CSC413 Tutorial11: Policy Gradient

by Sheng Jia

March 31st, 2020
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
  Reward-to-go based Policy Gradient Derivation
    (Exploit conditional independence. expected grad-log-prob equal 0)
  Reducing variance of policy gradient estimate by Baseline
    (\( \text{Var}(x - y) \) can be less than \( \text{Var}(x) \). Expected grad-log-prob equal 0)
Implementing Policy Gradient in Pytorch
  (Credit to the notebook from the last year’s CSC421)
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 $s$ 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.
“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 $\theta$.

$$p_{\theta}(a_t|s_t) = \pi_{\theta}(a_t|s_t) = \pi(a_t|s_t; \theta)$$
Problem Setup
Different implementations of a stochastic policy

\[ p_\theta(a_t|s_t) = \pi_\theta(a_t|s_t) = \pi(a_t|s_t; \theta) \]

Based on the problem, implement different types of stochastic policy

- If \( A \) is discrete, but \( S \) is continuous or too large (e.g. Atari), use a function approximator such as NN to map the state vector \( s \) to the distribution over actions using softmax for the output layer. i.e. The size of your network output will be \( A \), with each output denoting the probability of taking that action.
Problem Setup

Different implementations of a stochastic policy

\[ p_\theta(a_t|s_t) = \pi_\theta(a_t|s_t) = \pi(a_t|s_t; \theta) \]

Based on the problem, implement different types of stochastic policy:

- If both \( S \) and \( A \) are continuous or too large (e.g. Robot control), map \( s \) to parameters associated with distributions such as \( \mu \) and \( \sigma^2 \) for Gaussian distribution. Then sample the value, which we treat as the action, from this distribution under the mapped \( \mu \) and \( \sigma \).

  (A simpler solution is to discretize continuous action space. e.g. OpenAI Dota2 bot [1])
Trajectory is nothing but a set of random variables, and its distribution is a joint distribution over $2T + 1$ r.v.:

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

$$p(\tau; \theta) = p(s_1, a_1, s_2, ..., s_T, a_T, s_{T+1}; \theta) = (*)$$
We can simplify using conditional independences from DAG:

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

Remark: we will use \( p(\tau; \theta) \) to denote that changing our policy parameters \( \theta \) 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.)

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

\[
s_{t+1} \sim p(s_{t+1}|s_t, a_t)
\]

\[
a_t \sim \pi_\theta(a_t|s_t)
\]
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.
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 \( \tau \). (Consider finite-horizon undiscounted version in this tutorial)

\[
R(\tau) = \sum_{t=1}^{T} R(s_t, a_t)
\]
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 $\tau$. (Consider finite-horizon undiscounted version in this tutorial)

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

Return is also a random variable because it is a function of $2T$ random variables in the trajectory.
As $R(\tau)$ is random, the objective is to maximize the expected return $\mathbb{E} [R(\tau)]$ w.r.t $\theta$. By the law of the unconscious statistician, we can write it as the expectation under $\tau$ distribution $p(\tau; \theta)$:

$$\mathcal{J}(\theta) = \mathbb{E} [R(\tau)] = \mathbb{E}_{\tau \sim p(\tau; \theta)} [R(\tau)] = (*)$$
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 $\theta$. By the law of the unconscious statistician, we can write it as the expectation under $\tau$ distribution $p(\tau; \theta)$:

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

And by the ancestral sampling, we can further simplify:

$$(*') = \mathbb{E}_{s_1 \sim \rho_0(s)} \left[ \sum_{t=1}^{T} R(s_t, a_t) \right]$$

\[
\begin{align*}
    s_1 &\sim \rho_0(s) \\
    a_t &\sim \pi(\theta)(a_t|s_t) \\
    s_{t+1} &\sim p(s_{t+1}|s_t, a_t)
\end{align*}
\]
We can make a one-step optimization for the current policy $\pi_{\theta_k}(a_t|s_t)$ to $\pi_{\theta_{k+1}}(a_t|s_t)$ for maximizing $\mathcal{J}(\theta)$ by gradient ascent:

$$\theta_{k+1} = \theta_k + \alpha \nabla_{\theta} \mathcal{J}(\theta)|_{\theta_k}$$
Policy Optimization by Policy Gradient Ascent

