Policy Gradient

Felipe Glicério Gomes Marcelino

21 May 2020

1 Notes

Today's Lecture

1. The policy gradient algorithm
2. What does the policy gradient do?
3. Basic variance reduction: causality
4. Basic variance reduction: baselines
5. Policy gradient examples
6. Goals:
    - Understand policy gradient reinforcement learning
    - Understand practical considerations for policy gradients

Evaluating the objective

The objective of RL that we would like to optimize:

$$\theta_* = \underset{\theta}{\operatorname{argmax}} \underbrace{E_{\tau \sim p_\theta(\tau)}\Big[\sum_t r(s_t, a_t\Big]}_{J(\theta)} \tag{1}$$

Expanding π_θ results in:

$$\pi_\theta(s_1, a_1, \cdots, s_T, a_T) = p(s_1)\prod_{t=1}^{T}\pi_\theta(a_t|s_t)p(s_{t+1}|s_t, a_t) \tag{2}$$

The following equations are finite and infinite horizon cases.

Infinite Horizon Case

$$\theta_* = \underset{\text{argmax}}{\theta} E_{(s,a)\sim p_\theta(s,a)}[r(s,a)] \tag{3}$$

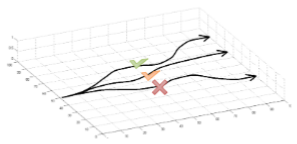

Figure 1: Trajectories: Calculate $J(\theta)$ summing the rewards of these trajectories.

Finite Horizon Case

$$\theta_* = \underset{\text{argmax}}{\theta} \sum_{t=1}^{T} E_{(s,a)\sim p_\theta(s,a)}[r(s,a)] \tag{4}$$

Now how it is possible to maximize the expectation of equation (1) under the trajectory distribution generated by the policy π_θ ? Before answering the question above, let us try something more fundamental. How do we obtain the estimate of this expected value without knowing the transition distribution and the initial state distribution?

- It is possible to generate samples from the distribution under which expectation is taken. Then calculate the average of the values of the quantity inside the expectation over those samples:

$$J(\theta) \approx \frac{1}{N} \sum_i \sum_t r(s_{i,t}, a_{i,t}) \tag{5}$$

Direct Policy Differentiation

The following notation is another notation for $J(\theta)$:

$$J(\theta) = E_{\tau\sim\pi_\theta(\tau)} \underbrace{[r(\tau)]}_{\sum_{t=1}^{T} r(s_t,a_t)} = \int \pi_\theta(\tau) r(\tau) d\tau \tag{6}$$

$$\nabla_\theta J(\theta) = \int \nabla_\theta \pi_\theta(\tau) r(\tau) d\tau \tag{7}$$

Now using the following identity: $\pi_\theta(\tau)\nabla_\theta log\pi_\theta(\tau) = \pi_\theta \frac{\nabla\pi_\theta(\tau)}{\pi_\theta(\tau)} = \nabla_\theta\pi_\theta(\tau)$, to transform the equation (7):

$$\nabla_\theta J(\theta) = \int \pi_\theta(\tau)\nabla_\theta log\pi_\theta(\tau) r(\tau) d\tau = E_{\tau\sim\pi_\theta(\tau)}[\nabla_\theta log\pi_\theta(\tau) r(\tau)] \tag{8}$$

And now expanding π_θ as showed in equation (2) transform equation (8) onto:

$$\nabla_\theta \Big[\cancel{log p(s_1)} + \sum_{t=1}^{T} log \pi_\theta(a_t|s_t) + \cancel{log p(s_{t+1}|s_t, a_t)} \Big] \tag{9}$$

The two terms without θ is zero

$$\nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \Big[\Big(\sum_{t=1}^{T} \nabla_\theta log \pi_\theta(a_t|s_t) \Big) \Big(\sum_{t=1}^{T} r(s_t, a_t) \Big) \Big] \tag{10}$$

How to derivate equation (10)? It is possible to use the same idea from equation (5):

- Generating samples throughout the policy iteration with the environment and evaluating trajectories rewards. Also, evaluate their $\nabla_\theta log \pi_\theta$, multiply them by the rewards and average the result's multiplication.

