Policy Gradients Methods

Easwar Subramanian

TCS Innovation Labs, Hyderabad

Email : easwar.subramanian@tcs.com / cs5500.2020@iith.ac.in

October 11, 2022

Overview

1. Policy Gradient Formulation

2. Variance Reduction Techniques

Policy Based Reinforcement Learning

- Last lecture, we parametrized value functions using parameter ϕ

$$V_\phi^\pi(s) = V^\pi(s)$$

$$Q_\phi^\pi(s,a) = Q^\pi(s,a)$$

- Policy was directly generated from value functions (greedy or ϵ greedy)

$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \arg\max_{a \in \mathcal{A}} Q_*(s,a) \\ 0 & \text{Otherwise} \end{cases}$$

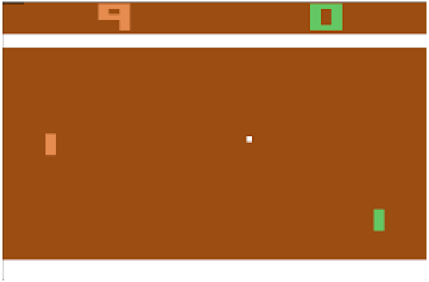
- In the next couple of lectures, we will directly parametrize the policy

$$\pi_\theta(a|s) = P(a|s,\theta)$$

- We will consider model free control with parametrized policies

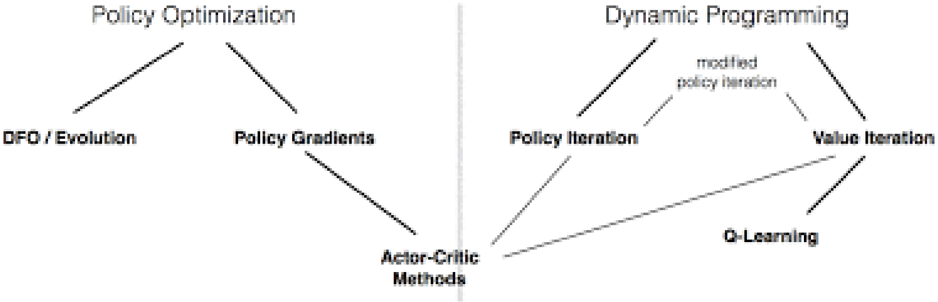
Why Policy Optimization ?

- Often policies (π) are simpler than value functions (V or Q)



- Computing optimal V is bit of problem (we did not see any control algorithms for V)
- With state-value functions Q, computing $\arg\max$ over actions gets tricky when action space is large or continuous
- Better convergence properties
- Can learn stochastic policies

RL Algorithms Landscape [2]



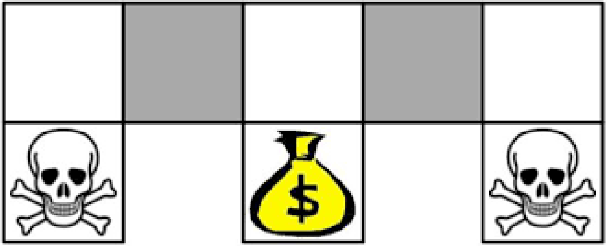
We will now actually look out for the optimal policies in the stochastic policy space !

[2] Slide content from Schulman

Example : Rock-Paper-Scissors

- Two player game of rock-paper-scissors
  - ★ Scissors beats paper
  - ★ Rock beats scissors
  - ★ Paper beats rock
- Consider policies for iterated rock-paper-scissors
  - ★ A deterministic policy is easily exploited
  - ★ A uniform random policy is optimal (i.e. Nash equilibrium)

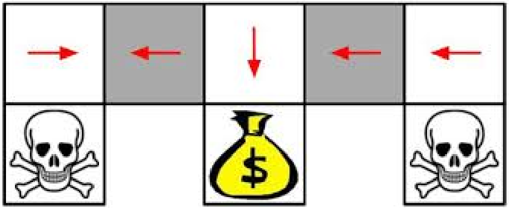
Example : Aliased Grid World



- The agent cannot differentiate the grey states
- For example, state could be represented by features of the following form