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

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

Gradient of the objective w.r.t policy

$$\nabla_\theta \mathcal{J}(\theta)|_{\theta_k} = \mathbb{E}_{s_1 \sim \rho_0(s)} \mathbb{E}_{a_t \sim \pi_{\theta_k}(a_t|s_t)} \mathbb{E}_{s_{t+1} \sim p(s_{t+1}|s_t,a_t)} \left[ \sum_{t=1}^{T} \nabla_\theta \log \pi_{\theta_k}(a_t|s_t) \left[ \sum_{t'=1}^{T} R(s_{t'},a_{t'}) \right] \right]$$
Step 1 using log-derivative trick

\[ \nabla_{\theta} J(\theta) = \nabla_{\theta} \mathbb{E}_{\tau \sim p(\tau; \theta)} [R(\tau)] \]
Step1 using log-derivative trick

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

\[ = \nabla_{\theta} \int p(\tau; \theta) R(\tau) \, d\tau \]
Step1 using log-derivative trick

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

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

$$= \int \nabla_\theta p(\tau; \theta) R(\tau) \, d\tau$$
Policy Optimization by Policy Gradient Ascent

Deriving policy gradient (Step1)

Step1 using log-derivative trick

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

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

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

$$= \int p(\tau; \theta)\nabla_{\theta} \log p(\tau; \theta) R(\tau) \, d\tau \quad : \quad \nabla_{\theta} \log p(\tau; \theta) = \frac{\nabla_{\theta} p(\tau; \theta)}{p(\tau; \theta)}$$
Policy Optimization by Policy Gradient Ascent

Deriving policy gradient (Step1)

**Step1 using log-derivative trick**

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

\[
= \nabla_{\theta} \int p(\tau; \theta) R(\tau) \, d\tau
\]

\[
= \int \nabla_{\theta} p(\tau; \theta) R(\tau) \, d\tau
\]

\[
= \int p(\tau; \theta) \nabla_{\theta} \log p(\tau; \theta) R(\tau) \, d\tau
\]

\[
= \mathbb{E}_{\tau \sim p(\tau; \theta)} \left[ \nabla_{\theta} \log p(\tau; \theta) R(\tau) \right]
\]

\[
\nabla_{\theta} \log p(\tau; \theta) = \frac{\nabla_{\theta} p(\tau; \theta)}{p(\tau; \theta)}
\]
Policy Optimization by Policy Gradient Ascent

Deriving policy gradient (Step 2)

Step 2 using conditional independences

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

$$= \mathbb{E}_{s_1 \sim \rho_0(s)} \left[ \nabla_\theta \log \left( \rho_0(s_1) \prod_{t=1}^{T} \pi_\theta(a_t | s_t) p(s_{t+1} | s_t, a_t) \right) \sum_{t'=1}^{T} R(s_{t'}, a_{t'}) \right]$$
Step2 using conditional independences

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

Now use ancestral sampling

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

where $\mathbb{E}_{\tau} \left[ \nabla_{\theta} \left( \log \rho_0(s_1) + \sum_{t=1}^{T} \log \pi_{\theta}(a_t|s_t) + \sum_{t=1}^{T} \log p(s_{t+1}|s_t, a_t) \right) \right]$
Policy Optimization by Policy Gradient Ascent

Deriving policy gradient (Step2)

Step2 using conditional independences

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

Now use ancestral sampling

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

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

$$= \nabla_\theta \rho_0(s_1) + \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_t | s_t) + \sum_{t=1}^T \nabla_\theta \log p(s_{t+1} | s_t, a_t)$$
Hence, the policy gradient w.r.t the current policy parameters is:

\[ \nabla_{\theta} J(\theta) |_{\theta_k} = \mathbb{E}_{s_1 \sim \rho_0(s), \ a_t \sim \pi_{\theta_k}(a_t|s_t), \ s_{t+1} \sim p(s_{t+1}|s_t,a_t)} \left[ \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta_k}(a_t|s_t) \left[ \sum_{t'=1}^{T} R(s_{t'}, a_{t'}) \right] \right] \]
Policy Optimization by Policy Gradient Ascent

Deriving policy gradient (Final form)

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

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

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

In practice, this gradient is estimated by executing the policy \( \pi_{\theta_k} \) in the environment \( N \) times (\( N \) times ancestral sampling).
Hence, the policy gradient w.r.t the current policy parameters is:

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

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

The log-derivative trick in step 1 allows for this type of gradient estimate of the expected value even though the thing inside expectation was a blackbox function using samples from the parameterized distribution. “score function estimator”
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]
State $S$: 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.**
- **Action $A$:** Continuous, but discretized into 8000-80000 actions.
Besides winning the game, intermediate rewards such as kill enemies are provided. PPO, an improved policy gradient method, is used to train the policy with Adam optimizer. [1]
Rollouts against the current bot means self-play [1]
Next: Deriving Reward-to-go Policy Gradient
Intuitively, the rewards $R(s_1, a_1), \ldots, R(s_{t-1}, a_{t-1})$ obtained before taking the action $a_t$ should not tell how good action $a_t$ is. This claim is saying:

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

$$= \mathbb{E}_{s_1 \sim \rho_0(s), a_t \sim \pi_{\theta_k}(a_t|s_t), s_{t+1} \sim p(s_{t+1}|s_t, a_t)} \left[ \sum_{t=1}^T \nabla_\theta \log \pi_{\theta_k}(a_t|s_t) \left[ \sum_{t'=t}^T R(s_{t'}, a_{t'}) \right] \right]$$
Proved using the DAG structure and expected grad-log-prob equal 0:

\[
\nabla_{\theta} J(\theta) |_{\theta_k} = \mathbb{E}_{s_1 \sim \rho_0(s)} \left[ \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta_k}(a_t | s_t) \right] \left[ \sum_{t'=1}^{T} R(s_{t'}, a_{t'}) \right]
\]
Proof of Reward-to-go Policy Gradient

Proved using the DAG structure and expected grad-log-prob equal 0:

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

by linearity
Proved using the DAG structure and expected grad-log-prob equal 0:

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

$$= \sum_{t=1}^{T} \sum_{t'=1}^{T} \mathbb{E}_{s_t,a_t,s_{t'},a_{t'}} \left[ \nabla_{\theta} \log \pi_{\theta_k}(a_t|s_t) R(s_{t'}, a_{t'}) \right] \text{ by linearity}$$

$$= \sum_{t=1}^{T} \sum_{t'=1}^{T} \mathbb{E}_{s_{t'},a_{t'}} \left[ \mathbb{E}_{s_t,a_t} \left[ \nabla_{\theta} \log \pi_{\theta_k}(a_t|s_t) R(s_{t'}, a_{t'}) | s_t, a_t \right] \right] \text{ by iterated } \mathbb{E}$$
Reward-to-go Policy Gradient (Proof)

Proved using the DAG structure and expected grad-log-prob equal 0:

\[
\nabla_\theta J(\theta)|_{\theta_k} = \mathbb{E}_{s_1 \sim \rho_0(s)} \left[ \sum_{t=1}^{T} \nabla_\theta \log \pi_{\theta_k}(a_t|s_t) \sum_{t'=1}^{T} R(s_{t'}, a_{t'}) \right]
\]

\[
= \sum_{t=1}^{T} \sum_{t'=1}^{T} \mathbb{E}_{s_t, a_t, s_{t'}, a_{t'}} \left[ \nabla_\theta \log \pi_{\theta_k}(a_t|s_t) R(s_{t'}, a_{t'}) \right]
\]

by linearity

\[
= \sum_{t=1}^{T} \sum_{t'=1}^{T} \mathbb{E}_{s_{t'}, a_{t'}} \left[ \mathbb{E}_{s_t, a_t} \left[ \nabla_\theta \log \pi_{\theta_k}(a_t|s_t) R(s_{t'}, a_{t'}) | s_{t'}, a_{t'} \right] \right]
\]

by iterated \( \mathbb{E} \)

\[
= \sum_{t=1}^{T} \sum_{t'=1}^{T} \mathbb{E}_{s_{t'}, a_{t'}} \left[ R(s_{t'}, a_{t'}) \mathbb{E}_{s_t, a_t} \left[ \nabla_\theta \log \pi_{\theta_k}(a_t|s_t) | s_{t'}, a_{t'} \right] \right]
\]

