Reinforcement Learning

申思杰

Outline

- Reinforcement Learning Background (model-free methods)
  - Value-based: **SARSA, Q-learning, Deep Q Network**
  - Policy-based: **REINFORCE, Actor-Critic**
- Asynchronous RL Framework
  - Asynchronous SARSA, Asynchronous Q-learning
  - Asynchronous Advantage Actor-Critic (A3C)

Markov Decision Process

- **Markov Decision Process:**

Definition

A *Markov Decision Process* is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}^a_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s, A_t = a\right]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}^a_s = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$
- γ is a discount factor $\gamma \in [0, 1]$.

***David Silver. Reinforcement Learning. Lesson 2: Markov Decision Process**

Markov Decision Process

- **Policy:** A policy π is a distribution over actions given states

$$\pi(a \mid s) = \mathbb{P}[A_t = a \mid S_t = s]$$

- **Episode:**
  - At state S_t
  - Perform action A_t according to policy π
  - Get reward R_{t+1} and get to state S_{t+1}
  - Loop until terminate or until time-step T

$$S_1, A_1, R_2, S_2, A_2, R_3, S_3, \ldots$$

Markov Decision Process

- **Return:** Total discounted reward from time step t

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- **Value function:**
  - State-value function: $V_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$
  - Action-value function: $Q_\pi(s, a) = \mathbb{E}[G_t \mid S_t = s, A_t = a]$

Markov Decision Process

- Value function can be decomposed into immediate reward plus discounted value of successor state

$$V_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma V_\pi(S_{t+1}) \,|\, S_t = s]$$

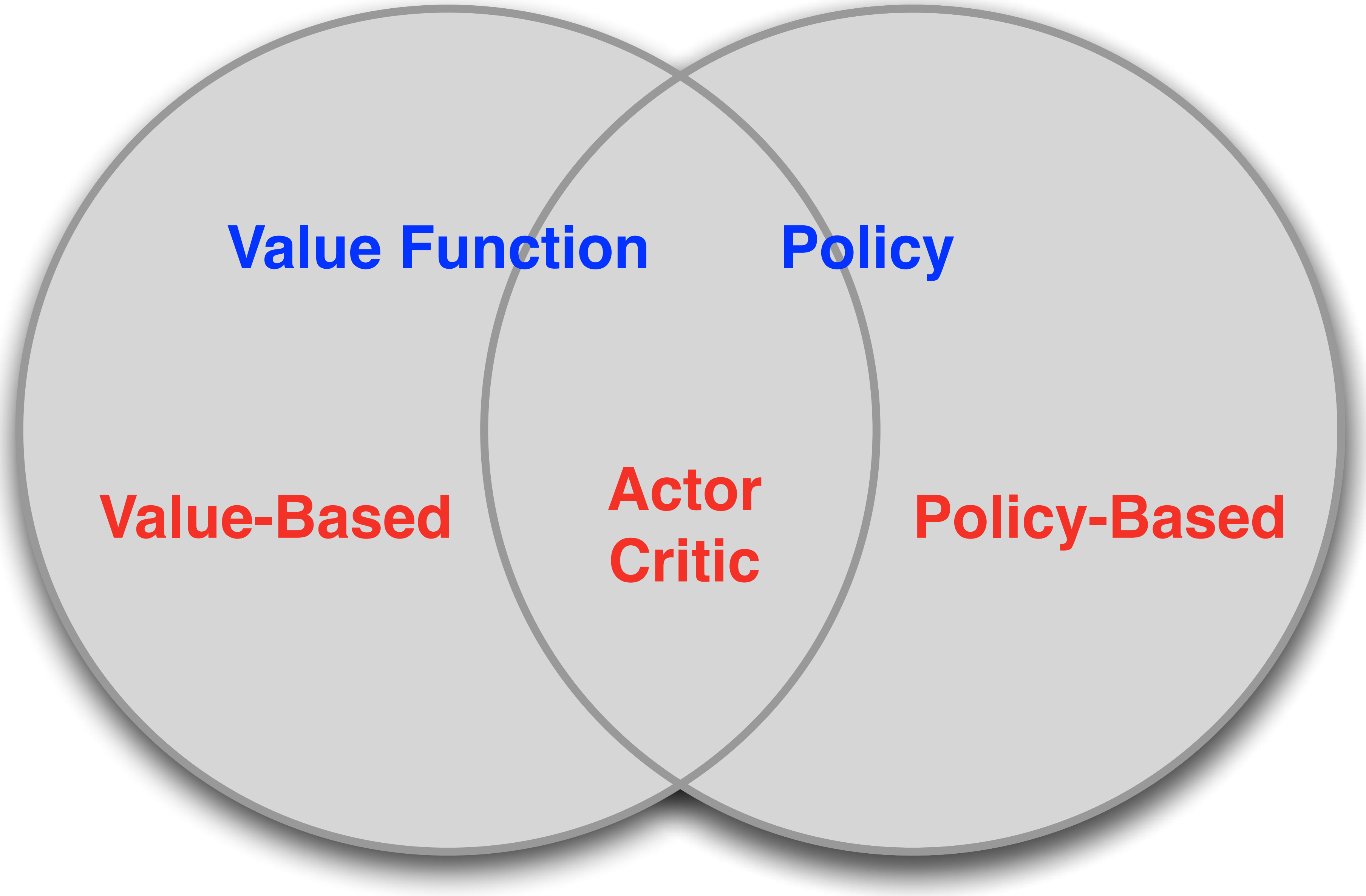
$$Q_\pi(s, a) = \mathbb{E}[R_{t+1} + \gamma Q_\pi(S_{t+1}, A_{t+1}) \,|\, S_t = s, A_t = a]$$

