Policy-Based Reinforcement Learning

Shusen Wang

Policy Function Approximation

Action-Value Function

Definition: Discounted return (aka cumulative discounted future reward).

- $R_t = r_t + \gamma\, r_{t+1} + \gamma^2\, r_{t+2} + \gamma^3\, r_{t+3} + \cdots$ (to infinity.)

Definition: Action-value function.

- $Q_\pi(s_t, a_t) = \mathbb{E}\left[R_t | s_t, a_t, \pi\right].$

- Taken w.r.t. actions $a_{t+1}, a_{t+2}, a_{t+3}, \cdots$ and states $s_{t+1}, s_{t+2}, s_{t+3}, \cdots$
- Actions are randomly sampled: $a_t \sim \pi(\cdot\,|s_t)$. (Policy function.)
- States are randomly sampled: $s_{t+1} \sim p(\cdot\,|s_t, a_t)$. (State transition.)

State-Value Function

Definition: Discounted return (aka cumulative discounted future reward).

- $R_t = r_t + \gamma\, r_{t+1} + \gamma^2\, r_{t+2} + \gamma^3\, r_{t+3} + \cdots$ (to infinity.)

Definition: Action-value function.

- $Q_\pi(s_t, a_t) = \mathbb{E}\left[R_t | s_t, a_t, \pi\right]$.

Definition: State-value function.

- $V_\pi(s_t) = \mathbb{E}_{a\sim\pi(\cdot|s_t)}\left[Q_\pi(s_t, a)\right] = \sum_a \pi(a|s_t) \cdot Q_\pi(s_t, a)$.

  - Integrate out action a.
  - Given s_t and π, state-value function can tell the expected return.

State-Value Function

Definition: State-value function.

- $V_\pi(s_t) = \mathbb{E}_{a \sim \pi(\cdot|s_t)}\left[Q_\pi(s_t, a)\right] = \sum_a \pi(a|s_t) \cdot Q_\pi(s_t, a).$

Policy Function Approximation

Definition: State-value function.

- $V_\pi(s_t) = \mathbb{E}_{a \sim \pi(\cdot|s_t)}\left[Q_\pi(s_t, a)\right] = \sum_a \pi(a|s_t) \cdot Q_\pi(s_t, a)$.

Policy-based learning: Learn a policy π that maximizes $\mathbb{E}_s[V_\pi(s)]$.

Policy Function Approximation

Definition: State-value function.

- $V_\pi(s_t) = \mathbb{E}_{a\sim\pi(\cdot|s_t)}[Q_\pi(s_t, a)] = \sum_a \pi(a|s_t) \cdot Q_\pi(s_t, a)$.

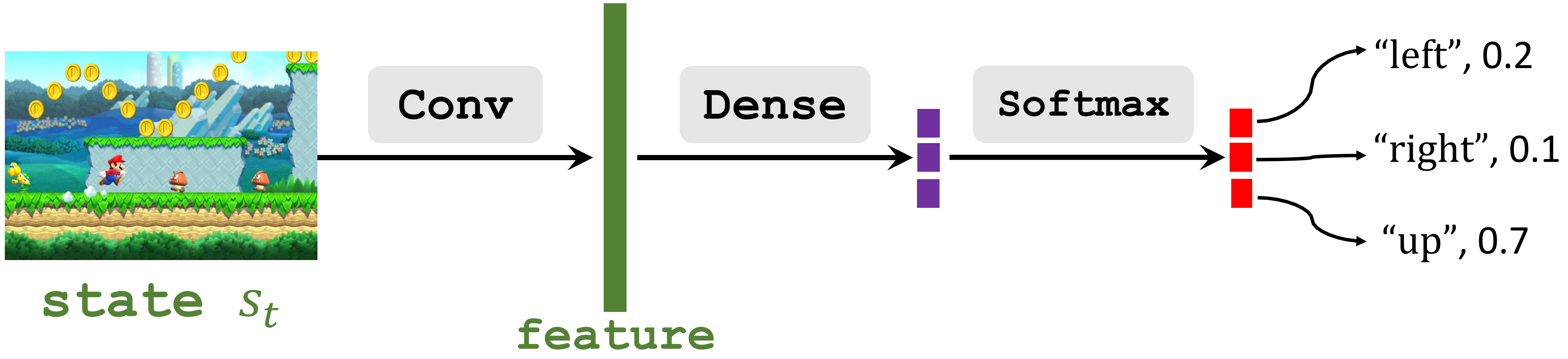
Policy-based learning: Learn a policy π that maximizes $\mathbb{E}_s[V_\pi(s)]$.

Policy network: Use a neural net to approximate $\pi(a|s)$.

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

Policy Network $\pi(a|s, \boldsymbol{\theta})$

- $\pi(a|s; \boldsymbol{\theta}) = 0.2$ means that observing s, the agent shall take action a with probability 0.2.
- Let $\mathcal{A}$ be the set all actions, e.g., $\mathcal{A} = \{\text{“left”}, \text{“right”}, \text{“up”}\}$.
- $\sum_{a \in \mathcal{A}} \pi(a|s, \boldsymbol{\theta}) = 1$. (That is why we use softmax activation.)



Policy Gradient

Policy Gradient

Definition: Approximate state-value function.

- $V(s;\boldsymbol{\theta}) = \sum_a \pi(a|s;\boldsymbol{\theta}) \cdot Q_\pi(s,a)$.

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

- $\frac{\partial\, V(s;\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}}$

Policy Gradient

Definition: Approximate state-value function.

- $V(s;\boldsymbol{\theta}) = \sum_{a} \pi(a|s;\boldsymbol{\theta}) \cdot Q_\pi(s,a)$.

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

- $\frac{\partial\, V(s;\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} = \sum_{a} \frac{\partial \pi(a|s;\boldsymbol{\theta}) \cdot Q_\pi(s,a)}{\partial\, \boldsymbol{\theta}}$

Push the differentiation into the summation.

Policy Gradient

Definition: Approximate state-value function.

