CSC2515 Tutorial 9: Policy Gradient

by Haotian Cui

March 23rd, 2021

Based on the slides by Sheng Jia

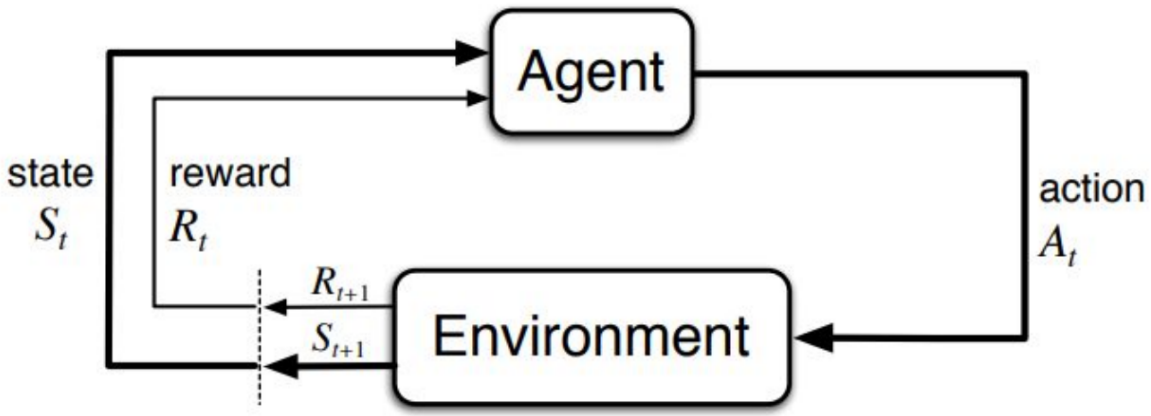
Content

- State and Action
- Policy
- Trajectory and how to sample it
- Objective in Reinforcement Learning
- Policy optimization by policy gradient ascent
  - **Trajectory-based Policy Gradient Derivation**
    (Log-derivative trick. Exploit conditional independence)
  - **Break: Apply policy gradient for playing Dota2**
  - **Reducing variance of policy gradient estimate by Baseline**
    ($Var(x - y)$ can be less than $Var(x)$. Expected grad-log-prob equal 0)
  -
- Implementing Policy Gradient in Pytorch

Problem Setup

State

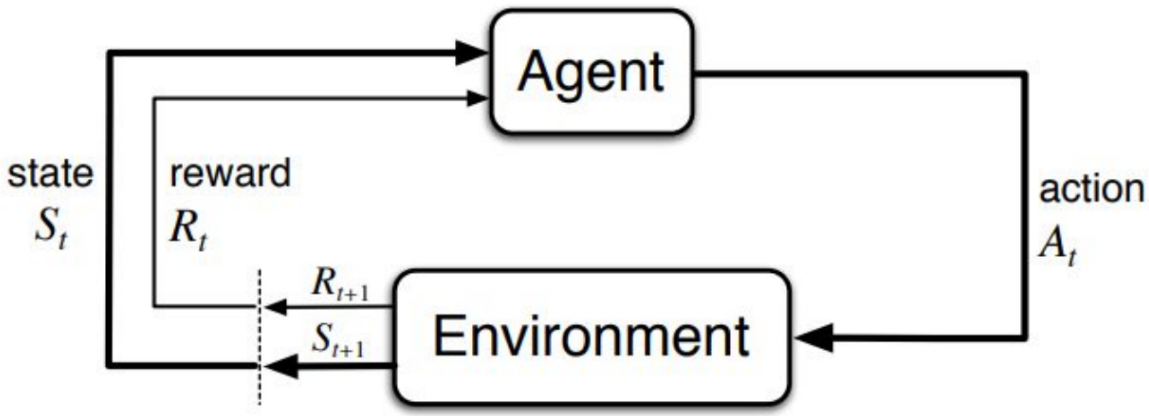
State s is the complete description of the task/environment from which the agent can make decisions for taking actions and receive rewards. Both state and action are indexed by the timestep as s_t, a_t during the agent-environment interaction.



