Lecture 6: Actor-Critic Method, Proximal Policy Optimization

GEARS Program
Summer 2024

Junxi Zhu

From previous lecture…

Tackle Non-Differentiable Conditions in RL

- https://stats.stackexchange.com/questions/340651/how-does-backpropagation-work-in-the-case-of-reinforcement-learning-for-games

The difficulty with RL is not that the network produces a sequence of outputs rather than just one output. The difficulty is that often, the environment is non-differentiable with respect to the action taken. For example, consider a simple game where you must go left or right, getting a reward of 1 for going left, and a reward of -1 for going right.

The network must output a discrete action, left or right, which is most likely sampled from a sigmoid/softmax layer, which immediately breaks the differentiability requirement. Even in continuous games, an RL environment is unlikely to be differentiable (suppose you output an action in [0,1], and you get 1 reward if your action $a > 0.5$ and -1 reward for $a \leq 0.5$ -- this is nondifferentiable).

Even if the environment happens to be differentiable, unless you know the dynamics (the physical laws) of the environment, you'd be unable to backpropagate through them.

Therefore in RL, we usually cannot train with vanilla backpropagation. Two popular classes of algorithms which work around this problem are:

1. Policy gradients: Use gradient estimation techniques which estimate the gradient of the reward wrt to weights, even if it is a nondifferentiable function.
2. Q-learning: Instead of training by maximizing reward directly, learn the expected reward of each state-action pair. This can be done with vanilla backpropagation. Then simply take the action with the highest expected reward to play the game.

See here for an interesting case of where vanilla BP can be used in RL.

Update Policy Network using Policy Gradient

1. Observe the state s_t
2. Randomly sample action a_t according to $\pi(\cdot\,|s;\theta)$
3. Compute $q_t \approx Q_\pi(s_t, a_t)$ (some estimate)
4. Differentiate policy network $d_{\theta,t} = \left.\frac{\partial \log \pi(a_t|s_t;\theta)}{\partial \theta}\right|_{\theta=\theta_t}$
5. Approximate policy gradient: $g(a_t, \theta_t) = q_t \cdot d_{\theta,t}$
6. Update policy network: $\theta_{t+1} = \theta_t + \beta \cdot g(a_t, \theta_t)$

Derivative-free Optimization

- https://en.wikipedia.org/wiki/Derivative-free_optimization

Algorithms [edit]

Notable derivative-free optimization algorithms include:

- Bayesian optimization
- Coordinate descent and adaptive coordinate descent
- Differential evolution, including multi-objective variants
- DONE
- Evolution strategies, Natural evolution strategies (CMA-ES, xNES, SNES)
- Genetic algorithms
- MCS algorithm
- Nelder-Mead method
- Particle swarm optimization
- Pattern search
- Powell's methods based on interpolation, e.g., COBYLA (PRIMA)
- Random search (including Luus–Jaakola)
- Simulated annealing
- Stochastic optimization
- Subgradient method
- various model-based algorithms like BOBYQA and ORBIT

This lecture…

Value-Based vs Policy-Based Learning

- **Question**: What is the difference between Value-Based Learning and Policy-Based Learning?

Value-Based Learning

- Return (discounted)

$$U_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \cdots$$

- Action-value function for policy π

$$Q_\pi(s_t, a_t) = \mathbb{E}[U_t | S_t = s_t, A_t = a_t]$$

- Optimal action-value function

$$Q^*(s_t, a_t) = \max_\pi Q_\pi(s_t, a_t)$$

- Goal of **value-based learning** is to learn or approximate $Q^*(s_t, a_t)$

Policy-Based Learning

- Return (discounted)

$$U_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \cdots$$

- Action-value function for policy π

$$Q_\pi(s_t, a_t) = \mathbb{E}[U_t | S_t = s_t, A_t = a_t]$$

- State-value function

$$V_\pi(s_t) = \mathbb{E}_A[Q_\pi(s_t, A)] = \sum_a \pi(a|s_t) \cdot Q_\pi(s_t, a)$$

- Approximate policy function $\pi(a|s_t)$ by policy network $\pi(a|s_t; \theta)$
- Approximate value function $V_\pi(s_t)$ by

$$V(s; \theta) = \sum_a \pi(a|s; \theta) \cdot Q_\pi(s, a)$$

- Goal of **policy-based learning** is to learn θ that maximizes $J(\theta) = \mathbb{E}_S[V(S; \theta)]$

Value-Based vs Policy-Based Learning

- Value-based learning: such as Q-learning, Deep Q Network (DQN)
- Learns a value function that estimates the expected return of taking a particular action in a given state and following the optimal policy thereafter
- It tries to derive an optimal policy indirectly by maximizing the value function
- Advantages:
  - Optimal action selection: Directly provides a way to select the best action based on the value function
  - Well-studied: Many theoretical guarantees and well-established algorithms exist for value-based methods
- Limitations:
  - Discrete actions: Typically more suited to environments with discrete action spaces
  - Exploration challenges: Often requires careful balance of exploration and exploitation
  - Function approximation issues: May suffer from instability and divergence issues when using function approximation, such as neural networks

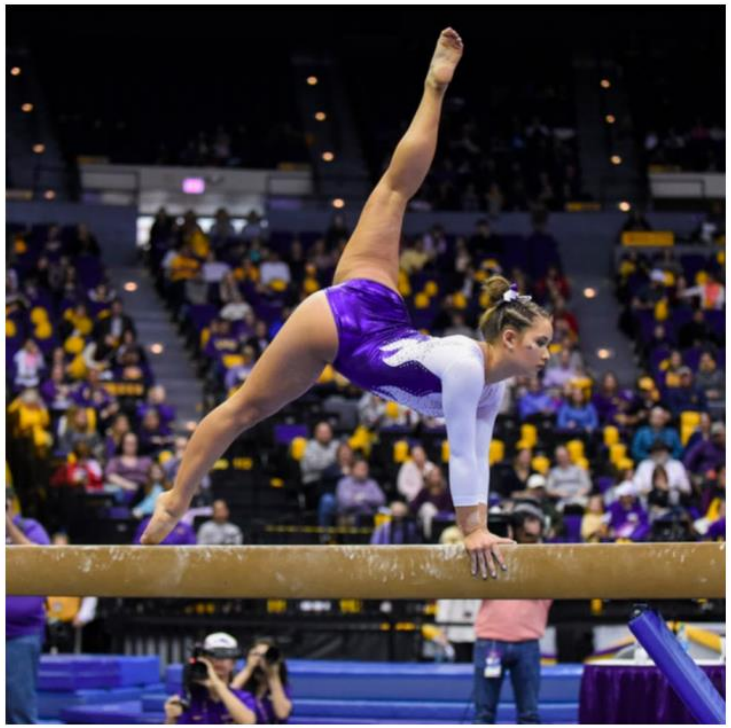
Value-Based vs Policy-Based Learning

- Policy-based learning: such as REINFORCE and Proximal Policy Optimization (PPO)
- Directly parameterize the policy and optimize it using gradient-based methods
- The policy determines the action to take in each state
- Advantages:
  - Continuous actions: Naturally handle environments with continuous action spaces
  - Stochastic policies: Can represent stochastic policies, which are beneficial for exploration and robustness
  - Stable update: Often more stable and easier to tune than value-based methods when using function approximation
  - Policy improvement: Directly optimize the policy, potentially leading to better performance in certain environments
- Limitations:
  - Sample efficiency: Typically less sample-efficient compared to value-based methods
  - Variance: Gradient estimates can have high variance, requiring techniques like baseline subtraction to reduce it

Actor-Critic Method

- Combination of value-based method and policy-based method

Actor (Policy network)



Critic (Value network)

Actor-Critic Method

- Advantage: Able to update the policy at each step, whereas policy gradient (REINFORCE) updates the policy at the end of each episode (next slide)
- Limitation:
  - Depend heavily on the judgement of the critic
  - Has convergence issue for the critic alone, even harder when including the update of the actor -> Google proposed Actor-Critic + DQN = Deep Deterministic Policy Gradient (DDPG, will not cover here)

(Recall) Update Policy Network using Policy Gradient

1. Observe the state s_t
2. Randomly sample action a_t according to $\pi(\cdot\,|s;\theta)$
3. Compute $q_t \approx Q_\pi(s_t, a_t)$ (some estimate)

How?

