Policy optimization

$Q_{opt}(s, a)$

$\Downarrow$

$\arg\max_a Q_{opt}$

- Why even estimate Q-values? **Can we learn the policy directly?**

- Parameterize $\boxed{\pi_\theta(a \mid s)}$ as a neural network and directly train this network to maximize the rewards

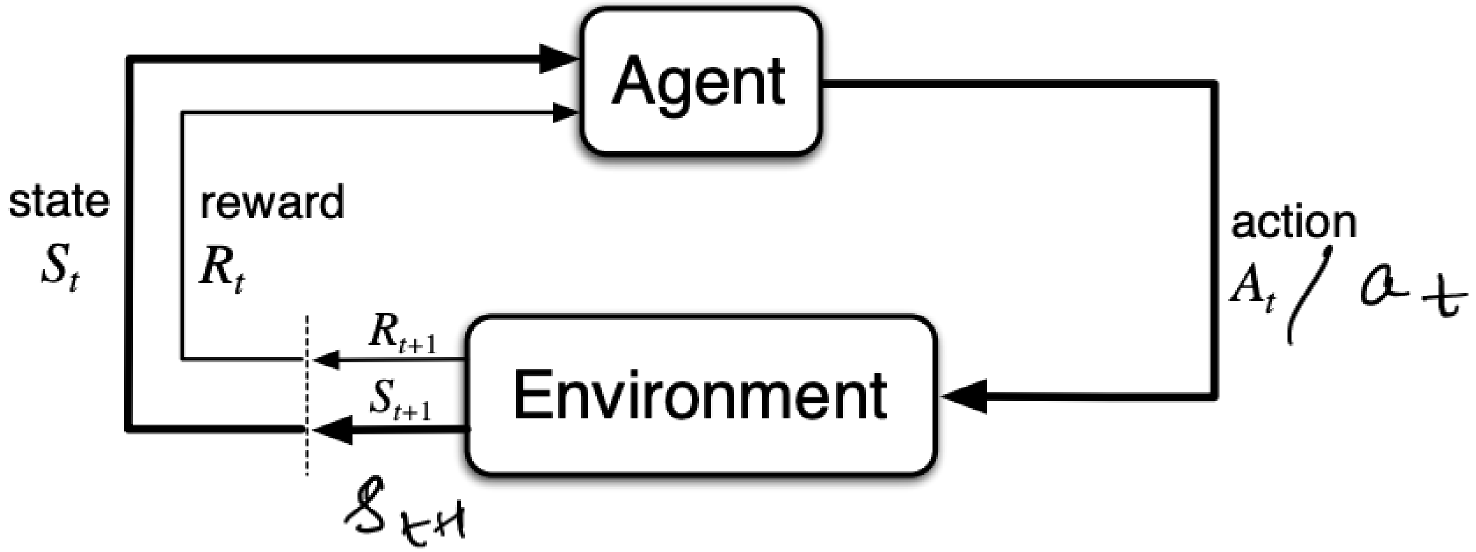
input: s

output: $a \in A$

- Should $\pi_\theta(a \mid s)$ be deterministic or stochastic?

