(Deep) Policy Gradient RL Algorithms

November 10, 2024

1 Introduction

Till now, in module 4, we have discussed Deep Q-Learning. In this lecture, we will deal with Policy Gradient RL algorithms. Proximal Policy optimization (PPO), as state-of-the-art RL algorithm that OpenAI uses, is a variant of Policy Gradient RL algorithms. But before going to details, there are two fundamental questions that needs to be answered.

What are Policy Gradient RL Algorithms?:

1. In Deep Q-Learning (and Q-Learning and SARSA in general), we are learning the Q-function which implicitly contains the policy. So Deep Q-Learning, is a value-based RL algorithm. We encountered something similar while studying Bandit algorithms where ε-greedy and UCB algorithms where value based Bandit algorithms.
2. In Policy Gradient RL, we directly learn the policy. Basically, we fix the structure of the policy and then tune the parameters θ of the policy using gradient ascent (NOT descent because we want to maximize reward). These RL algorithms are called policy-based RL algorithm. We countered something similar while studying contextual bandits.
3. Let $\pi(a \mid x, \theta)$ be the policy where x is the state and θ are the parameters of the policy. In Deep Policy Gradient, $\pi(a \mid x, \theta)$ is a neural network.

Why to learn Policy Gradient RL algorithms given the DQN is the most famous RL algorithm?:

1. Well, one answer is simple. Just like there are many ML/DL models, there are many RL algorithms. We may want to try a few RL algorithms and see which one works best for our problem.
2. In many applications, the policy may have a simpler form compared to the Q-value. In such application, we can use a simpler Neural network for policy gradient RL compared to the neural network required for DQN. Simpler neural network implies faster training.
   (a) Two such applications are *inventory control* and *packet scheduling in wireless networks*. In these applications, the policies are simple; basically a threshold policy, e.g. if the inventory is below a certain threshold, then we should buy more goods.
3. From theoretical point of view, there are NO proofs for convergence of DQN[1]. But, policy gradient will always converge atleast to a local optima.
4. Proximal policy optimization (PPO), which seems to be the state-of-the art algorithm that OpenAI uses, is a variant of policy gradient. Specifically, PPO is policy gradient where during updating the policy, we don't let the policy change a lot (hence proximal).

[1] When I say that DQN may not converge, I don't mean that DQN will not converge to the optimal policy. I mean that DQN may diverge, i.e. become unstable!

2 Derivation of Gradient for Policy Gradient RL

2.1 Is there any relation to policy gradient for contextual bandits?

Recall contextual bandits (you may refer to *lecture 6 to 9* slides). When discussing Policy Gradient for contextual bandits, we spent most of the time doing the following:

1. Deriving the gradient of the expected reward with respect to the parameters of the policy.
2. Finding how to compute this gradient using samples collected from interacting with the environment.

We are going to do the same thing for policy gradient RL but with one key difference: we want to compute the gradient of the expected return and NOT expected reward like we did for contextual bandits. This is because in RL, current actions effects future rewards. Because of this difference, a sample in RL is an entire episode (as shown in Figure 1) while a sample in contextual bandits is one time slot (as shown in Figure 2). In fact, one can think of contextual bandit as RL where every episode has just one time slot.

