

# CS285 (Deep Reinforcement Learning) Notes [Fall 2020]

Patrick Yin

Updated May 28, 2021

# Contents

<b>1</b>	<b>Note</b>	<b>2</b>
1.1	Note . . . . .	2
<b>2</b>	<b>Imitation Learning</b>	<b>3</b>
2.1	Imitation Learning . . . . .	3
2.2	Goal-Conditioned Behavioral Cloning . . . . .	4
<b>3</b>	<b>Reinforcement Learning</b>	<b>5</b>
3.1	Definitions . . . . .	5
3.2	RL Algorithm Anatomy . . . . .	6
3.3	Value Functions . . . . .	6
3.4	Types of Algorithms . . . . .	7
3.5	Tradeoffs Between Algorithms . . . . .	8
<b>4</b>	<b>Policy Gradient</b>	<b>10</b>
4.1	Direct Policy Differentiation . . . . .	10
4.2	Understanding Policy Gradients . . . . .	11
4.3	Reducing Variance . . . . .	12
4.4	Off-Policy Policy Gradients . . . . .	14
4.5	Advanced Policy Gradients . . . . .	15

# Chapter 1

## Note

### 1.1 Note

These course notes are my notes from CS 285 : Deep Reinforcement Learning taught by Professor Sergey Levine. The course is linked [here](#). I did not formally take this course, but self-studied the material via the lectures posted published on YouTube. These notes are currently in progress.

## Chapter 2

# Imitation Learning

### 2.1 Imitation Learning

Imitation learning has the issue of distributional drift. We can solve this in two ways.

The first way is just to mimic the expert so accurately so that it doesn't drift. Failing to fit the expert accurately could be due to non-markovian and/or multimodal behavior. For the former problem, we can consider history with an RNN. For the later problem, we can output a MoG, use latent variable models, or use autoregressive discretization.

The second way is to generate more data so that the training distribution matches the policy trajectory distribution. DAgger, Dataset Aggregation, does this by collecting data from  $p_{\pi_\theta}(o_t)$  instead of  $p_{data}(o_t)$ .

---

**Algorithm 1** DAgger

---

- 1: train  $\pi_\theta(a_t|o_t)$  from human data  $\mathcal{D} = \{o_1, a_1, \dots, o_N, a_N\}$
  - 2: run  $\pi_\theta(a_t|o_t)$  to get dataset  $\mathcal{D}_\pi = \{o_1, \dots, o_M\}$
  - 3: ask human to label  $\mathcal{D}_\pi$  with actions  $a_t$
  - 4: aggregate:  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$
- 

Let us prove theoretically why the error with DAgger is an order of magnitude lower than that of traditional behavioral cloning. Define

$$c(s, a) = \begin{cases} 0 & a = \pi^*(s) \\ 1 & \text{otherwise} \end{cases}$$

Assuming that  $\pi_\theta(a \neq \pi^*(s)|s) \leq \epsilon$  for all  $s$  in  $\mathcal{D}_{train}$ ,

$$\mathbb{E}[\sum_t c(s_t, a_t)] \leq \epsilon T + (1 - \epsilon)\epsilon T + (1 - \epsilon)^2 \epsilon T + \dots = \mathcal{O}(\epsilon T^2)$$

More generally, let us instead assume that  $\pi_\theta(a \neq \pi^*(s)|s) \leq \epsilon$  for all  $s \sim p_{train}(s)$ . With DAGger, since  $p_{train}(s) \rightarrow p_\theta(s)$ ,

$$\mathbb{E}[\sum_t c(s_t, a_t)] \leq \epsilon T$$

With behaviorial cloning, we can compute the probability distribution of the a state under the current policy in terms of the probability distribution of training data as such:

$$p_\theta(s_t) = (1 - \epsilon)^t p_{train}(s_t) + (1 - (1 - \epsilon)^t) p_{mistake}(s_t)$$

so,

$$|p_\theta(s_t) - p_{train}(s_t)| = (1 - (1 - \epsilon)^t) |p_{mistake}(s_t) - p_{train}(s_t)| \leq 2(1 - (1 - \epsilon)^t) \leq 2\epsilon t$$

