______________________

4. $\nabla_\theta \log p_\theta(x) =$ ______________________

   . $\implies \nabla_\theta p_\theta(x) =$ ______________________

5. $\nabla_\theta (R(x) \log p_\theta(x)) =$ ______________________

6. $\nabla_\theta \left(\int_{\mathcal{X}} R(x) \log p_\theta(x) dx \right) =$ ______________________

7. $\nabla_\theta \left(\int_{\mathcal{X}} R(x) p_\theta(x) dx \right) =$ ______________________

   . $= \mathbb{E}_{p_\theta(x)} \Big[\qquad \Big]$

8. $\nabla_\theta \mathbb{E}_{x \sim p_\theta(x)}[R(x)] =$ ______________________

   . $= \mathbb{E}_{p_\theta(x)} \Big[\qquad \Big]$

9. $\nabla_\theta \mathbb{E}_{\tau \sim p_\theta(\tau)}[R(\tau)] = \mathbb{E}_{p_\theta(\tau)} \Big[\qquad \Big]$

10. $\nabla_\theta \mathbb{E}_{\tau \sim p_\theta(\tau)}[\underbrace{\sum_{s_t, a_t \sim \tau} \gamma^t r(s_t, a_t)}_{R(\tau)}] = \mathbb{E}_{p_\theta(\tau)} \Big[\qquad \Big]$

1 Plan for today

Learning objectives

1. At the end of today's class, you will be able to take the derivative of an expectation w.r.t. parameters of the sampling distribution.
2. Derive the policy gradient and explain its intuition.
3. Implement an algorithm that maximizes rewards by doing gradient ascent using the policy gradient.

1.1 Review

1. What is the RL problem?
2. Do we really need RL?
   (a) Does time matter? If not, use a bandit method
   (b) Do you know the model of your system? [MPC, CEM]
   (c) Do you have expert data? If so, try imitation learning.
   (d) If all else fails, try RL!
3. The policy gradient and some simple RL methods
4. Value functions

2 Review: The RL Objective

Let's start by recalling the RL objective:

$$\max_\theta \mathbb{E}_{\substack{s_0 \sim p_0(s_0), a_t \sim \pi_\theta(a_t|s_t) \\ s_{t+1} \sim p(s_{t+1}|s_t, a_t)}} \left[\sum_t \gamma^t r(s_t, a_t) \right]. \tag{1}$$

This is sometimes call the *policy optimization* or *policy search* problem.

For simplicity, I'm going to introduce some notation that will make our lives easier. Let $\tau = (s_0, a_0, s_1, a_1, \cdots)$ be the sequence of states and actions, and let $\pi_\theta(\tau)$ denote the distribution over these trajectories. Formally, we can write this likelihood as

$$\pi_\theta(\tau) = p_0(s_0) \prod_{t=0}^{T} p(s_{t+1} \mid s_t, a_t) \pi_\theta(a_t \mid s_t). \tag{2}$$

We'll then define $R(\tau) = \sum_{t=0}^{T} \gamma^t r(s_t, a_t)$ as the discounted cumulative return of trajectory τ. With this notation in place, we can write the RL objective in a simpler form:

$$\max_\theta \mathbb{E}_{\pi_\theta(\tau)}[R(\tau)]. \tag{3}$$

3 The policy gradient [2]

Now, in today's class we're going to think about doing gradient ascent on this objective. The gradients are the derivatives, and they point towards increasing values of the function. If we knew the gradient $\nabla_x f(x)$ of function $f(x)$, we could optimize the function by iterating the following:

$$x \leftarrow x + \eta \nabla_x f(x). \tag{4}$$

Gradient ascent/descent is known as a **first-order method** because it uses the gradients (or *first*-derivatives) of the function.

To apply gradient ascent to the RL problem, we'll need to compute the gradient of the RL objective:

$$\nabla_\theta \mathbb{E}_{\pi_\theta(\tau)}[R(\tau)]. \tag{5}$$

Before going further, I want to emphasize that this should look different from what you typically do in supervised learning. In supervised learning, you typically sample data from a fixed dataset (e.g., $(x, y) \sim \mathcal{D}$) and then maximize the likelihood of that data:

$$\nabla_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}}[\log p_\theta(y \mid x)]. \tag{6}$$

For example, this is what we saw on Tuesday when we were looking at behavioral cloning. The key difference is that in supervised learning θ is inside the expectation; in reinforcement learning, θ controls the data distribution itself. This means that we'll have to be a bit careful when computing this derivative.

