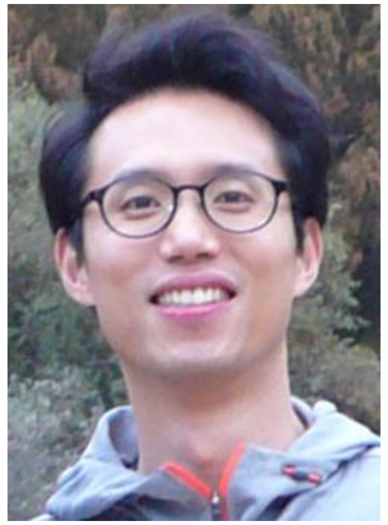
Pattern Recognition and
Machine Intelligence

HANKUK UNIVERSITY OF FOREIGN STUDIES

Introduction To Deep Reinforcement Learning - Part II

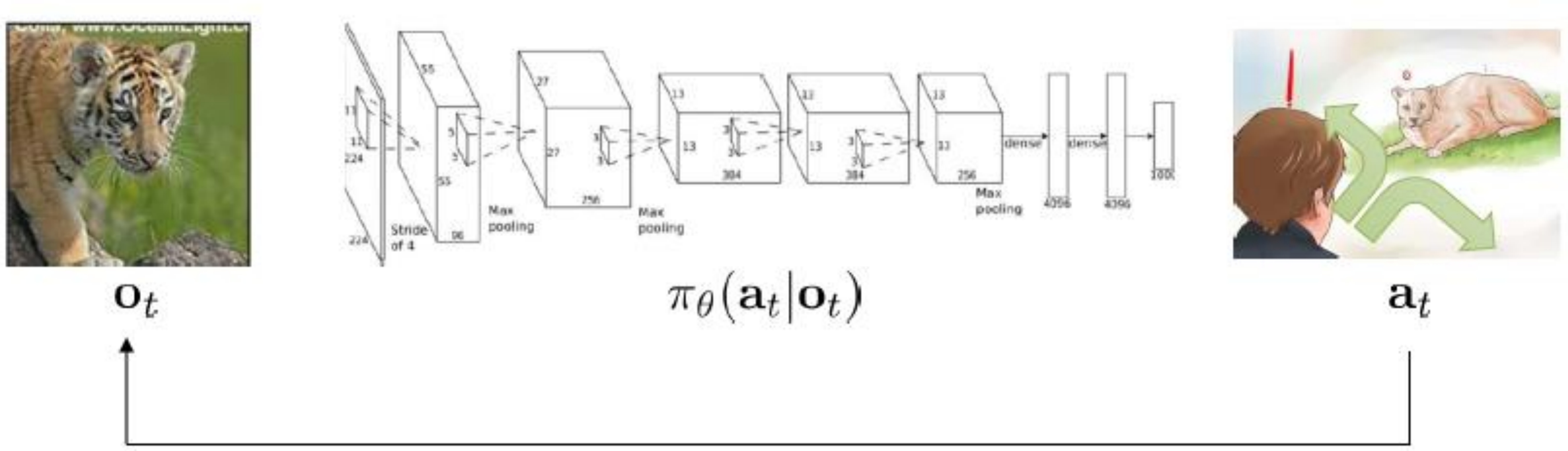


Prof. Jae Young Choi

Pattern Recognition and Machine Intelligence Lab. (PMI)

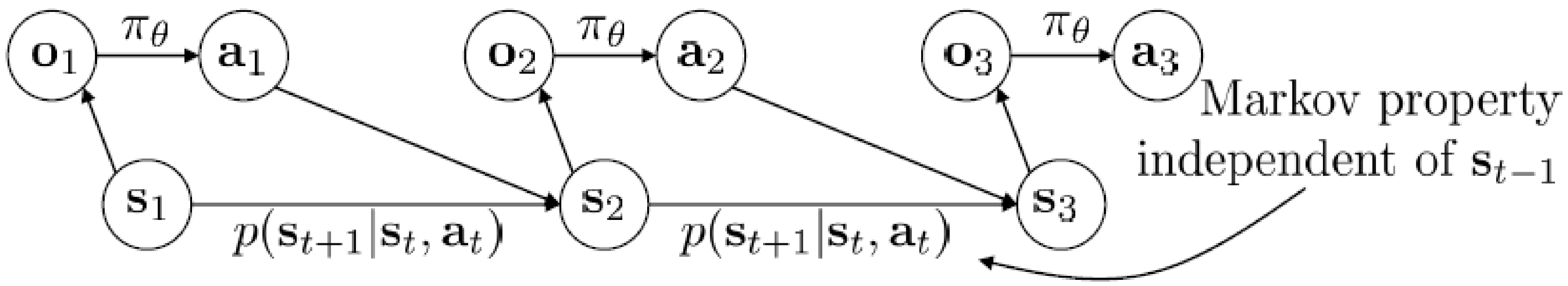
Hankuk University of Foreign Studies

❖ Terminology & Notation



$\mathbf{s}_t$ – state
$\mathbf{o}_t$ – observation
$\mathbf{a}_t$ – action

$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ – policy
$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ – policy (fully observed)



❖ Markov Chain

Markov chain

$\mathcal{M} = \{\mathcal{S}, \mathcal{T}\}$

$\mathcal{S}$ – state space states $s \in \mathcal{S}$ (discrete or continuous)

$\mathcal{T}$ – transition operator $p(s_{t+1}|s_t)$

why "operator"? let $\mu_{t,i} = p(s_t = i)$ $\vec{\mu}_t$ is a vector of probabilities

let $\mathcal{T}_{i,j} = p(s_{t+1} = i | s_t = j)$ then $\vec{\mu}_{t+1} = \mathcal{T}\vec{\mu}_t$

Andrey Markov

$$\mathbf{s}_1 \xrightarrow[p(\mathbf{s}_{t+1}|\mathbf{s}_t)]{} \mathbf{s}_2 \xrightarrow[p(\mathbf{s}_{t+1}|\mathbf{s}_t)]{} \mathbf{s}_3$$

Markov property
independent of $\mathbf{s}_{t-1}$

❖ Markov Decision Process

Markov decision process $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, r\}$

$\mathcal{S}$ – state space   states $s \in \mathcal{S}$ (discrete or continuous)

$\mathcal{A}$ – action space   actions $a \in \mathcal{A}$ (discrete or continuous)