Problem Setup

State

State s is the complete description of the task/environment from which the agent can make decisions for taking actions and receive rewards. Both state and action are indexed by the timestep as s_t, a_t during the agent-environment interaction.



State does not have to be the "physical location" of the agent.
E.g. s_t : how many ppl infected with covid19 today.
a_t : whether or not wash your hands now.

Problem Setup

Agent's Policy

- "Agent" is an abstract concept, but we can formulate how the agent behaves by, for example, a stochastic policy. This can be a conditional distribution that is parameterized by $\boldsymbol{\theta}$.

$$p_{\boldsymbol{\theta}}(a_t|s_t) = \pi_{\boldsymbol{\theta}}(a_t|s_t) = \pi(a_t|s_t;\boldsymbol{\theta})$$

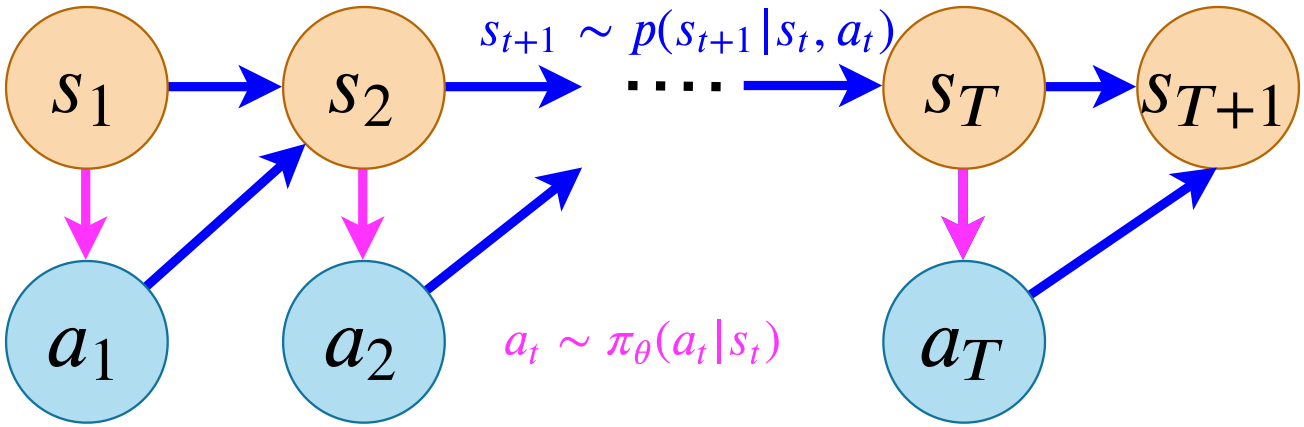
Problem Setup

Trajectory

- Trajectory is nothing but a set of random variables, and its distribution is a joint distribution over $2T + 1$ r.v.:

$$\tau = (s_1, a_1, s_2, ..., s_T, a_T, s_{T+1})$$

$$p(\tau; \theta) = p(s_1, a_1, s_2, ..., s_T, a_T, s_{T+1}; \theta) = (\star)$$



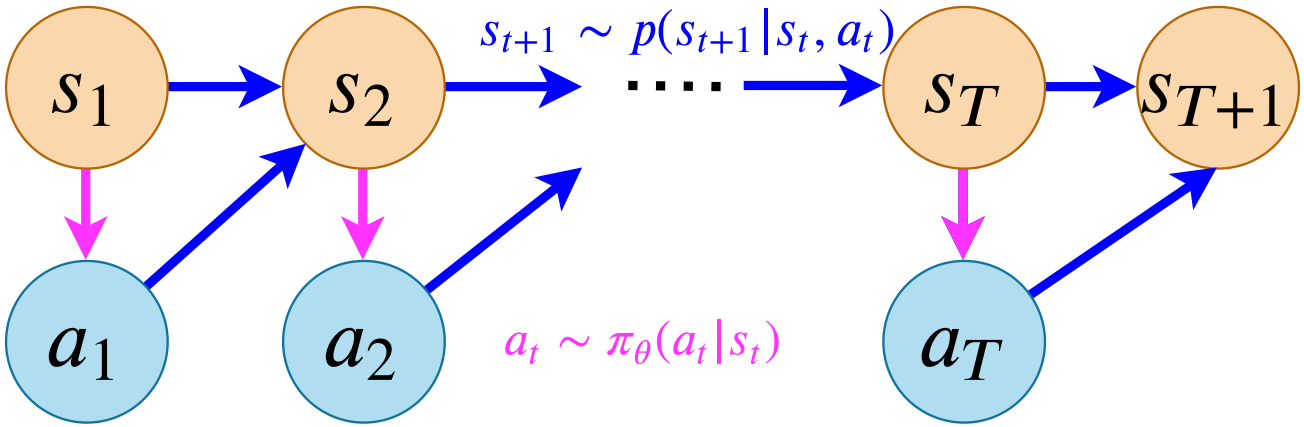
Problem Setup

Trajectory

- We can simplify using conditional independences from DAG:

$$(\star) = \rho_0(s_1)\Pi_{t=1}^{T}\pi_\theta(a_t|s_t)p(s_{t+1}|s_t, a_t)$$

- Remark: we will use $p(\tau;\theta)$ to denote that changing our policy parameters θ induce a different trajectory distribution.

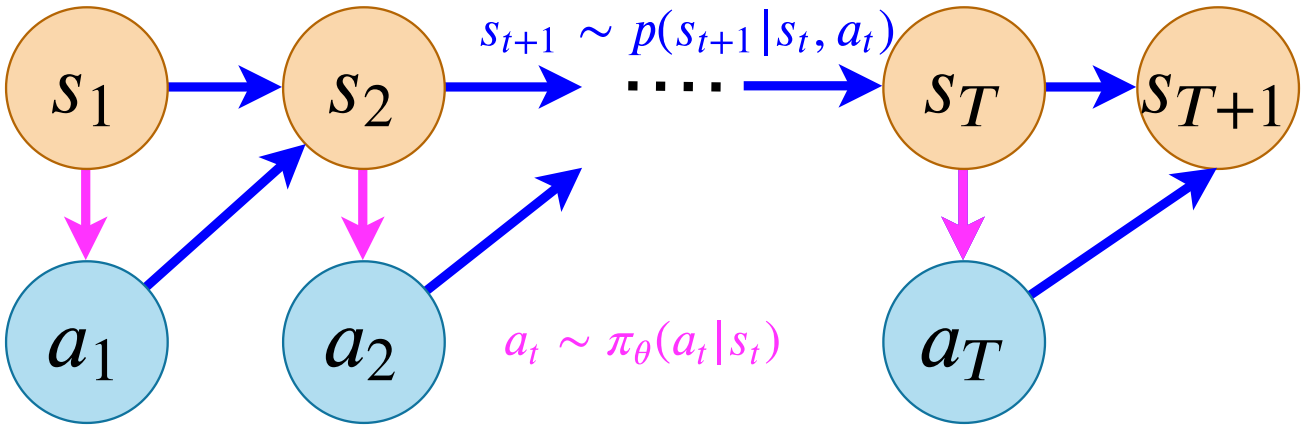


Problem Setup

How to sample a trajectory (Run/Execute an agent)

- "Running/Executing the agent in a environment" means ancestral sampling from this DAG. (Sample the parent node and successively sample the child nodes.)

$$s_1 \sim \rho_0(s) \quad a_t \sim \pi_\theta(a_t|s_t) \quad s_{t+1} \sim p(s_{t+1}|s_t, a_t)$$