- **Optimal value function:**

$$V^*(s) = \max_\pi V_\pi(s) \qquad Q^*(s, a) = \max_\pi Q_\pi(s, a)$$

An optimal policy can be found by maximizing over $Q^*(s, a)$

Model-Free Reinforcement Learning Methods



Value-based Methods

- Directly approximate the optimal action value function $Q^*(s, a)$
    - e.g. Using a table to store the approximation of $Q^*(s, a)$
    - e.g. Using neural network $Q(s, a; \theta)$ with parameters θ to fit $Q^*(s, a)$
- Policy can be derived from the learned $Q(s, a; \theta)$
    - e.g. ε-greedy policy

Value-based Methods

- Monte-Carlo control:

  Policy evaluation Monte-Carlo policy evaluation, $Q \approx q_\pi$

  Policy improvement ϵ-greedy policy improvement

- On-policy and off-policy learning:
  - On-policy learning
    - "Learn on the job"
    - Learn about policy π from experience sampled from π
  - Off-policy learning
    - "Look over someone's shoulder"
    - Learn about policy π from experience sampled from μ

***David Silver. Reinforcement Learning. Lesson 5: Model-Free Control**

Value-based Methods

- SARSA for on-policy control:

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Repeat (for each step of episode):
        Take action A, observe R, S'
        Choose A' from S' using policy derived from Q (e.g., ε-greedy)
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
        $S \leftarrow S'$; $A \leftarrow A'$;
    until S is terminal

***David Silver. Reinforcement Learning. Lesson 5: Model-Free Control**

Value-based Methods

- Q-learning for off-policy control

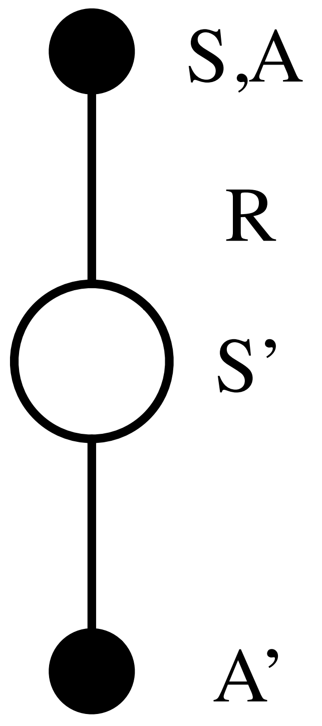
Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Repeat (for each step of episode):
        Choose A from S using policy derived from Q (e.g., ε-greedy)
        Take action A, observe R, S'
        $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
        $S \leftarrow S'$;
    until S is terminal

