Markov Decision Process

Markov Property

$$\mathbb{P}[S_{t+1}|S_t, A_t] = \mathbb{P}[S_{t+1}|S_1, A_1, S_2, A_2 \ldots, S_t, A_t]$$

$$\mathbb{P}[R_t|S_t, A_t] = \mathbb{P}[R_t|S_1, A_1, S_2, A_2 \ldots, S_t, A_t] = 1$$

Markov Decision Process $\langle \mathcal{S}, \mathcal{S}_F, \mathcal{A}, \mathcal{P}, \mathcal{P}_0, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is an infinite state space
- $\mathcal{S}_F$ is a set of final states
- $\mathcal{A}$ is a infinite (finite) action space
- $\mathcal{P}$ is an uknown transition probability function

$$\mathcal{P}(s'|s, a) = \mathbb{P}[S_{t+1} = s'|S_t = s, A_t = a]$$

- $\mathcal{P}_0$ is an uknown initial state probability function
- $\mathcal{R}$ is an uknown reward function

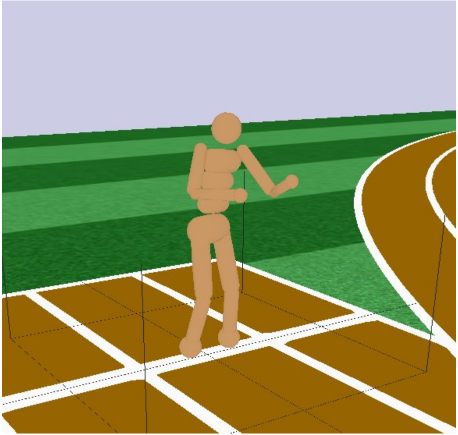
$$\mathcal{R}(s, a) = R_t \quad \Leftrightarrow \quad \mathbb{P}[R_t|S_t = s, A_t = a] = 1$$

- $\gamma \in [0, 1]$ is a discount coefficient

Example: Pendulum

- State space: $\mathbb{R}^2$ or screen pixels
- Action space: $[-2, 2]$
- Rewards: $-\psi^2 - 0.1\dot{\psi}^2 - 0.001a^2$

Example: Humanoid



- State Space: $\mathbb{R}^{26}$
- Action Space: $\mathbb{R}^{6}$
- Final states: fall

RL as an Finite-Dimensional Optimization Problem

Policy Approximation

$$\pi^\eta(a|s) \approx \pi_*(a|s),$$

where $\eta \in \mathbb{R}^N$ — parameter vector and $\exists \nabla_\eta \pi^\eta$ and, for example,

- $\pi^\eta(a|s) = \mathrm{Softmax}(F^\eta(s))$ in finite actions space
- $\pi^\eta(a|s) = \mathcal{N}(a|\nu^\eta(s), \sigma)$ in infinite actions space

Finite-dimensional optimization problem

$$J(\eta) = \mathbb{E}_{\pi^\eta}[G] \to \max_\eta,$$

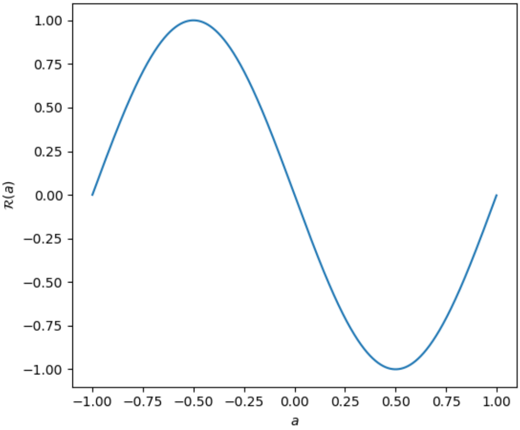
where

$$G = \sum_{t=0}^{\infty} \gamma^t \mathcal{R}(S_t, A_t)$$

Gradient Descent

$$\nabla_\eta J(\eta) =?$$

Continuous Bandit Example



- $\mathcal{S} = \{0, 1\}$
- $S_0 = 0$
- $S_F = \{1\}$
- $\mathcal{A} = [-1, 1]$
- $\mathcal{R}(0, a) = \mathcal{R}(a)$

