CSC413/2516 Tutorial 11

Reinforcement Learning, Policy Gradient

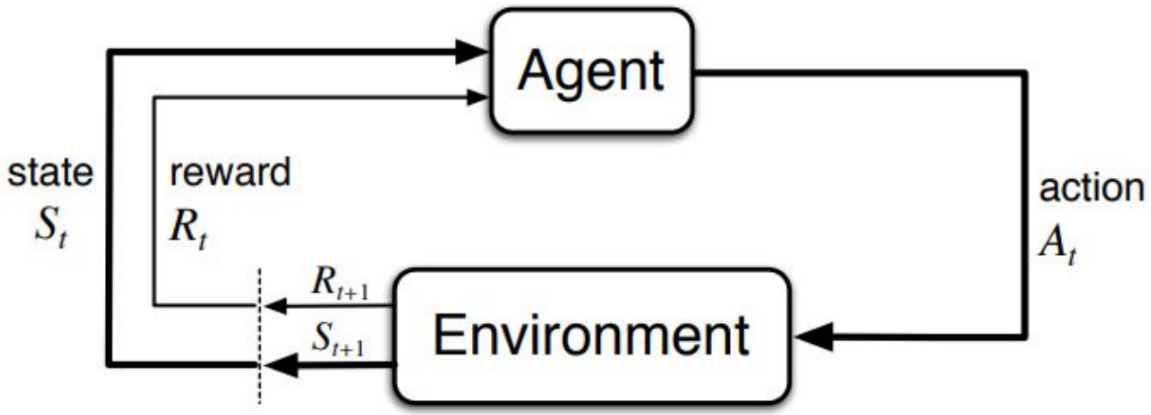
Based on Slides by Irene Zhang, Sheng Jia

March 29, 2022

Problem Setup

State

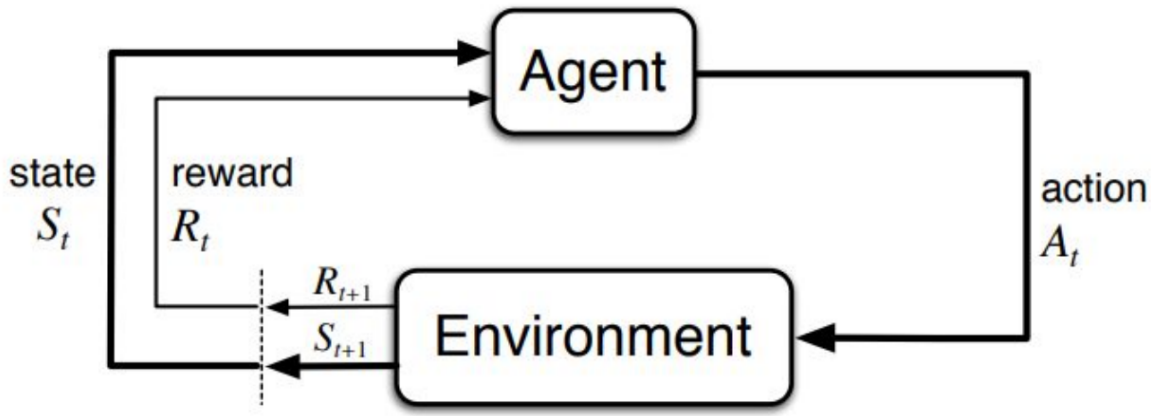
- s_t: state at time step t. Is a complete description of the task/environment (assume full observability for simplicity in this tutorial), and is input to the agent
- a_t: action taken by the agent at time step t (output from the agent)



Problem Setup

State

- s_t: state at time step t. Is a complete description of the task/environment (assume full observability for simplicity in this tutorial), and is input to the agent
- a_t: action taken by the agent at time step t (output from the agent)



- Examples:
  - s_t = agent location on grid, a_t = movement direction
  - s_t = financial data, a_t = buy or sell
  - s_t = sequence of frames from a video, a_t = game action / robot movement

Problem Setup

Agent's Policy

- "Agent" is an abstract concept, but we can formulate how the agent behaves by a 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