\(*\) apply iterated expectation again
\[
= \sum_{t=1}^{T} \sum_{t'=1}^{T} \mathbb{E}_{s_{t'}, a_{t'}} \left[ R(s_{t'}, a_{t'}) \mathbb{E}_{s_t} \left[ \mathbb{E}_{a_t \sim p(a_t | s_t, s_{t'}, a_{t'}, \theta_k)} \left[ \nabla_\theta \log \pi_{\theta_k}(a_t | s_t) | s_t \right] | s_{t'}, a_{t'} \right] \right]
\]
Reward-to-go Policy Gradient (Proof)

$$= \sum_{t=1}^{T} \sum_{t'=1}^{T} \mathbb{E}_{s_{t'}, a_{t'}} \left[ R(s_{t'}, a_{t'}) \mathbb{E}_{s_t} \left[ \mathbb{E}_{a_t \sim p(a_t|s_t, s_{t'}, a_{t'}; \theta_k)} \left[ \nabla_{\theta} \log \pi_{\theta_k}(a_t|s_t) | s_t \right] | s_{t'}, a_{t'} \right] \right]$$

The above step can be verified by:

$$(\star) = \int \int p(s_t, a_t|s_{t'}, a_{t'}; \theta_k) \nabla_{\theta} \log \pi_{\theta_k}(a_t|s_t) \, ds_t \, da_t$$
Reward-to-go Policy Gradient (Proof)

\[
\sum_{t=1}^{T} \sum_{t'=1}^{T} \mathbb{E}_{s_t', a_t'} \left[ R(s_t', a_t') \mathbb{E}_{s_t} \left[ \mathbb{E}_{a_t \sim p(a_t|s_t, s_t', a_{t'}; \theta_k)} \left[ \nabla_{\theta} \log \pi_{\theta_k}(a_t|s_t) | s_t \right] s_{t'}, a_{t'} \right] \right]
\]

The above step can be verified by:

\[
(\star) = \int \int p(s_t, a_t|s_t', a_{t'}; \theta_k) \nabla_{\theta} \log \pi_{\theta_k}(a_t|s_t) \; ds_t da_t
\]

\[
= \int \int p(a_t|s_t, s_t', a_{t'}; \theta_k) p(s_t|s_t', a_{t'}; \theta_k) \nabla_{\theta} \log \pi_{\theta_k}(a_t|s_t) \; ds_t da_t
\]
Reward-to-go Policy Gradient (Proof)

\[
= \sum_{t=1}^{T} \sum_{t'=1}^{T} \mathbb{E}_{s_{t'}, a_{t'}} \left[ R(s_{t'}, a_{t'}) \mathbb{E}_{s_t} \left[ \mathbb{E}_{a_t \sim p(a_t|s_t, s_{t'}, a_{t'}; \theta_k)} \left[ \nabla_{\theta} \log \pi_{\theta_k}(a_t | s_t) | s_{t}, a_{t'} \right] \right] \right]
\]

The above step can be verified by:

\[(\star) = \int \int p(s_t, a_t | s_{t'}, a_{t'}; \theta_k) \nabla_{\theta} \log \pi_{\theta_k}(a_t | s_t) \ ds_t \ da_t \]

\[= \int \int p(a_t | s_t, s_{t'}, a_{t'}; \theta_k)p(s_t | s_{t'}, a_{t'}; \theta_k) \nabla_{\theta} \log \pi_{\theta_k}(a_t | s_t) \ ds_t \ da_t \]

\[= \int p(s_t | s_{t'}, a_{t'}; \theta_k) \left( \int p(a_t | s_t, s_{t'}, a_{t'}; \theta_k) \nabla_{\theta} \log \pi_{\theta_k}(a_t | s_t) \ da_t \right) \ ds_t \]
Reward-to-go Policy Gradient (Proof)

\[
\sum_{t=1}^{T} \sum_{t'=1}^{T} \mathbb{E}_{s_t' , a_t'} \left[ R(s_t' , a_t') \mathbb{E}_{s_t} \left[ \mathbb{E}_{a_t \sim p(a_t|s_t, s_t', a_t'; \theta_k)} \left[ \nabla_{\theta} \log \pi_{\theta_k}(a_t|s_t) | s_t \right] | s_t' , a_t' \right] \right]
\]