Then, we know that

$$\begin{aligned} \sum_t \mathbb{E}_{p_\theta(s_t)}[c_t] &= \sum_t \sum_{s_t} p_\theta(s_t) c_t(s_t) \leq \sum_t \sum_{s_t} p_{train}(s_t) c_t(s_t) + |p_\theta(s_t) - p_{train}(s_t)| c_{max} \\ &\leq \sum_t \epsilon + 2\epsilon t = \mathcal{O}(\epsilon T^2) \end{aligned}$$

It turns out we can get the same bounds with the looser assumption that  $\mathbb{E}_{p_{train}(s)}[\pi_\theta(a \neq \pi^*(s)|s)] \leq \epsilon$ , but we won't prove this here.

## 2.2 Goal-Conditioned Behaviorial Cloning

For a policy to reach any goal  $p$ , which may not be in the training dataset, we can condition our policy on  $p$ . In other words, we collect data and train a goal conditioned policy with the last state being the goal state. In "Learning to Reach Goals via Iterated Supervised Learning", the authors start with a random policy, collect data with random goals, treat this data as demonstrations for the goals that were reached, used this to improve the policy, and repeated.

## Chapter 3

# Reinforcement Learning

### 3.1 Definitions

MDP is defined by  $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, r\}$ , and POMDP is defined by  $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{E}, r\}$  where  $\mathcal{E}$  gives the emission probability  $p(o_t|s_t)$ . Let us define

$$p_\theta(\tau) := p_\theta(s_1, a_1, \dots, s_T, a_T) = p(s_1) \prod_{t=1}^T \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$$

Then the optimal RL policy is

$$\theta^* = \arg \max_{\theta} \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[ \sum_t r(s_t, a_t) \right]$$

For convenient, let us rewrite this expression in terms of state-action marginals:

$$\theta^* = \arg \max_{\theta} \sum_{t=1}^T \mathbb{E}_{(s_t, a_t) \sim p_\theta(s_t, a_t)} [r(s_t, a_t)]$$

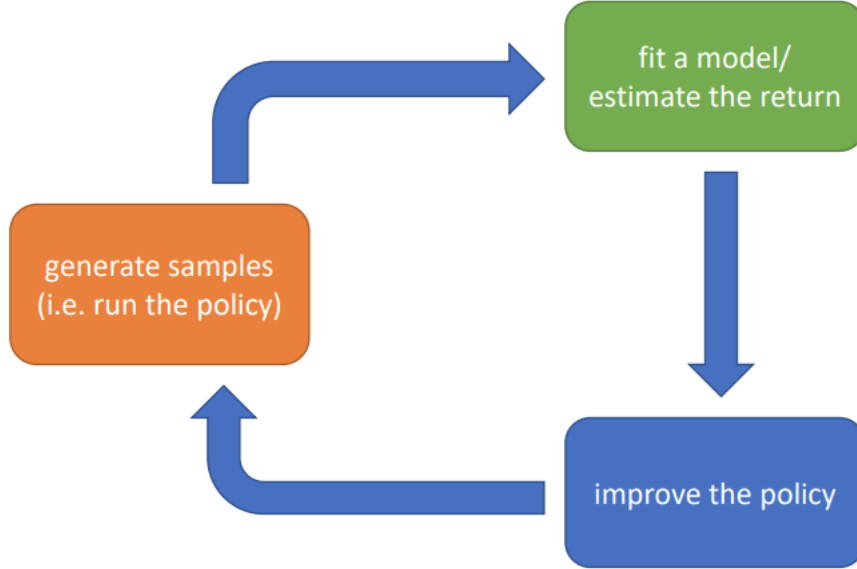
where  $p_\theta(s_t, a_t)$  is our state-action marginal. Our "new" MC now has the transition probability  $p((s_{t+1}, a_{t+1})|(s_t, a_t)) = p(s_{t+1}|s_t, a_t) \pi_\theta(a_{t+1}|s_{t+1})$ .