This expression also formalizes what we mean by "trial and error" learning: trials correspond to sampling trajectories, and "errors" correspond to learning from those trails and weighting each by their reward.

A common mistake. Many students attempt to compute the derivative of the policy gradient using the following, which is incorrect:

$$\nabla_\theta \mathbb{E}_{\pi_\theta(\tau)}[R(\tau)] = \mathbb{E}_{\pi_\theta(\tau)}[\cancelto{0}{\nabla_\theta R(\tau)}] = 0. \qquad \text{(Incorrect math.)}$$

The mistake here is that we cannot push the gradient operator inside the expectation because the expectation's distribution depends on θ, the variable that we are trying to take the gradient of.

3.1 The correct approach.

Instead, let's write out the expectation as an integral, which will make clear the dependence on θ

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{\pi_\theta(\tau)}[R(\tau)] &= \nabla_\theta \int R(\tau)\pi_\theta(\tau)d\tau & (7)\\
&= \int R(\tau)\nabla_\theta \pi_\theta(\tau)d\tau & (8)\\
&= \int R(\tau)\pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau)d\tau & (9)\\
&= \mathbb{E}_{\pi_\theta(\tau)}[R(\tau)\nabla_\theta \log \pi_\theta(\tau)]. & (10)
\end{aligned}$$

Note that we're allowed to push the gradient operator inside the integral because the bounds of the integral do not depend on θ. In the third line, we used the following identity, which you all derived at the beginning of class:

$$\nabla_\theta \log \pi_\theta(\tau) = \frac{1}{\pi_\theta(\tau)}\nabla_\theta \pi_\theta(\tau) \implies \nabla_\theta \pi_\theta(\tau) = \pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau). \tag{11}$$

Some people memorize this identity. I find it easier to just derive it on the fly. The most important thing to note about this objective is that the gradient is expressed as an expectation. This is important because it means that we can *approximate* this gradient using Monte Carlo samples:

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{\pi_\theta(\tau)}[R(\tau)] &= \mathbb{E}_{\pi_\theta(\tau)}[R(\tau)\nabla_\theta \log \pi_\theta(\tau)] & (12)\\
&\approx \frac{1}{k}\sum_{\tau\sim\pi_\theta(\tau)} R(\tau)\nabla_\theta \log \pi_\theta(\tau). & (13)
\end{aligned}$$

Note that we really needed to use the identity above to make this happen. If we just looked at the gradient of $\nabla_\theta \pi_\theta(\tau)$, this would not be true:

$$\int R(\tau)\nabla_\theta \pi_\theta(\tau)d\tau \approx \frac{1}{k}\sum_{\tau\sim\pi_\theta(\tau)} R(\tau)\nabla_\theta \pi_\theta(\tau). \qquad \text{(Incorrect math.)}$$

3.2 Breaking up time.

OK, so this gives us almost all of what we want. Our final step is to express this gradient in terms of actions, rather than entire trajectories. To do this, we start by noting that

$$\log \pi_\theta(\tau) = \log p_0(s_0) + \sum_{t=0}^{T} \left(\log p(s_{t+1} \mid s_t, a_t) + \log \pi_\theta(a_t \mid s_t)\right). \tag{14}$$

We'll next take the gradient w.r.t. θ. Note that many of these terms don't depend on θ, and hence their gradients are zero:

$$\begin{aligned}
\nabla_\theta \log \pi_\theta(\tau) &= \cancelto{0}{\nabla_\theta \log p_0(s_0)} + \sum_{t=0}^{T} \left(\cancelto{0}{\nabla_\theta \log p(s_{t+1} \mid s_t, a_t)} + \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right) && (15)\\
&= \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t). && (16)
\end{aligned}$$