$\mathcal{T}$ – transition operator (now a ter

r – reward function $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$

$r(s_t, a_t)$ – reward

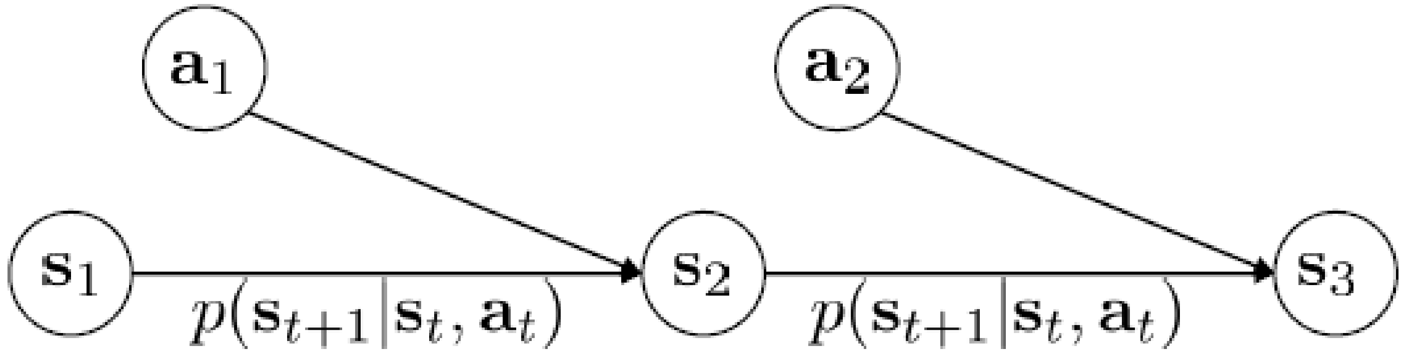
r – reward function $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$

$r(s_t, a_t)$ – reward

$$\mu_{t+1,i} = \sum_{j,k} \mathcal{T}_{i,j,k} \mu_{t,j} \xi_{t,k}$$

let $\mathcal{T}_{i,j,k} = p(s_{t+1} = i | s_t = j, a_t = k)$

Richard Bellman



❖ Partially Observed Markov Decision Process

partially observed Markov decision process $\qquad \mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{E}, r\}$

$\mathcal{S}$ – state space $\qquad$ states $s \in \mathcal{S}$ (discrete or continuous)

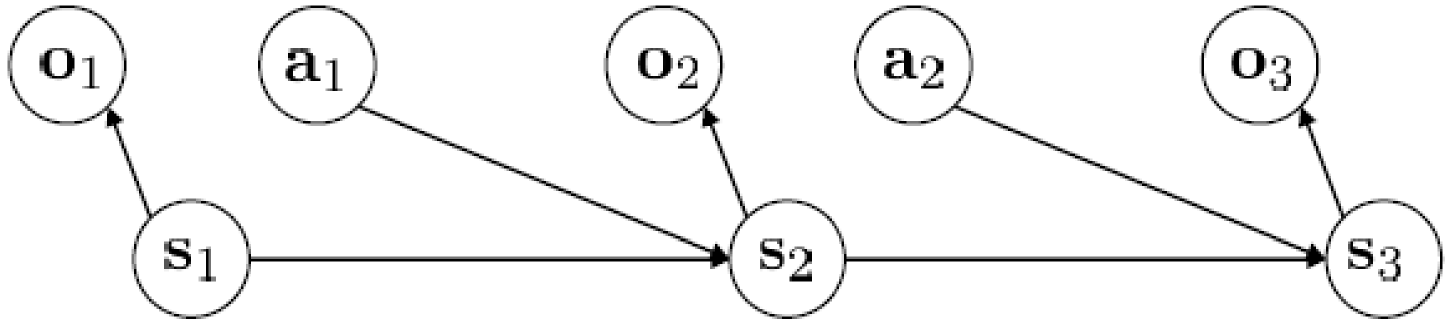
$\mathcal{A}$ – action space $\qquad$ actions $a \in \mathcal{A}$ (discrete or continuous)

$\mathcal{O}$ – observation space $\qquad$ observations $o \in \mathcal{O}$ (discrete or continuous)

$\mathcal{T}$ – transition operator (like before)

$\mathcal{E}$ – emission probability $p(o_t|s_t)$

r – reward function $\qquad r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$



Deep Reinforcement Learning

❖ Expected Discounted Return

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

γ is a parameter, $0 \leq \gamma \leq 1$, called the *discount rate*

- The agent tries to select actions to maximize the G_t
- γ determines the present value of future rewarsd

Deep Reinforcement Learning

❖ Returns at successive time steps

$$
\begin{aligned}
G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\
&= R_{t+1} + \gamma\left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots\right) \\
&= R_{t+1} + \gamma G_{t+1}
\end{aligned}
$$

- This is very important for the theory and algorithms of deep reinforcement learning

- $\underbrace{\text{**Current** (state) return}}_{G_t}$ = $\underbrace{\text{**immediate** reward}}_{R_{t+1}}$ + $\underbrace{\gamma \text{ **future** (state) return}}_{\gamma G_{t+1}}$

Deep Reinforcement Learning

❖ Bellman equation for state-value function with policy π

State-value function $v_\pi(s)$

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right], \text{ for all } s \in \mathcal{S}$$

Bellman equation for $v_\pi(s)$

$$G_t = R_{t+1} + \gamma G_{t+1}$$

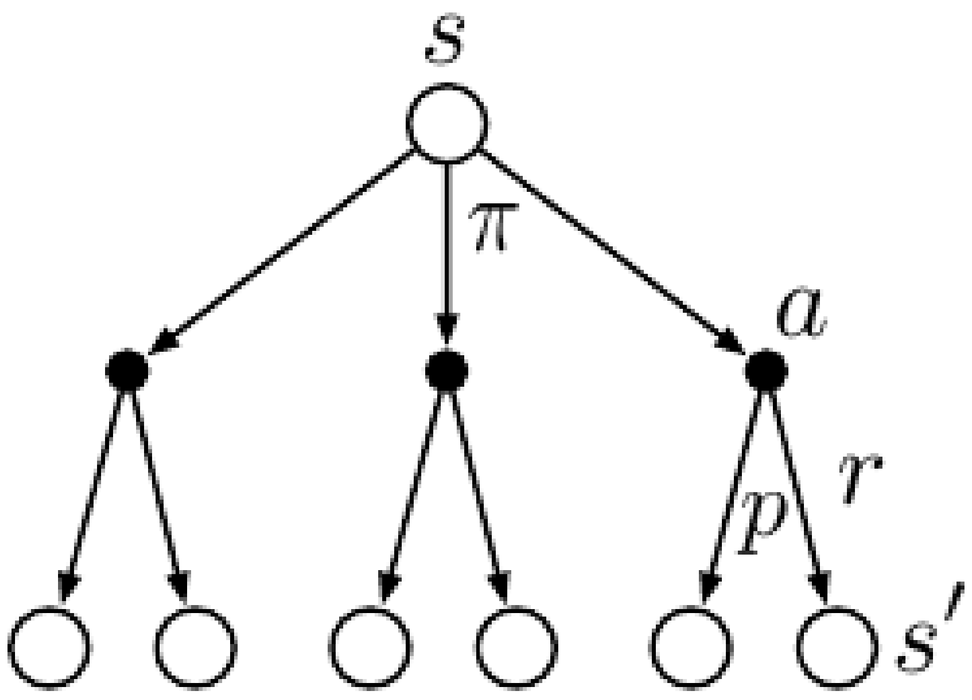
$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r | s, a)\Big[r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s']\Big] \\ &= \sum_a \pi(a|s) \sum_{s', r} p(s', r | s, a)\Big[r + \gamma v_\pi(s')\Big], \quad \text{for all } s \in \mathcal{S} \end{aligned}$$