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \Big(\sum_{t=1}^{T} \nabla_\theta log \pi_\theta(a_{i,t}|s_{i,t}) \Big) \Big(\sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \Big) \tag{11}$$

Comparison to maximum likelihood

$$\nabla_\theta J_{ML}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \Big(\sum_{t=1}^{T} \nabla_\theta log \pi_\theta(a_{i,t}|s_{i,t}) \Big) \tag{12}$$

This is just supervised learning maximum likelihood.

Evaluating the policy gradient

Finally, it is possible now to modify the parameters theta according to derivating of $J(\theta)$:

$$\theta \leftarrow \theta + \alpha J(\theta) \tag{13}$$

REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(a_t|s_t)$ (run the policy)
2. $\nabla_\theta J(\theta) \approx \sum_i \big(\sum_t \nabla_\theta log \pi_\theta(a_t^t|s_t^i a) \big) \big(\sum_t r(a_t^i, a_t^i) \big)$
3. $\theta \leftarrow \theta + \alpha J(\theta)$

- Doest not require the initial state distribution or the transition probabilities. See equation (11)
- Can be used in POMDP(Partially observed MDP) since Markov property is not use.

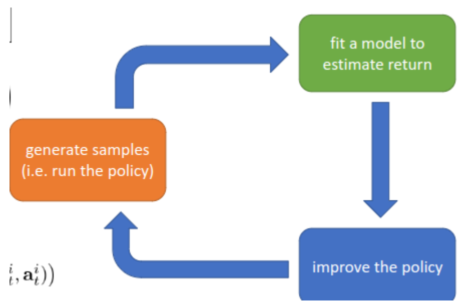
fit a model to estimate return
generate samples (i.e. run the policy)
improve the policy

Figure 2: The algorithm consists of three parts: generation of samples,fit a model to estimate return and improve the policy

What did we just do?

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta log\pi_\theta(\tau_i) r(\tau_i) \tag{14}$$

Equation (14) is a cleaner version of equation (11). It will take the take gradient of $log\pi_\theta(\tau)$ that points in the direction of trajectory τ and increase the probability of this trajectory – multiplying the τ trajectories by reward makes the high-reward trajectories more probable and the lower reward trajectories less probable. Compared to the maximum likelihood gradient that makes everything more probable, equation (14), on the other hand, penalize trajectories with bad reward expectation.

What is wrong with the policy gradient?

- High Variance - When the policy gradient gets different samples, it get very different gradients from these samples. The gradient becomes very noisy, taking zig zag path and then the algorithm cannot converge.

Reducing Variance

The first effort to reduce variance exploits the fact that the universe has causality. The past influences the future, but not otherwise.

- *Causality*: policy at time *t'* cannot affect reward at time *t* when t $\leq$ *t'*

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}\right)\left(\sum_{t'=t}^{T} r\left(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}\right)\right) \tag{15}$$

Equation (15) states that the action choosing now t' will not affect the past rewards. Take a look into the third $\sum$ and the subscript term. Consequently, the result of the multiplications of equation (15) is going to be smaller than

(11). Now, operations containing small numbers are going to have a smaller variance.

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta \left(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}\right) \hat{Q}_{i,t} \tag{16}$$

$$\hat{Q} = \text{rewards to go}$$

Baselines

Another problem with the reward function is the rewards can be a larger number and the values next to each other. For instance, if the rewards are between 999999 and 10000001, according to equation (14), they will be very probably. Because of this, the gradient does not have good separation for the real good one's trajectories. Solution: Adding a term b that is: $b = \frac{1}{N} \sum_{i=1}^{N} r(\tau)$

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\tau)[r(\tau) - b] \tag{17}$$