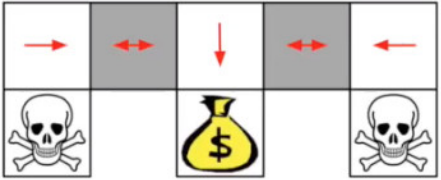
$$\psi(s, a) = 1(\text{wall to } \mathbf{S}, \text{ a=move } \mathbf{E})$$

Example : Aliased Grid World



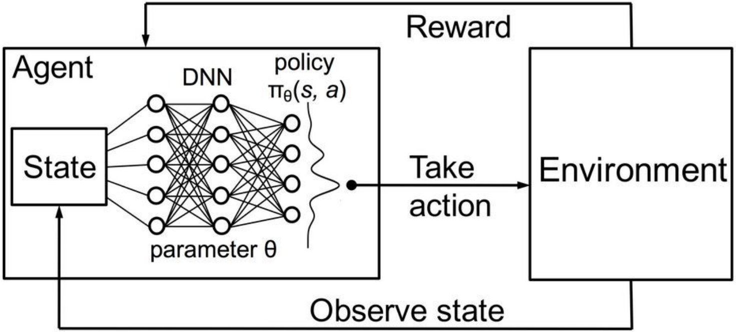
- Under aliasing, an optimal deterministic policy will either
  - ★ move W in both grey states (shown as above)
  - ★ move E in both grey states
- Either way, it can get stuck and never reach the money
- Value based RL learns a near deterministic policy (greedy or ϵ greedy)
- Such a policy will go back and forth on the grid for a long time before hitting money

Example : Aliased Grid World



- An optimal stochastic policy will randomly move E or W in grey states
- It will reach the goal state in a few steps with high probability
- Policy-based RL can learn the optimal stochastic policy

Policy Using Function Approximators



- If action space is discrete
  - ★ Network could output a vector of probabilities (softmax)
- If action space is continuous
  - ★ Network could output the parameters of a distribution (For e.g.. mean and variance of a Gaussian)

Continuous Action Space : Gaussian Policies

- Policy is Gaussian
- The mean (μ) of the Gaussian could be the output of the neural network
- The variance σ of the Gaussian could be constant or can be parametrized.
- One way to operate in continuous action space is to sample an action from the Gaussian distribution. i.e., $a \sim \mathcal{N}(\mu, \sigma)$
- Idea can be extended to any parametrized probability distribution (even multi-variable).

Policy Optimization

A policy $\pi(\cdot)$ is parametrized by parameter θ and denoted by π_θ

Performance of a policy π_θ is given by

$$J(\theta) = V^{\pi_\theta}(s) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s\right]$$

Goal of RL is to find a policy

$$\pi_\theta^* = \arg\max_{\pi_\theta} V^{\pi_\theta}(s) = \arg\max_{\pi_\theta} \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s\right]$$

We will look for π_θ^* in class of stochastic policies by finding θ that maximizes $J(\theta)$

Policy Gradient

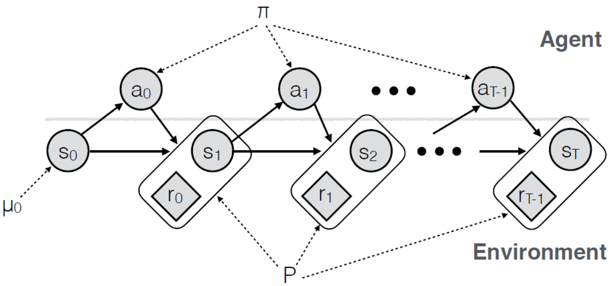
- Let $J(\theta)$ be the policy objective function
- Policy gradient algorithms search for a local maximum in $J(\theta)$ by ascending the gradient of the policy, w.r.t. parameters θ

$$\Delta\theta = \alpha \nabla_\theta J(\theta)$$

- $\nabla_\theta J(\theta)$ is the policy gradient and
- α is the step size parameter