Plugging this into Eq. 13, we get

$$\begin{aligned}
\nabla_\theta f(\theta) &= \int (\nabla_\theta \pi_\theta(\tau)) R(\tau)\, d\tau && (17)\\
&= \int \pi_\theta(\tau) \frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} R(\tau)\, d\tau && (18)\\
&= \mathbb{E}_{\tau \sim \pi_\theta(\tau)}[(\nabla_\theta \log \pi_\theta(\tau)) R(\tau)] && (19)\\
&= \mathbb{E}_{\tau \sim \pi_\theta(\tau)}[(\nabla_\theta \log \prod_{t=0}^{T} p(s_{t+1} \mid s_t, a_t) \pi_\theta(a_t|s_t)) p(s_0) R(\tau)] && (20)\\
&= \mathbb{E}_{\tau \sim \pi_\theta(\tau)}[(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t)) R(\tau)] && (21)\\
&= \mathbb{E}_{\tau \sim \pi_\theta(\tau)}[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \sum_{t'=0}^{T} \gamma^{t'} r(s_{t'}, a_{t'})] && (22)\\
&= \sum_{t=0}^{T} \mathbb{E}_{\tau \sim \pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(a_t \mid s_t) (\sum_{t'=0}^{t-1} \gamma^{t'} r(s_{t'}, a_{t'}) + \sum_{t'=t}^{T} \gamma^{t'} r(s_{t'}, a_{t'}))] && (23)\\
&= \sum_{t=0}^{T} \Big[\mathbb{E}_{\tau \sim \pi_\theta(\tau)}[\sum_{t'=t}^{T} \gamma^{t'} r(s_{t'}, a_{t'}) \nabla_\theta \log \pi_\theta(a_t \mid s_t)] && (24)\\
&\qquad + \cancelto{0}{\mathbb{E}_{\tau \sim \pi_\theta(\tau)}[\sum_{t'=0}^{t-1} \gamma^{t'} r(s_{t'}, a_{t'})] \mathbb{E}_{\tau \sim \pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(a_t \mid s_t)]} \Big] && (25)\\
&= \sum_{t=0}^{T} \mathbb{E}_{\tau \sim \pi_\theta(\tau)}[\sum_{t'=t}^{T} \gamma^{t'} r(s_{t'}, a_{t'}) \nabla_\theta \log \pi_\theta(a_t \mid s_t)] && (26)\\
&= \mathbb{E}_{\tau \sim \pi_\theta(\tau)}[\sum_{t'=t}^{T} \gamma^{t'} r(s_{t'}, a_{t'}) \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t)]. && (27)
\end{aligned}$$

In (21), the gradient of the transition terms goes to zero as they have no dependence on θ, so we remove those terms from the summation. In (22), we did a change of variables to avoid confusion about the time indices for the two summations (one for the reward, another for the grad log policy).

In (25), we note that the rewards we take before time step t should not influence our policy at time t, so we can ignore this term.[1]

One of the key challenges with policy gradient methods is decreasing the variance of the gradient estimator. The reason we removed the rewards before time t above, even though we still get a valid gradient estimator if we include that term, is because removing it decreases variance. In future lecture, we'll see how including a value function term can further reduce variance.

3.3 A Complete RL Algorithm

This is sometimes called the "likelihood ratio gradient" estimator. A complete RL algorithm based on this gradient estimator:

1. Initialize policy $\pi_\theta(a \mid s)$
2. Collect some trajectories τ using π_θ
3. Compute the policy gradient, ∇_θ
4. Do one step of gradient ascent: $\theta \leftarrow \theta + \eta \nabla_\theta$
5. Go back to step 2.

This algorithm is known as REINFORCE [2]. You'll implement it on your next homework. It works fairly well for simple problems.

3.4 Interpretation: reward weighted regression [1]

One way to interpret the policy gradient is as doing weighted imitation learning. Recall the behavioral cloning objective as its gradient:

$$\max_\theta \mathbb{E}_{(s,a)\sim\mathcal{D}}[\log \pi_\theta(a \mid s)] \tag{28}$$

$$\nabla_\theta \mathbb{E}_{(s,a)\sim\mathcal{D}}[\log \pi_\theta(a \mid s)] = \mathbb{E}_{(s,a)\sim\mathcal{D}}[\nabla_\theta \log \pi_\theta(a \mid s)]. \tag{29}$$

Thus, the policy gradient looks like a weighted version of behavioral cloning. Note that this is one of the ideas we saw in the last lecture for trying to beat the expert when doing imitation learning. One important difference is that, when doing the policy gradient, we're sampling the data from the policy itself and then doing the reweighting. It's because we sample data from the policy itself that we call the policy gradient method an *on-policy* RL algorithm.

This interpretation is also useful because it helps explain how you might implement REINFORCE: by implementing behavioral cloning, but then weighting each loss term by its corresponding reward. The "predictions" are the output of your policy, the "labels" are the actions actually taken, the "loss" is the cross entropy loss (for discrete actions) or maximum likelihood loss (for arbitrary action distributions).

4 Zero-Order Policy Optimization: Another Perspective on "Policy" Gradients

Let's return to zero order optimization from last week, where we were trying to optimize a function

$$\max_x f(x). \tag{30}$$

The key idea of CEM was that we kept a distribution over the variable that we were optimizing. Last week, we were optimizing sequences of actions, and we used a Gaussian distribution.