Deep Reinforcement Learning

❖ Interpretation of Bellman equation

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)\Big[r + \gamma \mathbb{E}_\pi[G_{t+1}|S_{t+1} = s']\Big] \\
&= \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a)\Big[r + \gamma v_\pi(s')\Big], \quad \text{for all } s \in \mathcal{S}
\end{aligned}
$$

- Express a relationship between the value of a state and the values of its successor states
  - State value must equal the (discounted) value of the expected next state plus the reward expected along the way



Backup diagram for v_π

Deep Reinforcement Learning

❖ Bellman equation for action-value function

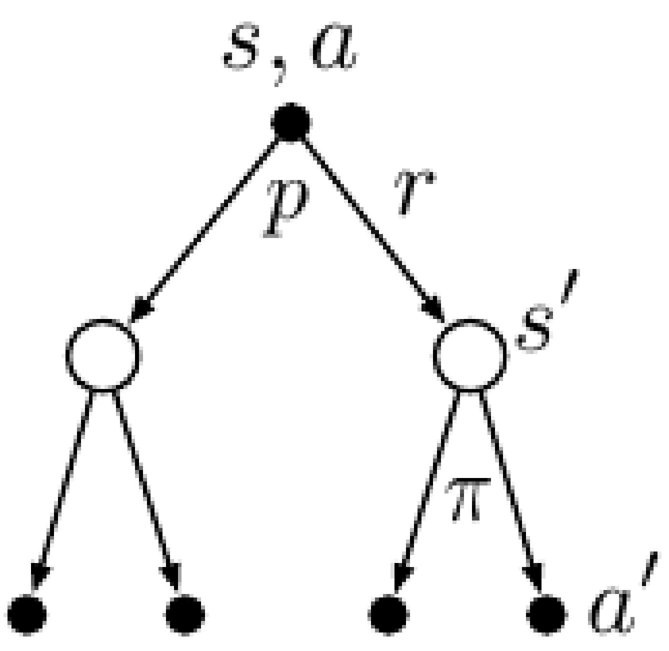
$G_t = R_{t+1} + \gamma G_{t+1}$

$S_{t+1} = s',\ A_{t+1} = a'$

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right]$$

$$= \mathrm{E}_\pi[R_{t+1} + \gamma q_\pi(s',a') \mid S_t = s, A_t = a]$$

$$= \sum_{s',r} P(s',r \mid s,a)\left[r + \gamma q_\pi(s',a')\right]$$



q_π backup diagram

❖ Optimal Policies

- Finding a policy that achieves a lot of rewards over the long run

- What is a better policy?

$$\pi \geq \pi' \text{ if and only if } v_\pi(s) \geq v_{\pi'}(s) \text{ for all } s \in \mathcal{S}$$

- There is always at least one policy that is better than or equal to all other policies

→ Optimal Policy

❖ Optimal Value Functions

- Optimal state-value function

$$v_*(s) \doteq \max_\pi v_\pi(s), \quad \text{for all } s \in \mathcal{S}$$

- Optimal action-value function

$$q_*(s,a) \doteq \max_\pi q_\pi(s,a), \text{ for all } s \in \mathcal{S} \text{ and } a \in \mathcal{A}(s).$$

$$q_*(s,a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a]$$

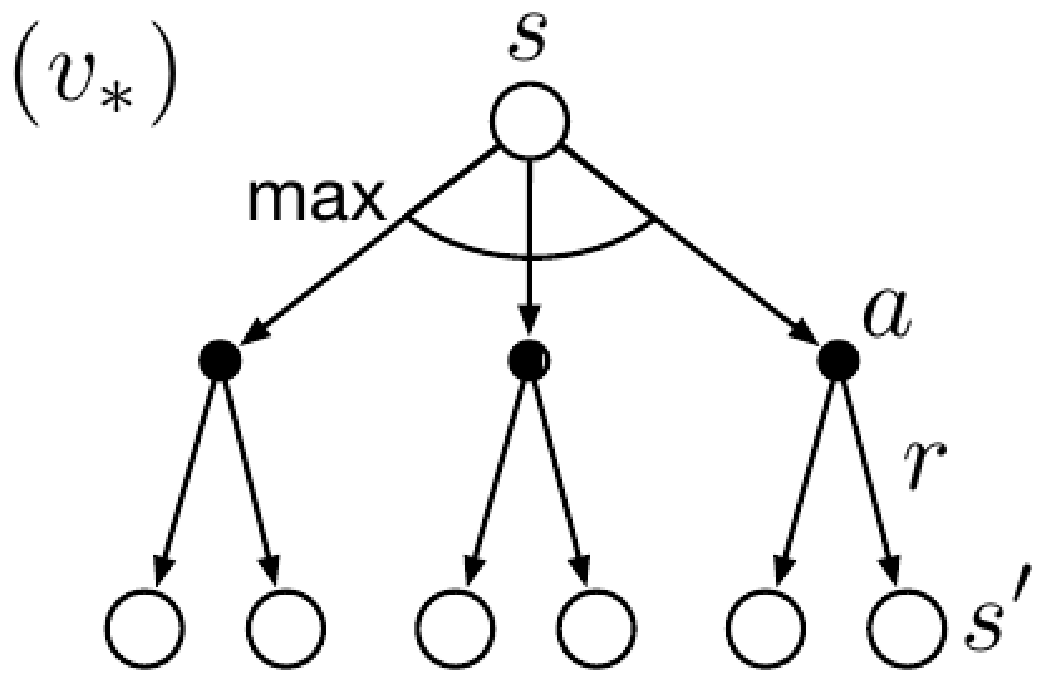
Expected return for taking action *a* in state *s* and thereafter following an optimal policy