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

- $\mathcal{S}$ = state space (set of possible states)
- $\mathcal{A}$ = action space (set of possible actions)
- If both $\mathcal{S}$ and $\mathcal{A}$ are discrete and small, can simply use a table of mappings from states to probability distributions over actions

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

- $\mathcal{S}$ = state space (set of possible states)
- $\mathcal{A}$ = action space (set of possible actions)
- If both $\mathcal{S}$ and $\mathcal{A}$ are discrete and small, can simply use a table of mappings from states to probability distributions over actions
- If $\mathcal{A}$ is discrete, but $\mathcal{S}$ 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

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

- $\mathcal{S}$ = state space (set of possible states)
- $\mathcal{A}$ = action space (set of possible actions)
- If both $\mathcal{S}$ and $\mathcal{A}$ are discrete and small, can simply use a table of mappings from states to probability distributions over actions
- If $\mathcal{A}$ is discrete, but $\mathcal{S}$ 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
- If both $\mathcal{S}$ and $\mathcal{A}$ are continuous or too large (e.g. Robot control), map s to parameters associated with distributions such as μ and σ^2 for Gaussian distribution. Then sample actions from the distribution (A simpler solution is to discretize continuous action space. e.g. OpenAI Dota2 bot [1])

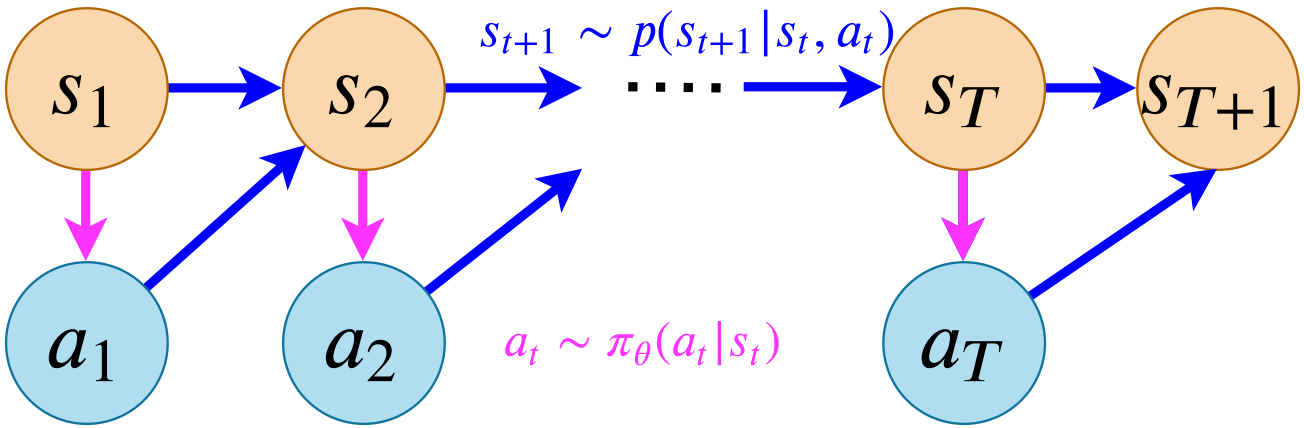
Problem Setup

Trajectory

- τ = trajectory, a record of states and actions over T time steps
- Trajectory is a set of random variables, and its distribution is a joint distribution over $2T + 1$ r.v.:

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

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



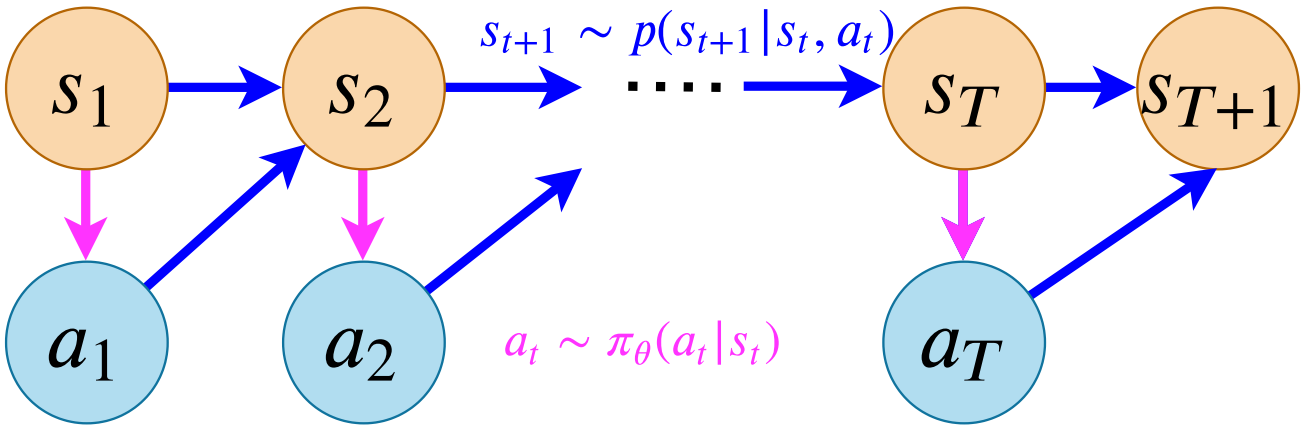
Problem Setup

Trajectory

- We can simplify using conditional independences from DAG (Markov assumption, state is a complete description):

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

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

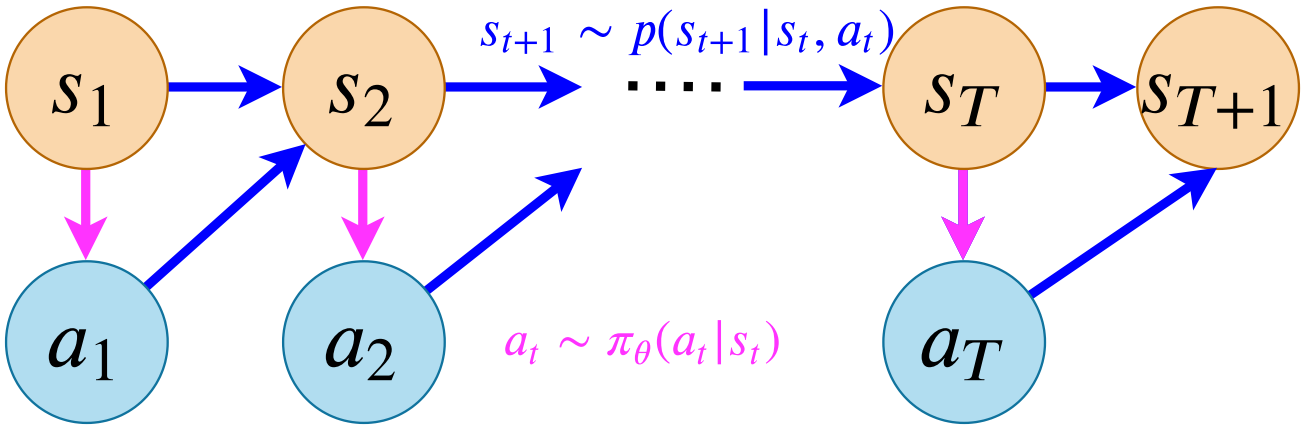


Problem Setup

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

- Reward $r_t = R(s_t, a_t)$ measures how well action a_t is in state s_t for the agent. This is computed by a blackbox function $R(s_t, a_t)$ from the environment

Objective in Reinforcement Learning

Reward, Return

- Reward $r_t = R(s_t, a_t)$ measures how well action a_t is in state s_t for the agent. 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

Reward, Return

- Reward $r_t = R(s_t, a_t)$ measures how well action a_t is in state s_t for the agent. 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)

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

Return is also a random variable because it is a function of $2T$ random variables in the trajectory

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 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

A method to "skill up" the agent

- Our goal: find the optimal policy $\theta^* = argmax_\theta \mathcal{J}(\theta)$

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:

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

Policy Optimization by Policy Gradient Ascent

A method to "skill up" the agent

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 (Policy Gradient)

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

Step1 using log-derivative trick

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

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

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

- Side note: We can take the gradient inside the expectation because of Leibniz's integral rule

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

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

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

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:

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

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).