Option 1: REINFORCE

- Play the game to the end and generate the trajectory

$$s_1, a_1, r_1, s_2, a_2, r_2, \cdots, s_T, a_T, r_T$$

- Compute the discounted return $u_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$, for all t
- Since $Q_\pi(s_t, a_t) = \mathbb{E}[U_t]$, we can use u_t to approximate $Q_\pi(s_t, a_t)$
- Thus, use $q_t = u_t$

State-Value Function Approximation

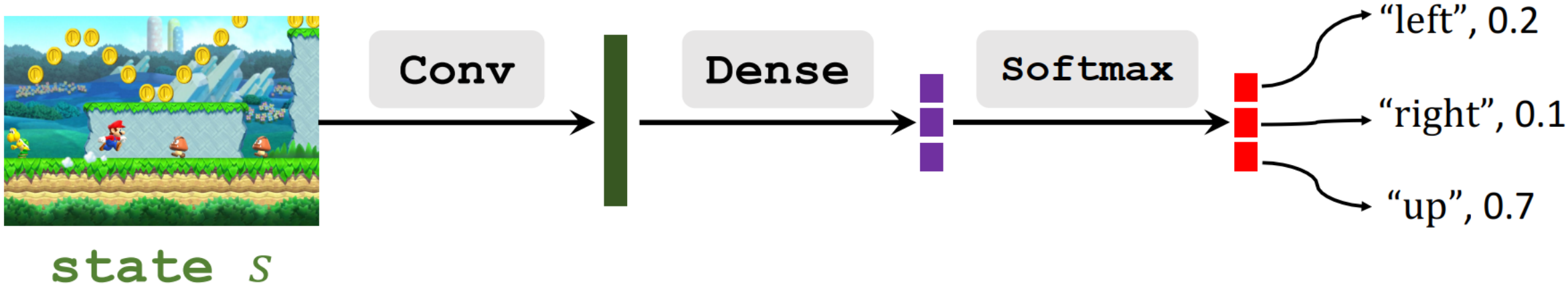
- State-value function

$$V_\pi(s) = \sum_a \pi(a|s) \cdot Q_\pi(s,a) \approx \sum_a \pi(a|s;\theta) \cdot q(s,a;w)$$

- Policy network (actor):
  - Use neural network $\pi(a|s;\theta)$ to approximate $\pi(a|s)$
  - θ: trainable parameters of the neural network
- Value network (critic):
  - Use neural network $q(s,a;w)$ to approximate $Q_\pi(s,a)$
  - w: trainable parameters of the neural network

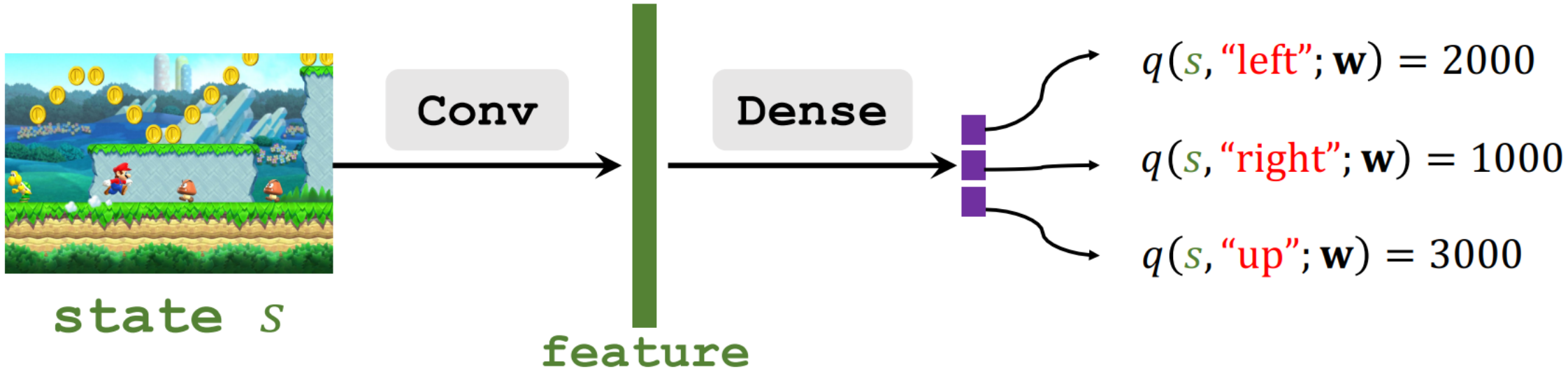
Policy Network (Actor): $\pi(a|s;\theta)$

- Input: state s, e.g., a screenshot of Super Mario
- Output: probability distribution over the actions
- Let $\mathcal{A}$ be the set of all actions, e.g., $\mathcal{A}$={left, right, up}
- $\sum_{a\in\mathcal{A}} \pi(a|s;\theta) = 1$



Value Network (Critic): $q(s, a; w)$

- Input: state s
- Output: action-values of all the actions



How to Train the Networks

- State-value function approximated using neural networks

$$V(s;\theta,w)=\sum_a \pi(a|s;\theta)\cdot q(s,a;w)$$

- Training: Update the parameters θ and w
  - Update policy network $\pi(a|s;\theta)$ to increase the state-value $V(s;\theta,w)$
    - Actor gradually performs better
    - Supervision is purely from the value network (critic)
  - Update value network $q(s,a;w)$ to better estimate the return
    - Critic's judgement becomes more accurate
    - Supervision is purely from the rewards

Update Value Network q using TD

- Compute $q(s_t, a_t; w_t)$ and $q(s_{t+1}, a_{t+1}; w_t)$
- TD target: $y_t = r_t + \gamma \cdot q(s_{t+1}, a_{t+1}; w_t)$
- Loss: $L(w) = \frac{1}{2}[q(s_t, a_t; w) - y_t]^2$
- Gradient descent: $w_{t+1} = w_t - \alpha \cdot \frac{\partial L(w)}{\partial w}\Big|_{w=w_t}$

