Reinforcement learning

Episode 10

Trust Region Policy Optimization

Let $\mathcal{Z}$ be a trajectory $(s_0, a_0, s_1, a_1, ...)$

Recall:

$$J(\Theta) = E_{\mathcal{Z}\sim\pi(\Theta)}\Big[\sum_{t=0}^{T} \gamma^t r(s_t)\Big]$$ - **goodness** of Θ

Policy gradient theorem:

$$\nabla_\Theta J(\Theta) = E_{(s,a)\sim\pi(\Theta)}\Big[Q^\Theta(s,a)\nabla_\Theta log\, \pi(a|s,\Theta)\Big]$$

Let $\mathcal{Z}$ be a trajectory $(s_0, a_0, s_1, a_1, ...)$

Recall:

$$J(\Theta) = E_{\mathcal{Z}\sim\pi(\Theta)}\left[\sum_{t=0}^{T} \gamma^t r(s_t)\right]$$ - **goodness** of Θ

Policy gradient theorem:

$$\nabla_\Theta J(\Theta) = E_{(s,a)\sim\pi(\Theta)}\left[Q^\Theta(s,a)\nabla_\Theta log\, \pi(a|s,\Theta)\right]$$

Problem: even if we change weights by small vector, our policy sometimes may change dramatically and thus become broken.

Let's define ρ_π as

$$\rho_\pi(s) = p(s_0 = s) + \gamma p(s_1 = s|\pi) + \gamma^2 p(s_2 = s|\pi) + ...$$

Suppose we have these uniformly distributed trajectories that may be generated by policy π

$$s^0 \rightarrow s^1 \rightarrow s^2$$

$$s^1 \rightarrow s^2$$

$$s^0 \rightarrow s^2$$

$$s^2$$

Suppose $\gamma = 0.8$

So $\rho_\pi(s^2) = 1/4 + \gamma 1/2 + \gamma^2 1/4 = 0.81$

There is a theorem...

$$J(\tilde{\pi}) = J(\pi) + E_{z\sim\tilde{\pi}}\left[\sum_{t=0}^{T} \gamma^t A_\pi(s_t, a_t)\right]$$

Then we obtain…

$$J(\tilde{\pi}) = J(\pi) + \sum_s \rho_{\tilde{\pi}}(s) \sum_a \tilde{\pi}(a|s) A_\pi(s, a)$$

Trust Region trick:

If $E_s\left[KL(\pi \,||\, \tilde{\pi})\right]$ is small,

$$J(\tilde{\pi}) \approx J(\pi) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A_\pi(s, a)$$

The TRPO Problem

Find policy $\tilde{\pi}$ that

1) $E_s\left[KL(\pi \,||\, \tilde{\pi})\right] < \alpha$

2) maximizes $\sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A_\pi(s,a)$

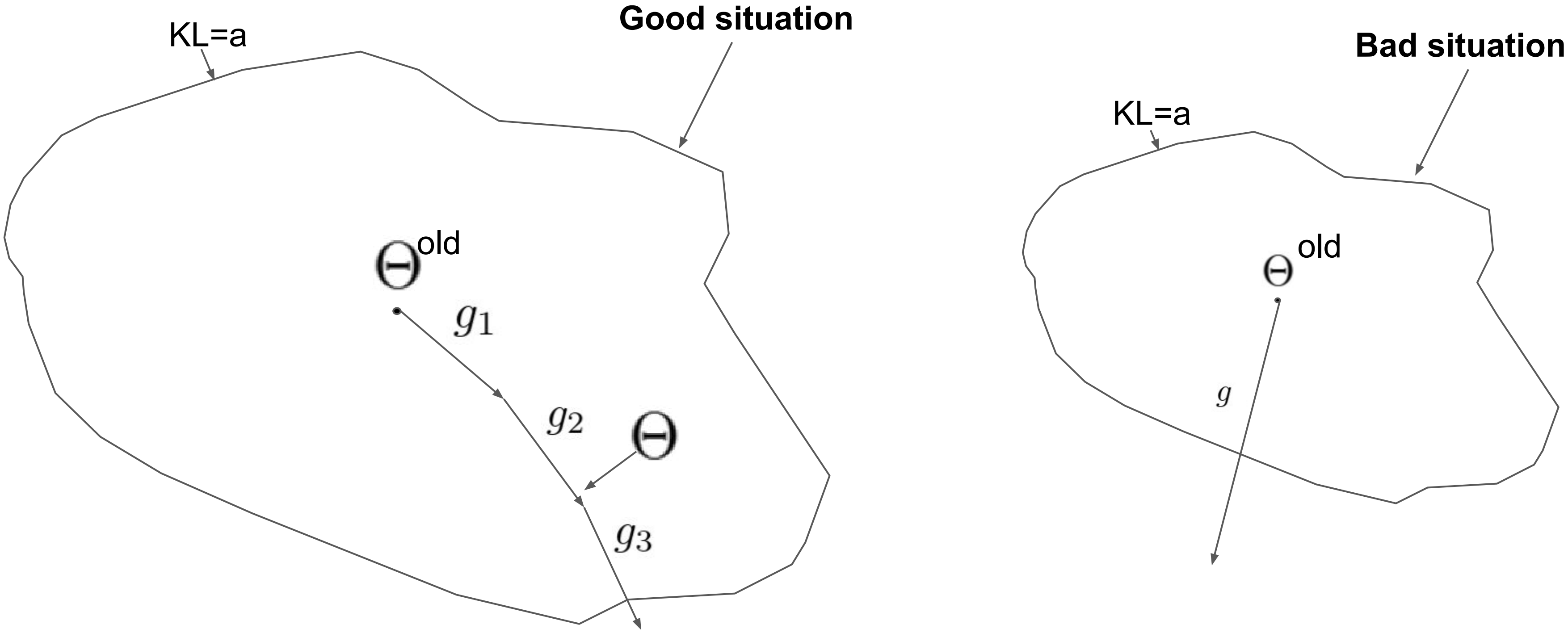
We can see that:

$$\sum_a \pi_\theta(a|s_n) A_{\theta_{\text{old}}}(s_n, a) = E_{a\sim\pi_{old}}\left[\sum_{i=0}^{N} \frac{\pi_\theta(s_i,a_i)}{\pi_{\theta_{old}}(s_i,a_i)} A_{\theta_{old}(s_i,a_i)}\right]$$

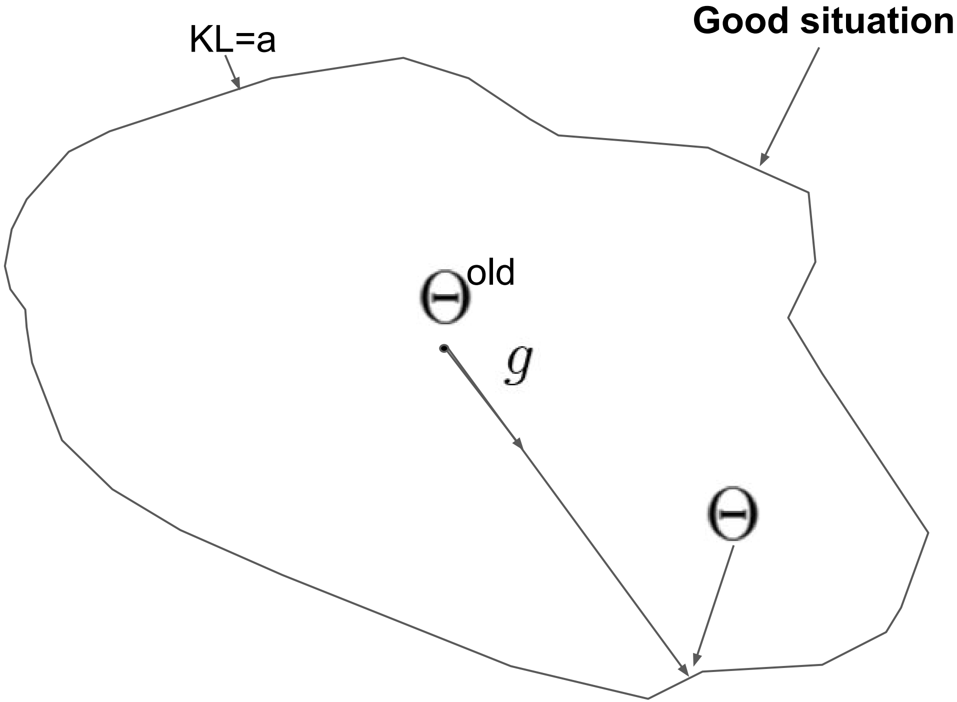
- can be sampled from on-policy distribution!

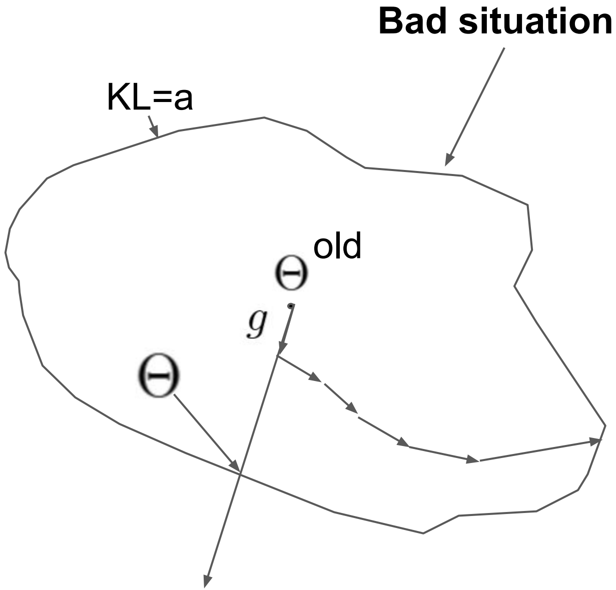
So let's optimize this function!