Deep Reinforcement Learning

❖ Bellman Optimality Equation

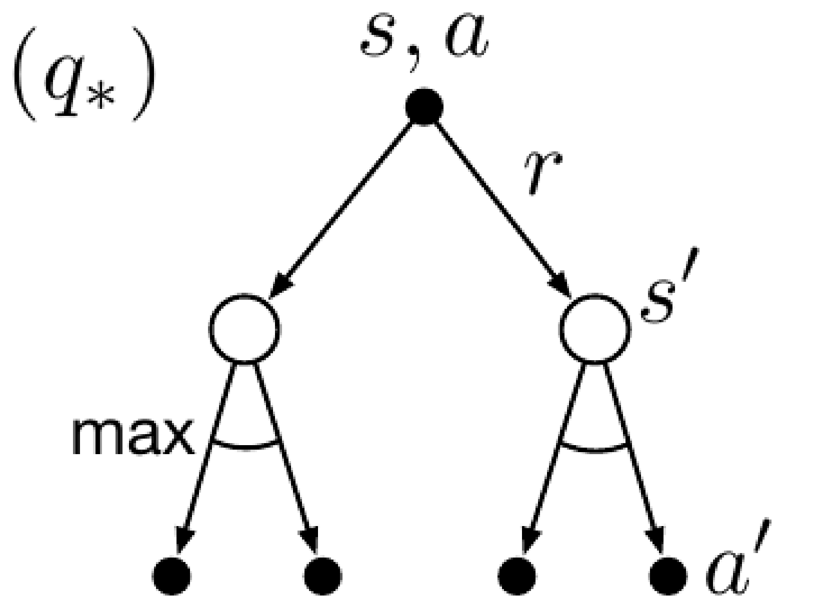
- Bellman optimal equation for v_*

$$\begin{aligned} v_*(s) &= \max_{a\in\mathcal{A}(s)} q_{\pi_*}(s,a) \\ &= \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t=s, A_t=a] \\ &= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} \mid S_t=s, A_t=a] \\ &= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t=s, A_t=a] \\ &= \max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma v_*(s')\big]. \end{aligned}$$



- Bellman optimal equation for q_*

$$\begin{aligned} q_*(s,a) &= \mathbb{E}\Big[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1},a') \;\Big|\; S_t = s, A_t = a\Big] \\ &= \sum_{s',r} p(s',r|s,a)\Big[r + \gamma \max_{a'} q_*(s',a')\Big]. \end{aligned}$$



Deep Reinforcement Learning

❖ Bellman Optimality Equation

- Interpreting Bellman optimal equation

$$\begin{aligned} v_*(s) &= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma v_*(s')\Big] \end{aligned}$$

$$\begin{aligned} q_*(s,a) &= \mathbb{E}\Big[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \Bigm| S_t = s, A_t = a\Big] \\ &= \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma \max_{a'} q_*(s', a')\Big], \end{aligned}$$

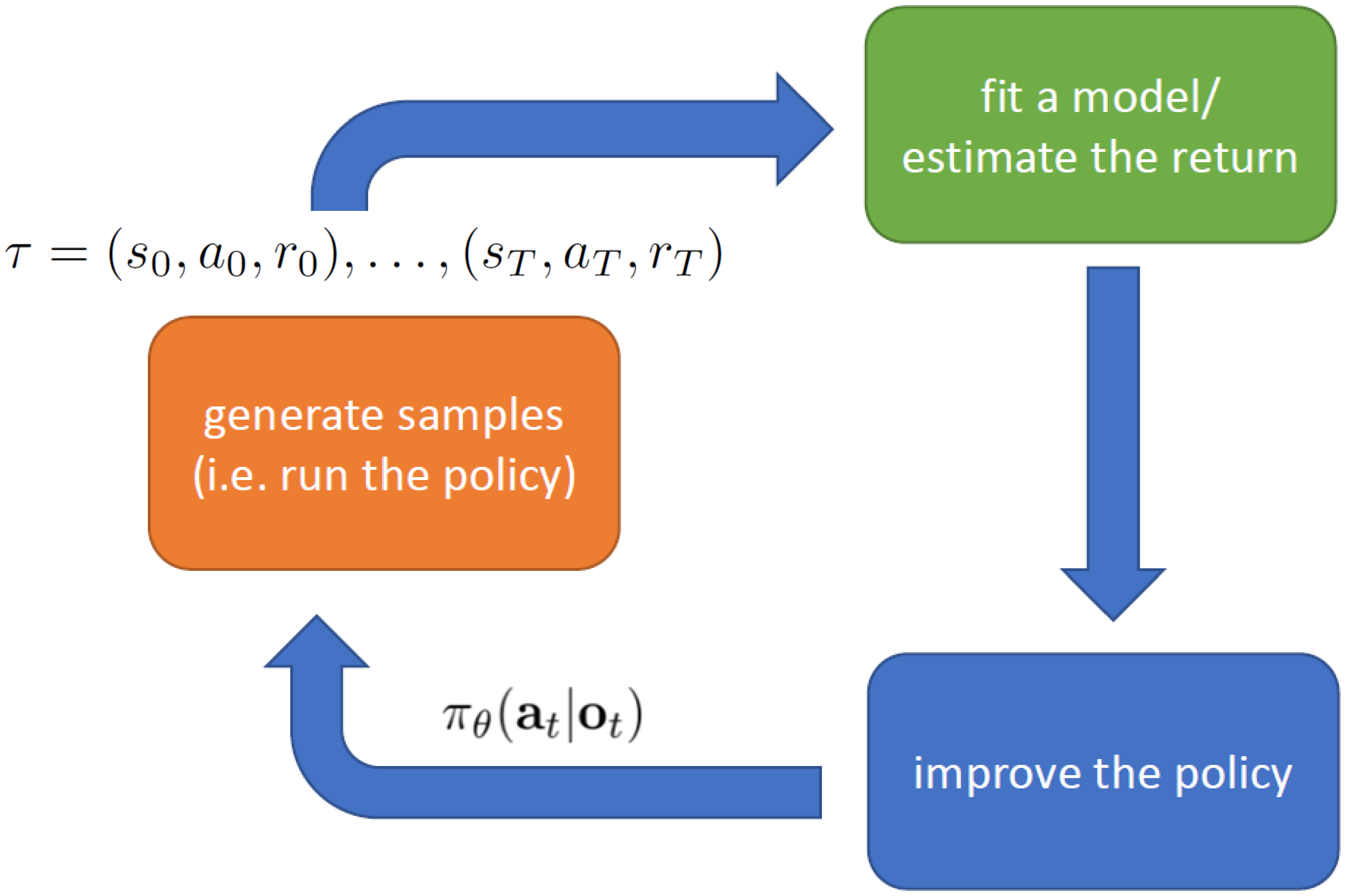
The value of a state under an optimal policy must equal the expected return for the best action from that state

