CSC413/2516 Tutorial11

Reinforcement Learning, Policy Gradient

Based on Slides by Sheng Jia

April 1st, 2021

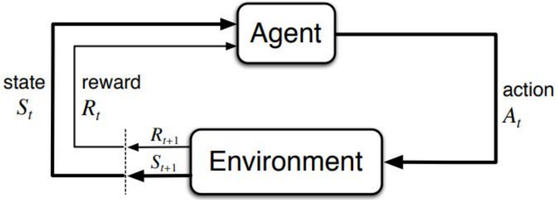
Content

- 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)
  - **Reward-to-go based Policy Gradient Derivation**
    (Exploit conditional independence. expected grad-log-prob equal 0)
  - **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
  (Credit to the notebook from CSC421 2019)

Problem Setup

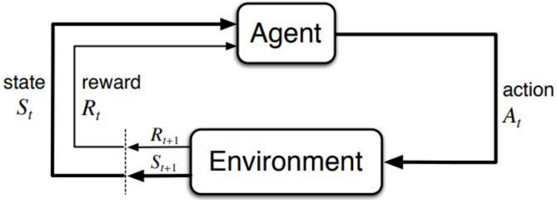
State

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 new cases of COVID-19 today.
a_t : whether or not wash your hands.

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

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

Based on the problem, implement different types of stochastic policy

- 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. i.e. The size of your network output will be $\mathcal{A}$, with each output denoting the probability of taking that action.

Problem Setup

Different implementations of a stochastic policy

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

Based on the problem, implement different types of stochastic policy

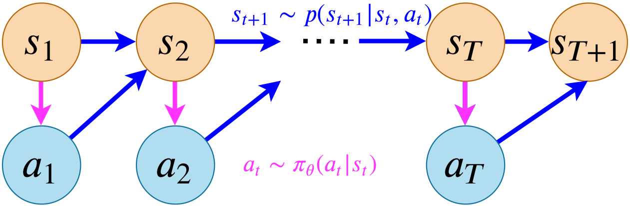
- If both $\mathcal{S}$ and $\mathcal{A}$ are continuous or too large (e.g. Robot control), map s to parameters associated with distributions such as μ and σ^2 for Gaussian distribution. Then sample the value, which we treat as the action, from this distribution under the mapped μ and σ.
(A simpler solution is to discretize continuous action space. e.g. OpenAI Dota2 bot [1])

Problem Setup
Trajectory

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

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

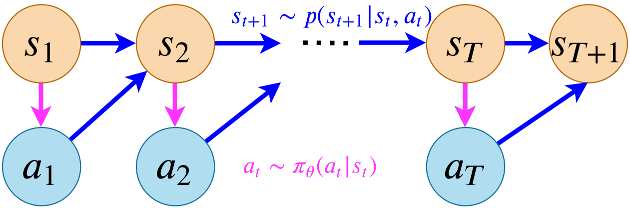
$$p(\boldsymbol{\tau}; \boldsymbol{\theta}) = p\left(s_1, a_1, s_2, ..., s_T, a_T, s_{T+1}; \boldsymbol{\theta}\right) = (\star)$$



- 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 $p(\boldsymbol{\tau};\boldsymbol{\theta})$ to denote that changing our policy parameters $\boldsymbol{\theta}$ induce a different trajectory distribution.

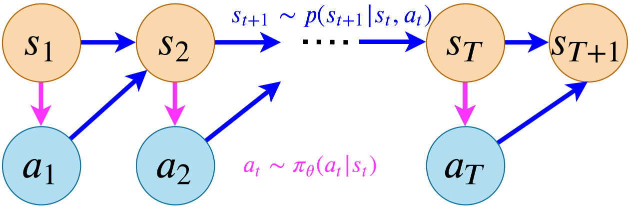


Problem Setup

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.)

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



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.