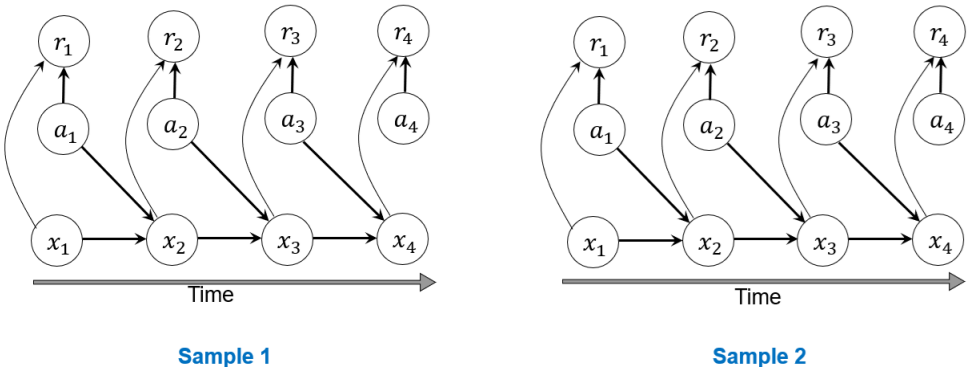

Figure 1: PGM of RL showing that an entire episode is one sample.

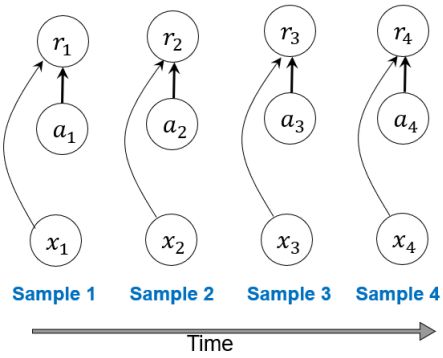

Figure 2: PGM of contextual bandit showing that a time slot is one sample.

2.2 Derivation of gradient of expected return

The contents of this section was not taught during lecture but is MANDATORY from exam/assignment purpose. You can easily follow it if you understand the contents of sections 2.3 which were taught during lecture.

In this section, we will derive an expression for the gradient of expected return for RL. The gradient that we derive in this section is not used in practice because it suffers from high variance (hence leads to slower convergence) but serves two purposes:

1. It helps in understanding why the gradient derived in sections 2.3 have lower variance.
2. The gradient derived in this section can be used if the return has any generic form. But, the gradient in sections 2.3 can be used only if the return is of the summation form.

2.3 Derivation of gradient of expected return while accounting for causality

Lets' consider a policy $\pi(a|x,\theta)$ parameterized by θ. Let $J(\theta)$ denote the expected return of this policy. We have,

$$\begin{aligned} J(\theta) &= E[G] \\ &= E\left[\sum_{k=0}^{T}\beta^k r_k\right] \\ &= \sum_{k=0}^{T}\beta^k E[r_k] \end{aligned}$$

where G is the return. We want to find the gradient of $J(\theta)$ with respect to θ,

$$\nabla_\theta J(\theta) = \sum_{k=0}^{T}\beta^k \nabla_\theta E[r_k] \tag{1}$$

Let $\eta_t = (x_0, a_0, r_0, x_1, a_1, r_1, \ldots, x_t, a_t, r_t)$ denote the trajectory till time t. Using law of total expectation and principle of causality we can write,

$$E[r_k] = \sum_{\eta_k} E[r_k|\eta_k]\, P[\eta_k] \tag{2}$$

$$= \sum_{\eta_k} r_k P[\eta_k] \tag{3}$$

Before moving forward with the derivation, let's take a moment to understand what happened in (2) because this is the main step that differentiates the derivation in the UC Berkeley lecture to that in this section:

1. Here, we are conditioning r_k to the trajectory till time k. This is simply because what happens in the future will NOT effect what happened in the past.
2. In UC Berkeley lecture, they are essentially conditioning r_k to the ENTIRE trajectory till time T, i.e. equation (2) will be $E[r_k] = \sum_{\eta_T} E[r_k|\eta_T]\, P[\eta_T]$ for UC Berkeley lectures. Note that according to law of total expectation, both $E[r_k] = \sum_{\eta_k} E[r_k|\eta_k]\, P[\eta_k]$ and $E[r_k] = \sum_{\eta_T} E[r_k|\eta_T]\, P[\eta_T]$ are correct. But, for the former case we are using causality.

Now, coming to (3), $E[r_k|\eta_k] = r_k$ because the trajectory $\eta_k = (x_0, a_0, r_0, x_1, a_1, r_1, \ldots, x_k, a_k, r_k)$ contains r_k. Hence, given r_k, $E[r_k|\eta_k] = r_k$ (this is more like a tautology!).