Objective in Reinforcement Learning

Reward, Return

- Consider reward $r_t = R(s_t, a_t)$ as something that measures how well action a_t is in state s_t. This is computed by a blackbox function $R(s_t, a_t)$ from the environment.
- Return is the cumulative reward for the trajectory τ. (Consider finite-horzion undiscounted version in this tutorial)

$$R(\tau) = \sum_{t=1}^{T} R(s_t, a_t)$$

Objective in Reinforcement Learning

Expected Return

- As $R(\tau)$ is random, the objective is to maximize the expected return $\mathbb{E}[R(\tau)]$ w.r.t $\boldsymbol{\theta}$. By the law of the unconscious statistician, we can write it as the expectation under τ distribution $p(\tau; \boldsymbol{\theta})$:

$$\mathcal{J}(\boldsymbol{\theta}) = \mathbb{E}[R(\boldsymbol{\tau})] = \mathbb{E}_{\boldsymbol{\tau} \sim p(\tau; \boldsymbol{\theta})}[R(\boldsymbol{\tau})] = (\star)$$

Objective in Reinforcement Learning

Expected Return

- As $R(\tau)$ is random, the objective is to maximize the expected return $\mathbb{E}[R(\tau)]$ w.r.t $\boldsymbol{\theta}$. By the law of the unconscious statistician, we can write it as the expectation under τ distribution $p(\tau; \boldsymbol{\theta})$:

$$\mathcal{J}(\boldsymbol{\theta}) = \mathbb{E}[R(\boldsymbol{\tau})] = \mathbb{E}_{\boldsymbol{\tau} \sim p(\tau; \boldsymbol{\theta})}[R(\boldsymbol{\tau})] = (\star)$$

And by the ancestral sampling, we can further simplify:

$$(\star) = \mathop{\mathbb{E}}_{\substack{s_1 \sim \rho_0(s) \\ a_t \sim \pi_\theta(a_t|s_t) \\ s_{t+1} \sim p(s_{t+1}|s_t, a_t)}} \left[\sum_{t=1}^{T} R(s_t, a_t) \right]$$

Policy Optimization by Policy Gradient Ascent

We can make a one-step optimization for the current policy $\pi_{\boldsymbol{\theta}_k}(a_t|s_t)$ to $\pi_{\boldsymbol{\theta}_{k+1}}(a_t|s_t)$ for maximizing $\mathcal{J}(\boldsymbol{\theta})$ by gradient ascent:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta})|_{\boldsymbol{\theta}_k}$$

Step1 using log-derivative trick

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{\boldsymbol{\tau} \sim p(\boldsymbol{\tau}; \boldsymbol{\theta})} [R(\boldsymbol{\tau})]$$

$$= \nabla_{\boldsymbol{\theta}} \int p(\boldsymbol{\tau}; \boldsymbol{\theta}) R(\boldsymbol{\tau}) \, d\boldsymbol{\tau}$$

Step1 using log-derivative trick

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{\boldsymbol{\tau} \sim p(\boldsymbol{\tau}; \boldsymbol{\theta})} [R(\boldsymbol{\tau})]$$

$$= \nabla_{\boldsymbol{\theta}} \int p(\boldsymbol{\tau}; \boldsymbol{\theta}) R(\boldsymbol{\tau}) \, d\boldsymbol{\tau}$$

$$= \int \nabla_{\boldsymbol{\theta}} p(\boldsymbol{\tau}; \boldsymbol{\theta}) R(\boldsymbol{\tau}) \, d\boldsymbol{\tau}$$

