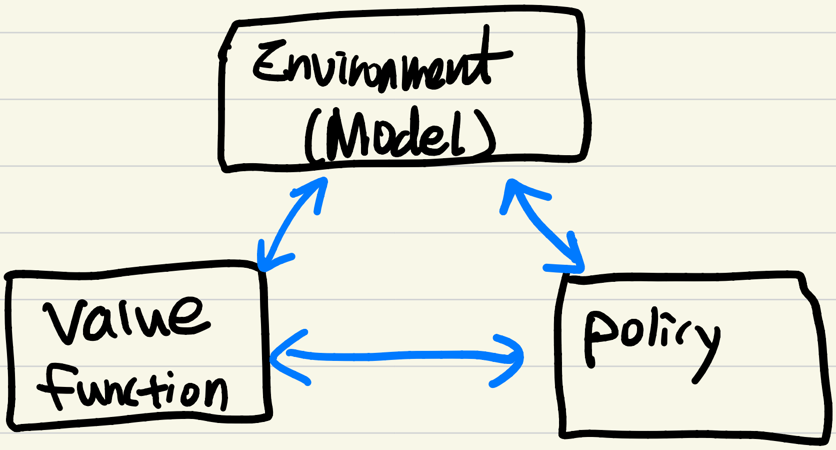
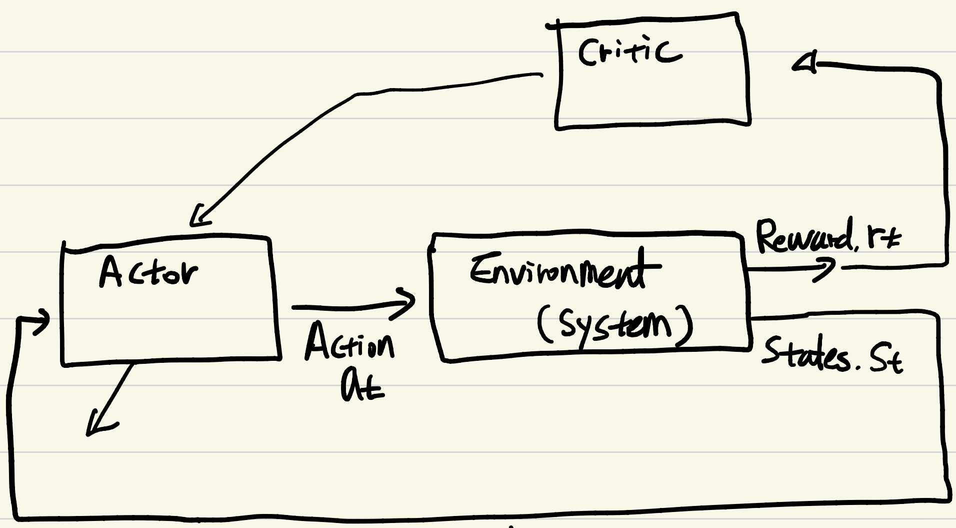
Lec 08 - Actor-Critic

Announcement: Homework 1 is released (Due: 7/30 PST)