Policy Gradient Formulation

Policy Gradient : Notation



- Let policy π be parametrized by θ and denoted by π_θ
- Let $\tau \sim \pi_\theta$ denote the state-action sequence given by $s_0, a_0, s_1, a_1, \cdots, s_t, a_t, \cdots$
- Then, $P(\tau;\theta)$ be the probability of finding a trajectory τ with policy π_θ

$$P(\tau;\theta) = P(s_0) \prod_{t=0}^{\infty} \pi_\theta(a_t|s_t) P(s_{t+1}|s_t, a_t)$$

Policy Gradient : Objective Function

We can define $G(\tau)$ discounted cumulative reward obtained by following trajectory τ

$$G(\tau) = \sum_{t=0}^{\infty} \gamma^t r_{t+1}$$

Objective function $J(\theta)$ for policy gradient approach is written as,

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | \pi_\theta\right] = \sum_{\tau \sim \pi_\theta} \left[P(\tau;\theta)G(\tau)\right]$$

Goal is to find θ^* such that

$$\theta^* = \arg\max_{\theta} J(\theta)$$

Policy Gradient Derivation

$$J(\theta) = \sum_{\tau} \left[P(\tau;\theta) G(\tau) \right]$$

Taking gradient with respect to θ gives

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \left(\sum_{\tau} \left[P(\tau;\theta) G(\tau) \right] \right) \\
&= \sum_{\tau} \nabla_\theta \left[P(\tau;\theta) G(\tau) \right] \\
&= \sum_{\tau} \frac{P(\tau;\theta)}{P(\tau;\theta)} [\nabla_\theta P(\tau;\theta)] G(\tau) \\
&= \sum_{\tau} \frac{\nabla_\theta P(\tau;\theta)}{P(\tau;\theta)} P(\tau;\theta) G(\tau) \\
&= \sum_{\tau} \nabla_\theta \log P(\tau;\theta) P(\tau;\theta) G(\tau) \qquad \left(\because \quad \nabla_\theta \log f(x) = \frac{\nabla_\theta f(x)}{f(x)} \right)
\end{aligned}$$

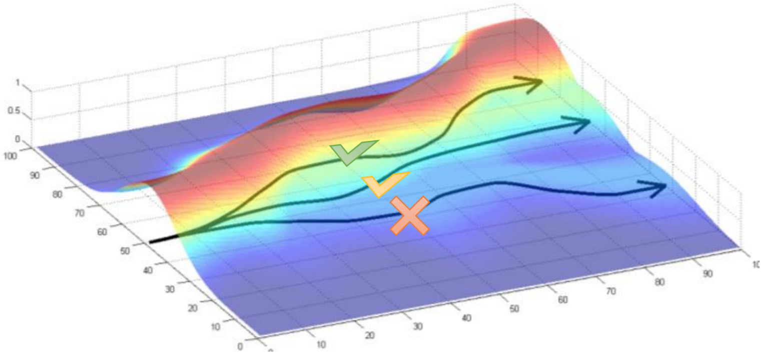
Policy Gradient : Sample Based Estimate

$$\nabla_\theta J(\theta) = \sum_{\tau \sim \pi_\theta} \nabla_\theta \log P(\tau;\theta) P(\tau;\theta) G(\tau) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\nabla_\theta \log P(\tau;\theta) G(\tau) \right]$$

Sample based estimate is given by

$$\nabla_\theta J(\theta) \approx \frac{1}{K} \sum_{i=1}^{K} \nabla_\theta \log P(\tau^{(i)};\theta) G(\tau^{(i)})$$

Policy Gradient : Intuition



- Increase the probability of paths with positive $G(\tau)$
- Decrease the probability of paths with negative $G(\tau)$
- Formalize the notion of 'trial and error'

Slide Credit: Pieter Abeel : Deep RL Boot Camp

Policy Gradient : Model Free Formulation

$$\nabla_\theta J(\theta) \approx \frac{1}{K}\sum_{i=1}^{K} \nabla_\theta \log P(\tau^{(i)};\theta) G(\tau^{(i)})$$