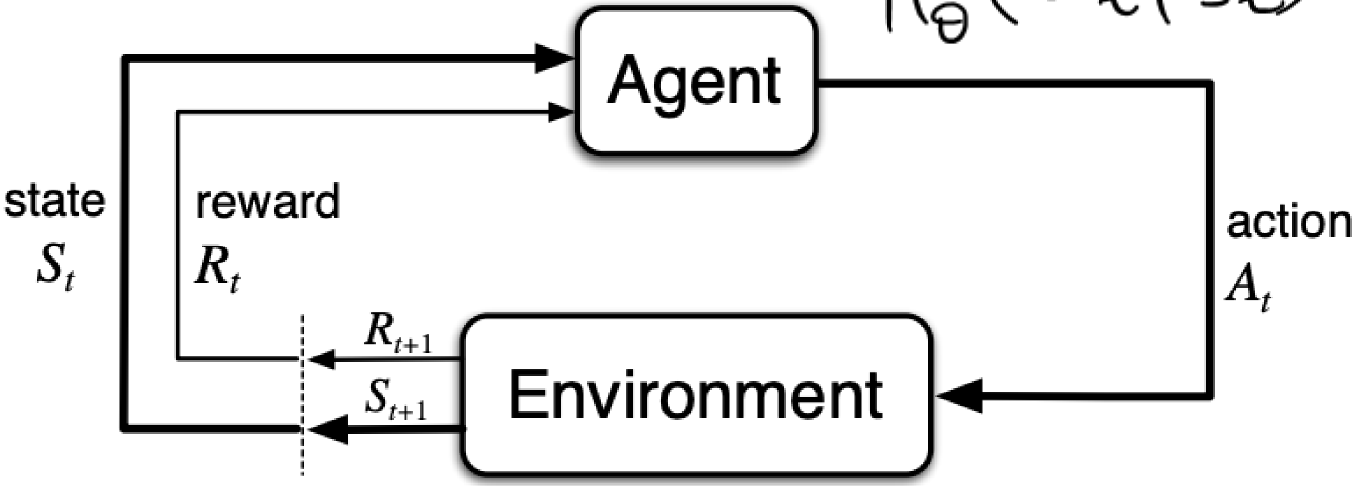
- The optimal policy is deterministic, but stochastic policies make the optimization process "smoother"
- Also helps with exploration

Reinforcement learning (recap)



Policy optimization

$\pi_\theta(a_t | s_t)$



Goal

$$\max \sum_{t=0}^{H} R(s_t, a_t)$$

→ MDP

could be discounted

Stochastic policy class

π_θ : distribution over actions given state

Why policy optimization?

- Can be simpler or faster to estimate optimal policy than Q or V
- If we compute V function, we still need to compute optimal policy by running Bellman update
- If we compute Q, we need to take argmax which can be challenging

$$\pi(s) = \arg\max_a Q_{opt}(s, a)$$

Likelihood ratio policy gradient

Notation: τ : state-action sequence, trajectory, roll-out

$s_0, a_0, s_1, a_1, s_2, \ldots\ldots s_H, a_H$

$R(\tau) : \sum_{t=0}^{H} R(s_t, a_t)$ $\qquad \pi_\theta(a|s)$ $\qquad \theta$: determine a policy

quality of policy corresponding to θ $\qquad \theta \to$ policy $\to$ trajectory

$$U(\theta) : = \mathbb{E}\left[\sum_{t=0}^{H} R(s_t, a_t) ; \theta\right] = \sum_{\tau} P(\tau;\theta) R(\tau)$$

Goal: $\max V(\theta) = \max \sum_{\tau} P(\tau;\theta)\, R(\tau)$ Expected reward

We need to compute $\nabla V(\theta)$ (gradient)

requires dynamics

$$\nabla_\theta V(\theta) = \nabla_\theta \left[\sum_\tau P(\tau;\theta)\, R(\tau) \right] = \sum_\tau \left[\nabla_\theta \underbrace{P(\tau;\theta) R(\tau)}_{\text{neural network}} \right]$$

For every gradient, we enumerate all trajectories

$$= \sum_\tau \frac{P(\tau;\theta)}{P(\tau;\theta)} \nabla_\theta P(\tau;\theta)\, R(\tau)$$

Sampling

$\nabla_\theta \log P(\tau;\theta)$

$$= \sum_\tau P(\tau;\theta) \frac{\nabla_\theta P(\tau;\theta)}{P(\tau;\theta)} \cdot R(\tau)$$

$$\nabla V(\theta) = \sum_{\tau} P(\tau;\theta)\, \nabla_\theta \log P(\tau;\theta)\, R(\tau)$$

unbiased estimate of gradient by sampling

$$\mathbb{E}_{\tau}\left[\nabla_\theta \log P(\tau;\theta)\, R(\tau) ; \theta \right]$$

trajectories sampled from current policy given by θ $(P(\tau;\theta))$

Empirical estimate

Sample $\tau^{(1)}, \tau^{(2)}, \cdots \tau^{(m)}$ according to π_θ

m paths under π_θ

$$\boxed{\nabla V(\theta) \approx \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log P(\tau^{(i)}; \theta)\, R(\tau^{(i)})}$$