- $V(s;\boldsymbol{\theta}) = \sum_a \pi(a|s;\boldsymbol{\theta}) \cdot Q_\pi(s,a)$.

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

- $\frac{\partial\, V(s;\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} = \sum_a \frac{\partial \pi(a|s;\boldsymbol{\theta}) \cdot Q_\pi(s,a)}{\partial\, \boldsymbol{\theta}} = \sum_a \frac{\partial \pi(a|s;\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} \cdot Q_\pi(s,a)$

Q_π is independent of $\boldsymbol{\theta}$.

Policy Gradient

Definition: Approximate state-value function.

- $V(s;\boldsymbol{\theta}) = \sum_a \pi(a|s;\boldsymbol{\theta}) \cdot Q_\pi(s,a)$.

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

- $$\frac{\partial\, V(s;\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} = \sum_a \frac{\partial \pi(a|s;\boldsymbol{\theta}) \cdot Q_\pi(s,a)}{\partial\, \boldsymbol{\theta}} = \sum_a \boxed{\frac{\partial \pi(a|s;\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}}} \cdot Q_\pi(s,a)$$
  $$= \sum_a \boxed{\pi(a|s;\boldsymbol{\theta}) \cdot \frac{\partial \log \pi(a|s;\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}}} \cdot Q_\pi(s,a)$$

- Chain rule: $\frac{\partial \log[f(x)]}{\partial\, x} = \frac{\partial\, f(x)}{\partial\, x} \cdot \frac{1}{f(x)}$.
- Thus $\frac{\partial\, f(x)}{\partial\, x} = f(x) \cdot \frac{\partial \log[f(x)]}{\partial\, x}$.

Policy Gradient

Definition: Approximate state-value function.

- $V(s;\boldsymbol{\theta}) = \sum_a \pi(a|s;\boldsymbol{\theta}) \cdot Q_\pi(s,a)$.

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

- $$\frac{\partial\, V(s;\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} = \sum_a \frac{\partial \pi(a|s;\boldsymbol{\theta}) \cdot Q_\pi(s,a)}{\partial\, \boldsymbol{\theta}} = \sum_a \frac{\partial \pi(a|s;\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} \cdot Q_\pi(s,a)$$
$$= \sum_a \pi(a|s;\boldsymbol{\theta}) \cdot \frac{\partial \log \pi(a|s;\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} \cdot Q_\pi(s,a)$$
$$= \mathbb{E}_a\left[\frac{\partial \log \pi(a|s;\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} \cdot Q_\pi(s,a)\right].$$

The expectation is taken w.r.t. the random variable $a \sim \pi(\cdot\,|s;\boldsymbol{\theta})$.

Policy Gradient

Definition: Approximate state-value function.

- $V(s;\boldsymbol{\theta}) = \sum_a \pi(a|s;\boldsymbol{\theta}) \cdot Q_\pi(s,a)$.

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

- $\frac{\partial\, V(s;\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} = \mathbb{E}_{a\sim\pi(\cdot|s;\boldsymbol{\theta})}\left[\frac{\partial \log \pi(a|s,\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} \cdot Q_\pi(s,a)\right]$.

Policy Gradient

Definition: Approximate state-value function.

- $V(s; \boldsymbol{\theta}) = \sum_a \pi(a|s; \boldsymbol{\theta}) \cdot Q_\pi(s, a)$.

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

- $\frac{\partial\, V(s;\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} = \mathbb{E}_{a\sim\pi(\cdot|s;\boldsymbol{\theta})}\left[\frac{\partial \log \pi(a|s,\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} \cdot Q_\pi(s, a)\right]$.

- Policy gradient ascent:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \beta \cdot \frac{\partial\, V(s_t;\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t}.$$

- Increasing the state-value means improving the policy.

Policy Gradient

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

- $\frac{\partial\, V(s;\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} = \mathbb{E}_{a\sim\pi(\cdot|s;\boldsymbol{\theta})}\left[\frac{\partial \log \pi(a|s,\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} \cdot Q_\pi(s,a)\right].$

Question: How to compute the policy gradient $\frac{\partial\, V(s;\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}}$?

Policy Gradient

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

- $\frac{\partial\, V(s;\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} = \mathbb{E}_{a\sim\pi(\cdot|s;\boldsymbol{\theta})}\left[\frac{\partial \log \pi(a|s,\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} \cdot Q_\pi(s,a)\right].$

Question: How to compute the policy gradient $\frac{\partial\, V(s;\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}}$?

- Sample actions: $a^{(1)}, a^{(2)}, \cdots, a^{(k)} \sim \pi(\cdot\,|s;\boldsymbol{\theta})$.
  - (The agent does not actually perform the actions.)
  - Sample only one action (i.e., $k=1$) also works.
- Compute $\tilde{\mathbf{g}}(\boldsymbol{\theta}) = \frac{1}{k}\sum_{i=1}^{k} \frac{\partial \log \pi(a^{(1)}|s_t,\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} \cdot Q_\pi\left(s_t, a^{(1)}\right)$.
- $\tilde{\mathbf{g}}(\boldsymbol{\theta})$ is unbiased estimate of $\frac{\partial\, V(s;\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}}$.

Update policy network using policy gradient

1. Observe the state s_t.
2. Randomly sample action a_t according to $\pi(\cdot \,|s_t; \boldsymbol{\theta}_t)$.

The agent may not actually perform action a_t.

Update policy network using policy gradient

1. Observe the state s_t.
2. Randomly sample action a_t according to $\pi(\cdot\,|s_t;\boldsymbol{\theta}_t)$.
3. Compute $q_t \approx Q_\pi(s_t, a_t)$ (some estimate).
4. Differentiate policy network: $\mathbf{d}_{\theta,t} = \frac{\partial \log \pi(a_t|s_t,\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}}\,\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t}$.
5. (Stochastic) policy gradient: $\tilde{\mathbf{g}}(\boldsymbol{\theta}_t) \approx q_t \cdot \mathbf{d}_{\theta,t}$.
6. Update policy network: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \beta \cdot \tilde{\mathbf{g}}(\boldsymbol{\theta}_t)$.

