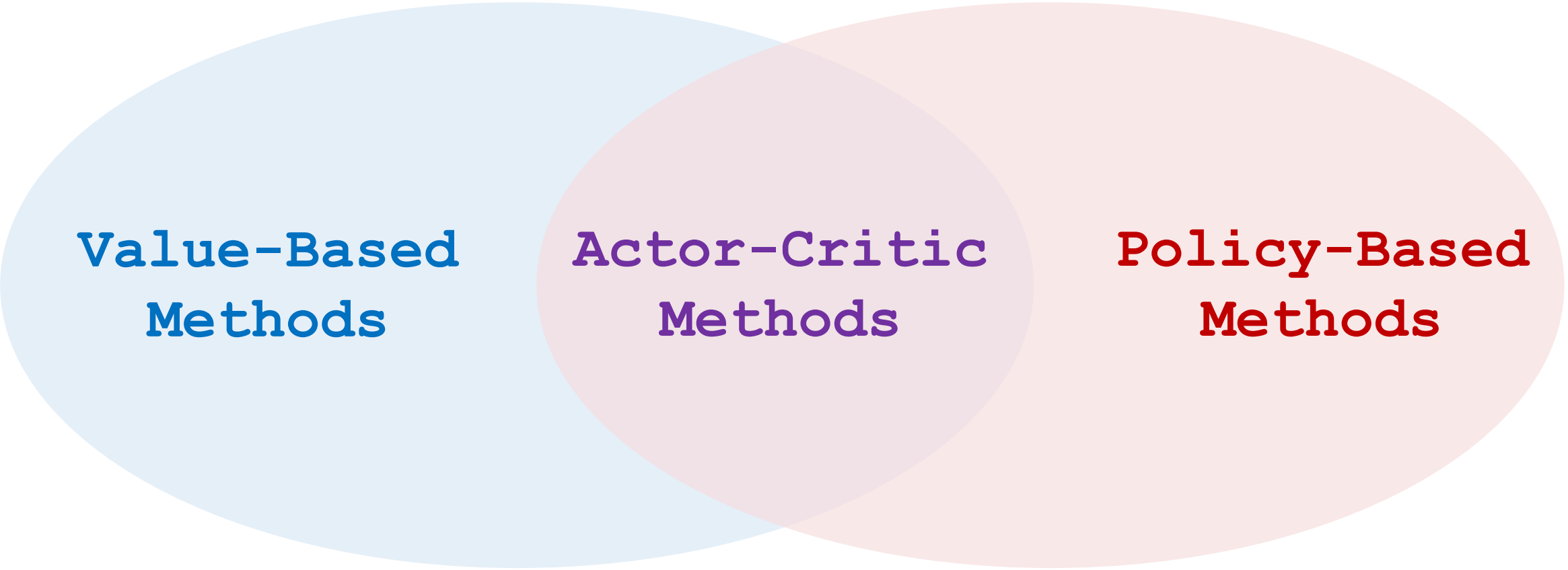


Actor-Critic Methods

Shusen Wang



A Venn diagram illustrating the relationship between three reinforcement learning method categories. It consists of three overlapping ellipses arranged horizontally. The leftmost ellipse is light blue and labeled 'Value-Based Methods' in blue text. The rightmost ellipse is light red and labeled 'Policy-Based Methods' in red text. The middle ellipse, which overlaps with both the blue and red ones, is light purple and labeled 'Actor-Critic Methods' in purple text. The intersection of the blue and red ellipses is the purple ellipse, indicating that Actor-Critic Methods are a combination of Value-Based and Policy-Based methods.

**Value-Based
Methods**

**Actor-Critic
Methods**

**Policy-Based
Methods**

Value Network and Policy Network

State-Value Function Approximation

Definition: State-value function.

- $V_{\pi}(s) = \sum_a \pi(a|s) \cdot Q_{\pi}(s, a).$

State-Value Function Approximation

Definition: State-value function.

- $V_{\pi}(s) = \sum_a \pi(a|s) \cdot Q_{\pi}(s, a).$

Policy network (actor):

- Use neural net $\pi(a|s; \theta)$ to approximate $\pi(a|s).$
- θ : trainable parameters of the neural net.

State-Value Function Approximation

Definition: State-value function.

- $V_{\pi}(s) = \sum_a \pi(a|s) \cdot Q_{\pi}(s, a).$

Policy network (actor):

- Use neural net $\pi(a|s; \theta)$ to approximate $\pi(a|s).$
- θ : trainable parameters of the neural net.