Update Policy Network π using Policy Gradient

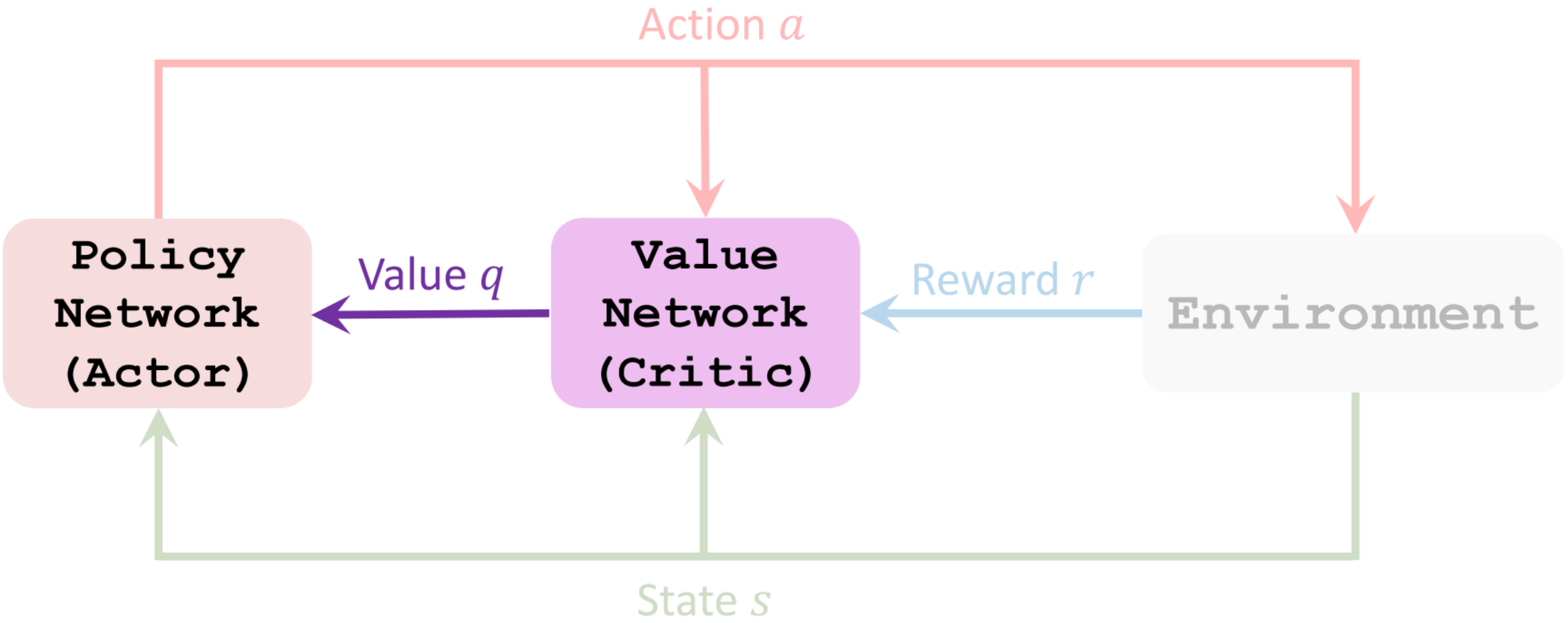
- State-value function approximated using neural networks

$$V(s;\theta,w)=\sum_a \pi(a|s;\theta)\cdot q(s,a;w)$$

- Policy gradient: Derivative of $V(s_t;\theta,w)$ w.r.t θ
  - Let $g(a,\theta)=\frac{\partial \log \pi(a|s_t;\theta)}{\partial \theta}\cdot q(s_t,a;w)$
  - $\frac{\partial V(s;\theta,w_t)}{\partial \theta}=\mathbb{E}_A[g(A,\theta)]$
- Algorithm: Update policy network using stochastic policy gradient
  - Random sampling: $a\sim\pi(\cdot\,|s_t;\theta_t)$
  - Stochastic gradient ascent: $\theta_{t+1}=\theta_t+\beta\cdot g(a,\theta_t)$