Update policy network using policy gradient

1. Observe the state s_t.
2. Randomly sample action a_t according to $\pi(\cdot\,|s_t;\boldsymbol{\theta}_t)$.
3. Compute $q_t \approx Q_\pi(s_t, a_t)$ (some estimate). How?
4. Differentiate policy network: $\mathbf{d}_{\theta,t} = \frac{\partial \log \pi(a_t|s_t,\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t}$.
5. (Stochastic) policy gradient: $\tilde{\mathbf{g}}(\boldsymbol{\theta}_t) \approx q_t \cdot \mathbf{d}_{\theta,t}$.
6. Update policy network: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \beta \cdot \tilde{\mathbf{g}}(\boldsymbol{\theta}_t)$.

Update policy network using policy gradient

1. Observe the state s_t.
2. Randomly sample action a_t according to $\pi(\cdot \mid s_t; \boldsymbol{\theta}_t)$.
3. $\boxed{\text{Compute } q_t \approx Q_\pi(s_t, a_t) \text{ (some estimate).}}$ How?

Option 1: Monte Carlo.

- Play the game to the end and generate the trajectory:

$$s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, \quad \cdots \quad , s_T, a_T, r_T.$$

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

Update policy network using policy gradient

1. Observe the state s_t.
2. Randomly sample action a_t according to $\pi(\cdot \mid s_t; \boldsymbol{\theta}_t)$.
3. Compute $q_t \approx Q_\pi(s_t, a_t)$ (some estimate). How?

Option 2: Approximate Q_π using a neural network.

- This leads to the actor-critic method.

Actor-Critic Method

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 a neural net to approximate $\pi(a|s)$.

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

Value network (critic): Use a neural net to approximate $Q_\pi(s, a)$.

- 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.

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

Policy network (actor): Use a neural net to approximate $\pi(a|s)$.

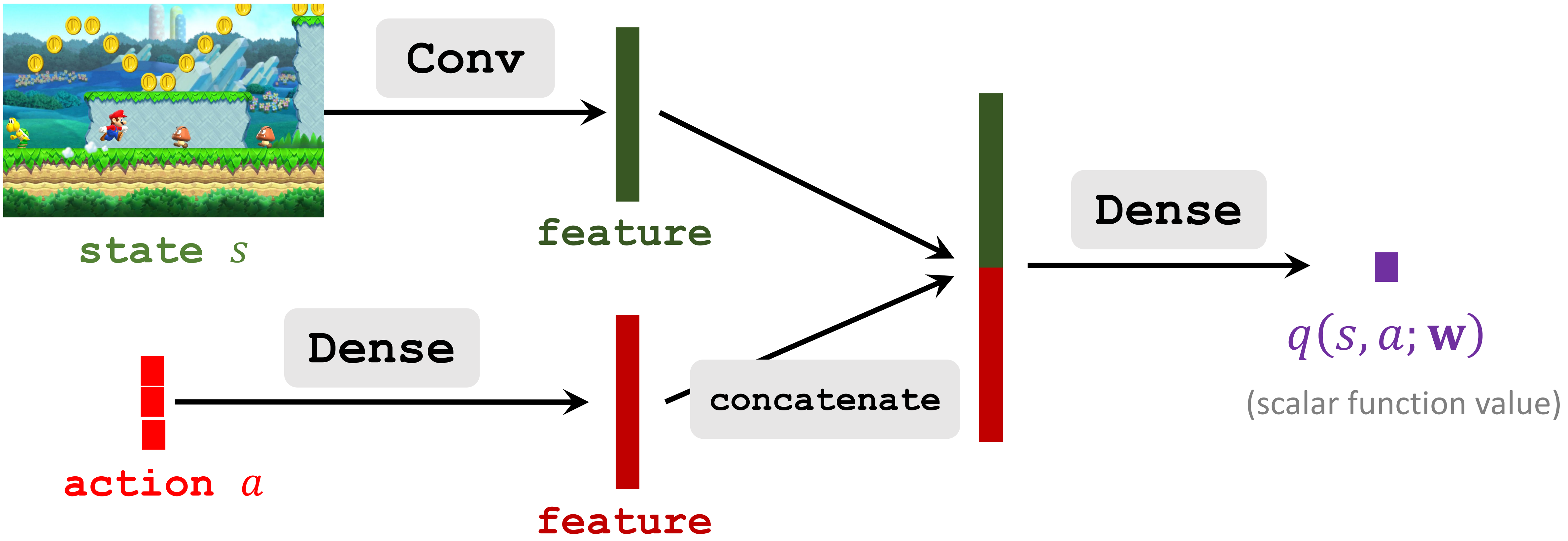
- Use neural net $\pi(a|s;\boldsymbol{\theta})$ to approximate $\pi(a|s)$.
- $\boldsymbol{\theta}$: trainable parameters of the neural net.

Value network (critic): Use a neural net to approximate $Q_\pi(s,a)$.

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

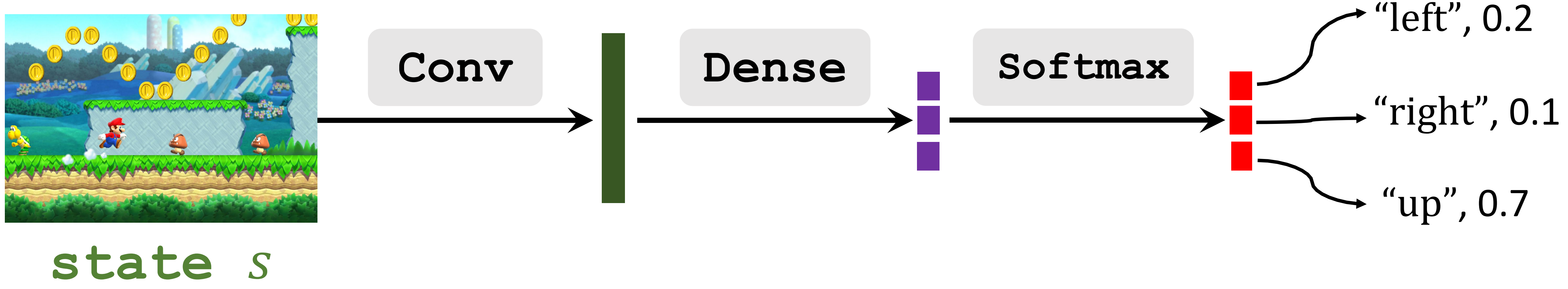
Value Network (Critic): $q(s, a; \mathbf{w})$

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



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

- 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., $\mathcal{A} = \{\text{“left”}, \text{“right”}, \text{“up”}\}$.
- $\sum_{a \in \mathcal{A}} \pi(a|s, \boldsymbol{\theta}) = 1$. (That is why we use softmax activation.)