Value network (critic):

- Use neural net $q(s, a; \mathbf{w})$ to approximate $Q_{\pi}(s, a).$
- \mathbf{w} : trainable parameters of the neural net.

State-Value Function Approximation

Definition: State-value function.

- $V_{\pi}(s) = \sum_a \pi(a|s) \cdot Q_{\pi}(s, a) \approx \sum_a \pi(a|s; \theta) \cdot q(s, a; \mathbf{w}).$

Policy network (actor):

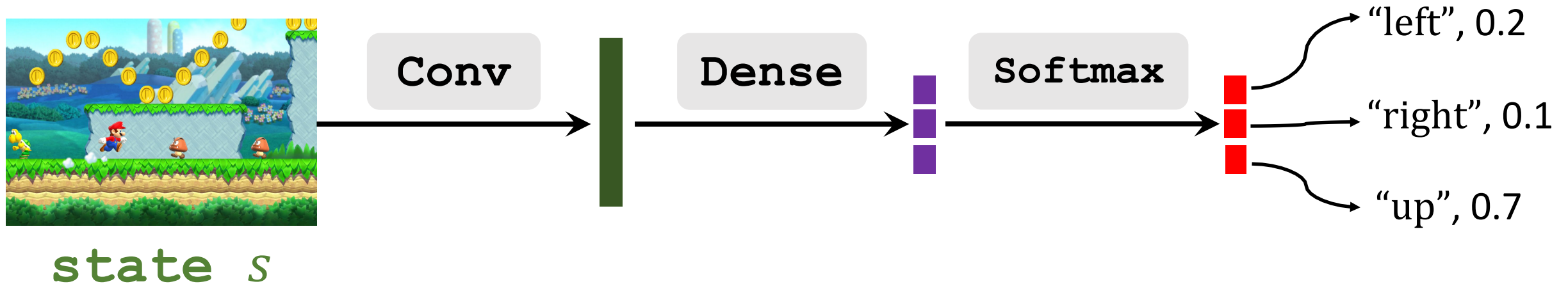
- Use neural net $\pi(a|s; \theta)$ to approximate $\pi(a|s)$.
- θ : trainable parameters of the neural net.

Value network (critic):

- Use neural net $q(s, a; \mathbf{w})$ to approximate $Q_{\pi}(s, a)$.
- \mathbf{w} : trainable parameters of the neural net.

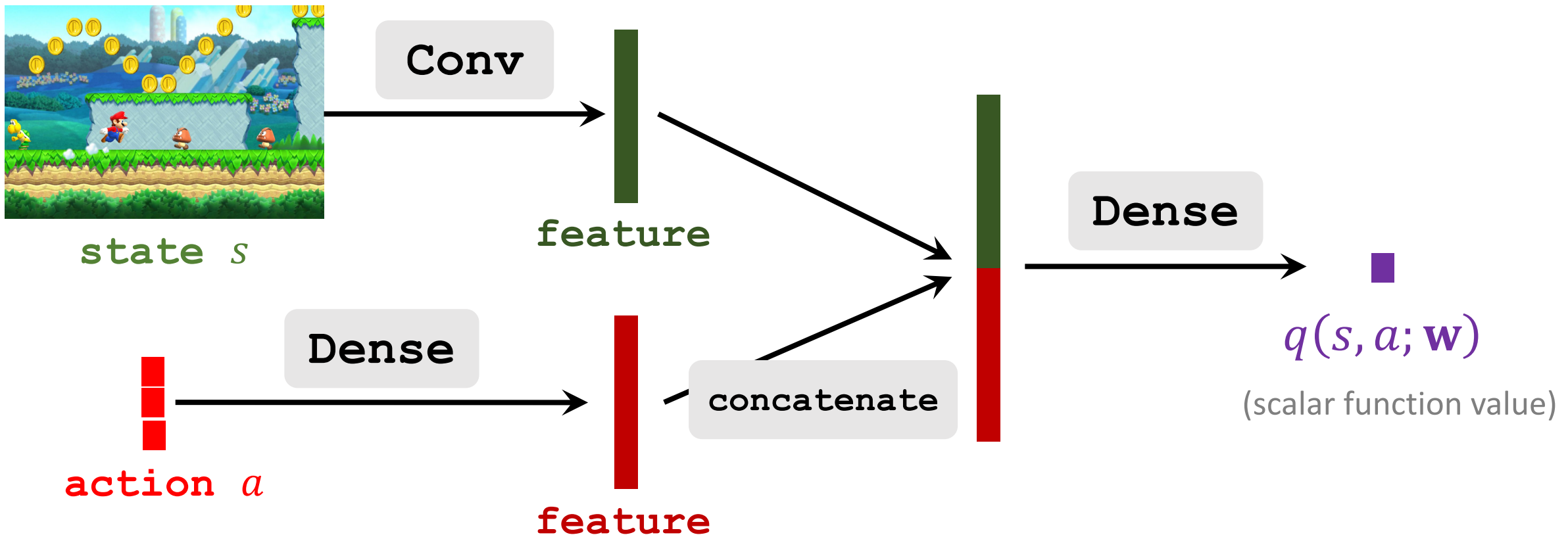
Policy Network (Actor): $\pi(a|s, \theta)$