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

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)

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

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 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

A method to "skill up" the agent

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

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 $\pi_{\boldsymbol{\theta}_k}(a_t|s_t)$ to $\pi_{\boldsymbol{\theta}_{k+1}}(a_t|s_t)$ for maximizing $\mathcal{J}(\boldsymbol{\theta})$ by gradient ascent:

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

Gradient of the objective w.r.t policy

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

Step1 using log-derivative trick

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

Policy Optimization by Policy Gradient Ascent

Deriving policy gradient (Step1)

Step1 using log-derivative trick

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

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

Policy Optimization by Policy Gradient Ascent

Deriving policy gradient (Step1)

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

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{\boldsymbol{\tau} \sim p(\boldsymbol{\tau};\boldsymbol{\theta})} [R(\boldsymbol{\tau})] \\
&= \nabla_{\boldsymbol{\theta}} \int p(\boldsymbol{\tau};\boldsymbol{\theta}) R(\boldsymbol{\tau}) \, d\boldsymbol{\tau} \\
&= \int \nabla_{\boldsymbol{\theta}} p(\boldsymbol{\tau};\boldsymbol{\theta}) R(\boldsymbol{\tau}) \, d\boldsymbol{\tau} \\
&= \int p(\boldsymbol{\tau};\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau};\boldsymbol{\theta}) R(\boldsymbol{\tau}) \, d\boldsymbol{\tau} \qquad \because \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau};\boldsymbol{\theta}) = \frac{\nabla_{\boldsymbol{\theta}} p(\boldsymbol{\tau};\boldsymbol{\theta})}{p(\boldsymbol{\tau};\boldsymbol{\theta})} \\
&= \mathop{\mathbb{E}}_{\tau \sim p(\boldsymbol{\tau};\boldsymbol{\theta})} [\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau};\boldsymbol{\theta}) R(\boldsymbol{\tau})]
\end{aligned}
$$

Step2 using conditional independences

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

Now use ancestral sampling

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

Policy Optimization by Policy Gradient Ascent

Deriving policy gradient (Step2)

Step2 using conditional independences

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

Now use ancestral sampling

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

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

Step2 using conditional independences

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

Now use ancestral sampling

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

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

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

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

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

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

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

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

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

Policy Optimization by Policy Gradient Ascent

Deriving policy gradient (Final form)

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

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

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

The log-derivative trick in step 1 allows for this type of gradient estimate of the expected value even though the thing inside 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 result in the most simple policy gradient method, the REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathrm{a}_t \mid \mathrm{s}_t)$ (run it 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. repeat the above

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

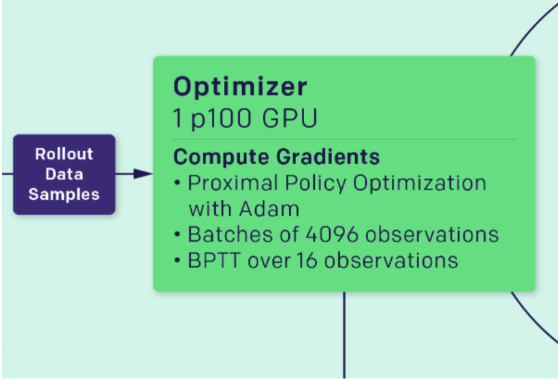
Action (Output of the policy)

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

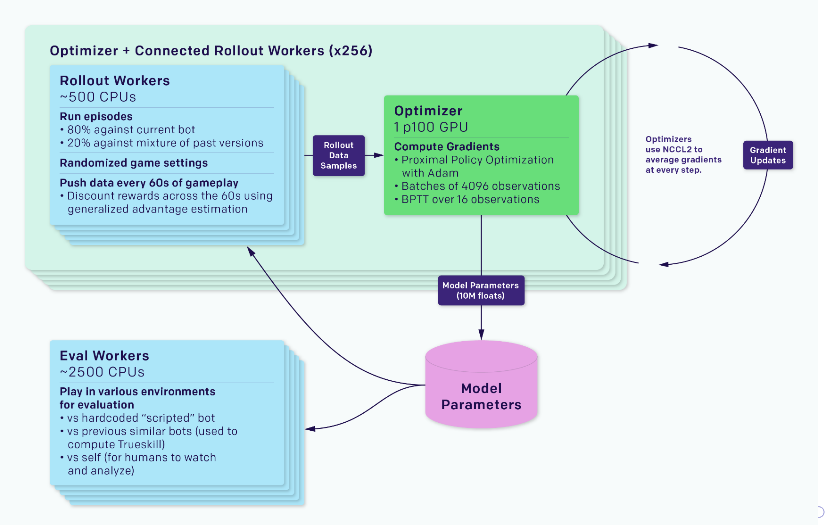
Break: Apply policy gradient for playing Dota2