State-Value Function Approximation

Definition: State-value function approximated using neural networks.

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

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

- Update policy network $\pi(a|s;\boldsymbol{\theta})$ to increase the state-value $V(s;\boldsymbol{\theta},\mathbf{w})$.
  - Actor gradually performs better.
  - (Not actually better; the actor just caters for the critic's taste.)
  - Supervision is purely from the 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.

State-Value Function Approximation

Definition: State-value function approximated using neural networks.

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

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

1. Observe the state s_t.
2. Randomly sample action a_t according to $\pi(\cdot\ |s_t; \boldsymbol{\theta}_t)$.
3. Observe new state s_{t+1} and reward r_t.
4. Update $\boldsymbol{\theta}$ (in policy network) using policy gradient.
5. Update $\mathbf{w}$ (in value network) using temporal difference (TD).

Update value network using TD

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

- Loss: $L_t(\mathbf{w}) = \frac{1}{2}[q(s_t, a_t; \mathbf{w}) - y_t]^2$.
- Gradient: $\mathbf{g}_t(\mathbf{w}) = \frac{\partial\, L_t(\mathbf{w})}{\partial\, \mathbf{w}} = [q(s_t, a_t; \mathbf{w}) - y_t] \cdot \frac{\partial\, q(s_t, a_t; \mathbf{w})}{\partial\, \mathbf{w}}$.
- Gradient descent: $\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \cdot \mathbf{g}_t(\mathbf{w}_t)$.

Update policy network using policy gradient

Definition: State-value function approximated using neural networks.

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

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

- Let $\tilde{\mathbf{g}}_{\theta}(\boldsymbol{\theta};a,s,\mathbf{w}) = \frac{\partial \log \pi(a|s,\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} \cdot q(s_t,a;\mathbf{w})$.
- $\frac{\partial\, V(s;\boldsymbol{\theta},\mathbf{w}_t)}{\partial\, \boldsymbol{\theta}} = \mathbb{E}_{a\sim\pi(\cdot|s;\boldsymbol{\theta})}\left[\, \tilde{\mathbf{g}}_{\theta}(\boldsymbol{\theta};a,s,\mathbf{w})\,\right]$.

Update policy network using policy gradient

Definition: State-value function approximated using neural networks.

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

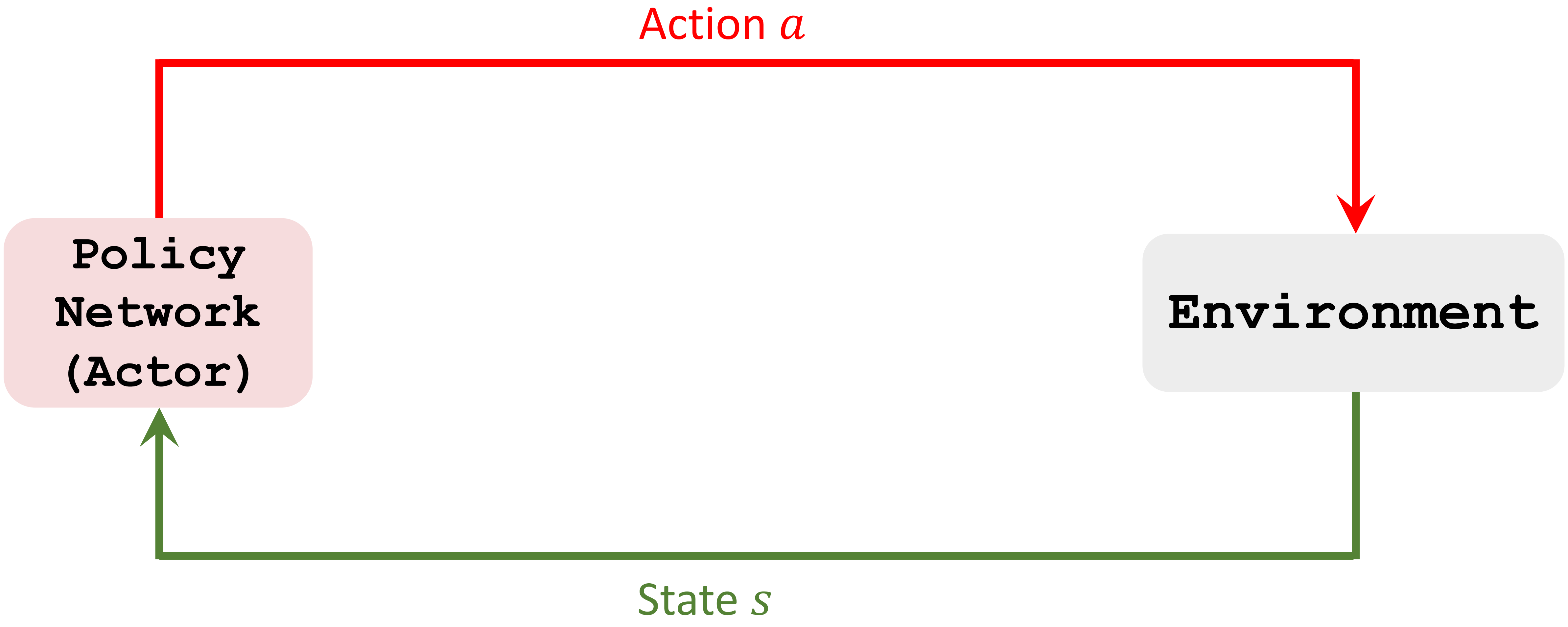
Policy gradient: Derivative of $V(s_t; \boldsymbol{\theta}, \mathbf{w})$ w.r.t. $\boldsymbol{\theta}$.