- Input: **state** s , e.g., a screenshot of Super Mario.
- Output: probability distribution over the **actions**.
- Let \mathcal{A} be the set all actions, e.g., $\mathcal{A} = \{\text{"left"}, \text{"right"}, \text{"up"}\}$.
- $\sum_{a \in \mathcal{A}} \pi(a|s, \theta) = 1$. (That is why we use softmax activation.)



Value Network (Critic): $q(s, a; \mathbf{w})$

- Inputs: **state** s and **action** a .
- Output: approximate action-value (scalar).

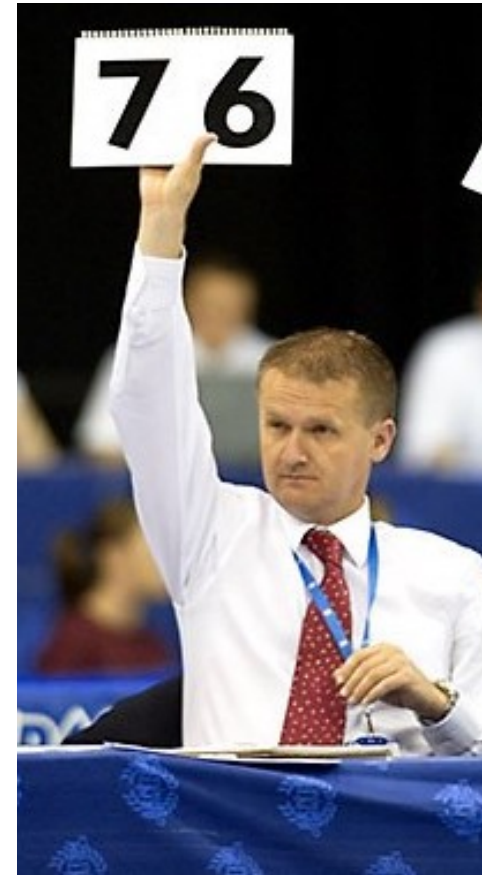


Actor-Critic Method

policy network (actor)



value network (critic)



Train the Neural Networks

Train the networks

Definition: State-value function approximated using neural networks.

- $V(\textcolor{green}{s}; \boldsymbol{\theta}, \mathbf{w}) = \sum_{\textcolor{red}{a}} \pi(\textcolor{red}{a} | \textcolor{green}{s}; \boldsymbol{\theta}) \cdot q(\textcolor{green}{s}, \textcolor{red}{a}; \mathbf{w})$.

Training: Update the parameters $\boldsymbol{\theta}$ and \mathbf{w} .

Train the networks

Definition: State-value function approximated using neural networks.

- $V(s; \theta, \mathbf{w}) = \sum_a \pi(a|s; \theta) \cdot q(s, a; \mathbf{w})$.

Training: Update the parameters θ and \mathbf{w} .

- Update policy network $\pi(a|s; \theta)$ to increase the state-value $V(s; \theta, \mathbf{w})$.
 - Actor gradually performs better.
 - Supervision is purely from the value network (critic).

Train the networks

Definition: State-value function approximated using neural networks.

- $V(s; \theta, \mathbf{w}) = \sum_a \pi(a|s; \theta) \cdot q(s, a; \mathbf{w})$.

Training: Update the parameters θ and \mathbf{w} .

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

Train the networks

Definition: State-value function approximated using neural networks.

- $V(\textcolor{green}{s}; \boldsymbol{\theta}, \mathbf{w}) = \sum_{\textcolor{red}{a}} \pi(\textcolor{red}{a} | \textcolor{green}{s}; \boldsymbol{\theta}) \cdot q(\textcolor{green}{s}, \textcolor{red}{a}; \mathbf{w})$.