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

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

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

The log-derivative trick in step 1 allows for this type of gradient estimate of the expected value even though the thing inside the expectation was a blackbox function using samples from the parameterized distribution.

This is known as the score function estimator, or the REINFORCE gradient estimator

REINFORCE Algorithm

Putting the above together, we get the most simple policy gradient method, the REINFORCE algorithm:

1. Sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t)$ (run in the environment)
2. Compute the gradient estimate: $\nabla_\theta \mathcal{J}(\boldsymbol{\theta})|_{\boldsymbol{\theta}_k} \approx \frac{1}{N}\sum_{i=1}^{N}\left[\sum_{t=1}^{T}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}_k}(a_t^{(i)}|s_t^{(i)})\left[\sum_{t'=1}^{T}R(s_{t'}^{(i)},a_{t'}^{(i)})\right]\right]$
3. Update the policy via gradient ascent
4. Repeat the above

Applying Policy Gradient for Playing Dota2

Successful application of policy optimization by policy gradient

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

Applying 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 (limited vision, cannot see whole map), so not actually a state. LSTM is used to make use of information from previous observations.

Applying Policy Gradient for Playing Dota2

Action (Output of the policy)

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

Applying Policy Gradient for Playing Dota2

Policy optimization by policy gradient ascent

- Besides winning the game, intermediate rewards (e.g. for killing enemies) are provided. PPO, an improved policy gradient method , is used to train the policy with the Adam optimizer
- Rollouts are done using self-play against a mixture of the current bot and previous versions [1]

Reward-to-go Policy Gradient

- The rewards $R(s_1, a_1), ...R(s_{t-1}, a_{t-1})$ obtained before taking the action a_t should not tell you how good action a_t is.
- Intuitively, "I should evaluate my current action based only on how it affects my future"

This claim is saying:

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

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

Proved using the DAG structure and expected grad-log-prob $= 0$ (Appendix)

Reducing Variance of Policy Gradient Estimate by Baseline

Gradient of the objective was an expectation, so we can only compute the gradient estimate (which is a random variable) from sampled trajectories:

$$\hat{\mathbf{g}} = \hat{\nabla}_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \left[\sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t^{(i)} | s_t^{(i)}) \left[\sum_{t'=t}^{T} R(s_{t'}^{(i)}, a_{t'}^{(i)}) \right] \right]$$

As $\hat{\mathbf{g}}$ is random, we can talk about **bias** and **variance**.
The estimator is unbiased: $\mathbb{E}[\hat{\mathbf{g}}] = \mathbf{g} = \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta})$.

Reducing Variance of Policy Gradient Estimate by Baseline

Gradient of the objective was an expectation, so we can only compute the gradient estimate (which is a random variable) from sampled trajectories:

$$\hat{\mathbf{g}} = \hat{\nabla}_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \left[\sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t^{(i)} | s_t^{(i)}) \left[\sum_{t'=t}^{T} R(s_{t'}^{(i)}, a_{t'}^{(i)}) \right] \right]$$

As $\hat{\mathbf{g}}$ is random, we can talk about **bias** and **variance**.
The estimator is unbiased: $\mathbb{E}[\hat{\mathbf{g}}] = \mathbf{g} = \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta})$.
Now consider a baseline of the state value function (true expected return):

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

where $V_{\pi_{\boldsymbol{\theta}}}(s_t)$ is random since s_t is random in this context.

Reducing Variance of Policy Gradient Estimate by Baseline

- This new gradient estimate $\hat{\mathbf{g}}'$ is unbiased so we can use it for policy gradient ascent (derivations in Appendix):

$$\mathbb{E}\left[\hat{\mathbf{g}}'\right] = \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta})$$

Reducing Variance of Policy Gradient Estimate by Baseline

- This new gradient estimate $\hat{\mathbf{g}}'$ is unbiased so we can use it for policy gradient ascent (derivations in Appendix):

$$\mathbb{E}\left[\hat{\mathbf{g}}'\right] = \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta})$$

- However, the variance is usually decreased, as there tends to be a strong positive correlation between the empirical rewards for $(s_t, a_t, ...)$ and the value function evaluation for the sampled state
  - (see Appendix for details)
- Intuitively, the advantage (return - baseline) measures how much better or worse any action is relative to the expected return in the current state under the current policy