Is the above formula good enough for implementation ?

$$P(\tau;\theta) = P(s_0)\prod_{t=0}^{\infty} \pi_\theta(a_t|s_t) P(s_t|s_{t-1},a_t)$$

$$\begin{aligned}
\nabla_\theta \log P(\tau^{(i)};\theta) &= \nabla_\theta \log \left[\prod_{t=0}^{\infty} \underbrace{P(s_{t+1}^{(i)}|s_t^{(i)},a_t^{(i)})}_{\text{dynamics model}} \cdot \underbrace{\pi(a_t^{(i)}|s_t^{(i)})}_{\text{policy}}\right] \\
&= \nabla_\theta \left[\sum_{t=0}^{\infty} \log P(s_{t+1}^{(i)}|s_t^{(i)},a_t^{(i)}) + \sum_{t=0}^{\infty} \log \pi(a_t^{(i)}|s_t^{(i)})\right] \\
&= \nabla_\theta \sum_{t=0}^{\infty} \log \pi(a_t^{(i)}|s_t^{(i)}) = \sum_{t=0}^{\infty} \nabla_\theta \log \pi(a_t^{(i)}|s_t^{(i)})
\end{aligned}$$

Policy Gradient : Model Free Formulation

The following formulation provides an unbiased estimate of the policy gradient and we can calculate it without using the dynamics model

$$\nabla_\theta J(\theta) \approx \frac{1}{K}\sum_{i=1}^{K}\left[\nabla_\theta \log P(\tau^{(i)};\theta)\right] G(\tau^{(i)})$$

$$\nabla_\theta J(\theta) \approx \frac{1}{K}\sum_{i=1}^{K}\left[\sum_{t=0}^{\infty}\nabla_\theta \log \pi(a_t^{(i)}|s_t^{(i)})\right]\left[\sum_{t=0}^{\infty}\gamma^t r_{t+1}^{(i)}\right]$$

REINFORCE : Monte-Carlo based Policy Gradient

Algorithm REINFORCE : MC based Policy Gradient

1: Initialize policy network π with parameters θ_1 and learning rate α
2: **for** $n = 1$ to N **do**
3: Sample K trajectories from π_{θ_n}
4: Calculate gradient estimate

$$\nabla_{\theta_n} J(\theta_n) \approx \frac{1}{K}\sum_{i=1}^{K}\left[\sum_{t=0}^{\infty}\nabla_{\theta_n}\log\pi(a_t^{(i)}|s_t^{(i)})\right]\left[\sum_{t=0}^{\infty}\gamma^t r_{t+1}^{(i)}\right]$$

5: Perform gradient update

$$\theta_{n+1} = \theta_n + \alpha\nabla_{\theta_n} J(\theta_n)$$

6: **end for**

Connections to Maximum Likelihood

TCS Research & Innovation

Policy Gradient

$$\nabla_\theta J(\theta) \approx \frac{1}{K} \sum_{i=1}^{K} \left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi(a_t^{(i)} | s_t^{(i)}) \right] \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1}^{(i)} \right]$$

Maximum Likelihood

$$\nabla_\theta J_{ML}(\theta) \approx \frac{1}{K} \sum_{i=1}^{K} \left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi(a_t^{(i)} | s_t^{(i)}) \right]$$

(Supervised Learning : Given s_t find a_t)

Issues with Gradient Estimate

- The gradient estimate, thus calculated, is unbiased but has high variance (reason : we are sampling stochastic paths)
- Hence the gradient descent is slow to converge
- Some variance reduction techniques are required in practice

Variance Reduction Techniques

Discount Factor and Variance Reduction

Gradient estimate is given by,

$$\nabla_\theta J(\theta) \approx \frac{1}{K}\sum_{i=1}^{K}\left[\sum_{t=0}^{\infty}\nabla_\theta \log \pi(a_t^{(i)}|s_t^{(i)})\right]\left[\sum_{t=0}^{\infty}\gamma^t r_{t+1}^{(i)}\right]$$