Training: Update the parameters $\boldsymbol{\theta}$ and \mathbf{w} .

1. Observe the state $\textcolor{green}{s}_t$.
2. Randomly sample action $\textcolor{red}{a}_t$ according to $\pi(\cdot | \textcolor{green}{s}_t; \boldsymbol{\theta}_t)$.
3. Perform $\textcolor{red}{a}_t$ and observe new state $\textcolor{green}{s}_{t+1}$ and reward $\textcolor{blue}{r}_t$.
4. Update \mathbf{w} (in value network) using temporal difference (TD).
5. Update $\boldsymbol{\theta}$ (in policy network) using policy gradient.

Update **value network** q using **TD**

- Compute $q(s_t, a_t; \mathbf{w}_t)$ and $q(s_{t+1}, a_{t+1}; \mathbf{w}_t)$.
- TD target: $y_t = r_t + \gamma \cdot q(s_{t+1}, a_{t+1}; \mathbf{w}_t)$.

Update **value network** q using **TD**

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

Update **policy network** π using **policy gradient**

Definition: State-value function approximated using neural networks.

- $V(\textcolor{green}{s}; \boldsymbol{\theta}, \mathbf{w}) = \sum_{\textcolor{red}{a}} \pi(\textcolor{red}{a} | \textcolor{green}{s}; \boldsymbol{\theta}) \cdot q(\textcolor{green}{s}, \textcolor{red}{a}; \mathbf{w})$.

Policy gradient: Derivative of $V(\textcolor{green}{s}_t; \boldsymbol{\theta}, \mathbf{w})$ w.r.t. $\boldsymbol{\theta}$.

- Let $\mathbf{g}(\textcolor{red}{a}, \boldsymbol{\theta}) = \frac{\partial \log \pi(\textcolor{red}{a} | \textcolor{green}{s}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cdot q(\textcolor{green}{s}_t, \textcolor{red}{a}; \mathbf{w})$.
- $\frac{\partial V(\textcolor{green}{s}; \boldsymbol{\theta}, \mathbf{w}_t)}{\partial \boldsymbol{\theta}} = \mathbb{E}_{\textcolor{red}{A}}[\mathbf{g}(\textcolor{red}{A}, \boldsymbol{\theta})]$.

Update **policy network** π using **policy gradient**

Definition: State-value function approximated using neural networks.

- $V(\textcolor{green}{s}; \boldsymbol{\theta}, \mathbf{w}) = \sum_{\textcolor{red}{a}} \pi(\textcolor{red}{a} | \textcolor{green}{s}; \boldsymbol{\theta}) \cdot q(\textcolor{green}{s}, \textcolor{red}{a}; \mathbf{w})$.

Policy gradient: Derivative of $V(\textcolor{green}{s}_t; \boldsymbol{\theta}, \mathbf{w})$ w.r.t. $\boldsymbol{\theta}$.

- Let $\mathbf{g}(\textcolor{red}{a}, \boldsymbol{\theta}) = \frac{\partial \log \pi(\textcolor{red}{a} | \textcolor{green}{s}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cdot q(\textcolor{green}{s}_t, \textcolor{red}{a}; \mathbf{w})$.
- $\frac{\partial V(\textcolor{green}{s}; \boldsymbol{\theta}, \mathbf{w}_t)}{\partial \boldsymbol{\theta}} = \mathbb{E}_{\textcolor{red}{A}}[\mathbf{g}(\textcolor{red}{A}, \boldsymbol{\theta})]$.

Algorithm: Update policy network using stochastic policy gradient.