*David Silver. Reinforcement Learning. Lesson 5: Model-Free Control

Value-based Methods

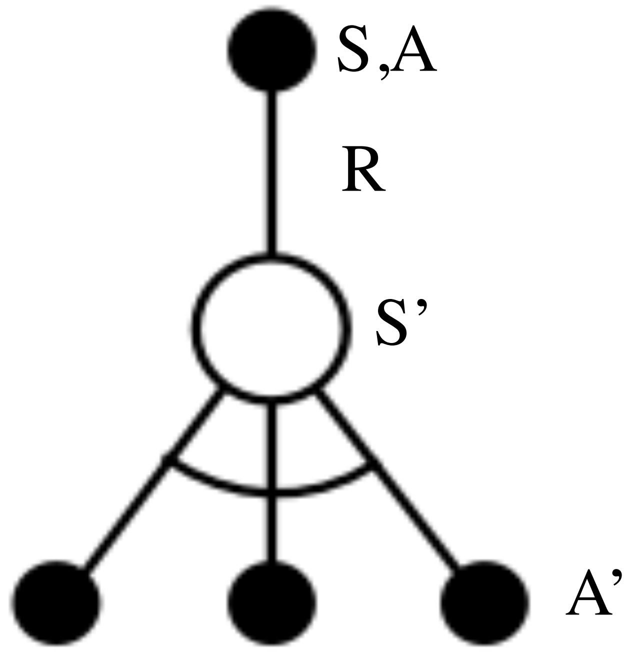
- **SARSA:**
  - Policy: ε-greedy

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left(R + \gamma Q(s',a')\right)$$



- **Q-learning:**
  - Target policy: greedy; Behavior policy: ε-greedy

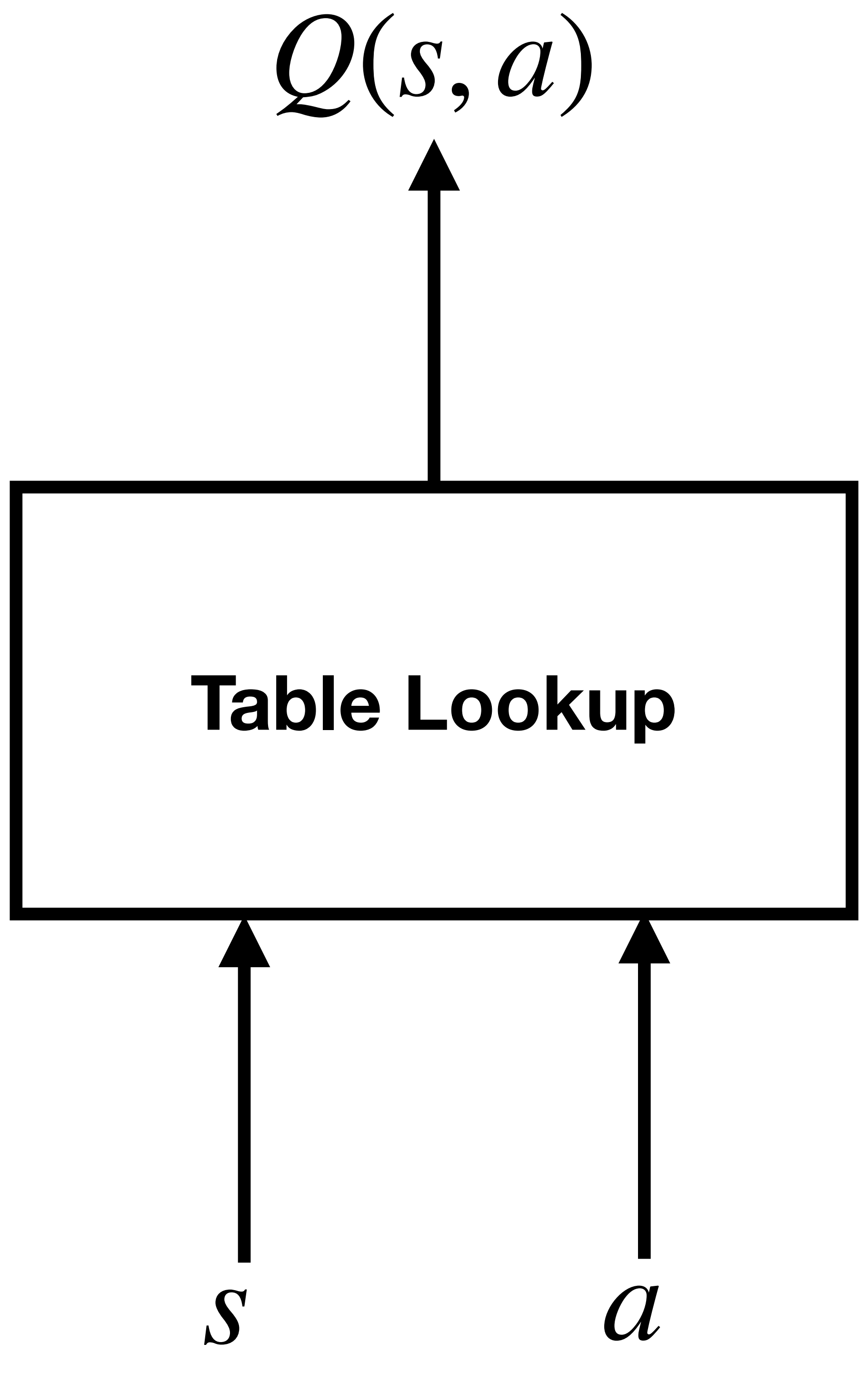
$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left(R + \gamma \max_{a'} Q(s',a')\right)$$