The above step can be verified by:

\[
(\star) = \int \int p(s_t, a_t|s_t', a_t'; \theta_k) \nabla_{\theta} \log \pi_{\theta_k}(a_t|s_t) \, ds_t \, da_t
\]

\[
= \int \int p(a_t|s_t, s_t', a_t'; \theta_k) p(s_t|s_t', a_t'; \theta_k) \nabla_{\theta} \log \pi_{\theta_k}(a_t|s_t) \, ds_t \, da_t
\]

\[
= \int p(\cdot|s_t, a_t'; \theta_k) \left( \int p(a_t|s_t, s_t', a_t'; \theta_k) \nabla_{\theta} \log \pi_{\theta_k}(a_t|s_t) \, da_t \right) \, ds_t
\]

\[
= \mathbb{E}_{s_t} \mathbb{E}_{a_t \sim p(a_t|s_t, s_t', a_t'; \theta_k)} \left[ \nabla_{\theta} \log \pi_{\theta_k}(a_t|s_t) | s_t \right] | s_t' , a_t'
\]
From the graphical model, we can observe the conditional independence when $t' < t$:
From the graphical model, we can observe the conditional independence when $t' < t$:

\[ a_t \perp s_{t'}, a_{t'} | s_t, a_t \]

Hence, if $t' < t$, $p(a_t | s_t, s_{t'}, a_{t'}; \theta) = p(a_t | s_t; \theta) = \pi_\theta(a_t | s_t)$
From the graphical model, we can observe the conditional independence when \( t' < t \):

\[
S_{t'} \quad \cdots \quad S_t
\]

\[
a_{t'} \perp s_{t'}, a_{t'} | s_t \quad \downarrow \quad a_t
\]

\[
S_t \quad \cdots \quad S_{t'}
\]

\[
a_t \not\perp s_{t'}, a_{t'} | s_t \quad \downarrow \quad a_{t'}
\]

Hence, if \( t' < t \),

\[
p(a_t|s_t, s', a'_t; \theta) = p(a_t|s_t; \theta) = \pi_\theta(a_t|s_t)
\]

\[
(\diamond) = \mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)} [\nabla_\theta \log \pi_\theta(a_t|s_t)|s_t] = \int \pi_\theta(a_t|s_t) \nabla_\theta \log \pi_\theta(a_t|s_t) \, da_t
\]

\[
= \int \nabla_\theta \pi_\theta(a_t|s_t) \, da_t = \nabla_\theta \int \pi_\theta(a_t|s_t) \, da_t = \nabla_\theta 1 = 0
\]
Hence, all the reward terms for $t' < t$ will naturally disappear when taking the expectation over $\tau = (s_1, a_1, ..., s_T, a_T, s_{T+1})$

\[
\nabla_{\theta} J(\theta)|_{\theta_k} = \mathbb{E}_{s_1 \sim \rho_0(s)} \mathbb{E}_{a_t \sim \pi_{\theta_k}(a_t|s_t)} \mathbb{E}_{s_{t+1} \sim p(s_{t+1}|s_t,a_t)} \left[ \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta_k}(a_t|s_t) \left[ \sum_{t'=t}^{T} R(s_{t'}, a_{t'}) \right] \right]
\]
Gradient of the objective was an expectation, so we can only compute the gradient estimate (which is a random variable) from sampled trajectories:

\[
\hat{g} = \hat{\nabla}_\theta J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left[ \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a^{(i)}_t | s^{(i)}_t) \left[ \sum_{t'=t}^{T} R(s^{(i)}_{t'}, a^{(i)}_{t'}) \right] \right]
\]

As \( \hat{g} \) is random, we can talk about **bias** and **variance**. It is easy to see that this estimator is unbiased, \( \mathbb{E} [\hat{g}] = g = \nabla_\theta J(\theta) \).
Gradient of the objective was an expectation, so we can only compute the gradient estimate (which is a random variable) from sampled trajectories:

\[
\hat{g} = \hat{\nabla}_\theta J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left[ \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a^{(i)}_t | s^{(i)}_t) \left[ \sum_{t'=t}^{T} R(s^{(i)}_{t'}, a^{(i)}_{t'}) \right] \right]
\]

As \( \hat{g} \) is random, we can talk about **bias** and **variance**. It is easy to see that this estimator is unbiased, \( \mathbb{E}[\hat{g}] = g = \nabla_\theta J(\theta) \). Consider

\[
\hat{g}' = \frac{1}{N} \sum_{i=1}^{N} \left[ \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a^{(i)}_t | s^{(i)}_t) \left[ \sum_{t'=t}^{T} R(s^{(i)}_{t'}, a^{(i)}_{t'}) - V_{\pi_\theta}(s^{(i)}_t) \right] \right]
\]

where \( V_{\pi_\theta}(s_t) \) is random since \( s_t \) is random in this context.
Reducing variance of policy gradient estimate by Baseline