[1] This can be proven formally: the expectation of the term that we're removing here is exactly zero. See OpenAI's Spinning Up for more information.

Here's another application of CEM: rather than optimizing action sequences, use it to optimize the parameters of a policy:

$$f(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_t \gamma^t r(s_t, a_t)\right]. \tag{31}$$

This works, and is very simple, and you'll implement it on your homework!

Last time we discussed CEM, I mentioned that it was a zero-order method: it didn't involve gradients. However, I hinted that it has close connections to first order methods. Let's dive into that.

To do that, let's think about optimizing the function $f(x)$ and using $p(x) = \mathcal{N}(\mu, \sigma = 1)$ as our sampling distribution. CEM is optimizing the mean parameter μ. CEM would ordinarily sample a bunch of candidate values $x^{(i)} \sim p(x)$, evaluate the function on each of them, and then update μ to match the subset of $x^{(i)}$ which had highest values of $f(x^{(i)})$.

We'll now see how we can reinterpret this in terms of the policy gradient:

$$\max_\mu \mathbb{E}_{p_\mu(x)}[f(x)]. \tag{32}$$

We now know how to compute the gradient of this objective:

$$\nabla_\mu \mathbb{E}_{p(x)}[f(x)] = \mathbb{E}_{p_\mu(x)}[f(x)\nabla_\mu p_\mu(x)] \tag{33}$$

$$= \mathbb{E}_{p_\mu(x)}[f(x)\frac{1}{2}\|\mu - x\|_2^2]. \tag{34}$$

Let's see what it looks like when we take the derivative of this function and set it to zero:

$$\nabla_\mu = 0 \implies \mathbb{E}[f(x)]\mu = \mathbb{E}[f(x)x] \tag{35}$$

$$\implies \mu = \frac{\mathbb{E}[f(x)x]}{\mathbb{E}[f(x)]} \approx \frac{\frac{1}{k}\sum_{i=1}^k f(x^{(i)})x^{(i)}}{\frac{1}{k}\sum_{i=1}^k f(x^{(i)})} \approx \sum_{i=1}^k \frac{\frac{1}{k}f(x^{(i)})}{\frac{1}{k}\sum_{j=1}^k f(x^{(j)})} x^{(i)}. \tag{36}$$

Thus, this is just doing a *weighted average* of the the samples that we've seen. Note that if $f \in \{0, 1\}$, this looks exactly like the CEM method that we discussed last time. So, in short, we can use the policy gradient to show that CEM is actually a first order method!

5 The Expectation Maximization (EM) Perspective.

There's a third perspective of the policy gradient, which I'll try to touch upon more in a future lecture, based upon expectation maximization [1]. The key idea is to write our the RL objective (again, in terms of trajectories) and compute an evidence lower bound on this objective. For this, we will assume (without loss of generality) that the returns are positive:

$$\arg\max_\theta \mathbb{E}_{\pi_\theta(\tau)}[R(\tau)] = \arg\max_\theta \log \mathbb{E}_{\pi_\theta(\tau)}[R(\tau)] \tag{37}$$

$$= \log \mathbb{E}_{\pi_\theta(\tau)}[R(\tau)\frac{q(\tau)}{q(\tau)}] \tag{38}$$

$$\geq \mathbb{E}_{q(\tau)}[\log R(\tau) + \log p_\theta(\tau) - \log q(\tau)]. \tag{39}$$

This lower bound holds for any choice of distribution $q(\tau)$, and achieves equality when $q(\tau) = \frac{R(\tau)p_\theta(\tau)}{\int R(\tau')p_\theta(\tau')d\tau'}$.

Thus, we can think about optimizing this lower bound on the RL objective w.r.t. two terms: the policy parameters θ and the data distribution $q(\tau)$. I'm calling this the data distribution because it's the thing we're using to sample the trajectories.

$$\max_\theta \log \mathbb{E}_{\pi_\theta(\tau)}[R(\tau)] \geq \max_{\theta, q(\tau)} \mathbb{E}_{q(\tau)}[\log R(\tau) + \log p_\theta(\tau) - \log q(\tau)] \tag{40}$$

This is neat because it highlights the key difference between RL and supervised learning: in RL you optimize the data distribution!

6 Outlook

Today's class was all about policy optimization. However, today's class is about as far as you can get without additional machinery. For practical problems, you'll need another ingredient: the Q-function (also known as the value function). We will introduce this next time.

References

[1] 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.

[2] Williams, R. J. and Peng, J. (1991). Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268.