In the infinite horizon case (i.e. when  $T = \infty$ ),  $p(s_t, a_t)$  converges to a stationary distribution if it is ergodic. If so, then the stationary distribution  $\mu$  satisfies  $\mu = \mathcal{T}\mu$ , so  $\mu := p_\theta(s, a)$  is the eigenvector of  $\mathcal{T}$  with eigenvalue 1. If rewards are positive, the reward sum can go to infinity, so we also divide by  $T$ . Thus, in the infinite horizon case, our optimal RL policy is

$$\theta^* = \arg \max_{\theta} \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{(s_t, a_t) \sim p_\theta(s_t, a_t)} [r(s_t, a_t)] \rightarrow \mathbb{E}_{(s, a) \sim p_\theta(s, a)} [r(s, a)]$$

Also note that we care about expectations over the reward accrued because it is smooth in  $\theta$ .

### 3.2 RL Algorithm Anatomy



### 3.3 Value Functions

Remember that our objective function is

$$\mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[ \sum_t r(s_t, a_t) \right]$$

This can be rewritten as

$$\mathbb{E}_{s_1 \sim p(s_1)} \left[ \mathbb{E}_{a_1 \sim \pi(a_1|s_1)} \left[ r(s_1, a_1) + \mathbb{E}_{s_2 \sim p(s_2|s_1, a_1)} \left[ \mathbb{E}_{a_2 \sim \pi(a_2|s_2)} \left[ r(s_2, a_2) + \dots |s_2 \right] |s_1, a_1 \right] |s_1 \right] \right]$$

If we define

$$Q(s_1, a_1) = r(s_1, a_1) + \mathbb{E}_{s_2 \sim p(s_2|s_1, a_1)} \left[ \mathbb{E}_{a_2 \sim \pi(a_2|s_2)} \left[ r(s_2, a_2) + \dots |s_2 \right] |s_1, a_1 \right]$$

then

$$\mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[ \sum_t r(s_t, a_t) \right] = \mathbb{E}_{s_1 \sim p(s_1)} \left[ \mathbb{E}_{a_1 \sim \pi(a_1|s_1)} \left[ Q(s_1, a_1) |s_1 \right] \right]$$

This "Q-function" is very useful as if we know it, we can easily improve the policy. So, we will define a Q-function as such

$$Q^{\pi}(s_t, a_t) := \sum_{t'=t}^T \mathbb{E}_{\pi_{\theta}} \left[ r(s_{t'}, a_{t'}) |s_t, a_t \right]$$

and a value function as such

$$V^\pi(s_t) := \sum_{t'=t}^T \mathbb{E}_{\pi_\theta} [r(s_{t'}, a_{t'}) | s_t] = \mathbb{E}_{a_t \sim \pi(a_t | s_t)} [Q^\pi(s_t, a_t)]$$

Note that  $\mathbb{E}_{s_1 \sim p(s_1)} [V^\pi(s_1)]$  is the RL objective.

Using Q-functions and value functions, we have two high-level ideas of how we can improve our policy  $\pi$  over time:

1. Set  $\pi'(a|s) = 1$  if  $a = \arg \max_a Q^\pi(s, a)$ . This is at least as good as  $\pi$ .
2. If  $Q^\pi(s, a) > V^\pi(s)$ , then  $a$  is better than average. Modify  $\pi(a|s)$  to increase probability of  $a$  if  $Q^\pi(s, a) > V^\pi(s)$ .

### 3.4 Types of Algorithms

RL Algorithm Types			
RL Type	Description	Fit model and/or estimate return	Improve policy
Policy gradient	directly differentiate the above objective	evaluate returns $R_\tau = \sum_t r(s_t, a_t)$	$\theta \leftarrow \theta + \alpha \nabla_\theta \mathbb{E}[\sum_t r(s_t, a_t)]$
Value-based	estimate value function or Q-function of the optimal policy (no explicit policy)	fit $V(s)$ or $Q(s, a)$	set $\pi(s) = \arg \max_a Q(s, a)$
Actor-critic	estimate value function or Q-function of the current policy, use it to improve policy	fit $V(s)$ or $Q(s, a)$	$\theta \leftarrow \theta + \alpha \nabla_\theta \mathbb{E}[Q(s_t, a_t)]$
Model-based	estimate transition model and use it for planning, to improve a policy, etc.	learn $p(s_{t+1}   s_t)$	<p>A few ways:</p> <ol style="list-style-type: none"> <li>1. Just use the model to plan (no policy), such as trajectory optimization/optimal control (e.g. backprop in continuous space or Monte Carlo tree search in discrete space)</li> <li>2. Backprop gradients into policy</li> <li>3. Use model to learn a value function with DP or by generating simulated experience for model-free learner</li> </ol>