The action that is the best after one-step search will be optimal action

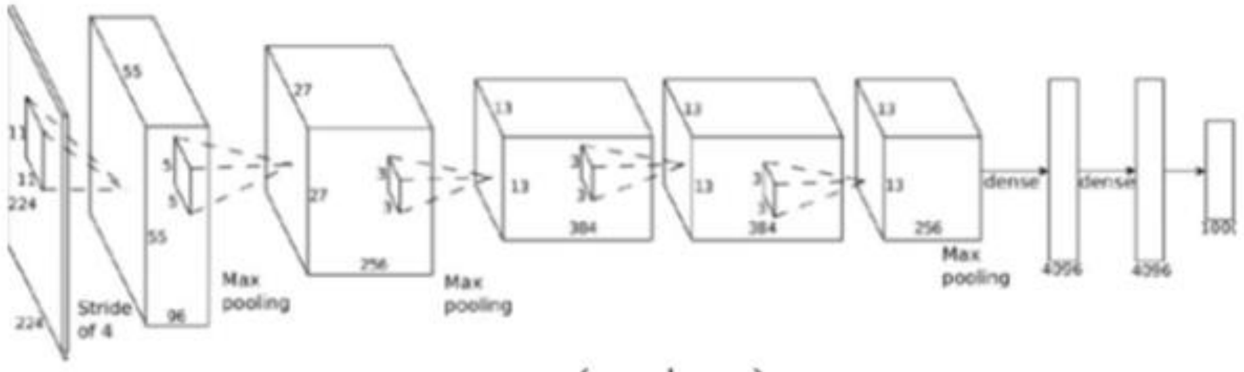
The action-value function effectively catches the result of all one-step ahead searches

Deep Learning for Reinforcement Learning

❖ The anatomy of deep reinforcement learning algorithm



$$J(\tau) = \mathbb{E}_{\tau\sim\pi}[R(\tau)] = \mathbb{E}_\tau\left[\sum_{t=0}^{T}\gamma^t r_t\right]$$

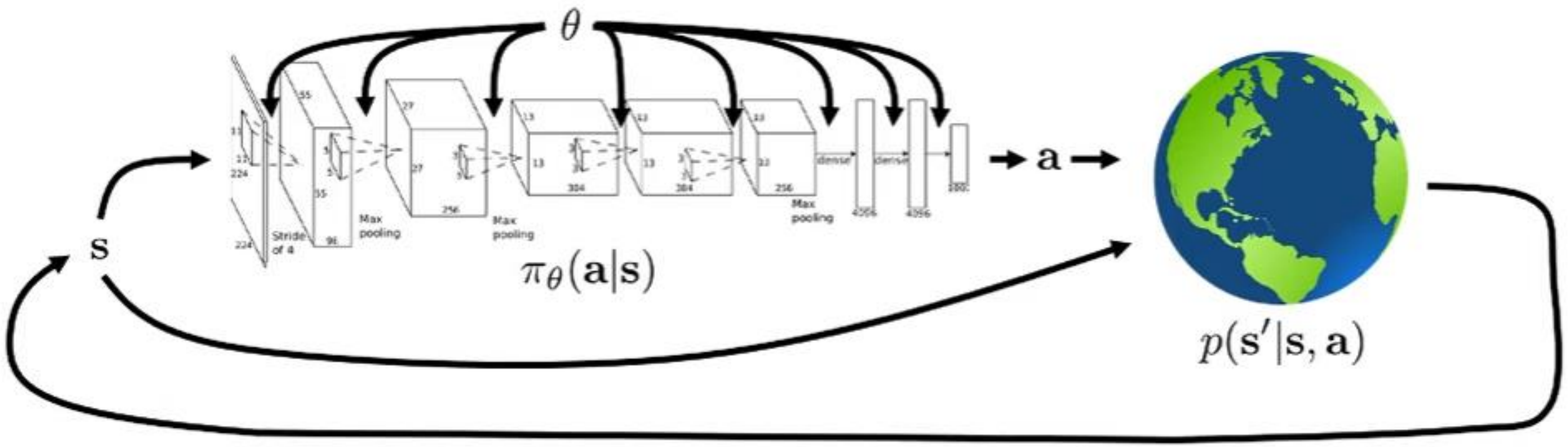


$$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$$

$$\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta)$$

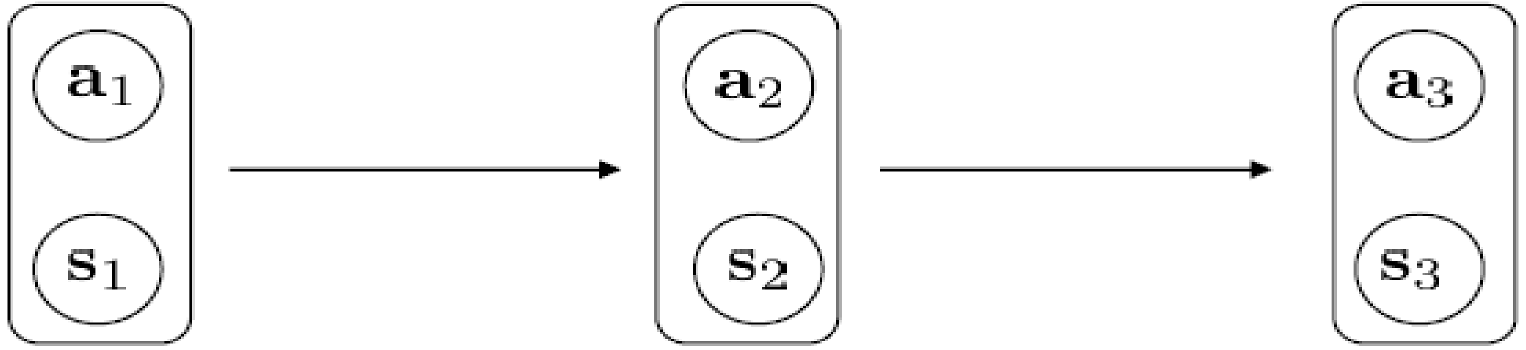
backprop through $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ to train $\pi_\theta(\mathbf{s}_t) = \mathbf{a}_t$

The Goal of Deep Reinforcement Learning



$$\underbrace{p_\theta(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T)}_{p_\theta(\tau)} = p(\mathbf{s}_1) \prod_{t=1}^{T} \underbrace{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t) p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)}_{\text{Markov chain on } (\mathbf{s}, \mathbf{a})}$$

$$p((\mathbf{s}_{t+1}, \mathbf{a}_{t+1})|(\mathbf{s}_t, \mathbf{a}_t)) = p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\pi_\theta(\mathbf{a}_{t+1}|\mathbf{s}_{t+1})$$