Policy optimization by policy gradient ascent

- 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]



Back to Policy Gradient

Next: Deriving Reward-to-go Policy Gradient

Deriving 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 how good action a_t is.
- Intuitively, "What I do today should not change what happened yesterday"

This claim is saying:

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

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

Reward-to-go Policy Gradient (Proof)

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

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

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:

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

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

Reward-to-go Policy Gradient (Proof)

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

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

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

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

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:

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

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

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

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

Reward-to-go Policy Gradient (Proof)

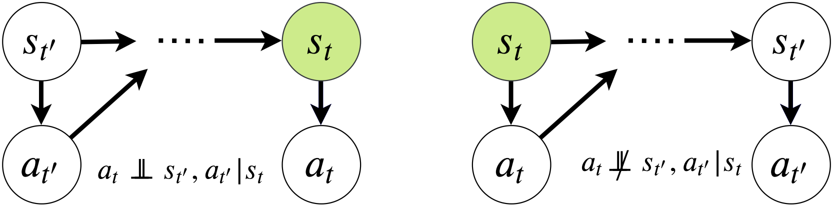
$$= \sum_{t=1}^{T} \sum_{t'=1}^{T} \mathop{\mathbb{E}}_{s_{t'}, a_{t'}} \left[R(s_{t'}, a_{t'}) \mathop{\mathbb{E}}_{s_t} \left[\underbrace{\mathop{\mathbb{E}}_{a_t \sim p(a_t | s_t, s_{t'}, a_{t'}; \boldsymbol{\theta}_k)} [\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}_k}(a_t | s_t) | s_t]}_{(\diamond)}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

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

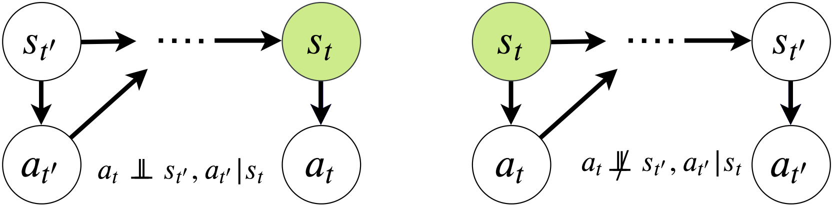
From the graphical model, we can observe the conditional independence when $t' < t$:



Reward-to-go Policy Gradient (Proof)

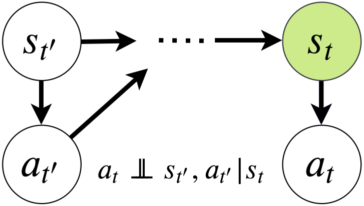
Final step using DAG structure and Expected grad-log-prob equal 0

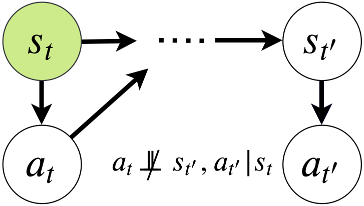
From the graphical model, we can observe the conditional independence when $t' < t$:



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

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





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

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

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

Reward-to-go Policy Gradient

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

Reward-to-go Policy Gradient

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

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:

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

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

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:

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

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

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

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

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

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

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

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

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

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

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.

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

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

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

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

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

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

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

$$\mathrm{Var}(\hat{\mathrm{g}}') = \mathrm{Var}(\hat{\mathrm{g}}) + \mathrm{Var}(\mathrm{f}) - 2\mathrm{Cov}(\hat{\mathrm{g}}, \mathrm{f}) \le \mathrm{Var}(\hat{\mathrm{g}})$$
$$\text{if } \mathrm{Cov}(\hat{\mathrm{g}}, \mathrm{f}) \ge \frac{1}{2}\mathrm{Var}(\mathrm{f})$$

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

Demo in PyTorch (Credit to 2019 CSC421 RL tutorial)

Reference

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