Now, the trajectory η_k is generated by the policy. Hence η_k depends on policy parameter θ[2]. So, we can re-write (3) as follows,

$$E[r_k] = \sum_{\eta_k} r_k P[\eta_k;\theta]$$

We therefore get,

$$\begin{aligned} \nabla_\theta E[r_k] &= \sum_{\eta_k} r_k \nabla_\theta P[\eta_k;\theta] \\ &= \sum_{\eta_k} r_k \frac{\nabla_\theta P[\eta_k;\theta]}{P[\eta_k;\theta]} P[\eta_k;\theta] \qquad (4) \\ &= \sum_{\eta_k} \left(r_k \nabla_\theta \log\left(P[\eta_k;\theta]\right)\right) P[\eta_k;\theta] \qquad (5) \\ &= E_{P[\eta_k;\theta]}\left[r_k \nabla_\theta \log\left(P[\eta_k;\theta]\right)\right] \qquad (6) \end{aligned}$$

[2] η_k also depends on *transition probability* distribution, *reward probability* distribution, and the *probability distribution that decides the initial state.*

In (4) and (5), $\frac{\nabla_\theta P[\eta_k;\theta]}{P[\eta_k;\theta]} = \nabla_\theta \log\left(P\left[\eta_k;\theta\right]\right)$ because $\frac{d}{dx}\log\left(x\right) = \frac{1}{f(x)}\frac{d}{dx}f\left(x\right)$.

The trajectory η_k can be also expressed as: $x_0 \to a_0 \to r_0 \to x_1 \to a_1 \to r_1 \to \ldots \to x_k \to a_k \to r_k$. Using the Markov property we can write,

$$\begin{aligned} P\left[\eta_k;\theta\right] &= P\left[x_0, a_0, r_0, x_1, a_1, r_1, \ldots, x_k, a_k, r_k;\theta\right] \\ &= f\left(x_0\right)\cdot\pi\left(a_0|x_0,\theta\right)\cdot f_r\left(r_0|x_0,a_0\right)\cdot f_{tr}\left(x_1|x_0,a_0\right)\cdot\pi\left(a_1|x_1,\theta\right)\cdot f_r\left(r_1|x_1,a_1\right)\cdot f_{tr}\left(x_2|x_1,a_1\right) \\ &= f\left(x_0\right)\prod_{t=0}^{k}\pi\left(a_t|x_t,\theta\right)\prod_{t=0}^{k}f_R\left(r_t|x_t,a_t\right)\prod_{t=0}^{k}f_{tr}\left(x_{t+1}|x_t,a_t\right) \end{aligned} \tag{7}$$

Taking log of (7) we get,

$$\log\left(P\left[\eta_k;\theta\right]\right) = \log\left(f\left(x_0\right)\right) + \sum_{t=0}^{k}\log\left(\pi\left(a_t|x_t,\theta\right)\right) + \sum_{t=0}^{k}\log\left(f_R\left(r_t|x_t,a_t\right)\right) + \sum_{t=0}^{k}\log\left(f_{tr}\left(x_{t+1}|x_t,a_t\right)\right) \tag{8}$$

Taking derivative of (8) we get,

$$\nabla_\theta \log\left(P\left[\eta_k;\theta\right]\right) = \sum_{t=0}^{k}\nabla_\theta \log\left(\pi\left(a_t|x_t,\theta\right)\right) \tag{9}$$

Now, substituting (9) back in (6) we get,

$$\begin{aligned} \nabla_\theta E\left[r_k\right] &= E_{P[\eta_k;\theta]}\left[r_k\nabla_\theta \log\left(P\left[\eta_k;\theta\right]\right)\right] \\ &= E_{P[\eta_k;\theta]}\left[r_k\sum_{t=0}^{k}\nabla_\theta \log\left(\pi\left(a_t|x_t,\theta\right)\right)\right] \end{aligned} \tag{10}$$