Step1 using log-derivative trick

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{\boldsymbol{\tau} \sim p(\boldsymbol{\tau}; \boldsymbol{\theta})} [R(\boldsymbol{\tau})]$$

$$= \nabla_{\boldsymbol{\theta}} \int p(\boldsymbol{\tau}; \boldsymbol{\theta}) R(\boldsymbol{\tau}) \, d\boldsymbol{\tau}$$

$$= \int \nabla_{\boldsymbol{\theta}} p(\boldsymbol{\tau}; \boldsymbol{\theta}) R(\boldsymbol{\tau}) \, d\boldsymbol{\tau}$$

$$= \int p(\boldsymbol{\tau}; \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau}; \boldsymbol{\theta}) R(\boldsymbol{\tau}) \, d\boldsymbol{\tau} \qquad \because \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau}; \boldsymbol{\theta}) = \frac{\nabla_{\boldsymbol{\theta}} p(\boldsymbol{\tau}; \boldsymbol{\theta})}{p(\boldsymbol{\tau}; \boldsymbol{\theta})}$$

Step1 using log-derivative trick

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{\boldsymbol{\tau} \sim p(\boldsymbol{\tau}; \boldsymbol{\theta})} [R(\boldsymbol{\tau})]$$

$$= \nabla_{\boldsymbol{\theta}} \int p(\boldsymbol{\tau}; \boldsymbol{\theta}) R(\boldsymbol{\tau}) \, d\boldsymbol{\tau}$$

$$= \int \nabla_{\boldsymbol{\theta}} p(\boldsymbol{\tau}; \boldsymbol{\theta}) R(\boldsymbol{\tau}) \, d\boldsymbol{\tau}$$

$$= \int p(\boldsymbol{\tau}; \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau}; \boldsymbol{\theta}) R(\boldsymbol{\tau}) \, d\boldsymbol{\tau} \qquad \because \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau}; \boldsymbol{\theta}) = \frac{\nabla_{\boldsymbol{\theta}} p(\boldsymbol{\tau}; \boldsymbol{\theta})}{p(\boldsymbol{\tau}; \boldsymbol{\theta})}$$

$$= \mathop{\mathbb{E}}_{\tau \sim p(\boldsymbol{\tau}; \boldsymbol{\theta})} [\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau}; \boldsymbol{\theta}) R(\boldsymbol{\tau})]$$

Step2 using conditional independences

$$\mathop{\mathbb{E}}_{\tau\sim p(\tau;\theta)}[\nabla_\theta \log p(\tau;\theta) R(\tau)] \qquad \text{Now use ancestral sampling}$$

$$= \mathop{\mathbb{E}}_{\substack{s_1\sim\rho_0(s)\\ a_t\sim\pi_\theta(a_t|s_t)\\ s_{t+1}\sim p(s_{t+1}|s_t,a_t)}} \left[\underbrace{\nabla_\theta \log\left(\rho_0(s_1)\Pi_{t=1}^{T}\pi_\theta(a_t|s_t)p(s_{t+1}|s_t,a_t)\right)}_{①} \left[\sum_{t'=1}^{T} R(s_{t'},a_{t'})\right]\right]$$

$$\text{where } ① = \nabla_\theta\left(\log\rho_0(s_1) + \sum_{t=1}^{T}\log\pi_\theta(a_t|s_t) + \sum_{t=1}^{T}\log p(s_{t+1}|s_t,a_t)\right)$$

Step2 using conditional independences

$$\mathop{\mathbb{E}}_{\tau\sim p(\tau;\theta)}[\nabla_\theta \log p(\tau;\theta) R(\tau)] \qquad \text{Now use ancestral sampling}$$

$$= \mathop{\mathbb{E}}_{\substack{s_1\sim\rho_0(s)\\ a_t\sim\pi_\theta(a_t|s_t)\\ s_{t+1}\sim p(s_{t+1}|s_t,a_t)}} \left[\underbrace{\nabla_\theta \log\left(\rho_0(s_1)\Pi_{t=1}^{T}\pi_\theta(a_t|s_t)p(s_{t+1}|s_t,a_t)\right)}_{①} \left[\sum_{t'=1}^{T} R(s_{t'},a_{t'})\right]\right]$$

$$\text{where } ① = \nabla_\theta \left(\log\rho_0(s_1) + \sum_{t=1}^{T}\log\pi_\theta(a_t|s_t) + \sum_{t=1}^{T}\log p(s_{t+1}|s_t,a_t)\right)$$

$$= \cancelto{0}{\nabla_\theta \rho_0(s_1)} + \sum_{t=1}^{T}\nabla_\theta \log\pi_\theta(a_t|s_t) + \sum_{t=1}^{T}\cancelto{0}{\nabla_\theta \log p(s_{t+1}|s_t,a_t)}$$