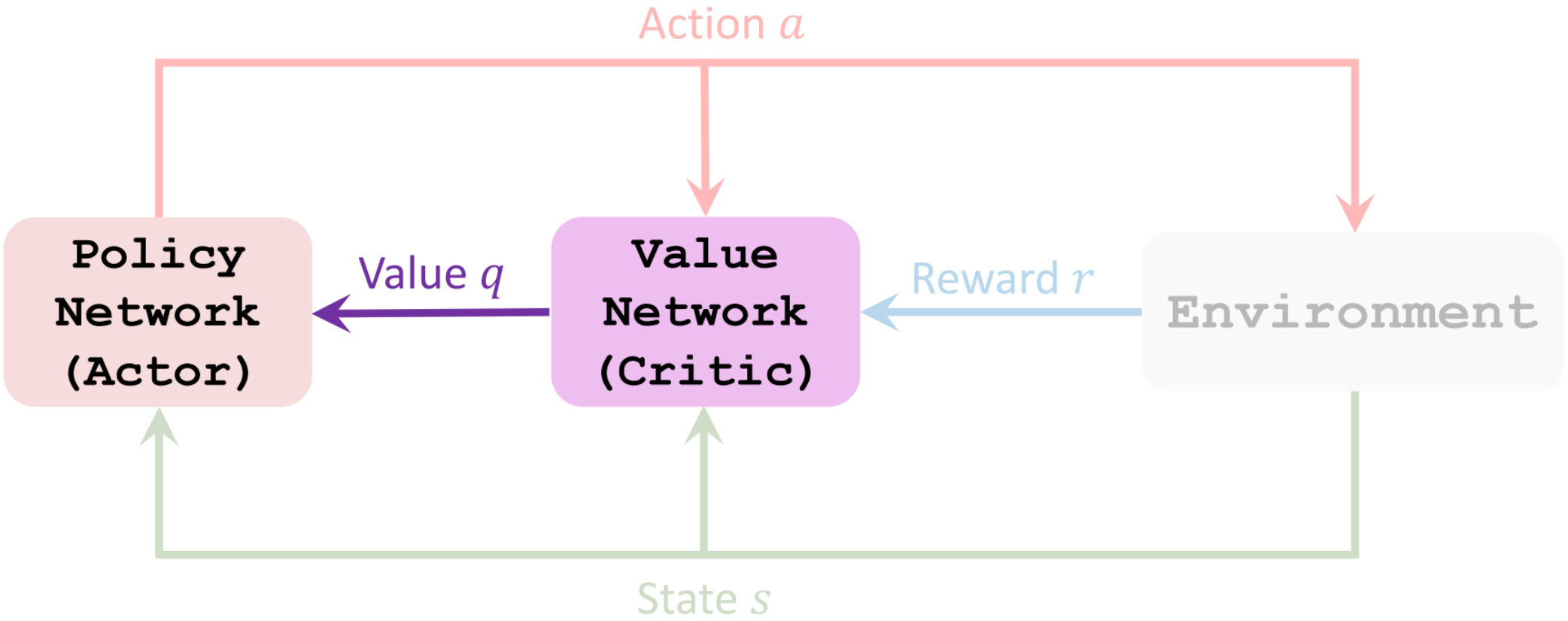
Actor-Critic Method

- Workflow



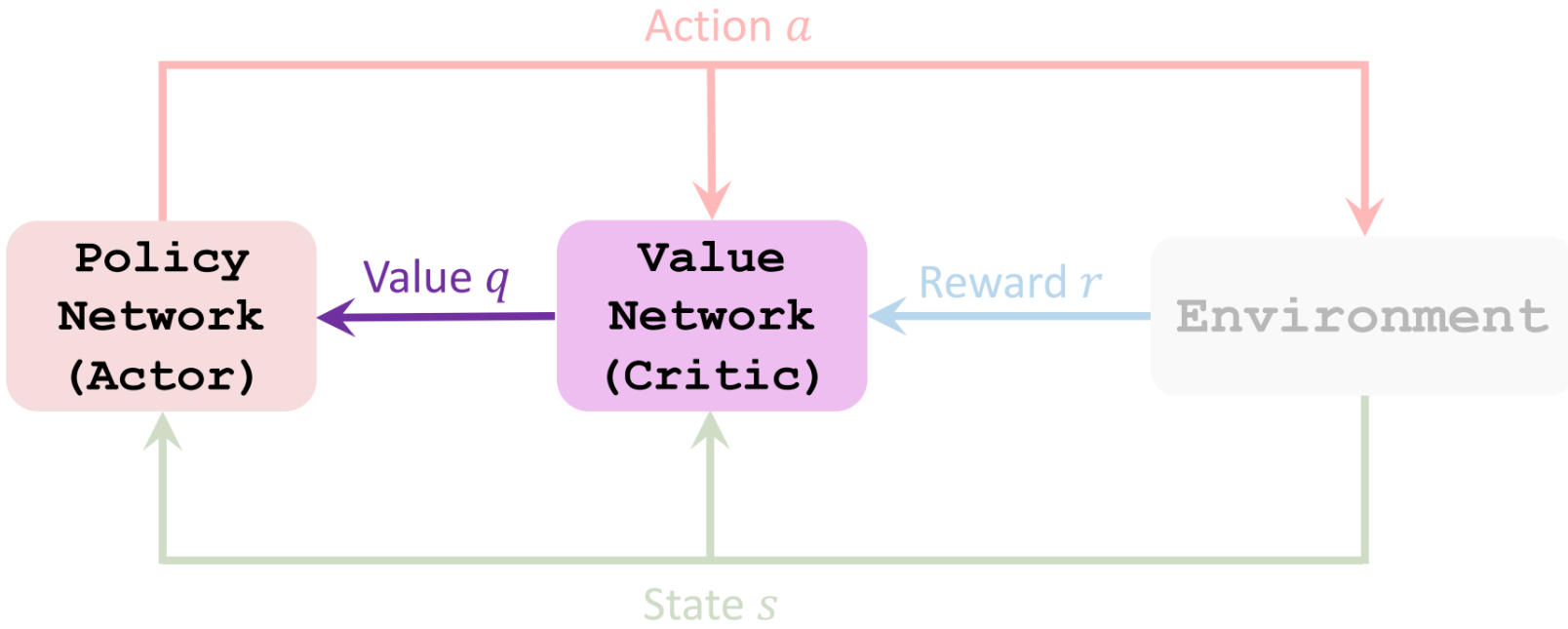
Actor-Critic Method Workflow

- **Question**: Why not use reward r as the input to the policy network?

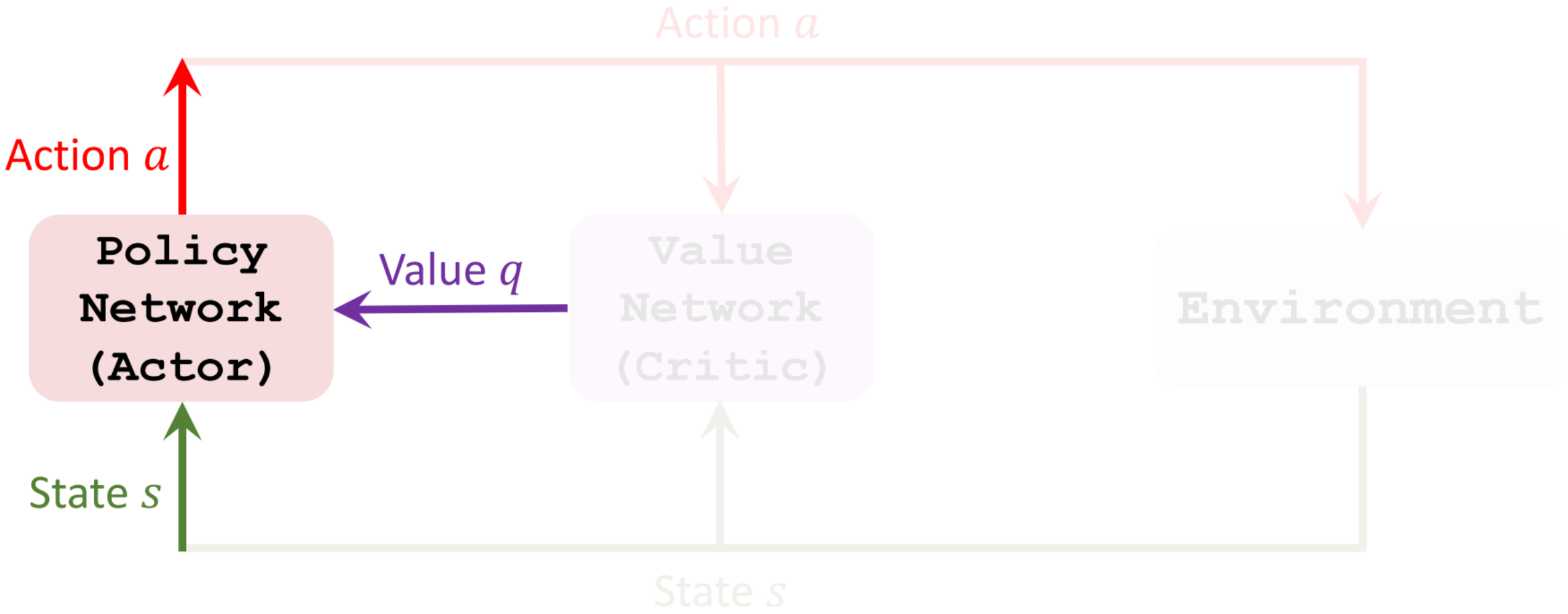


Actor-Critic Method Workflow

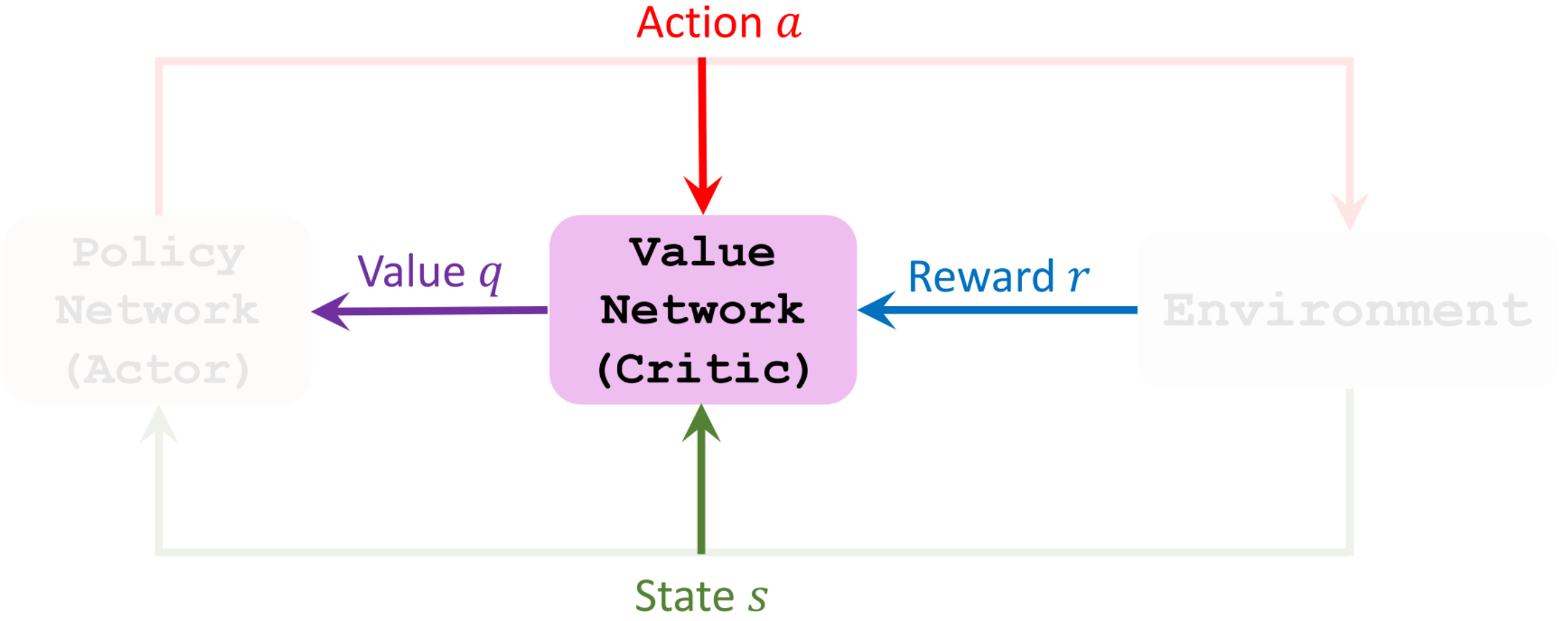
- **Question**: Why not use reward r as the input to the policy network?
- Recall that policy-based learning try to maximize $J(\theta) = \mathbb{E}_S[V(S;\theta)]$ where $V_\pi(s_t) = \mathbb{E}_A[Q_\pi(s_t, A)]$ and $Q_\pi(s_t, a_t) = \mathbb{E}[U_t | S_t = s_t, A_t = a_t]$
- Therefore policy-based learning target expectation of total return U, instead of reward r