One can rewrite the above equation as

$$\nabla_\theta J(\theta) \approx \frac{1}{K}\sum_{i=1}^{K}\left[\sum_{t=0}^{\infty}\nabla_\theta \log \pi(a_t^{(i)}|s_t^{(i)}) \cdot \left[\sum_{k=0}^{\infty}\gamma^k r_{k+1}^{(i)}\right]\right]$$

- For infinite horizon MDPs having $\gamma < 1$ not only helps in proving convergence of algorithms but also helps reduce variance of the policy gradient estimate
- Ignoring reward terms 'far' into the future gives us a reasonable approximation to policy gradient but with lower variance

Aside : Score Function

Score function in policy gradient is the term

$$\nabla_\theta \log \pi(a_t|s_t)$$

Expectation of the score function is zero

$$\begin{aligned}
\mathbb{E}_{a_t|s_t}\left[\nabla_\theta \log \pi(a_t|s_t)\right] &= \int_{a_t} \pi(a_t|s_t)\nabla_\theta \log \pi(a_t|s_t)\, da_t \\
&= \int_{a_t} \nabla_\theta \pi(a_t|s_t)\, da_t \\
&= \nabla_\theta \int_{a_t} \pi(a_t|s_t)\, da_t \\
&= \nabla_\theta 1 = 0
\end{aligned}$$

Principle of Causality

Causality : Policy at time t' cannot affect reward at time t when $t < t'$.

- When we take an action at timestep t, it can only affect the rewards from timesteps t and onwards.

Recall that,

$$\nabla_\theta J(\theta) \approx \frac{1}{K} \sum_{i=1}^{K} \left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi(a_t^{(i)} | s_t^{(i)}) \cdot \left[\sum_{k=0}^{\infty} \gamma^k r_{k+1}^{(i)} \right] \right]$$

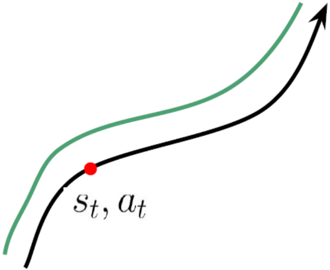
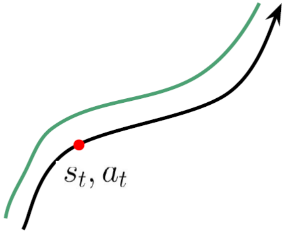


Figure Source: Jie-Han-Chen:SlideShare

Principle of Causality



$$G(\tau) = \sum_{t=0}^{\infty} \gamma^t r_{t+1}$$

Let $\tau_{a:b}$ denote the states and actions visited from time a to b and

$$G_{a:b}(\tau) = \sum_{t=a}^{b} \gamma^t r_{t+1}$$

Therefore for any time t, we have,

$$G(\tau) = G_{0:t-1}(\tau) + G_{t-1:\infty}(\tau)$$

Figure Source: Jie-Han-Chen:SlideShare

Temporal Structure

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{\infty}\nabla_\theta \log \pi(a_t|s_t)\cdot\left[\sum_{k=0}^{\infty}\gamma^k r_{k+1}\right]\right] \\
&= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{\infty}\nabla_\theta \log \pi(a_t|s_t)G(\tau)\right] \\
&= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{\infty}\nabla_\theta \log \pi(a_t|s_t)G_{0:t-1}(\tau) + \sum_{t=0}^{\infty}\nabla_\theta \log \pi(a_t|s_t)G_{t:\infty}(\tau)\right] \\
&= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{\infty}\nabla_\theta \log \pi(a_t|s_t)G_{0:t-1}(\tau)\right] \\
&\quad + \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{\infty}\nabla_\theta \log \pi(a_t|s_t)G_{t:\infty}(\tau)\right]
\end{aligned}
$$