R : need not be differentiable

$\log P(\tau^{(i)}; \theta)$: differentiable

Temporal decomposition

$$\nabla_\theta \log P(\tau^{(i)}; \theta) = \nabla_\theta \log \left[\prod_{t=0}^{H} \underbrace{P(s_{t+1}^{(i)} \mid s_t^{(i)}, a_t^{(i)})}_{\text{dynamics model (env)}} \cdot \underbrace{\pi_\theta(a_{t+1}^{(i)} \mid s_{t+1}^{(i)})}_{\text{policy}} \right]$$

$$= \nabla_\theta \left[\cancel{\sum_{t=0}^{H} \log P(s_{t+1}^{(i)} \mid s_t^{(i)}, a_t^{(i)})} + \sum_{t=0}^{H} \log \pi_\theta(a_{t+1}^{(i)} \mid s_{t+1}^{(i)}) \right]$$

$$= \sum_{t=0}^{H} \underbrace{\nabla_\theta \log \pi_\theta(a_{t+1}^{(i)} \mid s_{t+1}^{(i)})}_{\text{policy}}$$

$\nabla U(\theta)$: unbiased estimate : $\hat{g}$

$\tau^{(1)}, \tau^{(2)} \; \cdots \; \tau^{(m)}$

$$\hat{g} = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{H} \nabla_\theta \log \pi_\theta \left(a_{t+1}^{(i)} \mid s_{t+1}^{(i)} \right) R(\tau^{(i)})$$

$$E[\hat{g}] = \nabla U(\theta)$$

Intuition

→ if $R(\tau)$ is high, ~~we~~ want τ to be more likely under θ

→ if $R(\tau)$ is low, τ should be less likely

Decompose to states and actions

$s_1 - s_2 - s_3 \cdots \; s_k$ future reward

$\Downarrow$

$\pi_\theta(a_k | s_k)$

originally $R(\tau)$

For $\tau^{(i)}$:

$$\sum_{t=0}^{H} \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)}) \left[\sum_{k=t}^{H} R(s_k^{(i)}, a_k^{(i)}) \right]$$

Likelihood ration gradient estimate

Variance reduction: baseline

$\rightarrow$ if $R(\tau)$ is high, make τ more likely

high on average

For $\tau^{(i)}$

$$\sum_{t=0}^{H} \nabla_\theta \log \pi_\theta\left(a_t^{(i)} \mid s_t^{(i)}\right)\left[\sum_{k=t}^{H-1} R\left(s_k^{(i)}, a_k^{(i)}\right) - b\left(s_k^{(i)}\right)\right]$$

can be anything!

Baselines

→ adding baseline lowers variance, keeps bias unchanged

baseline doesn't depend on prob of action

Large family of methods to come up with baselines

⇒ "actor critic methods"

critic → baseline

baseline : $V^{\pi}(s_k^{(i)})$: value function from before

Practical update

How to estimate $V_\phi^\pi(s)$? $\rightarrow$ a neural network

Collect $\tau^{(1)}, \tau^{(2)}, \ldots \tau^{(m)}$

Regress against empirical estimate

$$\phi_{i+1} \leftarrow \arg\min_\phi \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{H-1} \Big(V_\phi^\pi(s_t^{(i)}) - \underbrace{\Big(\sum_{k=t}^{H-1} R(s_k^{(i)}, a_k^{(i)})\Big)}_{\text{target}} \Big)^2$$

Monte-carlo estimation of V^π

→ from prev slide

monte-carlo

↔ SARSA

Bootstrapped estimate

$$\phi_{i+1} \leftarrow \min \sum_{s,a,s',r} \| \underbrace{r + V^{\pi}_{\phi_i}(s')}_{\text{target}} - V^{\pi}_{\phi_i}(s) \|_2^2 + \lambda \| \phi - \phi_i \|_2^2$$

Algorithm 2 "Vanilla" policy gradient algorithm

Initialize policy parameter θ, baseline b

for iteration=1, 2, ... **do**

Collect a set of trajectories by executing the current policy

At each timestep in each trajectory, compute

the *return* $R_t = \sum_{t'=t}^{T-1} \cancel{\gamma^{t'-t}} r_{t'}$, and ϕ

lower noise the *advantage estimate* $\hat{A}_t = R_t - b(s_t)$.