Actor-Critic Method: Update Actor



Actor-Critic Method: Update Critic



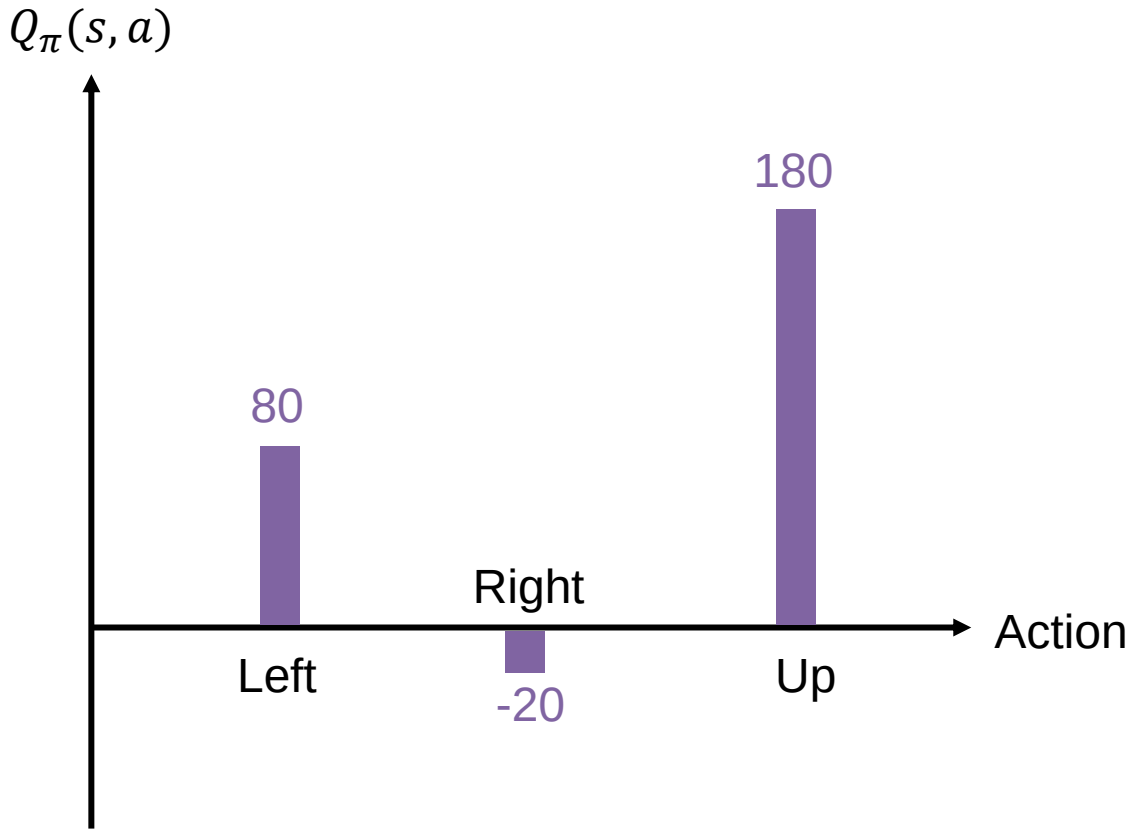
Roles of Actor and Critic

- During training
  - Agent is controlled by policy network (actor): $a_t \sim \pi(\cdot\,|s_t;\theta)$
  - Value network (critic) provides the actor with supervision
- After training
  - Agent is controlled by policy network (actor): $a_t \sim \pi(\cdot\,|s_t;\theta)$
  - Value network (critic) will not be used