\[ \hat{g}' = \frac{1}{N} \sum_{i=1}^{N} \left[ \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a^{(i)}_{t}|s^{(i)}_{t}) \left[ \sum_{t'=t}^{T} R(s^{(i)}_{t'}, a^{(i)}_{t'}) - V_{\pi_{\theta}}(s^{(i)}_{t}) \right] \right] \]

\[ = \frac{1}{N} \sum_{i=1}^{N} \left[ \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a^{(i)}_{t}|s^{(i)}_{t}) \left[ \sum_{t'=t}^{T} R(s^{(i)}_{t'}, a^{(i)}_{t'}) \right] \right] \]

\[ - \frac{1}{N} \sum_{i=1}^{N} \left[ \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a^{(i)}_{t}|s^{(i)}_{t}) \left[ V_{\pi_{\theta}}(s^{(i)}_{t}) \right] \right] \]

\[ = \hat{g} - f \]

\[ \mathbb{E} [f] = \mathbb{E} \left[ \frac{1}{N} \sum_{i=1}^{N} \left[ \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a^{(i)}_{t}|s^{(i)}_{t}) \left[ V_{\pi_{\theta}}(s^{(i)}_{t}) \right] \right] \right] \]

\[ = \mathbb{E} \left[ \frac{1}{N} \sum_{i=1}^{N} \left[ \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a_{t}|s_{t}) \left[ V_{\pi_{\theta}}(s_{t}) \right] \right] \right] \tau_{i} \text{ i.i.d} \]
Reducing variance of policy gradient estimate by Baseline

Similar to the derivation in Reward-to-go PG, the expected grad-log-prob equal 0 is also useful here.

\[
\frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \mathbb{E}_{s_t} \left[ V_{\pi_\theta}(s_t) \mathbb{E}_{a_t \sim \pi_\theta} \left[ \nabla_\theta \log \pi_\theta(a_t|s_t)|s_t \right] \right] \quad \text{out from inner } \mathbb{E}
\]
Reducing variance of policy gradient estimate by Baseline

Similar to the derivation in Reward-to-go PG, the expected grad-log-prob equal 0 is also useful here.

\[
\frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \mathbb{E}_{s_t} \left[ V_{\pi_\theta}(s_t) \mathbb{E}_{a_t \sim \pi_\theta} \left[ \nabla_\theta \log \pi_\theta(a_t|s_t)|s_t \right] \right]
\]

\[
= \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \mathbb{E}_{s_t} \left[ V_{\pi_\theta}(s_t) \int \pi_\theta(a_t|s_t) \nabla_\theta \log \pi_\theta(a_t|s_t) \, da_t \right]
\]
Reducing variance of policy gradient estimate by Baseline

Similar to the derivation in Reward-to-go PG, the expected grad-log-prob equal 0 is also useful here.

\[
\frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \mathbb{E}_{s_t} \left[ V_{\pi_\theta}(s_t) \mathbb{E}_{a_t \sim \pi_\theta} [\nabla_\theta \log \pi_\theta(a_t|s_t)|s_t] \right] = 0
\]

by Sheng Jia  
CSC413 Tutorial11: Policy Gradient  
March 31st, 2020  54 / 58
\[ \mathbb{E} [\hat{g}'] = \mathbb{E} [\hat{g} - f] = \mathbb{E} [\hat{g}] - \mathbb{E} [f] = g + 0 = \nabla_{\theta} J(\theta) \]

Hence, this new gradient estimate \( \hat{g}' \) is unbiased so we can use it for policy gradient ascent.
Reducing variance of policy gradient estimate by Baseline

\[ E[\hat{g}'] = E[\hat{g} - f] = E[\hat{g}] - E[f] = g + 0 = \nabla_\theta J(\theta) \]

Hence, this new gradient estimate \( \hat{g}' \) is unbiased so we can use it for policy gradient ascent. But the point is that we want to decrease the variance by:

\[
\text{Var}(\hat{g}') = \text{Var}(\hat{g}) + \text{Var}(f) - 2\text{Cov}(\hat{g}, f) \leq \text{Var}(\hat{g})
\]

if \( \text{Cov}(\hat{g}, f) \geq \frac{1}{2} \text{Var}(f) \)

In practice, we do see strong positive correlations between \( \hat{g} \) and \( f \) because the empirical rewards for \((s_t, a_t, \ldots)\) and the value function evaluation for the sampled state \( s_t \) do positively correlate.
Demo in PyTorch

(Credit to last year CSC421 RL tutorial)