Continuous Bandit Example: Gradient Calculation

$$\pi^\eta(a) = \pi^\eta(a|0), \quad J(\eta) = \mathbb{E}_{\pi^\eta}[G] = \int\limits_{a\in\mathcal{A}} \pi^\eta(a)\mathcal{R}(a)\mathrm{d}a$$

Monte-Carlo Gradient Calculation

$$\nabla_{\eta_i} J(\eta) \approx \frac{J(\eta + \delta e^i) - J(\eta)}{\delta}, \quad \delta > 0, \quad i \in \overline{1, N},$$

where $e_i \in \mathbb{R}^N$: $e^i_j = 1$, if $j = i$ and $e^i_j = 0$, otherwise,

$$J(\eta) \approx \frac{1}{n}\sum_{k=1}^{n} \mathcal{R}(a_k), \quad a_k \sim \pi^\eta(\cdot)$$

Continuous Bandit Example: Gradient Calculation

$$\pi^\eta(a) = \pi^\eta(a|0), \quad J(\eta) = \mathbb{E}_{\pi^\eta}[G] = \int_{a\in\mathcal{A}} \pi^\eta(a)\mathcal{R}(a)\mathrm{d}a$$

Gradient Calculation via Log Derivative Trick

$$\begin{aligned} \nabla_\eta J(\eta) &= \int_{a\in\mathcal{A}} \nabla_\eta \pi^\eta(a)\mathcal{R}(a)\mathrm{d}a \\ &\approx \frac{1}{n}\sum_{k=1}^{n} \mathcal{R}(a_k), \quad a_k \sim \nabla_\eta \pi^\eta(\cdot) \quad ? \end{aligned}$$

Continuous Bandit Example: Gradient Calculation

$$\pi^\eta(a) = \pi^\eta(a|0), \quad J(\eta) = \mathbb{E}_{\pi^\eta}[G] = \int\limits_{a\in\mathcal{A}} \pi^\eta(a)\mathcal{R}(a)\mathrm{d}a$$

Gradient Calculation via Log Derivative Trick

$$\begin{aligned}
\nabla_\eta J(\eta) &= \int\limits_{a\in\mathcal{A}} \nabla_\eta \pi^\eta(a)\mathcal{R}(a)\mathrm{d}a = \int\limits_{a\in\mathcal{A}} \pi^\eta(a)\frac{\nabla_\eta \pi^\eta(a)}{\pi^\eta(a)}\mathcal{R}(a)\mathrm{d}a \\
&= \int\limits_{a\in\mathcal{A}} \pi^\eta(a)\nabla_\eta \ln \pi^\eta(a)\mathcal{R}(a)\mathrm{d}a = \mathbb{E}_{\pi^\eta}[\nabla_\eta \ln \pi^\eta(a)\mathcal{R}(a)] \\
&\approx \frac{1}{n}\sum_{k=1}^{n} \nabla_\eta \ln \pi^\eta(a_k)\mathcal{R}(a_k), \quad a_k \sim \pi^\eta(\cdot)
\end{aligned}$$

Continuous Bandit Example: Two Remarks

Gradient Calculation via Log Derivative Trick

$$\nabla_\eta J(\eta) \approx \frac{1}{n}\sum_{k=1}^{n} \nabla_\eta \ln \pi^\eta(a_k)\mathcal{R}(a_k), \quad a_k \sim \pi^\eta(\cdot)$$

Remark 1: Constant Independence

If $\mathcal{R}(a) = \mathcal{R}_*(a) + c$ ($c = const$), then

$$\nabla_\eta J(\eta) \approx \frac{1}{n}\sum_{k=1}^{n} \nabla_\eta \ln \pi^\eta(a_k)\mathcal{R}_*(a_k), \quad a_k \sim \pi^\eta(\cdot)$$