Demo in PyTorch (Credit to 2019 CSC421 RL tutorial)

Appendix

- Reward-to-go Policy Gradient (Proof)

Reward-to-go Policy Gradient (Proof)

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

Some remarks before we get started

- **Remark: separating expectation over multiple R.V.s**

$$
\begin{aligned}
\mathop{\mathbb{E}}_{A,B}[f(A,B)] &= \int_{A,B} P(A,B) f(A,B) \\
&= \int_A \int_B P(B \mid A) P(A) f(A,B) \\
&= \int_A P(A) \int_B P(B \mid A) f(A,B) \\
&= \int_A P(A) \mathop{\mathbb{E}}_B [f(A,B) \mid A] \\
&= \mathop{\mathbb{E}}_A [[\mathop{\mathbb{E}}_B [f(A,B) \mid A]]
\end{aligned}
$$

Reward-to-go Policy Gradient (Proof)

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

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

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

Reward-to-go Policy Gradient (Proof)

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

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

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

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

recall Remark: $\mathbb{E}_{A,B}[f(A,B)] = \mathbb{E}_A[[\mathbb{E}_B[f(A,B) \mid A]]$

Reward-to-go Policy Gradient (Proof)

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

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

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

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

$$= \sum_{t=1}^{T} \sum_{t'=1}^{T} \mathop{\mathbb{E}}_{s_{t'}, a_{t'}} \left[R(s_{t'}, a_{t'}) \underbrace{\mathop{\mathbb{E}}_{s_t, a_t} [\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}_k}(a_t|s_t) | s_{t'}, a_{t'}]}_{(\star) \text{ apply Remark again}} \right]$$

Reward-to-go Policy Gradient (Proof)

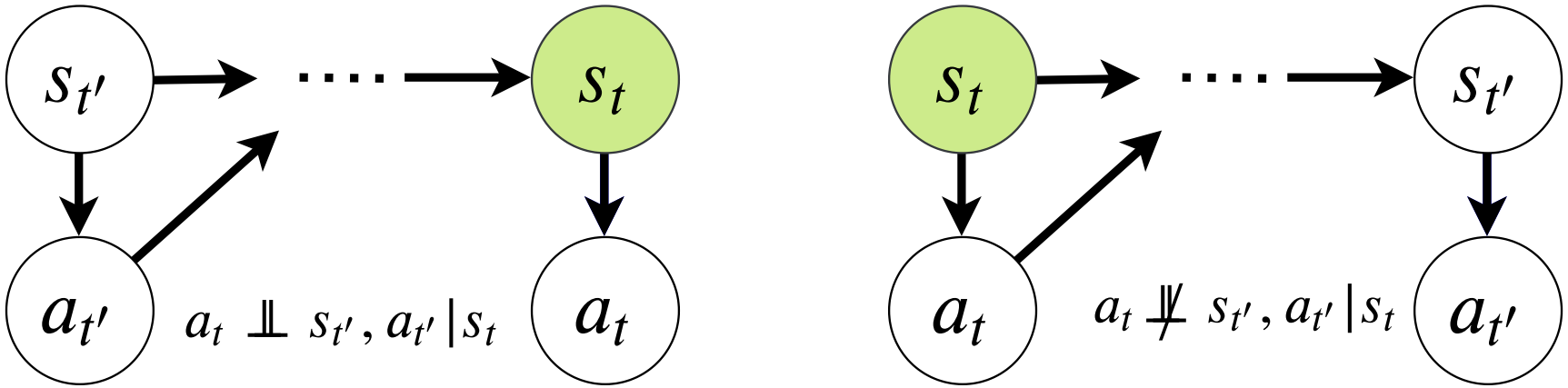
$$= \sum_{t=1}^{T} \sum_{t'=1}^{T} \mathop{\mathbb{E}}_{s_{t'}, a_{t'}} \left[R(s_{t'}, a_{t'}) \mathop{\mathbb{E}}_{s_t} \left[\underbrace{\mathop{\mathbb{E}}_{a_t \sim p(a_t | s_t, s_{t'}, a_{t'}; \theta_k)} [\nabla_\theta \log \pi_{\theta_k}(a_t | s_t) | s_t]}_{(\diamond)} | s_{t'}, a_{t'} \right]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

recall Remark: $\mathbb{E}_{A,B}[f(A,B)] = \mathbb{E}_A[[\mathbb{E}_B[f(A,B) \mid A]]$