Send your HW to { Saehong: sspark@berkeley.edu
Xinyi: zxyyx48@163.com

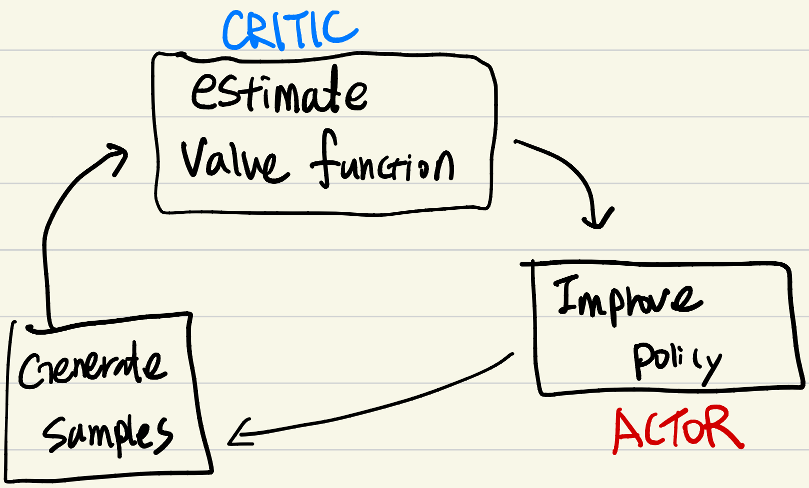
- Markov Decision Process
- Dynamic Programming
  - Learn value function
  - Implicit policy
- Policy Optimization
  - No value function
  - Learn policy
    ex) Policy gradient
- Actor Critic
  - Learn Value function
  - Learn policy





⟨ Schematic of Actor-Critic ⟩

In Actor-Critic, the actor improves the policy based on the value function that is estimated by Critic.



1. Critic

Value function estimation is equivalent to policy evaluation.
There are two approaches for policy evaluation.

- Monte-Carlo policy evaluation
- Bootstrap policy evaluation

1.1 Monte-Carlo policy evaluation

Suppose NN-type function approximator (FA) is used for value function,

$$V^{\pi}(s_t) = \sum_{t'=t}^{T-1} r(s_t, a_t)$$

Save the trajectory and add together the remaining rewards
The FA is going to average together along with other trajectories.

$$V^{\pi}(s_t) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t'=t}^{T-1} r(s_{i,t'}, a_{i,t'})$$

We fit the value function by Regression

- Traing data : $\underbrace{\{ s_{i,t} , \sum_{t'=t}^{T-1} r(s_{i,t'}, a_{i,t'}) \}}_{y_i}$
- Loss function : $\min_{\phi} \mathcal{L} = \sum_i \| y_i - V_\phi^\pi(s_t) \|^2$
- Run Stochastic Gradient Descent (SGD)

Ideal target should be $y_{i,t} = \sum_{t'=t}^{T-1} \mathbb{E}_{\pi_\theta} [r(s_{t'}, a_{t'}) | s_{i,t}]$

Monte-Carlo target : $y_{i,t} = \sum_{t'=t}^{T-1} r(s_{i,t'}, a_{i,t'})$
(Single Sample estimator)

1.2. Bootstrapped policy evaluation.

Start from ideal target,

$$
\begin{aligned}
y_{i,t} &= \sum_{t'=t}^{T-1} \mathbb{E}_{\pi_\theta} [r(s_{t'}, a_{t'}) | s_{i,t}] \\
&\approx r(s_{i,t}, a_{i,t}) + \sum_{t'=t+1}^{T-1} \mathbb{E}_{\pi_\theta} [r(s_{t'}, a_{t'}) | s_{i,t+1}] \\
&= r(s_{i,t}, a_{i,t}) + V^\pi(s_{i,t+1})
\end{aligned}
$$

It's approximately equal to current time step reward + next time step expectation.

We take NN, $\hat{V}_\phi^\pi$ as an approximation of the true value function, and plug in

<u>Bootstrapped</u>: $y_{i,t} \approx \underline{r(s_{i,t}, a_{i,t}) + \hat{V}_\phi^\pi(s_{i,t+1})}$

we can directly use previous fitted value function and plug it in for the second timestep. In training data, instead of summing all the rewards from that step until the end, we're gonna take just reward at current time step and add it the value function at the next time step.

Training data: $\{ (s_{i,t}, \underbrace{r(s_{i,t}, a_{i,t}) + \hat{V}_\phi^\pi(s_{i,t+1})}_{y_i}) \}$

Loss function: $\min_\phi \mathcal{L} = \sum_i \| y_i - V_\phi^\pi(s_{i,t}) \|^2$

Run SGD.