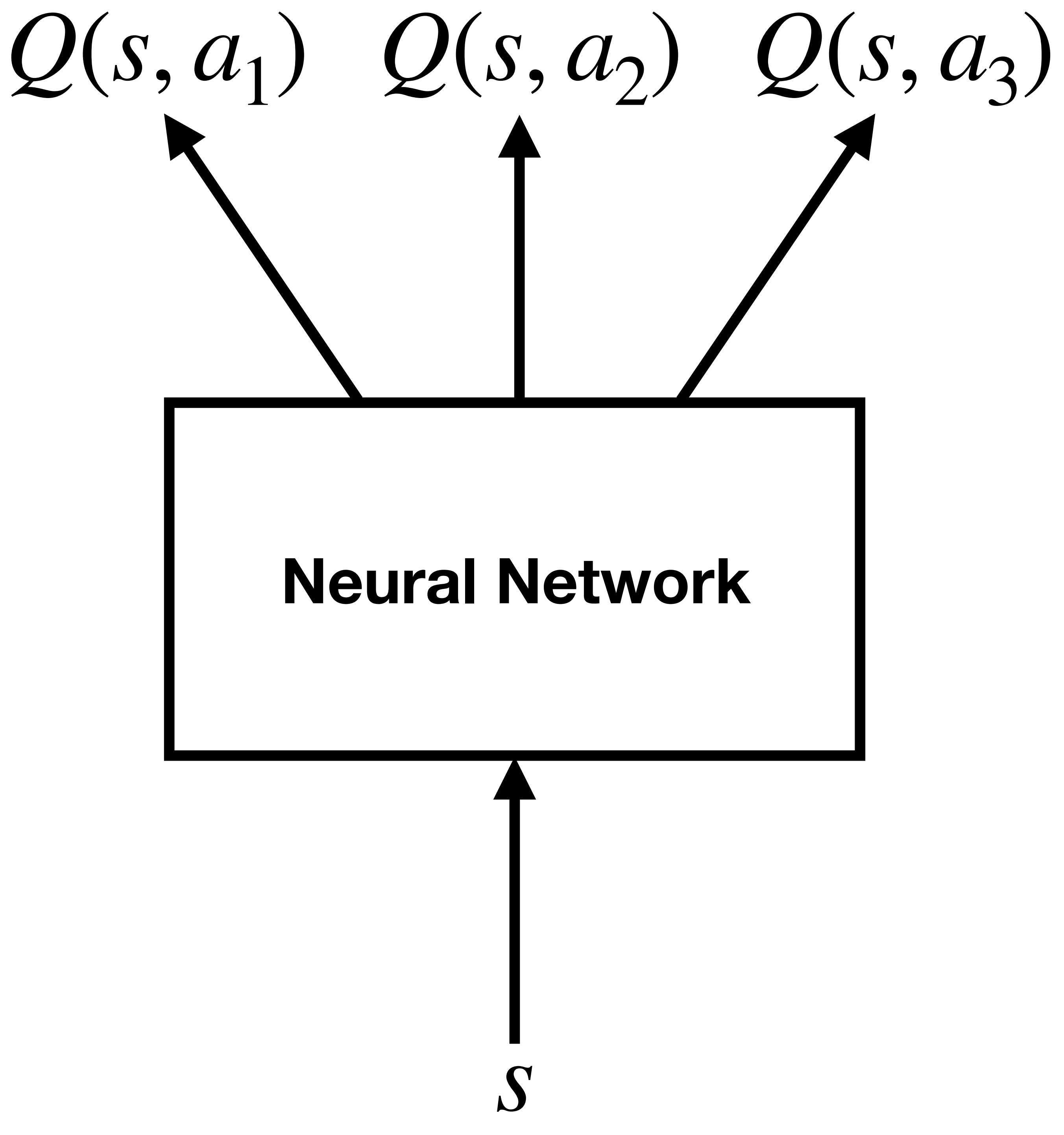
***David Silver. Reinforcement Learning. Lesson 5: Model-Free Control**