Policy Gradient with Baseline

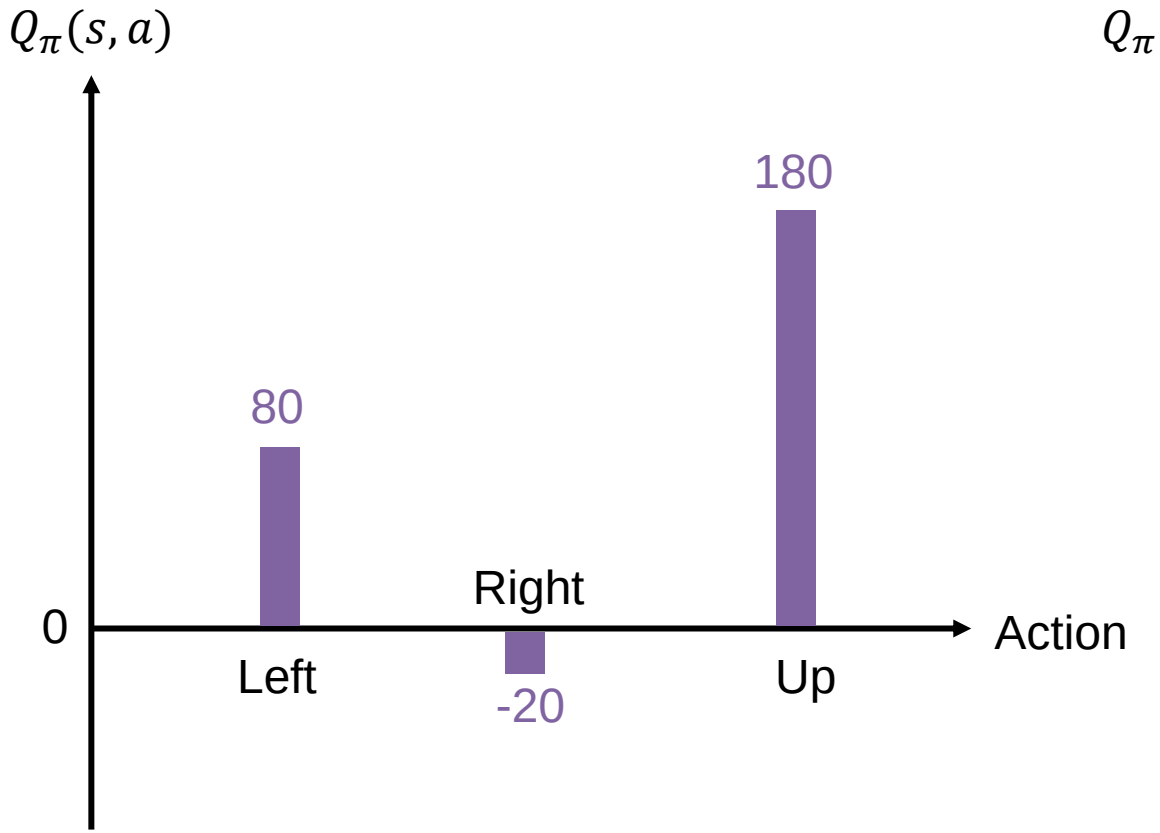
Challenge with Policy Gradient

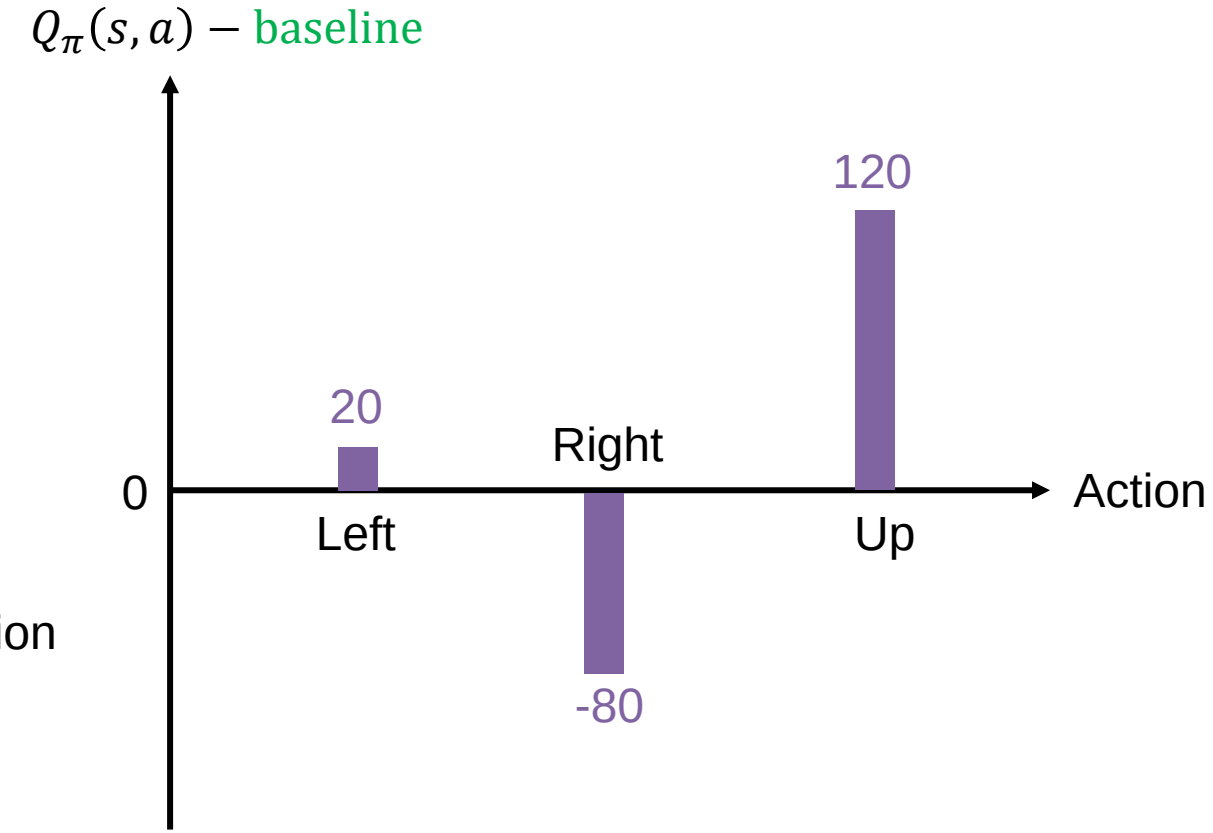
- Policy gradient has large variance and slow convergence
- Recall that policy gradient is: $\frac{\partial V(s;\theta)}{\partial \theta} = \mathbb{E}_{A\sim\pi(\cdot|s;\theta)}\left[\frac{\partial \log \pi(A|s;\theta)}{\partial \theta} \cdot Q_\pi(s,A)\right]$



Policy Gradient with Baseline

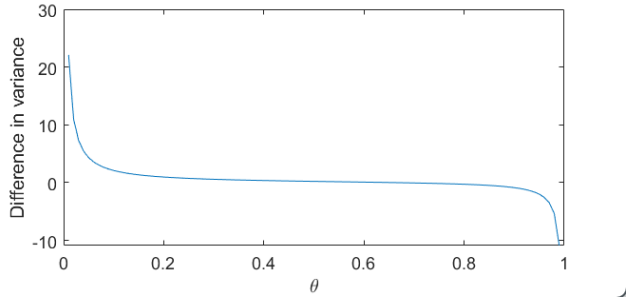
- Policy gradient has large variance and slow convergence
- Recall that policy gradient is: $\frac{\partial V(s;\theta)}{\partial \theta} = \mathbb{E}_{A\sim\pi(\cdot|s;\theta)}\left[\frac{\partial \log \pi(A|s;\theta)}{\partial \theta} \cdot Q_\pi(s,A)\right]$





Policy Gradient with Baseline

- Policy gradient with baseline is: $\frac{\partial V(s;\theta)}{\partial \theta} = \mathbb{E}_{A\sim\pi(\cdot|s;\theta)}\left[\frac{\partial \log \pi(A|s;\theta)}{\partial \theta} \cdot (Q_\pi(s,A) - b)\right]$
- Example: use the average reward as baseline
- Assume we have a policy $\pi(\cdot\,|s;\theta)$ that chooses between two actions a_1 and a_2 with probabilities θ and $1-\theta$, respectively
- Then $\frac{\partial \log \pi(a_1|s;\theta)}{\partial \theta} = \frac{1}{\theta}$, $\frac{\partial \log \pi(a_2|s;\theta)}{\partial \theta} = -\frac{1}{1-\theta}$
- The rewards for actions are $R(a_1) = 1$ and $R(a_2) = -1$; Baseline is $\frac{1+1+(-1)}{3} = \frac{1}{3}$
- Assume we sampled three trajectories: a_1, a_1, a_2
- Without baseline: $\frac{\partial V(s;\theta)}{\partial \theta} = \frac{1}{3}\left[\frac{1}{\theta}\cdot 1 + \frac{1}{\theta}\cdot 1 + \frac{-1}{1-\theta}\cdot(-1)\right] = \frac{1}{3}\cdot\left[\frac{2}{\theta} + \frac{1}{1-\theta}\right]$
- With baseline: $\frac{\partial V(s;\theta)}{\partial \theta} = \frac{1}{3}\left[\frac{1}{\theta}\cdot\frac{2}{3} + \frac{1}{\theta}\cdot\frac{2}{3} + \frac{-1}{1-\theta}\cdot\left(-1-\frac{1}{3}\right)\right] = \frac{1}{3}\cdot\left[\frac{4}{3\theta} + \frac{4}{3(1-\theta)}\right]$



- The variance of policy gradient with baseline is reduce compared to policy gradient without baseline over multiple trajectories
- More detailed discussion can be seen here: https://www.youtube.com/watch?v=e20EY4tFC_Q&t=1067s

Policy Gradient with Baseline

- Policy gradient with baseline is: $\frac{\partial V(s;\theta)}{\partial \theta} = \mathbb{E}_{A\sim\pi(\cdot|s;\theta)}\left[\frac{\partial \log \pi(A|s;\theta)}{\partial \theta} \cdot (Q_\pi(s,A) - b)\right]$
- **Question**: Any requirement for the baseline b?