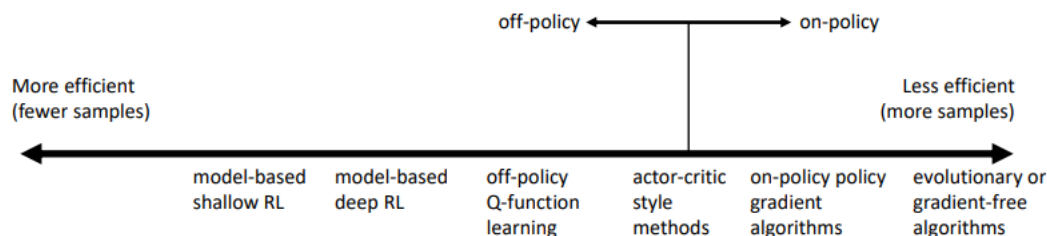
## 3.5 Tradeoffs Between Algorithms

There are many RL algorithms because there are

- Different tradeoffs: sample efficiency, stability and ease of use
- Different assumptions: stochastic or deterministic, continuous or discrete, episodic or infinite horizon
- Different things are easy or hard in different settings: easier to represent the policy, easier to represent the model

### 3.5.1 Sample Efficiency

Sample efficiency spectrum:



Off-policy means being able to improve the policy without generating new samples from the policy. On-policy means that each time the policy is changed, even a little bit, we need to generate new samples. Note that wall clock time is not the same as sample efficiency.

### 3.5.2 Stability and Ease of Use

Unlike supervised learning, RL often does not use gradient descent so there are not as much convergence guarantees:

- Value function fitting: At best, Bellman error (i.e. error of fit) is minimized. At worst, nothing is optimized (many not guaranteed to converge to anything in nonlinear case)
- Model-based RL: Model minimizes error of fit. But no guarantee better model = better policy.
- Policy gradient: The only one that actually performs gradient descent (or ascent) on the true objective.

### 3.5.3 Common Assumptions

1. Full observability
  - (a) Generally assumed by value function fitting methods
  - (b) Can be mitigated by adding recurrence
2. Episodic learning
  - (a) Often assumed by pure policy gradient methods
  - (b) Assumed by some model-based RL methods
3. Continuity/Smoothness
  - (a) Assumed by some continuous value function learning methods
  - (b) Often assumed by some model-based RL methods

## Chapter 4

# Policy Gradient

### 4.1 Direct Policy Differentiation

Remember that in RL

$$\theta^* = \arg \max_{\theta} \mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[ \sum_t r(s_t, a_t) \right]$$

For simplicity, define

$$J(\theta) := \mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[ \sum_t r(s_t, a_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(s_{i,t}, a_{i,t})$$

and define

$$r(\tau) = \sum_{t=1}^T r(s_t, a_t)$$

so, then

$$J(\theta) := \mathbb{E}_{\tau \sim p_{\theta}(\tau)} [r(\tau)] = \int p_{\theta}(\tau) r(\tau) d\tau$$

A convenient identity we will use quite a bit for the future is that

$$p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau) = p_{\theta}(\tau) \frac{\nabla_{\theta} p_{\theta}(\tau)}{p_{\theta}(\tau)} = \nabla_{\theta} p_{\theta}(\tau)$$

Using this identity, we find that

$$\nabla_{\theta} J(\theta) = \int \nabla_{\theta} p_{\theta}(\tau) r(\tau) d\tau = \int p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau) r(\tau) d\tau = \mathbb{E}_{\tau \sim p_{\theta}(\tau)} [\nabla_{\theta} \log p_{\theta}(\tau) r(\tau)]$$