ϕ is updated

value fn network

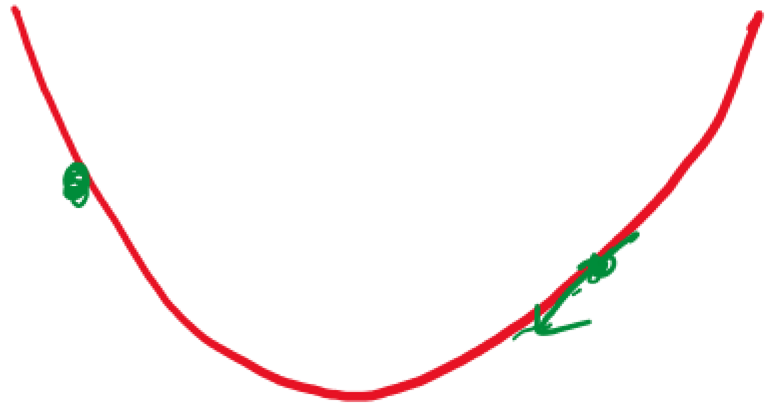
Re-fit the baseline, by minimizing $\|b(s_t) - R_t\|^2$, summed over all trajectories and timesteps.

Update the policy, using a policy gradient estimate $\hat{g}$, which is a sum of terms $\nabla_\theta \log \pi(a_t \mid s_t, \theta)\hat{A}_t$

policy is updated

end for

Step sizing



- Cannot move too far or too less

- How to re-formulate policy gradient to allow a natural notion of "how far to move"?

gradients

rewards come from current policy

$U(\theta)$:

$\nabla U(\theta)$ from policy grad equation

Surrogate loss interpretation

$$V(\theta) = \sum_{\tau} P(\tau|\theta) R(\tau)$$

$$V(\theta) = \mathop{\mathbb{E}}_{\tau \sim \theta_{old}} \left[\underbrace{\frac{P(\tau|\theta)}{P(\tau|\theta_{old})}}_{\gamma} R(\tau) \right]$$

θ_{old}: fixed policy (current)

collect trajectories using θ_{old}, we estimating expected reward of a different θ

"importance sampling"

< importance weights

	0.2	0.8
π_1	A	B
	5	4

	0.8	0.2
π_2	A	B
	5	4

Trust-region Policy Optimization

constrained optimization

TRPO:

$$\max_{\pi} L(\pi) = \mathbb{E}_{\pi_{old}}\left[\frac{\pi(a|s)}{\pi_{old}(a|s)} \underbrace{A^{\pi_{old}}(s,a)}_{\text{advantage rather than reward}}\right]$$

s.t. constraint that

trust region

$$\underbrace{KL(\pi \,\|\, \pi_{old})}_{\text{distance b/w prob distributions}} \le \epsilon$$

original TRPO: CG for optimization

KL divergence : independent of dynamics

PPO : v1

Constrained optimization is hard

TRPO

$$\max_{\theta} \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} \hat{A}_t \right]$$

s.t. $KL(\pi_{\theta_{old}} \,||\, \pi_\theta) < \epsilon$

$$\max_{\theta} \left[\hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} \hat{A}_t \right] - \beta \, KL(\pi_{old} \,||\, \pi_\theta) \right]$$

PPO V_2

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \qquad r_t(\theta_{old}) = 1$$

"Don't deviate too much from 1"

$$L^{CLIP}(\theta) = \hat{E}_t\left[\min\left(r_t(\theta)\hat{A}_t,\ clip\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right]$$