Policy Gradient with Baseline

- Policy gradient with baseline is: $\frac{\partial V(s;\theta)}{\partial \theta} = \mathbb{E}_{A\sim\pi(\cdot|s;\theta)}\left[\frac{\partial \log \pi(A|s;\theta)}{\partial \theta} \cdot (Q_\pi(s,A) - b)\right]$
- **Question**: Any requirement for the baseline b?
- Baseline b can be any function as long as it does not depend on action a_t

Policy Gradient with Baseline

- Policy gradient with baseline is: $\frac{\partial V(s;\theta)}{\partial \theta} = \mathbb{E}_{A\sim\pi(\cdot|s;\theta)}\left[\frac{\partial \log \pi(A|s;\theta)}{\partial \theta} \cdot (Q_\pi(s,A) - b)\right]$
- We want to prove that $\mathbb{E}_{A\sim\pi(\cdot|s;\theta)}\left[b \cdot \frac{\partial \log \pi(A|s;\theta)}{\partial \theta}\right] = 0$
- Proof:

$$\mathbb{E}_{A\sim\pi(\cdot|s;\theta)}\left[b \cdot \frac{\partial \log \pi(A|s;\theta)}{\partial \theta}\right] = b \cdot \mathbb{E}_{A\sim\pi(\cdot|s;\theta)}\left[\frac{\partial \log \pi(A|s;\theta)}{\partial \theta}\right]$$

Because b does not depend on A

$$= b \cdot \sum_{a\in\mathcal{A}} \pi(a|s;\theta) \cdot \boxed{\frac{1}{\pi(a|s;\theta)} \cdot \frac{\partial \pi(A|s;\theta)}{\partial \theta}}$$

Chain rule

$$= b \cdot \sum_{a\in\mathcal{A}} \frac{\partial \pi(A|s;\theta)}{\partial \theta}$$

Summation is w.r.t a, while partial derivative is w.r.t θ

$$= b \cdot \frac{\partial}{\partial \theta} \sum_{a\in\mathcal{A}} \pi(a|s;\theta)$$

$\sum_{a\in\mathcal{A}} \pi(a|s;\theta) = 1$ by definition

$$= b \cdot \frac{\partial 1}{\partial \theta}$$

$$= 0$$

Policy Gradient with Baseline

- Policy gradient with baseline is: $\frac{\partial V(s;\theta)}{\partial \theta} = \mathbb{E}_{A\sim\pi(\cdot|s;\theta)}\left[\frac{\partial \log \pi(A|s;\theta)}{\partial \theta} \cdot (Q_\pi(s,A) - b)\right]$
- **Question**: What to choose for the baseline b?

Policy Gradient with Baseline

- Policy gradient with baseline is: $\frac{\partial V(s;\theta)}{\partial \theta} = \mathbb{E}_{A\sim\pi(\cdot|s;\theta)}\left[\frac{\partial \log \pi(A|s;\theta)}{\partial \theta} \cdot (Q_\pi(s,A) - b)\right]$
- **Question**: What to choose for the baseline b?
- Let the baseline b equal to the state-action value $V_\pi(s_t)$!

(Recall) Update Policy Network using Policy Gradient

1. Observe the state s_t
2. Randomly sample action a_t according to $\pi(\cdot\,|s;\theta)$
3. Compute $q_t \approx Q_\pi(s_t, a_t)$ (some estimate)

How?

Option 1: REINFORCE

- Play the game to the end and generate the trajectory

$$s_1, a_1, r_1, s_2, a_2, r_2, \cdots, s_T, a_T, r_T$$

- Compute the discounted return $u_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$, for all t
- Since $Q_\pi(s_t, a_t) = \mathbb{E}[U_t]$, we can use u_t to approximate $Q_\pi(s_t, a_t)$
- Thus, use $q_t = u_t$

REINFORCE with Baseline

- Policy gradient is:

$$g(s_t, a_t; \theta) = \frac{\partial \log \pi(a_t|s_t;\theta)}{\partial \theta} \cdot (U_t - V_\pi(s_t))$$

Actual return through Monte-Carlo simulation

Estimated state value from value network

- Note that this is **not** an actor-critic method because $v_\pi(s_t)$ is only used as a baseline to reduce variance and accelerate convergence. What actually helps improve the policy network (actor) is actual return u_t, instead of $v_\pi(s_t)$!

Actor-Critic with Baseline

- Policy gradient is:

$$g(s_t, a_t; \theta) = \frac{\partial \log \pi(a_t|s_t;\theta)}{\partial \theta} \cdot (\overbrace{Q(s_t, a_t; w) - V_\pi(s_t; w)}^{\text{Advantage } A_t})$$

Estimated action value

Estimated state value from value network

- This method is called Advantage Actor-Critic (A2C)

$$g(s_t, a_t; \theta) = \frac{\partial \log \pi(a_t|s_t;\theta)}{\partial \theta} \cdot \big(Q(s_t, a_t; w) - V_\pi(s_t; w)\big)$$

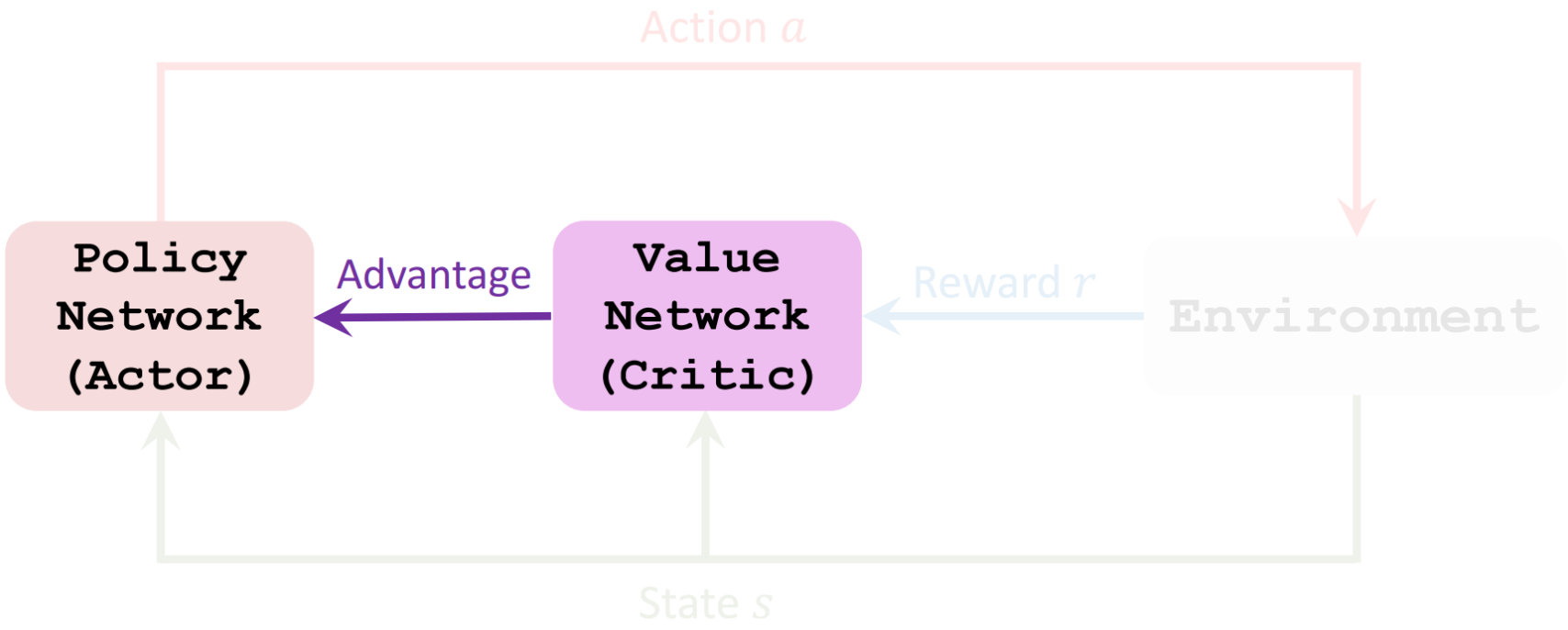
$$\approx \frac{\partial \log \pi(a_t|s_t;\theta)}{\partial \theta} \cdot (\underbrace{r_t + \gamma \cdot v_\pi(s_{t+1}; w)}_{\text{TD target}} - v_\pi(s_t; w))$$

Actor-Critic with Baseline

- Approximated policy gradient is:

$$g(s_t, a_t; \theta) = \frac{\partial \log \pi(a_t|s_t;\theta)}{\partial \theta} \cdot (r_t + \gamma \cdot v_\pi(s_{t+1}; w) - v_\pi(s_t; w))$$

- **Question**: How does the critic judge on the goodness of the action a_t if it only sees two states s_t and s_{t+1}?