We know that

$$\begin{aligned}
p_\theta(\tau) &= p(s_1) \prod_{t=1}^T \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t) \\
\log p_\theta(\tau) &= \log p(s_1) + \sum_{t=1}^T \log \pi_\theta(a_t|s_t) + \log p(s_{t+1}|s_t, a_t) \\
\nabla_\theta \log p_\theta(\tau) &= \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_t|s_t)
\end{aligned}$$

Now, we have a nicer expression for  $\nabla_\theta J(\theta)$ :

$$\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[ \left( \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_t|s_t) \right) \left( \sum_{t=1}^T r(s_t, a_t) \right) \right] \\
&\approx \frac{1}{N} \sum_{i=1}^N \left( \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left( \sum_{t=1}^T r(s_{i,t}, a_{i,t}) \right)
\end{aligned}$$

Now if we simply do gradient ascent using this gradient, we have our most basic policy gradient algorithm REINFORCE:

---

**Algorithm 2** REINFORCE

---

```

1: loop
2:   sample  $\{\tau^i\}$  from  $\pi_\theta(a_t|s_t)$  (run the policy)
3:    $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left( \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left( \sum_{t=1}^T r(s_{i,t}, a_{i,t}) \right)$ 
4:    $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$ 
5: end loop

```

---

## 4.2 Understanding Policy Gradients

In policy gradients, the gradient is

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left( \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left( \sum_{t=1}^T r(s_{i,t}, a_{i,t}) \right)$$

In MLE, the gradient is

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left( \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right)$$

Intuitively, REINFORCE is then just trial and error learning where good stuff is made more likely and bad stuff is made less likely.

Also note that if we follow the same derivation with partial observability, we get

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left( \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_{i,t} | o_{i,t}) \right) \left( \sum_{t=1}^T r(s_{i,t}, a_{i,t}) \right)$$

so we can use policy gradient on POMDPs without modification.

## 4.3 Reducing Variance

The current policy gradient won't work in practice because the variance of the gradient is too big. We can implement some tricks to reduce the variance of the gradient while keeping the estimate unbiased.

### 4.3.1 Causality

Through some derivation, we can show that

$$\mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[ \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \left( \sum_{t'=1}^t r(s'_{t'}, a'_{t'}) \right) \right] = 0$$

Intuitively, this makes sense since the policy at time  $t'$  cannot affect reward at time  $t$  when  $t < t'$ . So,

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_{i,t} | s_{i,t}) \left( \sum_{t'=t}^T r(s_{i,t'}, a_{i,t'}) \right)$$

is the same as our original approximation in expectation and now has smaller variance. We will define the right-hand expression as our "reward to go"  $\hat{Q}_{i,t} = \sum_{t'=t}^T r(s_{i,t'}, a_{i,t'})$ .

### 4.3.2 Baselines

#### Average Reward

It turns out that

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log p_{\theta}(\tau) [r(\tau) - b]$$

is also a valid approximation where

$$b = \frac{1}{N} \sum_{i=1}^N r(\tau)$$

This is because

$$\begin{aligned}
\mathbb{E}[\nabla_{\theta} \log p_{\theta}(\tau)b] &= \int p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau) b d\tau \\
&= \int \nabla_{\theta} p_{\theta}(\tau) b d\tau \\
&= b \nabla_{\theta} \int p_{\theta}(\tau) d\tau \\
&= b \nabla_{\theta} 1 = 0
\end{aligned}$$

In other words, subtracting a baseline is also unbiased in expectation. Note that average reward is not the best baseline, but it's still pretty good.