- Let $\tilde{\mathbf{g}}_\theta(\boldsymbol{\theta}; a, s, \mathbf{w}) = \frac{\partial \log \pi(a|s, \boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} \cdot q(s_t, a; \mathbf{w})$.
- $\frac{\partial\, V(s; \boldsymbol{\theta}, \mathbf{w}_t)}{\partial\, \boldsymbol{\theta}} = \mathbb{E}_{a \sim \pi(\cdot|s; \boldsymbol{\theta})}\left[\, \tilde{\mathbf{g}}_\theta(\boldsymbol{\theta}; a, s, \mathbf{w})\, \right]$.

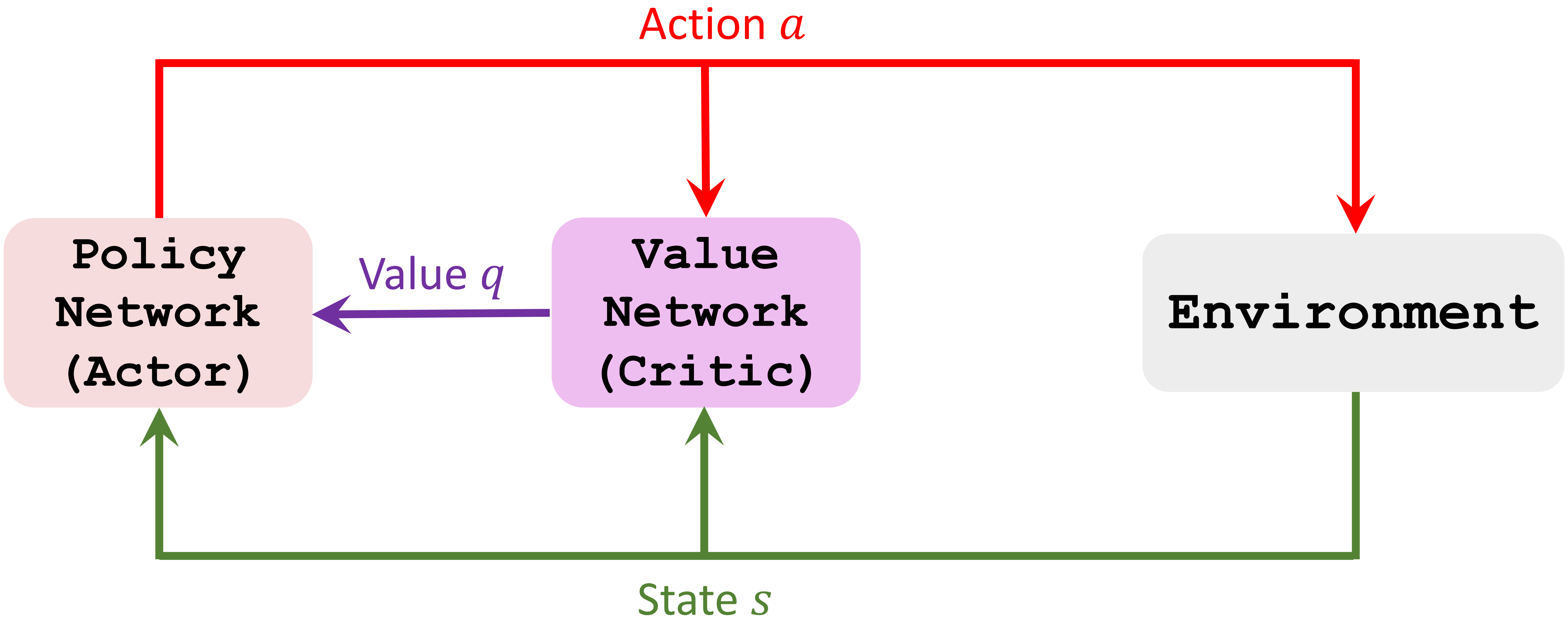
Algorithm: Update policy network using stochastic policy gradient.