Vanilla implementation



Linear search





Constrained optimization

We can't compute constrained update analytically but there is a **trick**.

We can approximate KL by $\frac{1}{2}(\theta - \theta_{\text{old}})^T F(\theta - \theta_{\text{old}})$ where F is the **Hessian matrix** of KL.

Then we obtain that if **g** is the gradient of $\mathbb{E}_{a \sim \pi_{\theta_{old}}}\left[\frac{\pi_\theta(a|s_n)}{\pi(a|s_n)} A_{\theta_{\text{old}}}(s_n, a)\right]$

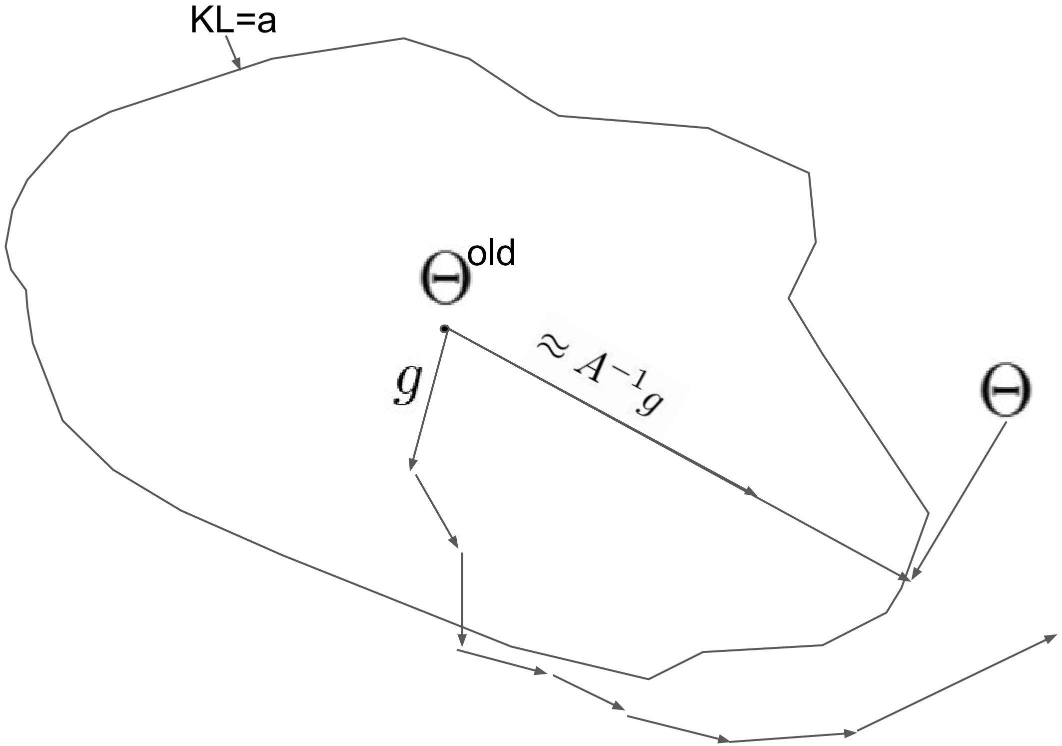
our constrained update has the same direction as the solution of $Fx = g$

We don't want to compute F^{-1} because of complexity.

But we can find approximate solution of $Fx = g$ using **conjugate gradients!**

Constrained optimization

Then we do linesearch!

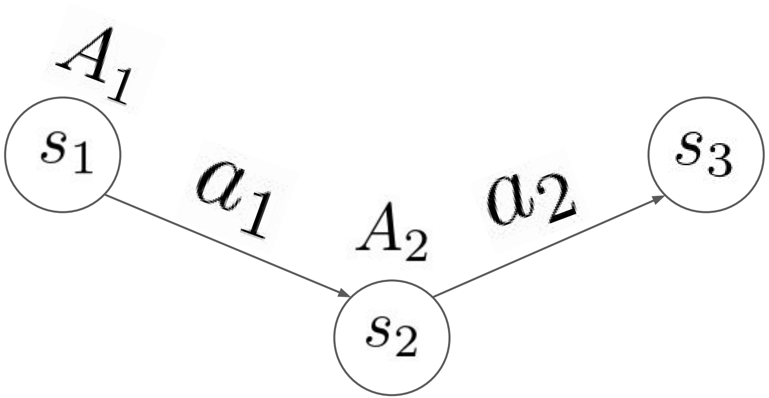


Sampling

Single path (naive approach)