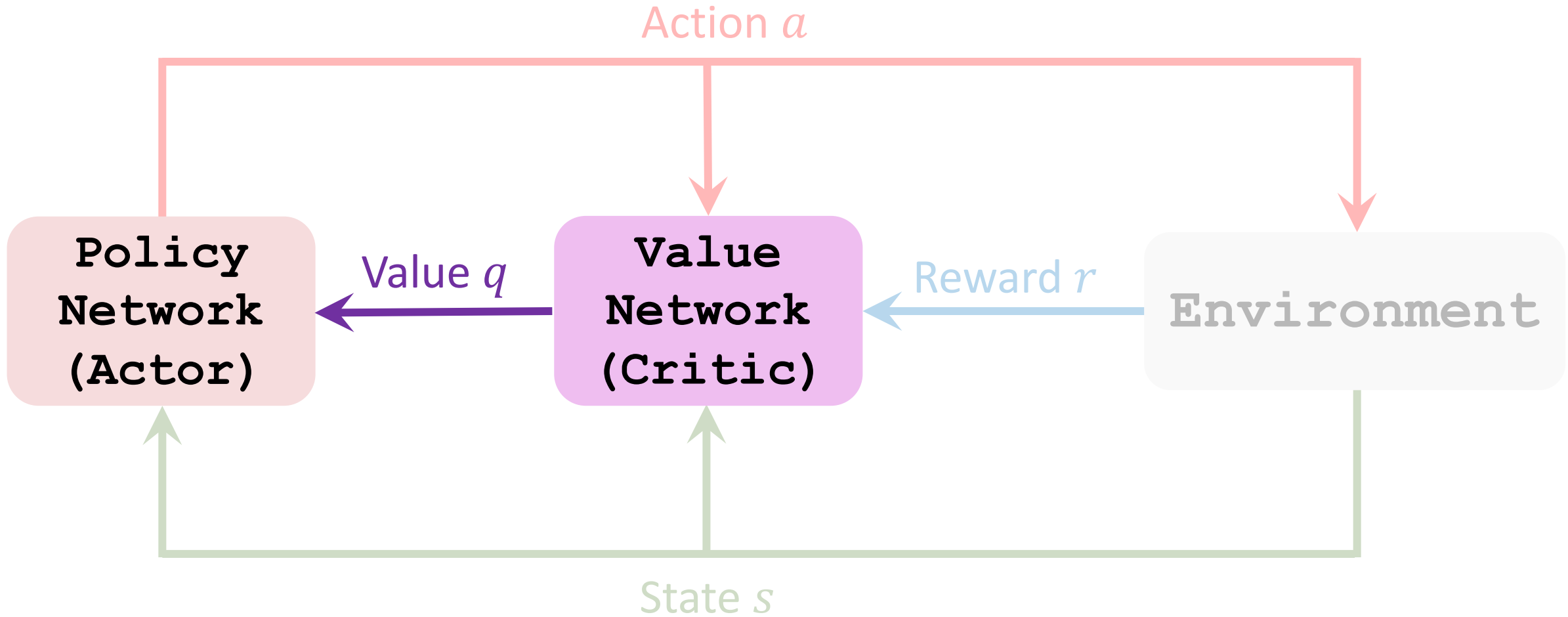
- Random sampling: $\textcolor{red}{a} \sim \pi(\cdot | \textcolor{green}{s}_t; \boldsymbol{\theta}_t)$. (Thus $\mathbf{g}(\textcolor{red}{a}, \boldsymbol{\theta})$ is unbiased.)
- Stochastic gradient ascent: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \beta \cdot \mathbf{g}(\textcolor{red}{a}, \boldsymbol{\theta}_t)$.