Remark 2: Off-policy case

If $\beta(a)$ is a policy, then

$$\nabla_\eta J(\eta) \approx \frac{1}{n}\sum_{k=1}^{n} \frac{\nabla_\eta \pi^\eta(a_k)}{\beta(a_k)}\mathcal{R}_*(a_k), \quad a_k \sim \beta(\cdot)$$

Policy Gradient Theorem

Discounted State Distribution

$$\rho_\pi(s) = \sum_{t=0}^{\infty} \gamma^t \mathbb{P}[S_t = s|\pi]$$

On-Policy Gradient Theorem

Let $\exists \nabla_\eta \pi^\eta(a|s)$ and $\pi^\eta(a|s) \neq 0$ for any $s \in \mathcal{S}$, $a \in \mathcal{A}$. Then

$$\nabla_\eta J(\eta) = \mathbb{E}_{s\sim\rho_{\pi^\eta}, a\sim\pi^\eta}\left[\nabla_\eta \ln \pi^\eta(a|s)\Big(q_{\pi^\eta}(s,a) - c(s)\Big)\right]$$

Off-Policy Gradient Theorem

Let $\exists \nabla_\eta \pi^\eta(a|s)$ and $\beta(a|s) \neq 0$ for any $s \in \mathcal{S}$, $a \in \mathcal{A}$. Then

$$\nabla_\eta J(\eta) = \mathbb{E}_{s\sim\rho_\beta, a\sim\beta}\left[\frac{\nabla_\eta \pi^\eta(a|s)}{\beta(a|s)}\Big(q_{\pi^\eta}(s,a) - c(s)\Big)\right]$$

Reinforce: Main Points

Policy Approximation

$$\pi^{\eta}(a|s) \approx \pi_{*}(a|s),$$

Policy Gradient Theorem

$$\nabla_{\eta} J(\eta) = \mathbb{E}_{s \sim \rho_{\pi^{\eta}}, a \sim \pi^{\eta}} [\nabla_{\eta} \ln \pi^{\eta}(a|s) q_{\pi^{\eta}}(s,a)]$$

Monte-Carlo Estimate

$$q_{\pi^{\eta}}(s,a) = \mathbb{E}_{\pi^{\eta}}[G_t | S_t = s,\ A_t = a] \approx G_t = \sum_{i=t}^{T-1} \gamma^{i-t} R_i$$

Reinforce

Initialize an approximation $\pi^\eta(a|s)$.
For each episode, do

- According to π^η, get trajectory $\tau = (S_0, A_0, \ldots, S_T)$ and rewards $(R_0, \ldots, R_{T-1})$. Define $(G_0, \ldots, G_{T-1})$:

$$G_t = \sum_{i=t}^{T-1} \gamma^{i-t} R_i$$

- For each $t \in \overline{0, T-1}$, update η:

$$\eta \leftarrow \eta - \alpha \nabla_\eta \ln \pi^\eta(A_t|S_t)\, G_t$$

Advantage Actor-Critic (A2C): Main Points

Policy and Value Function Approximations

$$\pi^\eta(a|s) \approx \pi_*(a|s), \quad V^\theta(s,a) \approx v_{\pi^\eta}(s,a)$$

On-Policy Gradient Theorem

$$\nabla_\eta J(\eta) = \mathbb{E}_{s\sim\rho_{\pi^\eta}, a\sim\pi^\eta}\Big[\nabla_\eta \ln \pi^\eta(a|s) a_{\pi^\eta}(s,a)\Big],$$

where $a_{\pi^\eta}(s,a) = q_{\pi^\eta}(s,a) - v_{\pi^\eta}(s)$ or

$$a_{\pi^\eta}(s,a) = \mathbb{E}\big[R_t + \gamma v_{\pi^\eta}(S_{t+1}) - v_{\pi^\eta}(s)\big|S_t = s, A_t = a\big]$$