The term b helps the gradient. The good trajectories become most likely than the bad ones. This term is the average reward of the samples. So b increases the probability of the trajectories that is better than the average and decreases probability of trajectories that is worst than the average. The expectation $E|\nabla_\theta \log \pi_\theta(\tau)b]$ is going to be the same value that the expectation has before, but with less variance. Proof that the term b does not affect the expectation:

$$E\left[\nabla_\theta \log \pi_\theta(\tau)b\right] = \int \pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) b d\tau \tag{18}$$

$$\int \nabla_\theta \pi_\theta(\tau) b d\tau = b \nabla_\theta \int \pi_\theta(\tau) d\tau \text{ (Integral of all probabilities is one)} \tag{19}$$

$$b \nabla_\theta 1 = 0 \tag{20}$$

The identity: $\pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) = \nabla_\theta \pi_\theta(\tau)$, transform right side of the equation (18) into equation left side of the equation (19). These proof affirms that it is possible to subtract any constant into expectation, and the expectation will not change.
The optimal baseline is better than average baseline, but for implementation, average baseline are good enough:

$$b = \frac{E\left[g(\tau)^2 r(\tau)\right]}{E\left[g(\tau)^2\right]} \tag{21}$$

The optimal baseline is the expected reward weighted by gradient magnitudes.

Proof:

$$\begin{aligned}
\text{Var}(x) &= E(x^2) - E(x)^2 \\
&= E_{\tau\sim\pi_\theta(\tau)}\left[\left(\nabla_\theta \log \pi_\theta(\tau)(r(\tau)-b)\right)^2\right] - E_{\tau\sim\pi_\theta(\tau)}\left[\nabla_\theta \log \pi_\theta(\tau)(r(\tau)-b)\right]^2 \\
&= E_{\tau\sim\pi_\theta(\tau)}\left[\left(\nabla_\theta \log \pi_\theta(\tau)(r(\tau)-b)\right)^2\right] - \cancel{\underbrace{E_{\tau\sim\pi_\theta(\tau)}\left[\nabla_\theta \log \pi_\theta(\tau) r(\tau)\right]^2}_{\text{this bit is just} E_{\tau\sim\pi_\theta(\tau)}[\nabla_\theta log\pi_\theta(\tau)r(\tau)]}}
\end{aligned}$$

Since b is unbiased, the second term can be ignored.
Let $g(\tau) = \nabla_\theta log\pi_\theta(\tau)$, compute the derivate of variance with respect to b.

$$\begin{aligned}
\frac{d\text{Var}}{db} &= \frac{d}{db}E\left[g(\tau)^2(r(\tau)-b)^2\right] \\
&= \frac{d}{db}\left(E\left[g(\tau)^2 r(\tau)^2\right] - 2E\left[g(\tau)^2 r(\tau) b\right] + b^2 E\left[g(\tau)^2\right]\right) \\
&= -2E\left[g(\tau)^2 r(\tau)\right] + 2bE\left[g(\tau)^2\right] = 0
\end{aligned}$$

Solve the optimal value for b:

$$b = \frac{E\left[g(\tau)^2 r(\tau)\right]}{E\left[g(\tau)^2\right]} \tag{22}$$

Policy Gradient is on-policy

On-policy definition: Every time the policy changes, it is necessary to generate new samples in order to improve the policy.
$\nabla_\theta J(\theta) = \underbrace{E_{\tau\sim\pi_\theta(\tau)}}_{\text{This is trouble ...}} \left[\nabla_\theta \log \pi_\theta(\tau) r(\tau)\right]$
What does it mean?
REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(a_t|s_t)$ (run the policy) $\leftarrow$ **THIS STEP CAN'T BE SKIPPED**
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta log\pi_\theta(a_t^t|s_t^i a)\right)\left(\sum_t r(a_t^i, a_t^i)\right)$
3. $\theta \leftarrow \theta + \alpha J(\theta)$

- Neural networks change only a little bit with each gradient step. For this reason, it is necessary a large number of gradient steps
- On-policy learning can be extremely inefficient!