Actor-Critic Method

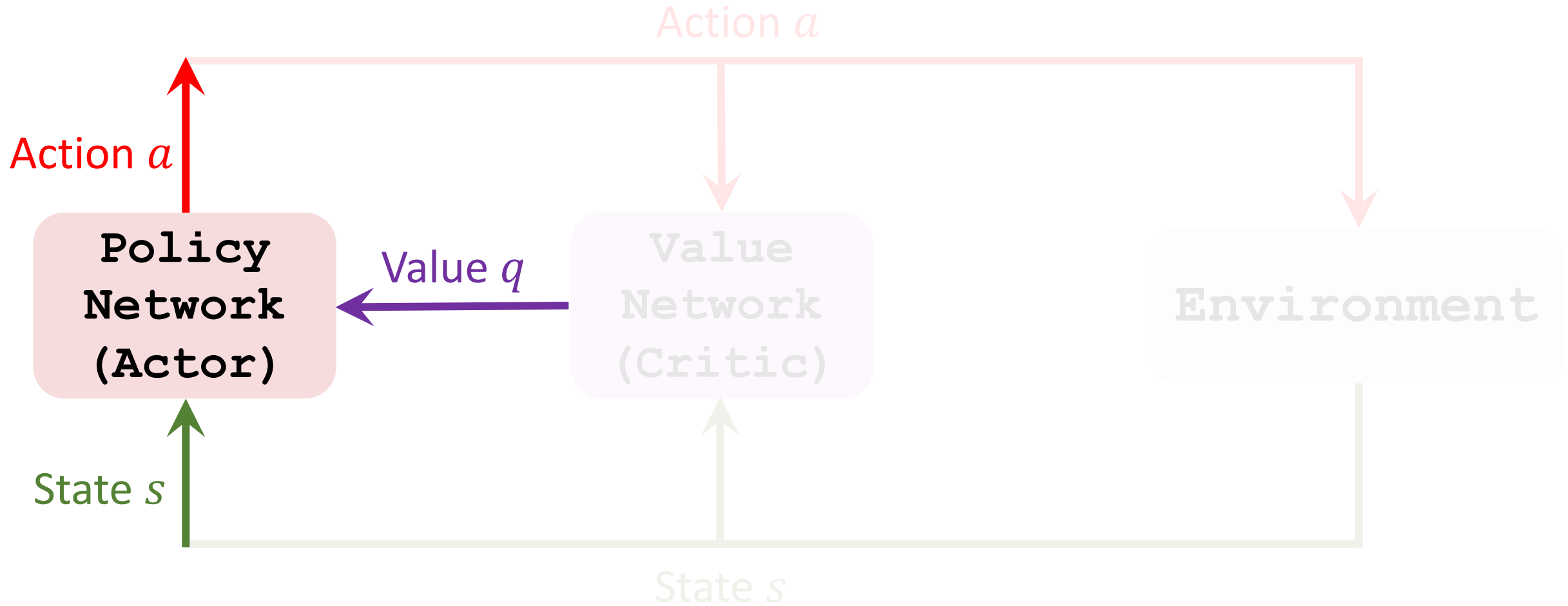
Action a



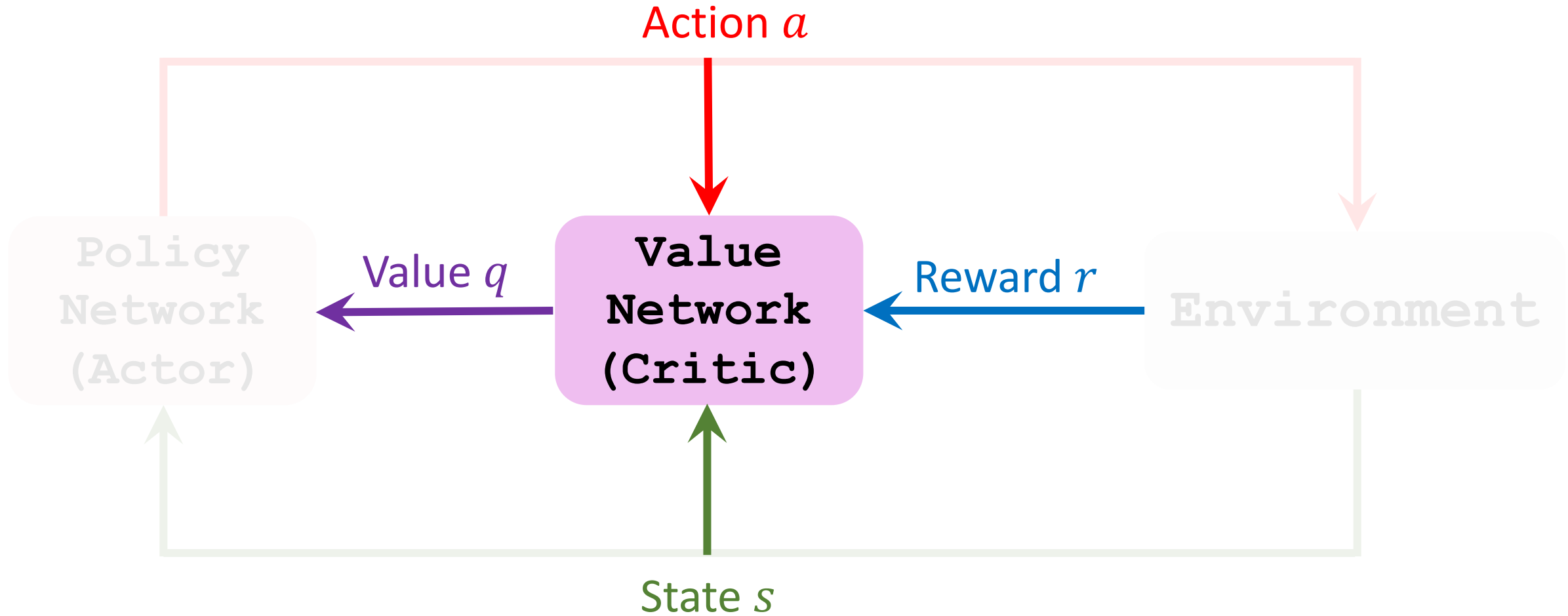
Actor-Critic Method



Actor-Critic Method: Update Actor



Actor-Critic Method: Update Critic



Summary of Algorithm

1. Observe state s_t and randomly sample $a_t \sim \pi(\cdot | s_t; \theta_t)$.
2. Perform a_t ; then environment gives new state s_{t+1} and reward r_t .
3. Randomly sample $\tilde{a}_{t+1} \sim \pi(\cdot | s_{t+1}; \theta_t)$. (Do not perform \tilde{a}_{t+1} !)