Value Function

❖ How do we deal with all these expectations?

$$E_{\tau \sim p_\theta(\tau)}\left[\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t)\right]$$

$$E_{\mathbf{s}_1 \sim p(\mathbf{s}_1)}\left[E_{\mathbf{a}_1 \sim \pi(\mathbf{a}_1|\mathbf{s}_1)}\left[r(\mathbf{s}_1, \mathbf{a}_1) + \underbrace{E_{\mathbf{s}_2 \sim p(\mathbf{s}_2|\mathbf{s}_1,\mathbf{a}_1)}\left[E_{\mathbf{a}_2 \sim \pi(\mathbf{a}_2|\mathbf{s}_2)}\left[r(\mathbf{s}_2, \mathbf{a}_2) + \ldots|\mathbf{s}_2\right]|\mathbf{s}_1, \mathbf{a}_1\right]}_{\text{what if we knew this part?}} \, |\mathbf{s}_1\right]\right]$$

$$Q(\mathbf{s}_1, \mathbf{a}_1) = r(\mathbf{s}_1, \mathbf{a}_1) + E_{\mathbf{s}_2 \sim p(\mathbf{s}_2|\mathbf{s}_1,\mathbf{a}_1)}\left[E_{\mathbf{a}_2 \sim \pi(\mathbf{a}_2|\mathbf{s}_2)}\left[r(\mathbf{s}_2, \mathbf{a}_2) + \ldots|\mathbf{s}_2\right]|\mathbf{s}_1, \mathbf{a}_1\right]$$

$$E_{\tau \sim p_\theta(\tau)}\left[\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t)\right] = E_{\mathbf{s}_1 \sim p(\mathbf{s}_1)}\left[E_{\mathbf{a}_1 \sim \pi(\mathbf{a}_1|\mathbf{s}_1)}\left[Q(\mathbf{s}_1, \mathbf{a}_1)|\mathbf{s}_1\right]\right]$$

easy to modify $\pi_\theta(\mathbf{a}_1|\mathbf{s}_1)$ if $Q(\mathbf{s}_1, \mathbf{a}_1)$ is known!

example: $\pi(\mathbf{a}_1|\mathbf{s}_1) = 1$ if $\mathbf{a}_1 = \arg\max_{\mathbf{a}_1} Q(\mathbf{s}_1, \mathbf{a}_1)$

Value Function

❖ Definition : Q-function

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_t, \mathbf{a}_t\right]\text{: total reward from taking } \mathbf{a}_t \text{ in } \mathbf{s}_t$$

❖ Definition : Value function

$$V^\pi(\mathbf{s}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_t\right]\text{: total reward from } \mathbf{s}_t$$

$$V^\pi(\mathbf{s}_t) = E_{\mathbf{a}_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t)}[Q^\pi(\mathbf{s}_t, \mathbf{a}_t)]$$

$E_{\mathbf{s}_1 \sim p(\mathbf{s}_1)}[V^\pi(\mathbf{s}_1)]$ is the RL objective!

Value Function

❖ Using Q-functions and value functions

Idea 1: if we have policy π, and we know $Q^\pi(\mathbf{s}, \mathbf{a})$, then we can *improve* π:

set $\pi'(\mathbf{a}|\mathbf{s}) = 1$ if $\mathbf{a} = \arg\max_{\mathbf{a}} Q^\pi(\mathbf{s}, \mathbf{a})$

this policy is at least as good as π (and probably better)!

and it doesn't matter what π is

Idea 2: compute gradient to increase probability of good actions $\mathbf{a}$:

if $Q^\pi(\mathbf{s}, \mathbf{a}) > V^\pi(\mathbf{s})$, then $\mathbf{a}$ is *better than average* (recall that $V^\pi(\mathbf{s}) = E[Q^\pi(\mathbf{s}, \mathbf{a})]$ under $\pi(\mathbf{a}|\mathbf{s})$)

modify $\pi(\mathbf{a}|\mathbf{s})$ to increase probability of $\mathbf{a}$ if $Q^\pi(\mathbf{s}, \mathbf{a}) > V^\pi(\mathbf{s})$

These ideas are *very* important in RL; we'll revisit them again and again!

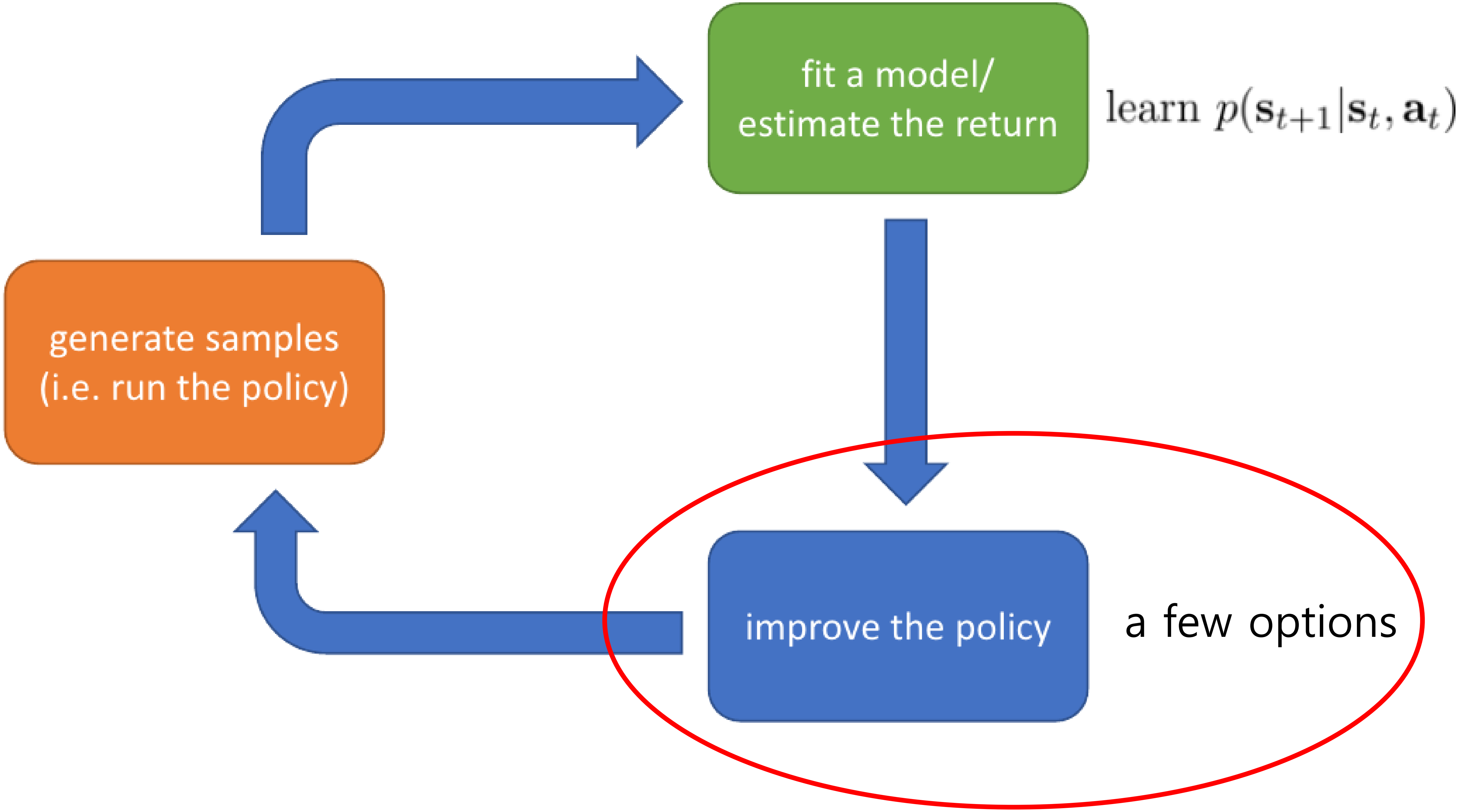
Types of Reinforcement Learning Algorithms