### Optimal Baseline

The best baseline minimizes the variance of the gradient, which is

$$\text{Var} = \mathbb{E}_{\tau \sim p_{\theta}(\tau)}[(\nabla_{\theta} \log p_{\theta}(\tau)(r(\tau) - b))^2] - \mathbb{E}_{\tau \sim p_{\theta}(\tau)}[\nabla_{\theta} \log p_{\theta}(\tau)(r(\tau) - b)]^2$$

For convenience, let

$$g(\tau) := \nabla_{\theta} \log p_{\theta}(\tau)$$

To minimize this variance, we take a derivative of the variance and set it to zero. Note that the latter part of the variance expression is just  $\mathbb{E}_{\tau \sim p_{\theta}(\tau)}[\nabla_{\theta} \log p_{\theta}(\tau)r(\tau)]$  because baselines are unbiased in expectation, so its derivative in terms of  $b$  is just zero.

$$\begin{aligned}
\frac{\partial \text{Var}}{\partial b} &= \frac{\partial}{\partial b} \mathbb{E}[g(\tau)^2(r(\tau) - b)^2] \\
&= \frac{\partial}{\partial b} (\mathbb{E}[g(\tau)^2 r(\tau)^2] - 2\mathbb{E}[g(\tau)^2 r(\tau)b] + b^2 \mathbb{E}[g(\tau)^2]) \\
&= -2\mathbb{E}[g(\tau)^2 r(\tau)] + 2b \mathbb{E}[g(\tau)^2] = 0 \\
b &= \frac{\mathbb{E}[g(\tau)^2 r(\tau)]}{\mathbb{E}[g(\tau)^2]}
\end{aligned}$$

So the optimal baseline is just the expected reward, but weighted by gradient magnitudes.

## 4.4 Off-Policy Policy Gradients

In off-policy policy gradients, we want to improve our policy with data generated from a different policy. We can do this with importance sampling:

$$\begin{aligned}\mathbb{E}_{x \sim p(x)}[f(x)] &= \int p(x)f(x)dx \\ &= \int \frac{q(x)}{p(x)}p(x)f(x)dx \\ &= \int q(x)\frac{p(x)}{q(x)}f(x)dx \\ &= \mathbb{E}_{x \sim q(x)}\left[\frac{p(x)}{q(x)}f(x)\right]\end{aligned}$$

In our case, we want to improve policy  $\theta'$  with data generated from policy  $\theta$ . So,

$$\begin{aligned}\nabla_{\theta'} J(\theta') &= \mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[ \frac{p_{\theta'}(\tau)}{p_{\theta}(\tau)} \nabla_{\theta'} \log \pi_{\theta'}(\tau) r(\tau) \right] \\ &= \mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[ \left( \prod_{t=1}^T \frac{\pi_{\theta'}(a_t | s_t)}{\pi_{\theta}(a_t | s_t)} \right) \left( \sum_{t=1}^T \nabla_{\theta'} \log \pi_{\theta'}(a_t | s_t) \right) \left( \sum_{t=1}^T r(s_t, a_t) \right) \right] \\ &= \mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[ \sum_{t=1}^T \nabla_{\theta'} \log \pi_{\theta'}(a_t | s_t) \left( \prod_{t'=1}^t \frac{\pi_{\theta'}(a_{t'} | s_{t'})}{\pi_{\theta}(a_{t'} | s_{t'})} \right) \left( \sum_{t'=t}^T r(s_{t'}, a_{t'}) \left( \prod_{t''=t}^{t'} \frac{\pi_{\theta'}(a_{t''} | s_{t''})}{\pi_{\theta}(a_{t''} | s_{t''})} \right) \right) \right]\end{aligned}$$

If we ignore the importance weight ratios over  $t''$ , we get a policy iteration algorithm (more on this later), so we can simplify the gradient to

$$\nabla_{\theta'} J(\theta') = \mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[ \sum_{t=1}^T \nabla_{\theta'} \log \pi_{\theta'}(a_t | s_t) \left( \prod_{t'=1}^t \frac{\pi_{\theta'}(a_{t'} | s_{t'})}{\pi_{\theta}(a_{t'} | s_{t'})} \right) \left( \sum_{t'=t}^T r(s_{t'}, a_{t'}) \right) \right]$$