Value-based Methods

- Deep Q Network (DQN):



Tradition Methods



Deep Methods

Value-based Methods

DQN uses experience replay and fixed Q-targets

- Take action a_t according to ϵ-greedy policy
- Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in replay memory $\mathcal{D}$
- Sample random mini-batch of transitions (s, a, r, s') from $\mathcal{D}$
- Compute Q-learning targets w.r.t. old, fixed parameters w^-
- Optimise MSE between Q-network and Q-learning targets

$$\mathcal{L}_i(w_i) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}_i}\left[\left(r + \gamma \max_{a'} Q(s', a'; w_i^-) - Q(s, a; w_i)\right)^2\right]$$

- Using variant of stochastic gradient descent

*David Silver. Reinforcement Learning. Lesson 5: Model-Free Control

Policy-based Methods

- Directly parameterize the policy $\pi_\theta(a \mid s)$, find the best θ
- Measure the quality of policy: policy objective functions
  - Start value (episodic): $J_1(\theta) = V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$
  - Average value (continuing): $J_{avV}(\theta) = \sum_s d^{\pi_\theta}(s) V^{\pi_\theta}(s)$
  - Average reward per time-step (continuing): $J_{avR}(\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s, a) \mathcal{R}_s^a$

d^{π_θ} is stationary distribution of Markov chain for π_θ

Policy-based Methods

- Policy gradient to update parameters:

$$\Delta\theta = \alpha \nabla_\theta J(\theta)$$

$\nabla_\theta J(\theta)$ is called policy gradient

- Compute policy gradient: Likelihood ratios:

$$\begin{aligned}\nabla_\theta \pi_\theta(s, a) &= \pi_\theta(s, a) \frac{\nabla_\theta \pi_\theta(s, a)}{\pi_\theta(s, a)} \\ &= \pi_\theta(s, a) \nabla_\theta \log \pi_\theta(s, a)\end{aligned}$$

Policy-based Methods

- An example: Gradient of one-step MDPs:

$$
\begin{aligned}
J(\theta) &= \mathbb{E}_{\pi_\theta}\left[r\right] \\
&= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi_\theta(s, a) \mathcal{R}_{s,a} \\
\nabla_\theta J(\theta) &= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi_\theta(s, a) \nabla_\theta \log \pi_\theta(s, a) \mathcal{R}_{s,a} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s, a) r\right]
\end{aligned}
$$

***David Silver. Reinforcement Learning. Lesson 7: Policy Gradient**

Policy-based Methods

- Policy Gradient Theorem:

Theorem

For any differentiable policy $\pi_\theta(s, a)$,
for any of the policy objective functions $J = J_1, J_{avR},$ *or* $\frac{1}{1-\gamma} J_{avV}$,
the policy gradient is

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(s, a) \; Q^{\pi_\theta}(s, a) \right]$$

*David Silver. Reinforcement Learning. Lesson 7: Policy Gradient

Policy-based Methods

- REINFORCE (Monte-Carlo Policy Gradient):
  - Using return v_t as a sample of $Q^{\pi_\theta}(s_t, a_t)$

function REINFORCE
    Initialise θ arbitrarily
    for each episode $\{s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**
        for $t = 1$ to $T - 1$ **do**
            $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$
        end for
    end for
    return θ
end function

*David Silver. Reinforcement Learning. Lesson 7: Policy Gradient**

Policy-based Methods

- Reduce variance using a critic:
  - We use a critic to estimate the action-value function,

$$Q_w(s, a) \approx Q^{\pi_\theta}(s, a)$$

  - Actor-critic algorithms maintain *two* sets of parameters

    Critic Updates action-value function parameters w

    Actor Updates policy parameters θ, in direction suggested by critic

  - Actor-critic algorithms follow an *approximate* policy gradient

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s, a)\ Q_w(s, a)\right]$$
$$\Delta\theta = \alpha \nabla_\theta \log \pi_\theta(s, a)\ Q_w(s, a)$$

***David Silver. Reinforcement Learning. Lesson 7: Policy Gradient**

Policy-based Methods

- Actor-Critic:
  - Using linear value function to approximate $Q_w(s,a) = \phi(s,a)^\mathsf{T} w$

```
function QAC
    Initialise s, θ
    Sample a ~ π_θ
    for each step do
        Sample reward r = R_s^a; sample transition s' ~ P_{s,·}^a
        Sample action a' ~ π_θ(s', a')
        δ = r + γQ_w(s', a') − Q_w(s, a)
        θ = θ + α∇_θ log π_θ(s, a)Q_w(s, a)
        w ← w + βδφ(s, a)
        a ← a', s ← s'
    end for
end function
```

*David Silver. Reinforcement Learning. Lesson 7: Policy Gradient

Policy-based Methods

- Reduce variance using a baseline:

$$\begin{aligned}\mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)B(s)\right] &= \sum_{s\in\mathcal{S}} d^{\pi_\theta}(s)\sum_a \nabla_\theta \pi_\theta(s,a)B(s)\\ &= \sum_{s\in\mathcal{S}} d^{\pi_\theta} B(s)\nabla_\theta \sum_{a\in\mathcal{A}} \pi_\theta(s,a)\\ &= 0\end{aligned}$$

- **Advantage Actor-Critic:** Using state-value function as a baseline:
  - Advantage function: $A^{\pi_\theta}(s,a)$

$$A^{\pi_\theta}(s,a) = Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)$$
$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ A^{\pi_\theta}(s,a)\right]$$

***David Silver. Reinforcement Learning. Lesson 7: Policy Gradient**

Outline

- Reinforcement Learning Background (model-free methods)
    - Value-based: **SARSA, Q-learning, Deep Q Network**
    - Policy-based: **REINFORCE, Actor-Critic**
- Asynchronous RL Framework
    - Asynchronous SARSA, Asynchronous Q-learning
    - Asynchronous Advantage Actor-Critic (A3C)

Asynchronous Methods for Deep Reinforcement Learning

Volodymyr Mnih[1] VMNIH@GOOGLE.COM
Adrià Puigdomènech Badia[1] ADRIAP@GOOGLE.COM
Mehdi Mirza[1,2] MIRZAMOM@IRO.UMONTREAL.CA
Alex Graves[1] GRAVESA@GOOGLE.COM
Tim Harley[1] THARLEY@GOOGLE.COM
Timothy P. Lillicrap[1] COUNTZERO@GOOGLE.COM
David Silver[1] DAVIDSILVER@GOOGLE.COM
Koray Kavukcuoglu [1] KORAYK@GOOGLE.COM
[1] Google DeepMind
[2] Montreal Institute for Learning Algorithms (MILA), University of Montreal