$$\theta^\star = \arg\max_\theta E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right]$$

- Policy gradients: directly differentiate the above objective
- Value-based: estimate value function or Q-function of the optimal policy (no explicit policy)
- Actor-critic: estimate value function or Q-function of the current policy, use it to improve policy
- Model-based RL: estimate the transition model, and then…
  - Use it for planning (no explicit policy)
  - Use it to improve a policy

Types of Reinforcement Learning Algorithms

- Model-based RL algorithms



Types of Reinforcement Learning Algorithms

❖ Model-based RL algorithms

improve the policy — a few options

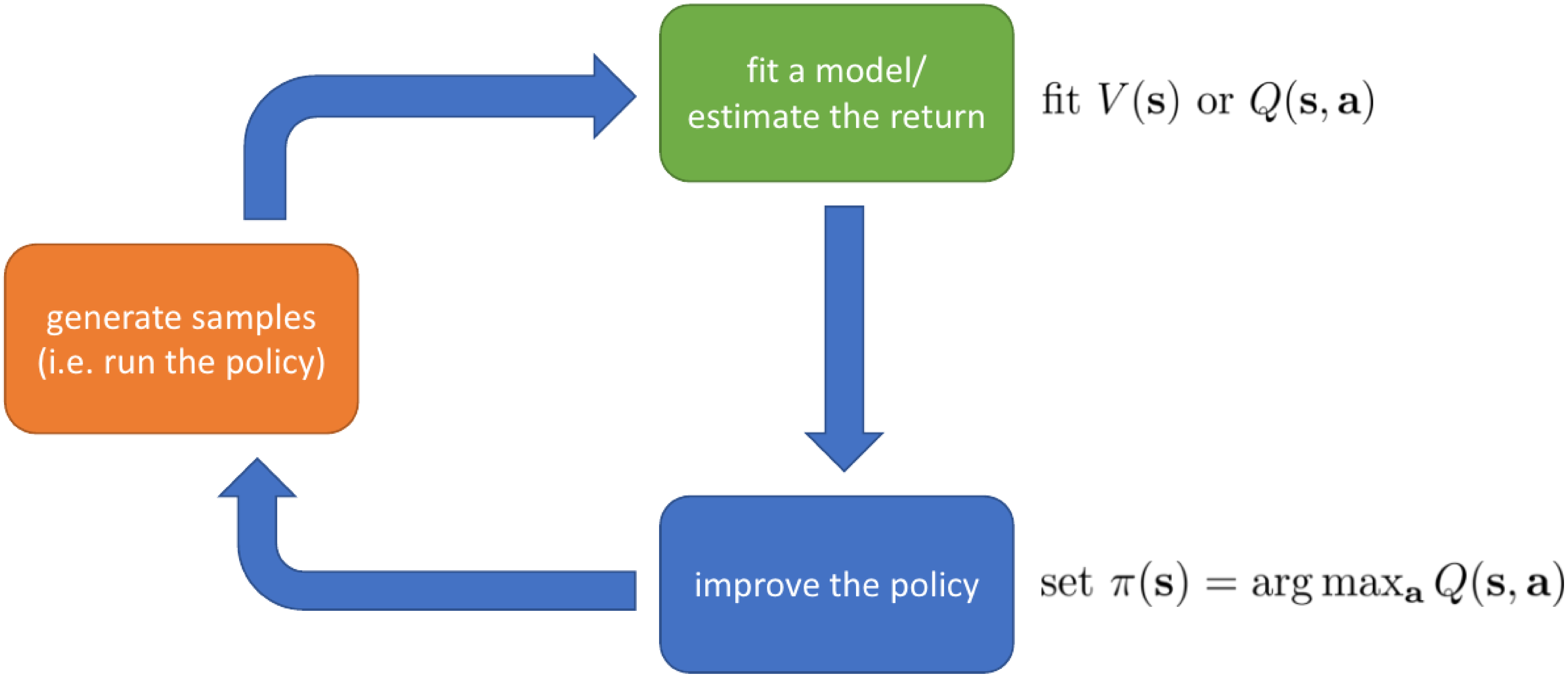
1. Just use the model to plan (no policy)
   - Trajectory optimization/optimal control (primarily in continuous spaces) – essentially backpropagation to optimize over actions
   - Discrete planning in discrete action spaces – e.g., Monte Carlo tree search

2. Backpropagate gradients into the policy
   - Requires some tricks to make it work

3. Use the model to learn a value function
   - Dynamic programming
   - Generate simulated experience for model-free learner