Reward-to-go Policy Gradient (Proof)

Final step using DAG structure and Expected grad-log-prob equal 0

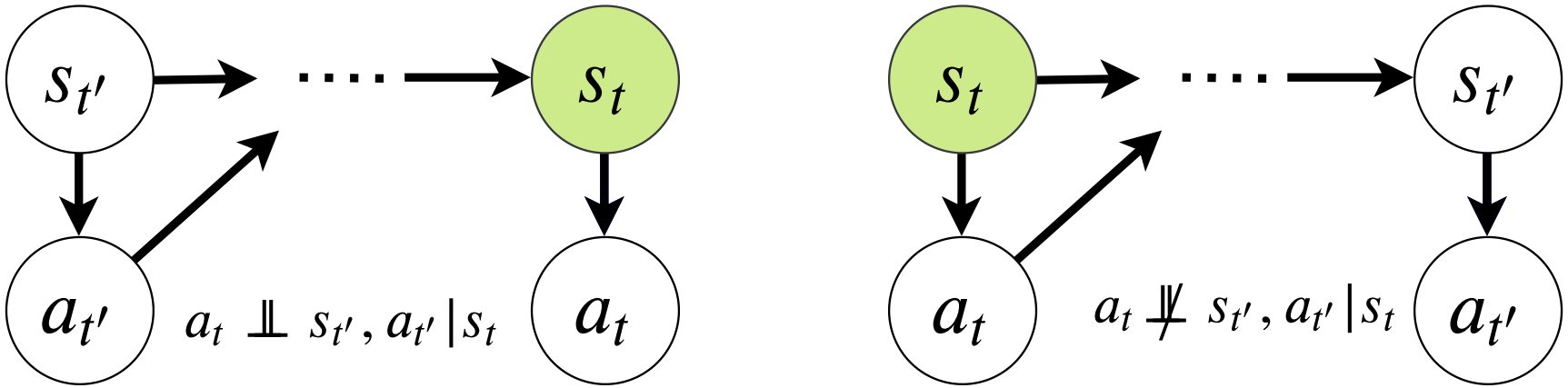
From the graphical model, we can observe the conditional independence when $t' < t$:



Reward-to-go Policy Gradient (Proof)

Final step using DAG structure and Expected grad-log-prob equal 0

From the graphical model, we can observe the conditional independence when $t' < t$:

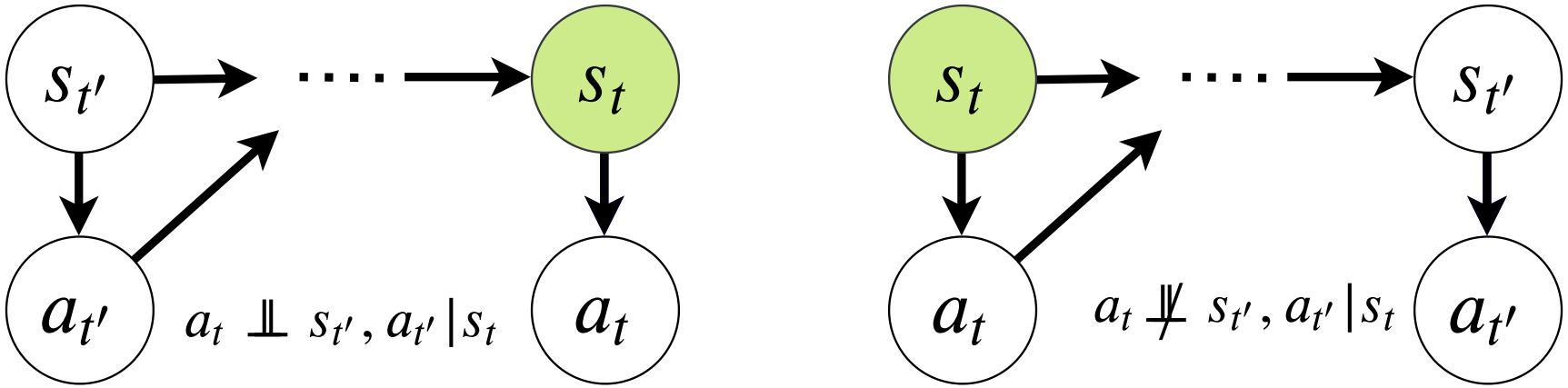


Hence, if $t' < t$, $p(a_t|s_t, s_{t'}, a_{t'}; \theta) = p(a_t|s_t; \theta) = \pi_\theta(a_t|s_t)$

Reward-to-go Policy Gradient (Proof)

Final step using DAG structure and Expected grad-log-prob equal 0

From the graphical model, we can observe the conditional independence when $t' < t$:



Hence, if $t' < t$, $p(a_t|s_t, s_{t'}, a_{t'}; \theta) = p(a_t|s_t; \theta) = \pi_\theta(a_t|s_t)$

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

$$= \int \nabla_\theta \pi_\theta(a_t|s_t)\, da_t = \nabla_\theta \int \pi_\theta(a_t|s_t)\, da_t = \nabla_\theta 1 = 0$$

Reward-to-go Policy Gradient

Hence, all the reward terms for $t' < t$ will naturally disappear when taking the expectation over $\tau = (s_1, a_1, ..., s_T, a_T, s_{T+1})$

Reward-to-go Policy Gradient

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

Appendix

- Reducing Variance of Policy Gradient Estimate by Baseline (Derivations)

Reducing Variance of Policy Gradient Estimate by Baseline

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

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

$$- \frac{1}{N}\sum_{i=1}^{N}\left[\sum_{t=1}^{T}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a_t^{(i)}|s_t^{(i)})\left[V_{\pi_{\boldsymbol{\theta}}}(s_t^{(i)})\right]\right]$$

$$= \hat{\mathbf{g}} - \mathbf{f}$$

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

$$= \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N}\left[\sum_{t=1}^{T}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a_t|s_t)\left[V_{\pi_{\boldsymbol{\theta}}}(s_t)\right]\right]\right] \qquad \boldsymbol{\tau}_i \text{ i.i.d}$$

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.

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

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.

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

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

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.

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

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

$$= \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbb{E}_{s_t}\left[V_{\pi_\theta}(s_t)\int \nabla\pi_\theta(a_t|s_t)\, da_t\right] = \cdots$$

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

$$= 0$$

Reducing Variance of Policy Gradient Estimate by Baseline

$$\mathbb{E}\left[\hat{\mathbf{g}}'\right] = \mathbb{E}\left[\hat{\mathbf{g}} - \mathbf{f}\right] = \mathbb{E}\left[\hat{\mathbf{g}}\right] - \mathbb{E}\left[\mathbf{f}\right] = \mathbf{g} + 0 = \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta})$$

Hence, this new gradient estimate $\hat{\mathbf{g}}'$ is unbiased so we can use it for policy gradient ascent.

Reducing Variance of Policy Gradient Estimate by Baseline

$$\mathbb{E}\left[\hat{\mathbf{g}}'\right] = \mathbb{E}\left[\hat{\mathbf{g}} - \mathbf{f}\right] = \mathbb{E}\left[\hat{\mathbf{g}}\right] - \mathbb{E}\left[\mathbf{f}\right] = \mathbf{g} + 0 = \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta})$$

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

$$\mathrm{Var}(\hat{\mathbf{g}}') = \mathrm{Var}(\hat{\mathbf{g}}) + \mathrm{Var}(\mathbf{f}) - 2\mathrm{Cov}(\hat{\mathbf{g}}, \mathbf{f}) \leq \mathrm{Var}(\hat{\mathbf{g}})$$

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

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

Reference

[1] Christopher Berner et al. “Dota 2 with Large Scale Deep Reinforcement Learning”. In: *arXiv preprint arXiv:1912.06680* (2019).