Approximation at Each Step

If $V^\theta \approx v_{\pi^\eta}$, then

$$\nabla_\eta J(\eta) \approx \nabla_\eta \ln \pi^\eta(A_t|S_t)\big(R_t + \gamma V^\theta(S_{t+1}) - V^\theta(S_t)\big)$$

Advantage Actor-Critic (A2C): Main Points

Policy and Value Function Approximations

$$\pi^\eta(a|s) \approx \pi_*(a|s), \quad V^\theta(s,a) \approx v_{\pi^\eta}(s,a)$$

Bellman Expectation Equation for v_π

$$v_{\pi^\eta}(s) = \mathbb{E}_{\pi^\eta}[R_t + \gamma v_{\pi^\eta}(S_{t+1})|S_t = s]$$

Approximation at Each Step

If

$$V^\theta(S_t) \approx R_t + \gamma V^\theta(S_{t+1}), \quad S_{t+1} \sim \mathcal{P}(\cdot|S_t, A_t), \quad A_t \sim \pi^\eta(\cdot|S_t)$$

then $V^\theta \approx v_{\pi^\eta}$

Advantage Actor-Critic (A2C)

Initialize $\pi^\eta(a|s)$ and $V^\theta(s)$
For each episode, do
During the episode, do

- Being in a state S_t, Agent acts $A_t \sim \pi^\eta(\cdot|S_t)$, gets a reward R_t, and goes to the next state S_{t+1}.
- By (S_t, A_t, R_t, S_{t+1}), define the Loss functions

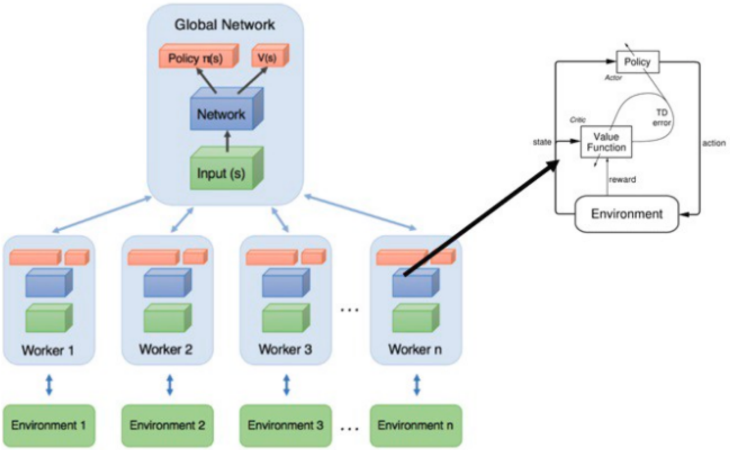
$$Loss_1(\theta) = \big(R_t + \gamma V^\theta(S_{t+1}) - V^\theta(S_t)\big)^2$$

$$Loss_2(\eta) = \ln \pi^\eta(A_t|S_t)\big(R_t + \gamma V^\theta(S_{t+1}) - V^\theta(S_t)\big)$$

and update parameters

$$\theta \leftarrow \theta - \alpha \nabla_\theta Loss_1(\theta), \quad \eta \leftarrow \eta + \beta \nabla_\eta Loss_2(\eta)$$

Asynchronous Advantage Actor-Critic (A3C)



Proximal Policy Optimization (PPO): Main Points

Policy and Value Function Approximations

$$\pi^\eta(a|s) \approx \pi_*(a|s), \quad V^\theta(s,a) \approx v_{\pi^\eta}(s,a)$$

Off-Policy Gradient Theorem

$$\nabla_\eta J(\eta) = \mathbb{E}_{s\sim\rho_\beta, a\sim\beta}\left[\frac{\nabla_\eta \pi^\eta(a|s)}{\beta(a|s)} a_{\pi^\eta}(s,a)\right]$$