Policy Optimization by Policy Gradient Ascent

Deriving policy gradient (Final form)

Hence, the policy gradient w.r.t the current policy parameters is:

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta})|_{\boldsymbol{\theta}_k} = \mathop{\mathbb{E}}_{\substack{s_1 \sim \rho_0(s) \\ a_t \sim \pi_{\boldsymbol{\theta}_k}(a_t|s_t) \\ s_{t+1} \sim p(s_{t+1}|s_t, a_t)}} \left[\sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}_k}(a_t|s_t) \left[\sum_{t'=1}^{T} R(s_{t'}, a_{t'}) \right] \right]$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} \left[\sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}_k}(a_t^{(i)}|s_t^{(i)}) \left[\sum_{t'=1}^{T} R(s_{t'}^{(i)}, a_{t'}^{(i)}) \right] \right]$$

In practice, this gradient is estimated by executing the policy $\pi_{\boldsymbol{\theta}_k}$ in the environment N times (N times ancestral sampling).

Break: Apply policy gradient for playing Dota2

Successful application of policy optimization by policy gradient

- In Dota2, each team have five players controlling their unique agents. Players gather golds by killing monsters and enemies to buy items. The final objective is destroy an enemy structure called Ancient. OpenAI agents recently won against the best team in the world. [1]

Break: Apply policy gradient for playing Dota2

Observation (Input of the policy)

- State $\mathcal{S}$: 16000-dimensional vector with information such as the distances to the observed enemies. But it is partially observable because teams don't see the map far from the current locations even if they went there before. LSTM is used to memorize previous states.

Break: Apply policy gradient for playing Dota2

Action (Output of the policy)

- Action $\mathcal{A}$: Continuous, but discretized into 8000-80000 actions.

Reducing Variance Using a Baseline

- We subtract a baseline function $B(s)$ from the policy gradient
- This can reduce variance, without changing expectation

$$\begin{aligned}\mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) B(s)\right] &= \sum_{s\in\mathcal{S}} d^{\pi_\theta}(s) \sum_a \nabla_\theta \pi_\theta(s,a) B(s) \\ &= \sum_{s\in\mathcal{S}} d^{\pi_\theta} B(s) \nabla_\theta \sum_{a\in\mathcal{A}} \pi_\theta(s,a) \\ &= 0\end{aligned}$$

- A good baseline is the state value function $B(s) = V^{\pi_\theta}(s)$
- So we can rewrite the policy gradient using the advantage function $A^{\pi_\theta}(s,a)$

$$\begin{aligned}A^{\pi_\theta}(s,a) &= Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s) \\ \nabla_\theta J(\theta) &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ A^{\pi_\theta}(s,a)\right]\end{aligned}$$

Estimating the Advantage Function (1)

- The advantage function can significantly reduce variance of policy gradient
- So the critic should really estimate the advantage function
- For example, by estimating *both* $V^{\pi_\theta}(s)$ *and* $Q^{\pi_\theta}(s, a)$
- Using two function approximators and two parameter vectors,

$$V_v(s) \approx V^{\pi_\theta}(s)$$
$$Q_w(s, a) \approx Q^{\pi_\theta}(s, a)$$
$$A(s, a) = Q_w(s, a) - V_v(s)$$

- And updating *both* value functions by e.g. TD learning

Reference

 Christopher Berner et al. "Dota 2 with Large Scale Deep Reinforcement Learning". In: *arXiv preprint arXiv:1912.06680* (2019).