Note that the importance weight ratios are exponential in  $T$ , so its variance will grow exponentially in  $T$ . Instead, let us write the objective in terms of state-action marginals, so

$$\begin{aligned}\nabla_{\theta'} J(\theta') &\approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \frac{\pi_{\theta'}(s_{i,t}, a_{i,t})}{\pi_{\theta}(s_{i,t}, a_{i,t})} \nabla_{\theta'} \log \pi_{\theta'}(a_{i,t} | s_{i,t}) \hat{Q}_{i,t} \\ &\approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \frac{\pi_{\theta'}(s_{i,t})}{\pi_{\theta}(s_{i,t})} \frac{\pi_{\theta'}(a_{i,t} | s_{i,t})}{\pi_{\theta}(a_{i,t} | s_{i,t})} \nabla_{\theta'} \log \pi_{\theta'}(a_{i,t} | s_{i,t}) \hat{Q}_{i,t}\end{aligned}$$

If we ignore the ratio of the state priors, we have

$$\nabla_{\theta'} J(\theta') \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \frac{\pi_{\theta'}(a_{i,t} | s_{i,t})}{\pi_{\theta}(a_{i,t} | s_{i,t})} \nabla_{\theta'} \log \pi_{\theta'}(a_{i,t} | s_{i,t}) \hat{Q}_{i,t}$$

This is no longer unbiased, but its error can be bounded in terms of the difference between  $\theta$  and  $\theta'$  (more on this later). Under this approximation, we get rid of the exponentially growing variance.

## 4.5 Advanced Policy Gradients

### 4.5.1 Covariant/Natural Policy Gradient

One issue with policy gradients is that most likely than not the gradients dominate in terms of some parameters, leading it to be hard to converge to optimal parameters. So instead, we need to rescale the gradient so that this doesn't happen.

Recall that with vanilla gradient ascent, we have

$$\theta_{k+1} \leftarrow \theta_k + \alpha \nabla_{\theta_k} J(\theta_k)$$

This is equivalent to

$$\theta_{k+1} \leftarrow \arg \max_{\theta} \alpha \nabla_{\theta_k} J(\theta_k)^T \theta - \frac{1}{2} \|\theta - \theta_k\|^2$$

This makes sense since the above expression is just maximizing the linear approximation of  $J(\theta)$  at  $\theta_k$  with a quadratic regularization. Using Lagrangian form, we can rewrite the above expression as

$$\theta' \leftarrow \arg \max_{\theta'} (\theta' - \theta)^T \nabla_{\theta} J(\theta) \text{ s.t. } \|\theta' - \theta\| \leq \epsilon$$

Currently the constraint is in parameter-space. As a result, it doesn't account for the fact that some parameters influence the policy more than others. So we would like a constraint in policy-space. One parameterization-independent divergence measure is KL-divergence:  $\mathcal{D}_{KL}(\pi_{\theta'} \parallel \pi_{\theta}) = \mathbb{E}_{\pi_{\theta'}} [\log \pi_{\theta} - \log \pi_{\theta'}]$ .

$$\theta' \leftarrow \arg \max_{\theta'} (\theta' - \theta)^T \nabla_{\theta} J(\theta) \text{ s.t. } \mathcal{D}_{KL}(\pi_{\theta'} \parallel \pi_{\theta}) \leq \epsilon$$

We can approximate the KL-divergence with its second-order Taylor approximation around  $\theta' = \theta$ ,

$$\mathcal{D}_{KL}(\pi_{\theta'} \parallel \pi_{\theta}) \approx (\theta' - \theta)^T \mathbf{F} (\theta' - \theta)$$

$\mathbf{F}$  is the Fischer-information matrix, where

$$\mathbf{F} = \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(\mathbf{a}|\mathbf{s}) \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}|\mathbf{s})^T]$$

which can be estimated with samples taken from  $\pi_{\theta}$ .

After writing out the Lagrangian and solving for the optimal solution, we find that the update rule is

$$\theta \leftarrow \theta + \alpha \mathbf{F}^{-1} \nabla_{\theta} \mathbf{J}(\theta)$$

The natural policy gradient selects  $\alpha$ . Trust region policy optimization (TRPO) selects  $\epsilon$  and then derives  $\alpha$  using conjugate gradient.