Actor-Critic with Baseline

- Approximated policy gradient is:

$$g(s_t, a_t; \theta) = \frac{\partial \log \pi(a_t|s_t;\theta)}{\partial \theta} \cdot (r_t + \gamma \cdot v_\pi(s_{t+1}; w) - v_\pi(s_t; w))$$

Both are approximation to $\mathbb{E}[U_t]$

Both evaluate how good s_t is

Actor-Critic with Baseline

- Approximated policy gradient is:

$$g(s_t, a_t; \theta) = \frac{\partial \log \pi(a_t|s_t;\theta)}{\partial \theta} \cdot (r_t + \gamma \cdot v_\pi(s_{t+1}; w) - v_\pi(s_t; w))$$

Depends on a_t Independent of a_t

If a_t is good, their difference is positive

Generalized Advantage Estimation (GAE)

- GAE uses a weighted sum of exponentially discounted TD errors to estimate the advantage

$$\mathrm{Adv}_t^{GAE(\gamma,\lambda)} = \sum_{k=0}^{\infty} (\gamma\lambda)^k \, \delta_{t+k}$$

Here, δ_t represents the TD error, γ is the discount factor, and λ is a new parameter that determines the weighting of the TD errors in the sum, providing a way to balance the bias-variance tradeoff in the advantage estimation.

- In practical terms, GAE can be computed efficiently using a recursive formula:

$$\mathrm{Adv_t} = \delta_t + (\gamma\lambda)\mathrm{Adv_{t+1}}$$

This recursive calculation starts from the end of an episode or batch and moves backward, similar to the way TD(λ) updates are computed.

Generalized Advantage Estimation (GAE)

- Advantages of GAE
  - **Reduced Variance**: By averaging over multiple time steps, GAE smooths out the variations in advantage estimates that might result from singular transitions, reducing the variance of the estimates without introducing significant bias.
  - **Tunable Parameters**: With λ, GAE allows the practitioner to adjust how much weight is given to longer versus shorter horizon estimates. A lower λ results in advantage estimates that rely more heavily on immediate rewards and less on future values, similar to a lower-horizon TD estimate, while a higher λ provides estimates closer to a full Monte Carlo return.

On-Policy vs Off-Policy and Importance Sampling

- On-policy: the agent learned and the agent interacting with the environment is the same
  - Use $\pi(\theta)$ to collect data. When θ is updated, we have to sample training data again.
  - $\nabla_\theta J(\theta) = \mathbb{E}_S\left[\mathbb{E}_{A\sim\pi(\cdot|S;\,\theta)}[A_t \nabla_\theta \ln \pi(a|s;\theta)]\right]$
- Off-policy: the agent learned and the agent interacting with the environment is different
  - Sample the data from θ'
  - Use the data to train θ many times → more efficient
  - $\nabla_\theta J(\theta) = \mathbb{E}_S\left[\mathbb{E}_{A\sim\pi(\cdot|S;\,\theta')}\left[\frac{\pi(\cdot|S;\theta)}{\pi(\cdot|S;\theta')} A_t \nabla_\theta \ln \pi(a|s;\theta)\right]\right]$
- Importance sampling

$$\mathbb{E}_{x\sim p}[f(x)] = \mathbb{E}_{x\sim q}\left[\frac{p(x)}{q(x)} f(x)\right]$$

See https://www.youtube.com/watch?v=OAKAZhFmYoI for derivation

Challenges of Policy-Based Learning

- High variance (subtract baseline to reduce variance)
- Sample inefficiency (use off-policy learning to improve efficiency)
- Learning rate highly affects training
- **Question**: How to adaptively choose appropriate learning rate?

Trust Region Policy Optimization (TRPO)

- To ensure that the policy won't move too far, we add a constraint to our optimization problem in terms of making sure that the updated policy lies within a trust region
- The KL divergence between the new and the old policy must be lower than a predefined value δ
- Now the objective function becomes:

$$\max_{\theta} J(\theta) = \mathbb{E}_S\left[\mathbb{E}_{A\sim\pi(\cdot|S;\theta_{old})}\left[\frac{\pi(\cdot\,|S;\theta_{new})}{\pi(\cdot\,|S;\theta_{old}\,)}A_t\right]\right]$$

$$\text{subject to}\quad \mathbb{E}[KL(\pi(\cdot\,|s_t;\theta_{old}),\pi(\cdot\,|s_t;\theta_{new})] \le \delta$$

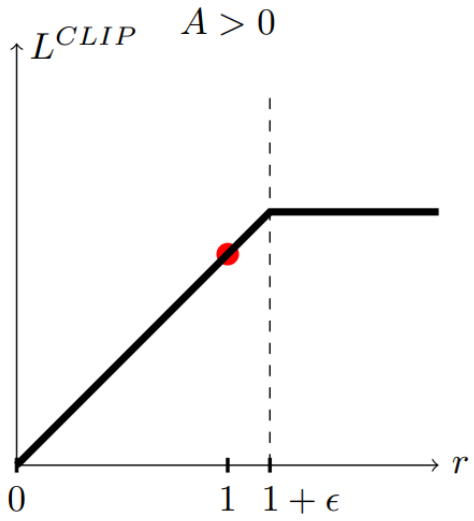
- How to solve a constrained optimization problem?
- Use conjugate gradient method. But it is inefficient!

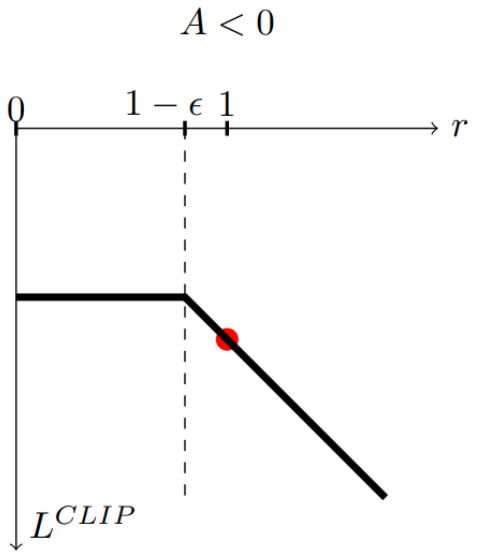
Proximal Policy Optimization (PPO)

- Instead of having a separate constraint to penalize the big change between the old and new policy, we can add a clipping function to the objective function
- Now the objective function becomes:

$$\max_{\theta} J(\theta) = \mathbb{E}_S\left[\mathbb{E}_{A\sim\pi(\cdot|S;\theta)}[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t]\right]$$

$$\text{where } r_t(\theta) = \frac{\pi(\cdot\,|S;\theta_{new})}{\pi(\cdot\,|S;\theta_{old})}$$





Resources

- https://www.youtube.com/watch?v=HTONz4ZLGxw
- https://www.youtube.com/watch?v=tvMu5oPYpsw&lc=Ugi9M5wmVDmwKHgCoAEC.8SSsSAm7XeJ8SVJfwC0WoR

Let’s check the code…