Sample **(state, action, return)** from on-policy distribution

$$L = -\sum_{i=0}^{T} \frac{\pi_\theta(s_i,a_i)}{\pi_{\theta_{old}}(s_i,a_i)} A_{\theta_{old}(s_i,a_i)} \Big]$$

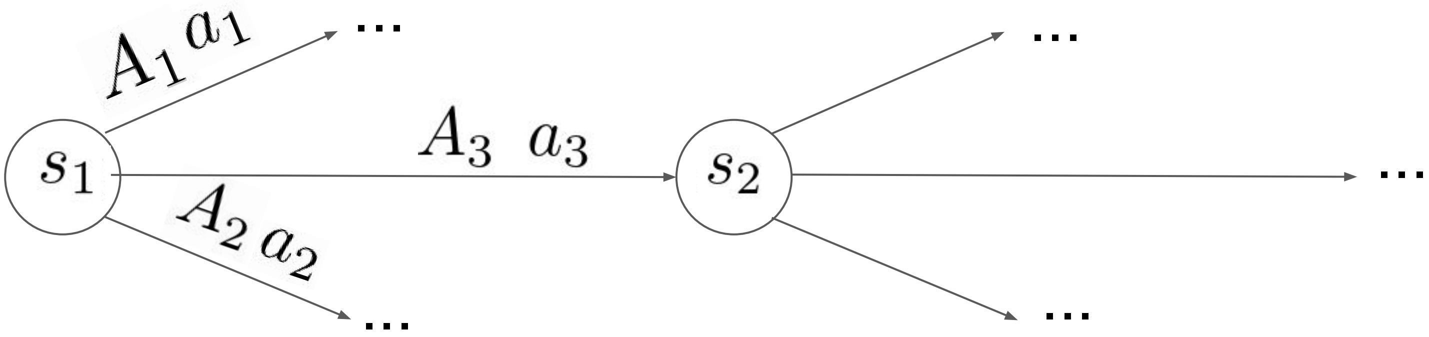


Sampling

Vine (works only if we may use checkpoints)

Sample **(state, returns for all a)** from on-policy distribution

$$L = -\sum_{i=0}^{T}\sum_{j=0}^{n} \frac{\pi_\theta(s_i,a_j)}{\pi_{\theta_{old}}(s_i,a_j)} A_{\theta_{old}(s_i,a_j)}\Bigg]$$

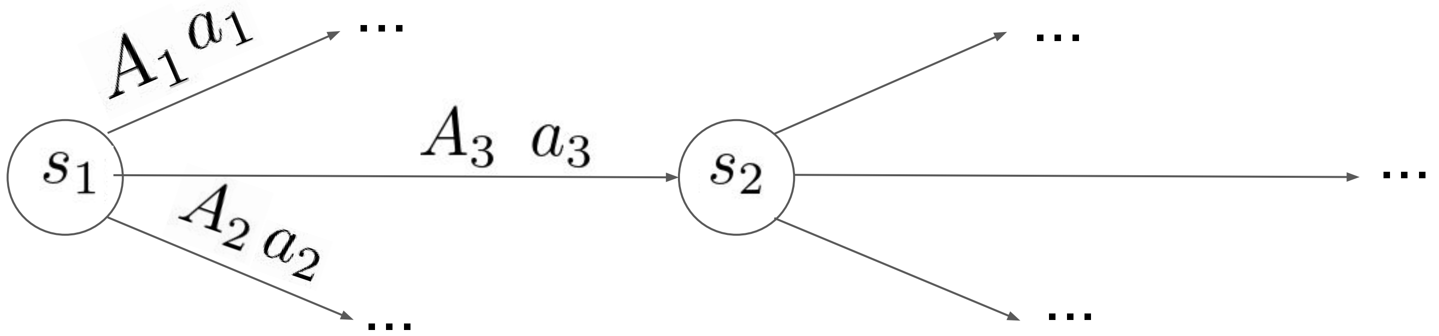


Sampling

Vine (works only if we may use checkpoints)

Sample **(state, returns for all a)** from on-policy distribution

$$L = -\sum_{i=0}^{T}\sum_{j=0}^{n} \frac{\pi_\theta(s_i,a_j)}{\pi_{\theta_{old}}(s_i,a_j)} A_{\theta_{old}(s_i,a_j)}\Big]$$



It's your homework!

TRPO

Advantages

- Very stable training
- Good result

Disadvantages

- Cheap sampling is necessary
- Not easy to implement

Thank you for your attention!

Questions?