Off-policy learning & importance sampling

Importance Sampling:

Importance sampling is a technique to estimate the expectation of one distribution using a different distribution.

Importance sampling

$$\begin{aligned} E_{x\sim p(x)}[f(x)] &= \int p(x)f(x)dx \\ &= \int \frac{q(x)}{q(x)}p(x)f(x)dx \\ &= \int q(x)\frac{p(x)}{q(x)}f(x)dx \\ &= E_{x\sim q(x)}\left[\frac{p(x)}{q(x)}f(x)\right] \end{aligned} \tag{23}$$

Plugging it into policy gradient

$$J(\theta) = E_{\tau\sim\bar{\pi}(\tau)}\left[\frac{\pi_\theta(\tau)}{\bar{\pi}(\tau)}r(\tau)\right] \tag{24}$$

In that case, the samples come from $\bar{\pi}$ because we do not have samples from π. As a result, the policy becomes off-policy.

$$\frac{\pi_\theta(\tau)}{\bar{\pi}(\tau)} = \frac{p(\mathbf{s}_1)\prod_{t=1}^{T}\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\,p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)}{p(\mathbf{s}_1)\prod_{t=1}^{T}\bar{\pi}(\mathbf{a}_t|\mathbf{s}_t)\,p(\mathbf{s}_{t+1}+\mathbf{s}_t,\mathbf{a}_t)} = \frac{\prod_{t=1}^{T}\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}{\prod_{t=1}^{T}\bar{\pi}(\mathbf{a}_t|\mathbf{s}_t)} \tag{25}$$

Equation (25) states that only the action probability of the old and new policy is necessary for calculating the expectation.

The off-policy policy gradient

$$\begin{aligned} \nabla_{\theta'}J(\theta') &= E_{\tau\sim\pi_\theta(\tau)}\left[\frac{\pi_{\theta'}(\tau)}{\pi_\theta(\tau)}\nabla_{\theta'}\log\pi_{\theta'}(\tau)r(\tau)\right] \quad \text{when } \theta\neq\theta' \\ &= E_{\tau\sim\pi_\theta(\tau)}\left[\left(\prod_{t=1}^{T}\frac{\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}\right)\left(\sum_{t=1}^{T}\nabla_{\theta'}\log\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)\right)\left(\sum_{t=1}^{T}r(\mathbf{s}_t,\mathbf{a}_t)\right)\right] \end{aligned}$$

The problem with the equation above is that if T is too large, the numbers will be enormous or too small.Consequently, it causes the problem of increasing the variance of the expectation, and the all effort using the baseline is useless. To solve this, we are going to use the first-order approximation for importance sampling.

$$\text{on-policy policy gradient:} \quad \nabla_\theta J(\theta) \approx \frac{1}{N}\sum^{N}\sum^{T}\nabla_\theta\log\pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\,\hat{Q}_{i,t}$$

$$\text{off-policy policy gradient:}\ \nabla_{\theta'}J(\theta') \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\frac{\pi_{\theta'}(\mathbf{s}_{i,t},\mathbf{a}_{i,t})}{\pi_\theta(\mathbf{s}_{i,t},\mathbf{a}_{i,t})}\nabla_{\theta'}\log\pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\,\hat{Q}_{i,t}$$

$$\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\underbrace{\cancel{\frac{\pi_{\theta'}\left(\mathbf{s}_{i,t}\right)}{\pi_{\theta}\left(\mathbf{s}_{i,t}\right)}}}_{\text{Ignore this part}}\frac{\pi_{\theta'}\left(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}\right)}{\pi_{\theta}\left(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}\right)}\nabla_{\theta'}\log\pi_{\theta'}\left(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}\right)\hat{Q}_{i,t} \tag{26}$$

The marginal distribution can be ignored, and policy is going to continue to improve **IF** θ' is close to θ.

Policy gradient in practice

- Using much larger batches will help reducing variance.
- Tweaking learning rates is very hard. Adaptive step size rules like ADAM is okay.