Summary of Algorithm

1. Observe state s_t and randomly sample $a_t \sim \pi(\cdot | s_t; \theta_t)$.
2. Perform a_t ; then environment gives new state s_{t+1} and reward r_t .
3. Randomly sample $\tilde{a}_{t+1} \sim \pi(\cdot | s_{t+1}; \theta_t)$. (Do not perform \tilde{a}_{t+1} !)
4. Evaluate value network: $q_t = q(s_t, a_t; \mathbf{w}_t)$ and $q_{t+1} = q(s_{t+1}, \tilde{a}_{t+1}; \mathbf{w}_t)$.
5. Compute TD error: $\delta_t = q_t - \underbrace{(r_t + \gamma \cdot q_{t+1})}_{\text{TD Target}}$.

TD Target

Summary of Algorithm

1. Observe state s_t and randomly sample $a_t \sim \pi(\cdot | s_t; \theta_t)$.
2. Perform a_t ; then environment gives new state s_{t+1} and reward r_t .
3. Randomly sample $\tilde{a}_{t+1} \sim \pi(\cdot | s_{t+1}; \theta_t)$. (Do not perform \tilde{a}_{t+1} !)
4. Evaluate value network: $q_t = q(s_t, a_t; \mathbf{w}_t)$ and $q_{t+1} = q(s_{t+1}, \tilde{a}_{t+1}; \mathbf{w}_t)$.
5. Compute TD error: $\delta_t = q_t - (r_t + \gamma \cdot q_{t+1})$.
6. Differentiate value network: $\mathbf{d}_{w,t} = \frac{\partial q(s_t, a_t; \mathbf{w})}{\partial \mathbf{w}} \Big|_{\mathbf{w}=\mathbf{w}_t}$.
7. Update value network: $\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \cdot \delta_t \cdot \mathbf{d}_{w,t}$.

Summary of Algorithm

1. Observe state s_t and randomly sample $a_t \sim \pi(\cdot | s_t; \theta_t)$.
2. Perform a_t ; then environment gives new state s_{t+1} and reward r_t .
3. Randomly sample $\tilde{a}_{t+1} \sim \pi(\cdot | s_{t+1}; \theta_t)$. (Do not perform \tilde{a}_{t+1} !)
4. Evaluate value network: $q_t = q(s_t, a_t; \mathbf{w}_t)$ and $q_{t+1} = q(s_{t+1}, \tilde{a}_{t+1}; \mathbf{w}_t)$.
5. Compute TD error: $\delta_t = q_t - (r_t + \gamma \cdot q_{t+1})$.
6. Differentiate value network: $\mathbf{d}_{w,t} = \frac{\partial q(s_t, a_t; \mathbf{w})}{\partial \mathbf{w}} \Big|_{\mathbf{w}=\mathbf{w}_t}$.
7. Update value network: $\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \cdot \delta_t \cdot \mathbf{d}_{w,t}$.
8. Differentiate policy network: $\mathbf{d}_{\theta,t} = \frac{\partial \log \pi(a_t | s_t, \theta)}{\partial \theta} \Big|_{\theta=\theta_t}$.
9. Update policy network: $\theta_{t+1} = \theta_t + \beta \cdot q_t \cdot \mathbf{d}_{\theta,t}$.

Summary of Algorithm

1. Observe state s_t and randomly sample $a_t \sim \pi(\cdot | s_t; \boldsymbol{\theta}_t)$.
2. Perform a_t ; then environment gives new state s_{t+1} and reward r_t .
3. Randomly sample $\tilde{a}_{t+1} \sim \pi(\cdot | s_{t+1}; \boldsymbol{\theta}_t)$. (Do not perform \tilde{a}_{t+1} !)
4. Evaluate value network: $q_t = q(s_t, a_t; \mathbf{w}_t)$ and $q_{t+1} = q(s_{t+1}, \tilde{a}_{t+1}; \mathbf{w}_t)$.
5. Compute TD error: $\delta_t = q_t - (r_t + \gamma \cdot q_{t+1})$.
6. Differentiate value network: $\mathbf{d}_{w,t} = \frac{\partial q(s_t, a_t; \mathbf{w})}{\partial \mathbf{w}} \Big|_{\mathbf{w}=\mathbf{w}_t}$.
7. Update value network: $\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \cdot \delta_t \cdot \mathbf{d}_{w,t}$.
8. Differentiate policy network: $\mathbf{d}_{\theta,t} = \frac{\partial \log \pi(a_t | s_t, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t}$.
9. Update policy network: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \beta \cdot q_t \cdot \mathbf{d}_{\theta,t}$.