Now, substituting (10) back in (1) we get,

$$\begin{aligned} \nabla_\theta J\left(\theta\right) &= \sum_{k=0}^{T}\beta^k\nabla_\theta E\left[r_k\right] \\ &= \sum_{k=0}^{T}\beta^k E_{P[\eta_k;\theta]}\left[r_k\sum_{t=0}^{k}\nabla_\theta \log\left(\pi\left(a_t|x_t,\theta\right)\right)\right] \\ &= \sum_{k=0}^{T}E_{P[\eta_k;\theta]}\left[\beta^k r_k\sum_{t=0}^{k}\nabla_\theta \log\left(\pi\left(a_t|x_t,\theta\right)\right)\right] \\ &= \sum_{k=0}^{T}E_{P[\eta_T;\theta]}\left[\beta^k r_k\sum_{t=0}^{k}\nabla_\theta \log\left(\pi\left(a_t|x_t,\theta\right)\right)\right] \\ &= E_{P[\eta_T;\theta]}\left[\sum_{k=0}^{T}\beta^k r_k\sum_{t=0}^{k}\nabla_\theta \log\left(\pi\left(a_t|x_t,\theta\right)\right)\right] \\ &= E_{P[\eta_T;\theta]}\left[\sum_{k=0}^{T}\sum_{t=0}^{k}\beta^k r_k\nabla_\theta \log\left(\pi\left(a_t|x_t,\theta\right)\right)\right] \\ &= E_{P[\eta_T;\theta]}\left[\sum_{t=0}^{T}\sum_{k=t}^{T}\beta^k r_k\nabla_\theta \log\left(\pi\left(a_t|x_t,\theta\right)\right)\right] \\ &= E_{P[\eta_T;\theta]}\left[\sum_{t=0}^{T}\nabla_\theta \log\left(\pi\left(a_t|x_t,\theta\right)\right)\left(\sum_{k=t}^{T}\beta^k r_k\right)\right] \\ &= E_{P[\eta_T;\theta]}\left[\sum_{t=0}^{T}\nabla_\theta \log\left(\pi\left(a_t|x_t,\theta\right)\right)\hat{Q}_t\right] \end{aligned} \tag{11}$$

where,

$$\begin{aligned}\hat{Q}_t &= \sum_{k=t}^{T} \beta^k r_k \\ &= \beta^t \sum_{k=t}^{T} \beta^{k-t} r_k\end{aligned}$$

Now, we can't compute $\nabla_\theta J(\theta)$ using (11) because $P[\eta_T;\theta]$ depends on initial state distribution $f(x_0)$, transition probability distribution $f_{tr}(x_{t+1}|x_t,a_t)$ which are NOT known (because we are dealing with a learning setup). So, just like in contextual bandit, we can learn $\nabla_\theta J(\theta)$ given by (11) by sampling trajectories η_T,

$$\begin{aligned}\nabla_\theta J(\theta) &\approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=0}^{T} \nabla_\theta \log\left(\pi\left(a_{n,t}|x_{n,t},\theta\right)\right)\hat{Q}_{n,t} \\ &= \nabla_\theta\left(\frac{1}{N}\sum_{n=1}^{N}\sum_{t=0}^{T} \log\left(\pi\left(a_{n,t}|x_{n,t},\theta\right)\right)\hat{Q}_{n,t}\right) \quad (12) \\ &= \nabla_\theta\left(\frac{1}{NT}\sum_{n=1}^{N}\sum_{t=0}^{T} \log\left(\pi\left(a_{n,t}|x_{n,t},\theta\right)\right)T\hat{Q}_{n,t}\right) \quad (13)\end{aligned}$$

where,

$$\hat{Q}_{n,t} = \beta^t \sum_{k=t}^{T} \beta^{k-t} r_{n,k} \quad (14)$$