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

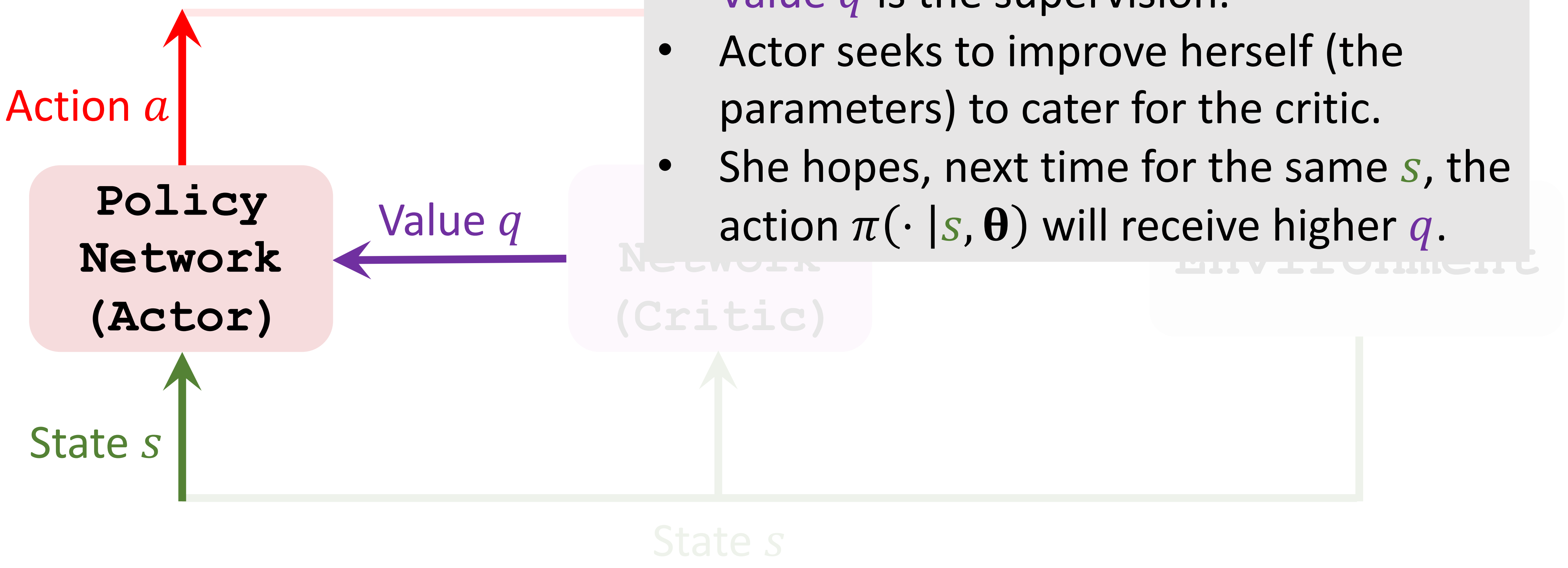
Actor Critic Method



Actor Critic Method

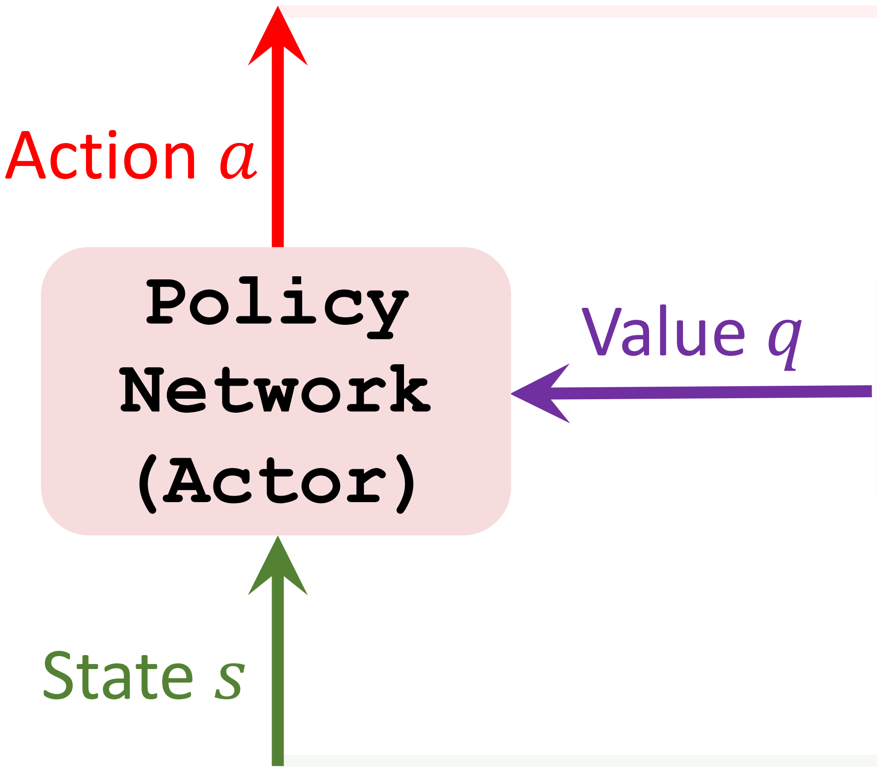


Actor Critic Method: Update Actor



- Value q is the supervision.
- Actor seeks to improve herself (the parameters) to cater for the critic.
- She hopes, next time for the same s, the action $\pi(\cdot\,|s, \boldsymbol{\theta})$ will receive higher q.

Actor Critic Method: Update Actor

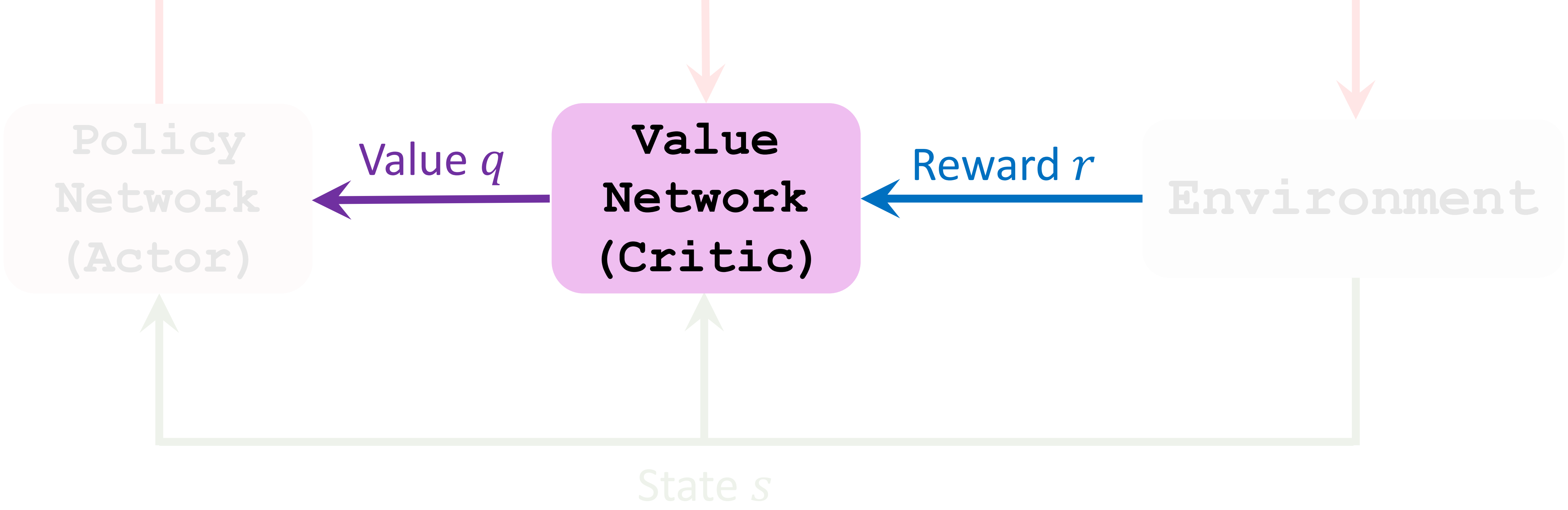


- Value q is the supervision.
- Actor seeks to improve herself (the parameters) to cater for the critic.
- She hopes, next time for the same s, the action $\pi(\cdot\,|s, \boldsymbol{\theta})$ will receive higher q.