Temporal Structure

Consider evaluating the expectation of the first term

$$
\begin{aligned}
\mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{\infty}\nabla_\theta \log \pi(a_t|s_t) G_{0:t-1}(\tau)\right] &= \left[\sum_{t=0}^{\infty} G_{0:t-1}(\tau)\, \mathbb{E}_{\pi_\theta}\, \nabla_\theta \log \pi(a_t|s_t)\right] \\
&= \sum_{t=0}^{\infty} G_{0:t-1}\cdot 0 = 0
\end{aligned}
$$

Therefore, the policy gradient estimate with temporal structure is given by

$$
\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{\infty}\nabla_\theta \log \pi(a_t|s_t) G_{t:\infty}(\tau)\right]
$$

Temporal Structure

The sample estimate of the gradient expression is given by

$$\nabla_\theta J(\theta) \approx \frac{1}{K} \sum_{i=1}^{K} \left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi(a_t^{(i)} | s_t^{(i)}) \cdot \left[\sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'+1}^{(i)} \right] \right]$$

- The above policy gradient estimate with temporal structure is also an unbiased estimate of the true policy gradient but has **lower variance** since it has '*thrown out*' a few terms

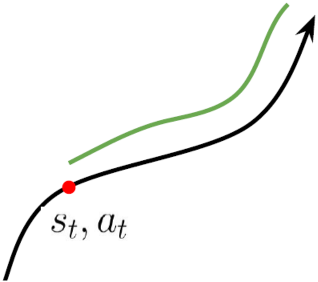
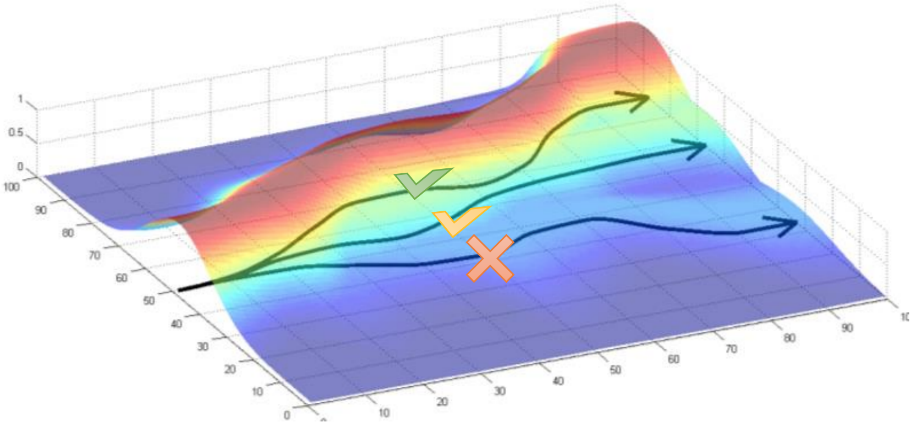


Figure Source:
Jie-Han-Chen:SlideShare

Need for a Baseline



What if all paths have positive reward sum ?

Baseline

Can we subtract a baseline without biasing the gradient ?

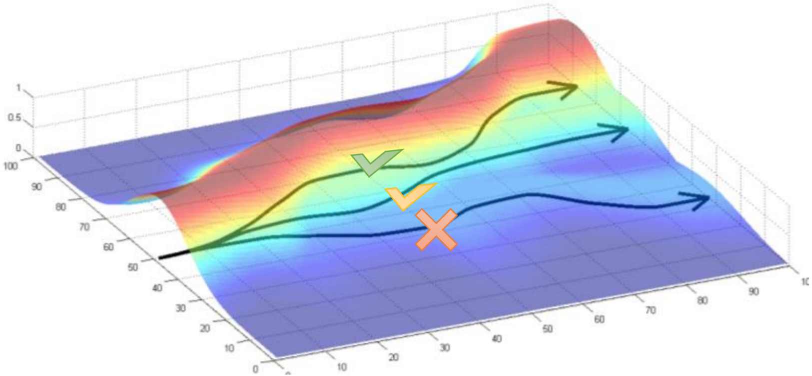
Let $b(s_t)$ be a baseline that is conditioned on s_t. Then,

$$\mathbb{E}_{a_t|s_t}\left[b(s_t)\nabla_\theta \log \pi(a_t|s_t)\right] = b(s_t)\mathbb{E}_{a_t|s_t}\left[\nabla_\theta \log \pi(a_t|s_t)\right] = 0$$