Policy Gradient with Baseline

1. Observe state s_t and randomly sample $a_t \sim \pi(\cdot | s_t; \boldsymbol{\theta}_t)$.
2. Perform a_t ; then environment gives new state s_{t+1} and reward r_t .
3. Randomly sample $\tilde{a}_{t+1} \sim \pi(\cdot | s_{t+1}; \boldsymbol{\theta}_t)$. (Do not perform \tilde{a}_{t+1} !)
4. Evaluate value network: $q_t = q(s_t, a_t; \mathbf{w}_t)$ and $q_{t+1} = q(s_{t+1}, \tilde{a}_{t+1}; \mathbf{w}_t)$.
5. Compute TD error: $\delta_t = q_t - \underbrace{(r_t + \gamma \cdot q_{t+1})}_{\text{Baseline}}$.
6. Differentiate value network: $\mathbf{d}_{w,t} = \frac{\partial q(s_t, a_t, \mathbf{w})}{\partial \mathbf{w}} \big|_{\mathbf{w}=\mathbf{w}_t}$.
7. Update value network: $\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \cdot \delta_t \cdot \mathbf{d}_{w,t}$.
8. Differentiate policy network: $\mathbf{d}_{\theta,t} = \frac{\partial \log \pi(a_t | s_t, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t}$.
9. Update policy network: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \beta \cdot \delta_t \cdot \mathbf{d}_{\theta,t}$.

Summary

Policy Network and Value Network

Definition: State-value function.

- $V_{\pi}(s) = \sum_a \pi(a|s) \cdot Q_{\pi}(s, a).$

Definition: function approximation using neural networks.

- Approximate policy function $\pi(a|s)$ by $\pi(a|s; \theta)$ (actor).
- Approximate value function $Q_{\pi}(s, a)$ by $q(s, a; \mathbf{w})$ (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 q (**critic**) provides the **actor** with supervision.

Roles of **Actor** and **Critic**

During training

- Agent is controlled by policy network (**actor**): $a_t \sim \pi(\cdot | s_t; \theta)$.
- Value network q (**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 q (**critic**) will not be used.

Training

Learning: Update the **policy network (actor)** by **policy gradient**.

- Seek to increase state-value: $V(\mathbf{s}; \boldsymbol{\theta}, \mathbf{w}) = \sum_a \pi(a|\mathbf{s}; \boldsymbol{\theta}) \cdot q(\mathbf{s}, a; \mathbf{w})$.
- Compute policy gradient: $\frac{\partial V(\mathbf{s}; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \mathbb{E}_A \left[\frac{\partial \log \pi(A|\mathbf{s}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cdot q(\mathbf{s}, A; \mathbf{w}) \right]$.
- Perform gradient ascent.

Training

Learning: Update the **policy network (actor)** by **policy gradient**.

- Seek to increase state-value: $V(\mathbf{s}; \boldsymbol{\theta}, \mathbf{w}) = \sum_a \pi(a|\mathbf{s}; \boldsymbol{\theta}) \cdot q(\mathbf{s}, a; \mathbf{w})$.
- Compute policy gradient: $\frac{\partial V(\mathbf{s}; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \mathbb{E}_A \left[\frac{\partial \log \pi(A|\mathbf{s}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cdot q(\mathbf{s}, A; \mathbf{w}) \right]$.
- Perform gradient ascent.

Learning: Update the **value network (critic)** by **TD learning**.

- Predicted action-value: $q_t = q(\mathbf{s}_t, \mathbf{a}_t; \mathbf{w})$.
- TD target: $y_t = r_t + \gamma \cdot q(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}; \mathbf{w})$
- Gradient: $\frac{\partial (q_t - y_t)^2 / 2}{\partial \mathbf{w}} = (q_t - y_t) \cdot \frac{\partial q(\mathbf{s}_t, \mathbf{a}_t; \mathbf{w})}{\partial \mathbf{w}}$.
- Perform gradient descent.

Thank you!