- Compute policy gradient using s, a, q.
- Update the actor's parameters using "stochastic gradient ascent".

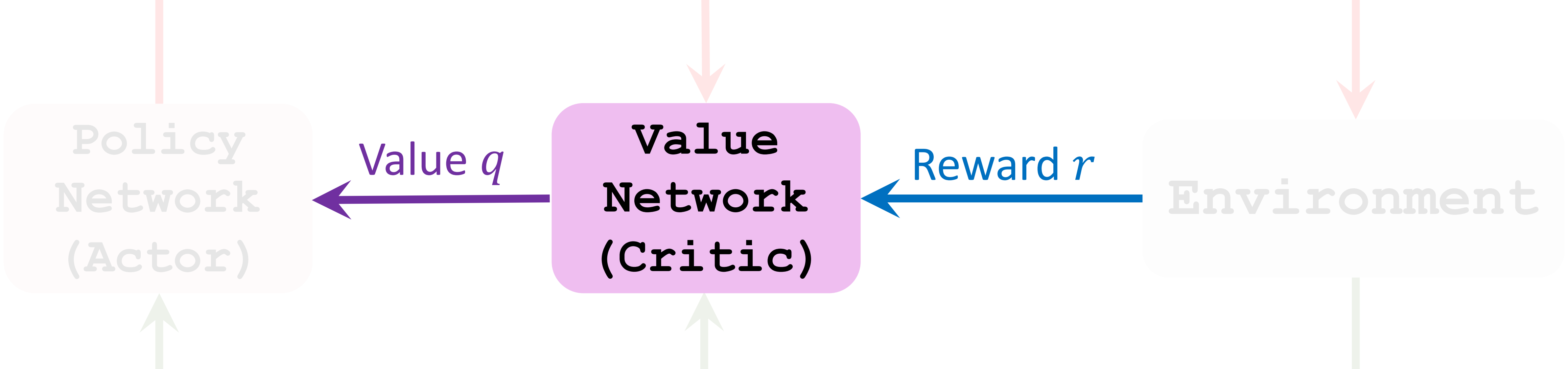
Actor Critic Method: Update Critic

- In the beginning, the critic makes random judgement.
- How to improve the critic?



Actor Critic Method: Update Critic

- In the beginning, the critic makes random judgement.
- How to improve the critic?



- Make use the fact that q_t should be close to $r_t + \gamma \cdot q_{t+1}$; if not, a loss.
- Compute q_t using (s_t, a_t) and q_{t+1} using (s_{t+1}, a_{t+1}).
- Update the critic's parameters using TD learning.

Summary of Algorithm

1. Observe the state s_t; randomly sample action a_t according to $\pi(\cdot \mid s_t; \boldsymbol{\theta}_t)$.
2. Perform a_t; observe new state s_{t+1} and reward r_t.
3. Randomly sample a_{t+1} according to $\pi(\cdot \mid s_{t+1}; \boldsymbol{\theta}_t)$. (Do not perform a_{t+1}.)
4. Evaluate value network: $q_t = q(s_t, a_t; \mathbf{w}_t)$ and $q_{t+1} = q(s_{t+1}, a_{t+1}; \mathbf{w}_t)$.
5. Compute the 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

Policy Gradient with Baseline

Definition: Approximated state-value function.

- $V(s;\boldsymbol{\theta}) - b = \sum_a \pi(a|s;\boldsymbol{\theta}) \cdot [Q_\pi(s,a) - b]$.
- Here, the baseline b must be independent of $\boldsymbol{\theta}$ and a.

Policy Gradient with Baseline

Definition: Approximated state-value function.

- $V(s;\boldsymbol{\theta}) - b = \sum_a \pi(a|s;\boldsymbol{\theta}) \cdot [Q_\pi(s,a) - b]$.
- Here, the baseline b must be independent of $\boldsymbol{\theta}$ and a.

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

- $\frac{\partial\, V(s;\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} = \frac{\partial\, [V(s;\boldsymbol{\theta}) - b]}{\partial\, \boldsymbol{\theta}} = \mathbb{E}_{a \sim \pi(\cdot|s;\boldsymbol{\theta})}\left[\frac{\partial \log \pi(a|s,\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} \cdot [Q_\pi(s,a) - b]\right]$.
- The baseline b does not affect correctness.
- A good baseline b can reduce variance.
- We can use $b = r_t + \gamma \cdot q_{t+1}$ (TD target) as the baseline.

Actor Critic Update (without baseline)

1. Observe the state s_t; randomly sample action a_t according to $\pi(\cdot \mid s_t; \boldsymbol{\theta}_t)$.
2. Perform a_t; observe new state s_{t+1} and reward r_t.
3. Randomly sample a_{t+1} according to $\pi(\cdot \mid s_{t+1}; \boldsymbol{\theta}_t)$. (Do not perform a_{t+1}.)
4. Evaluate value network: $q_t = q(s_t, a_t; \mathbf{w}_t)$ and $q_{t+1} = q(s_{t+1}, a_{t+1}; \mathbf{w}_t)$.
5. Compute the 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: $\mathbf{d}_{\theta,t} = \frac{\partial \log \pi(a_t \mid s_t, \boldsymbol{\theta})}{\partial \, \boldsymbol{\theta}} \big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t}$.
9. Update policy network: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \beta \cdot q_t \cdot \mathbf{d}_{\theta,t}$.