Therefore,

$$\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\tau\sim\pi_\theta}\left[\nabla_\theta \log \pi(a_t|s_t)\cdot G_{t:\infty}(\tau)\right] \\
&= \mathbb{E}_{\tau\sim\pi_\theta}\left[\nabla_\theta \log \pi(a_t|s_t)\cdot G_{t:\infty}(\tau)\right] - \mathbb{E}_{\tau\sim\pi_\theta}\left[b(s_t)\nabla_\theta \log \pi(a_t|s_t)\right] \\
&= \mathbb{E}_{\tau\sim\pi_\theta}\Big[\nabla_\theta \log \pi(a_t|s_t)\cdot\left[G_{t:\infty}(\tau) - b(s_t)\right]\Big]
\end{aligned}$$

Need for a Baseline



A good choice for baseline :

$$b = \mathbb{E}(G(\tau)) \approx \frac{1}{K}\sum_{i=1}^{K} G(\tau^{(i)})$$

Slide Credit: Pieter Abeel : Deep RL Boot Camp

Popular choices of Baseline

- Constant Baseline

$$b = \mathbb{E}(G(\tau)) \approx \frac{1}{K}\sum_{i=1}^{K} G(\tau^{(i)})$$

- Time Dependent Baseline

$$b_t = \frac{1}{K}\sum_{i=1}^{K} G_{t:\infty}(\tau^{(i)})$$

- Optimal Baseline

$$b = \frac{\mathbb{E}_{\tau}(\nabla_\theta \log \pi(a_t|s_t)^2 G_{t:\infty}(\tau))}{\mathbb{E}_{\tau}(\nabla_\theta \log \pi(a_t|s_t)^2)}$$

- State dependent expected return

$$b(s) = \mathbb{E}_{\pi_\theta}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | s_t = s] = V^{\pi}(s)$$

State dependent expected return

$$b(s) = \mathbb{E}_{\pi_\theta}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | s_t = s] = V^\pi(s)$$

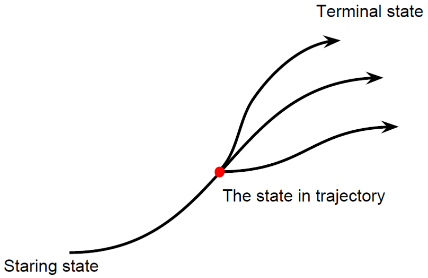


Figure Source:
Jie-Han-Chen:SlideShare

Vanilla Policy Gradient Algorithm

Algorithm Vanilla Policy Gradient Algorithm

1: Initialize policy network π with parameters θ_1 learning rate α and baseline b
2: **for** $n = 1$ to N **do**
3: Sample K trajectories by executing the policy π_{θ_n}
4: At each time step of each trajectory compute $G_t = \sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t+1}$ and advantage estimate $A_t = G_t - b(s_t)$
5: Calculate gradient estimate

$$\nabla_{\theta_n} J(\theta_n) \approx \frac{1}{K} \sum_{i=1}^{K} \left[\sum_{t=0}^{\infty} \nabla_{\theta_n} \log \pi(a_t^{(i)} | s_t^{(i)}) A_t \right]$$

6: Perform gradient update

$$\theta_{n+1} = \theta_n + \alpha \nabla_{\theta_n} J(\theta_n)$$

7: **end for**

Improvements to Vanilla Policy Gradient

- The REINFORCE and Vanilla policy gradient as described above is on-policy
  - ★ There is an off-policy way to do policy gradient algorithms
- We do learning by Monte-Carlo roll-outs
  - ★ Will be addressed by Actor-Critic method