After Each Trajectory

$$\min\left\{\frac{\pi^\eta(A_t|S_t)}{\pi^{\eta_{old}}(A_t|S_t)} A^\theta(S_t,A_t), g_\varepsilon(A^\theta(S_t,A_t))\right\} \to \max_\eta,$$

where $A^\theta(S_t,A_t) = R_t + \gamma V^\theta(S_{t+1}) - V^\theta(S_t)$ and

$$g_\varepsilon(A^\theta(S_t,A_t)) = \begin{cases} 1+\varepsilon, & A^\theta(S_t,A_t) \geq 0 \\ 1-\varepsilon, & A^\theta(S_t,A_t) < 0 \end{cases}$$

Proximal Policy Optimization (PPO): Main Points

Policy and Value Function Approximations

$$\pi^{\eta}(a|s) \approx \pi_*(a|s), \quad V^{\theta}(s,a) \approx v_{\pi^{\eta}}(s,a)$$

Definition

$$v_{\pi^{\eta}}(s) = \mathbb{E}_{\pi^{\eta}}[G_t | S_t = s]$$

Approximation

$$V^{\theta}(s) \approx G_t$$

Proximal Policy Optimization (PPO)

Define the policy $\pi^\eta(a|s)$ and value $V^\theta(s)$ neural network structure.
Initialize parameters η_0 and θ_0
For k in $\overline{1, K}$

- According to π^η, get trajectory $\tau = (S_0, A_0, \dots, S_T)$ and rewards $(R_0, \dots, R_{T-1})$. Define $(G_0, \dots, G_{T-1})$.
- By τ, define the Loss functions

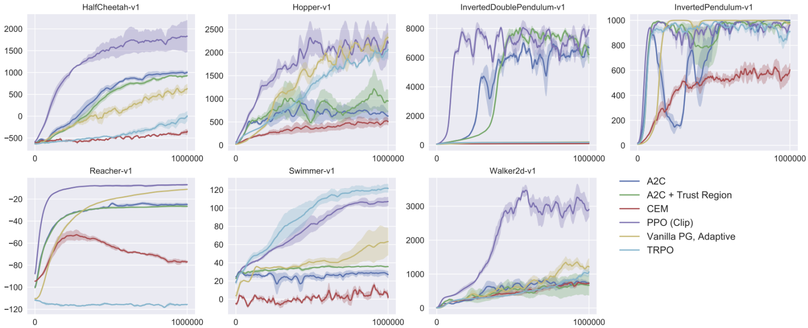
$$Loss_1(\eta) = -\frac{1}{T}\sum_{t=0}^{T-1} \min\left\{\frac{\pi^\eta(A_t|S_t)}{\pi^{\eta_k}(A_t|S_t)} A^{\theta_k}(S_t, A_t), g_\varepsilon(A^{\theta_k}(S_t, A_t))\right\}$$

$$Loss_2(\theta) = \frac{1}{T}\sum_{t=0}^{T-1} \left(V^\theta(S_t) - G_t\right)^2$$

where $A^\theta(S_t, A_t) = R_t + \gamma V^\theta(S_{t+1}) - V^\theta(S_t)$,
and update parameters

$$\eta_{k+1} \leftarrow \eta_k - \alpha_1 \nabla_\eta Loss_1(\eta_k), \quad \theta_{k+1} \leftarrow \theta_k - \alpha_2 \nabla_\theta Loss_1(\theta)$$

PPO Results


HalfCheetah-v1
Hopper-v1
InvertedDoublePendulum-v1
InvertedPendulum-v1
Reacher-v1
Swimmer-v1
Walker2d-v1
0
1000000
A2C
A2C + Trust Region
CEM
PPO (Clip)
Vanilla PG, Adaptive
TRPO

InstructGPT

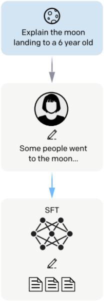
Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.



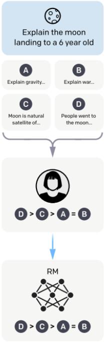
Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.