Actor Critic Update (with baseline)

1. Observe the state s_t; randomly sample action a_t according to $\pi(\cdot\,|s_t;\boldsymbol{\theta}_t)$.
2. Perform a_t; observe new state s_{t+1} and reward r_t.
3. Randomly sample a_{t+1} according to $\pi(\cdot\,|s_{t+1};\boldsymbol{\theta}_t)$. (Do not perform a_{t+1}.)
4. Evaluate value network: $q_t = q(s_t, a_t; \mathbf{w}_t)$ and $q_{t+1} = q(s_{t+1}, a_{t+1}; \mathbf{w}_t)$.
5. Compute the 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

Actor Critic Method

Definition: State-value function.

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

Definition: State-value function approximation using neural networks.

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

Actor Critic Method

- Value network (critic) is useful in training; it provides the policy network (actor) with supervision.

- After training, the value network (critic) will not be used.
- Agent is controlled by policy network (actor):

$$a_t \sim \pi(\cdot \mid s_t; \boldsymbol{\theta}).$$

Training Policy and Value Networks

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

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

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

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

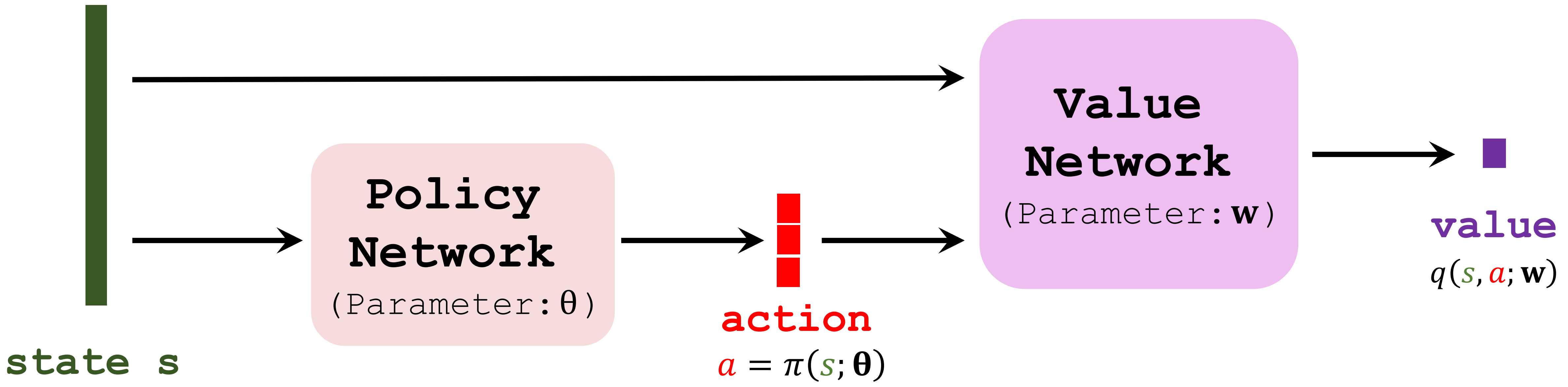
Deterministic Policy Gradient (DPG)

Reference

- Lillicrap and others: Continuous control with deep reinforcement learning. arXiv:1509.02971. 2015.

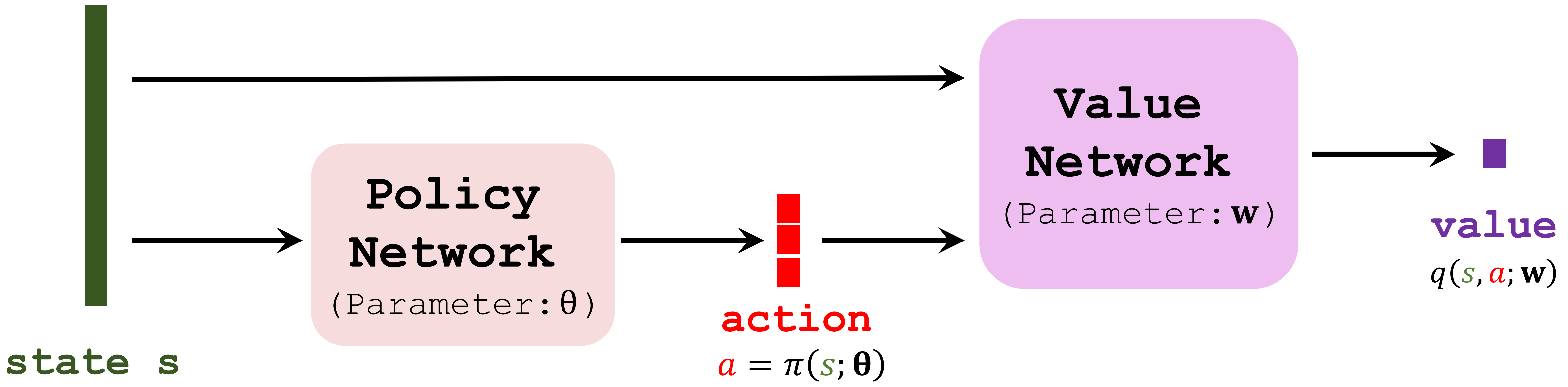
Deterministic Policy Gradient (DPG)

- DPG is a actor-critic method.
- The policy network is deterministic: $a = \pi(s; \boldsymbol{\theta})$.



Deterministic Policy Gradient (DPG)

- DPG is a actor-critic method.
- The policy network is deterministic: $a = \pi(s; \boldsymbol{\theta})$.
- Trained value network by TD learning.
- Train policy network to maximize the value $q(s, a; \mathbf{w})$.



Train Policy Network

- Train policy network to maximize the value $q(s, a; \mathbf{w})$.
- Gradient: $\frac{\partial\, q(s,a;\mathbf{w})}{\partial\, \boldsymbol{\theta}} = \frac{\partial\, a}{\partial\, \boldsymbol{\theta}} \cdot \frac{\partial\, q(s,a;\mathbf{w})}{\partial\, a}$.
- Update $\boldsymbol{\theta}$ using gradient ascent.