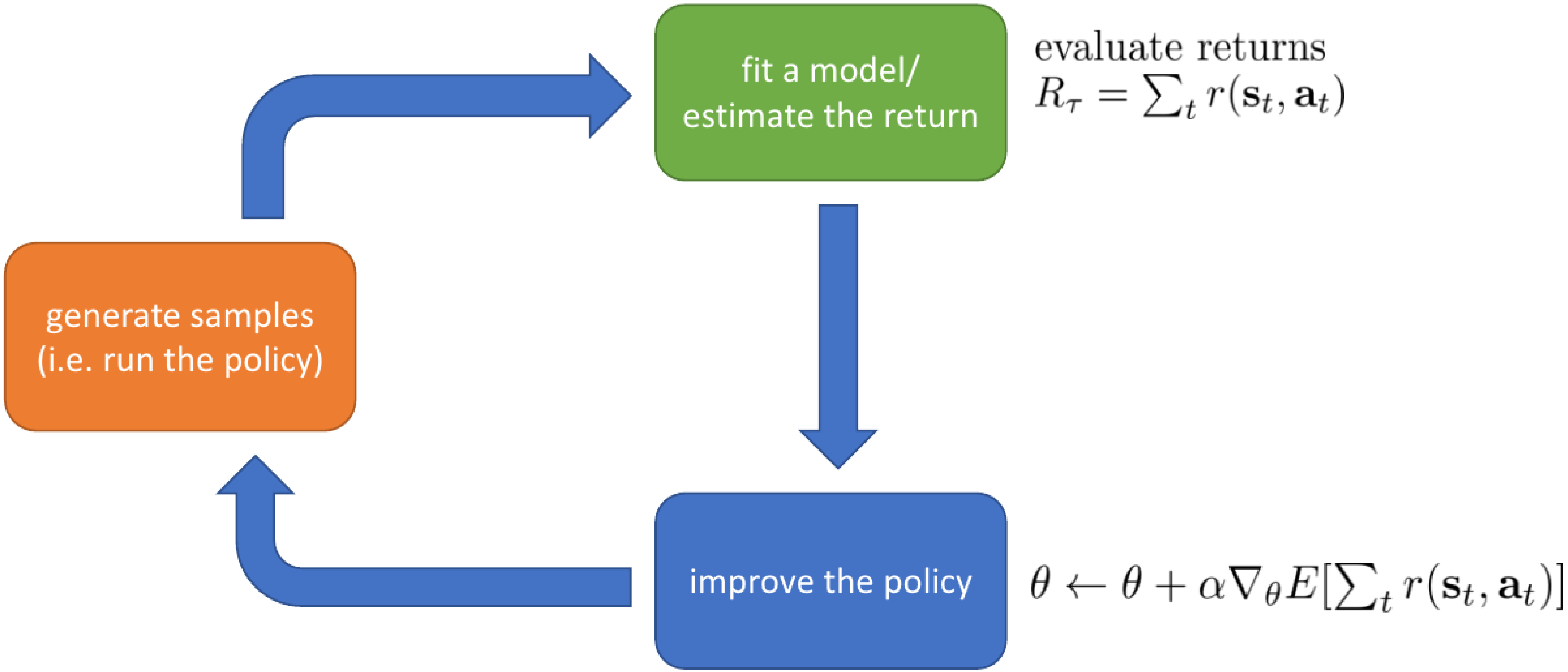
Types of Reinforcement Learning Algorithms

- ❖ Value function based RL algorithms



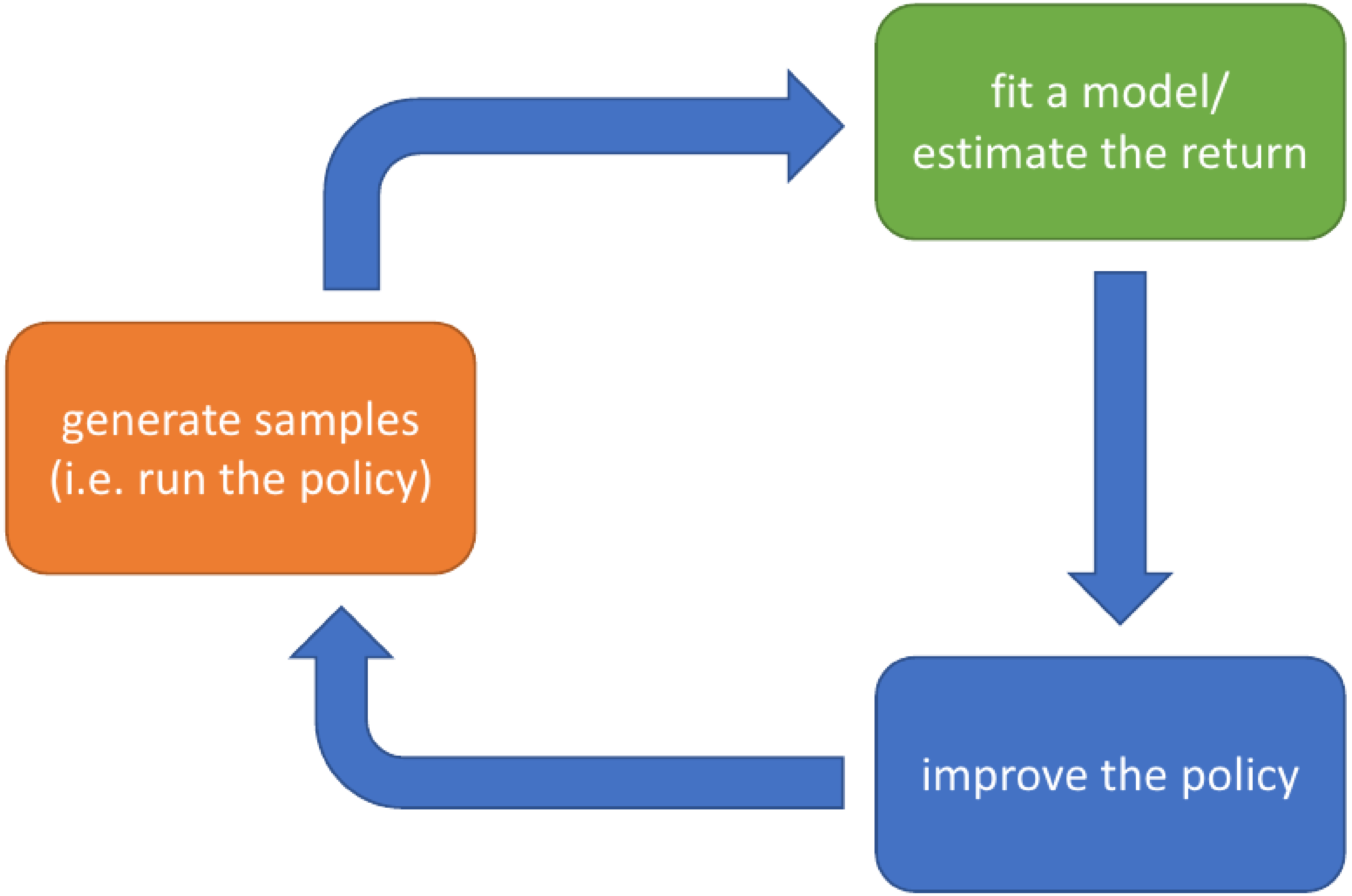
Types of Reinforcement Learning Algorithms

- Direct policy gradients



Types of Reinforcement Learning Algorithms

❖ Actor-critic: value function + policy gradients



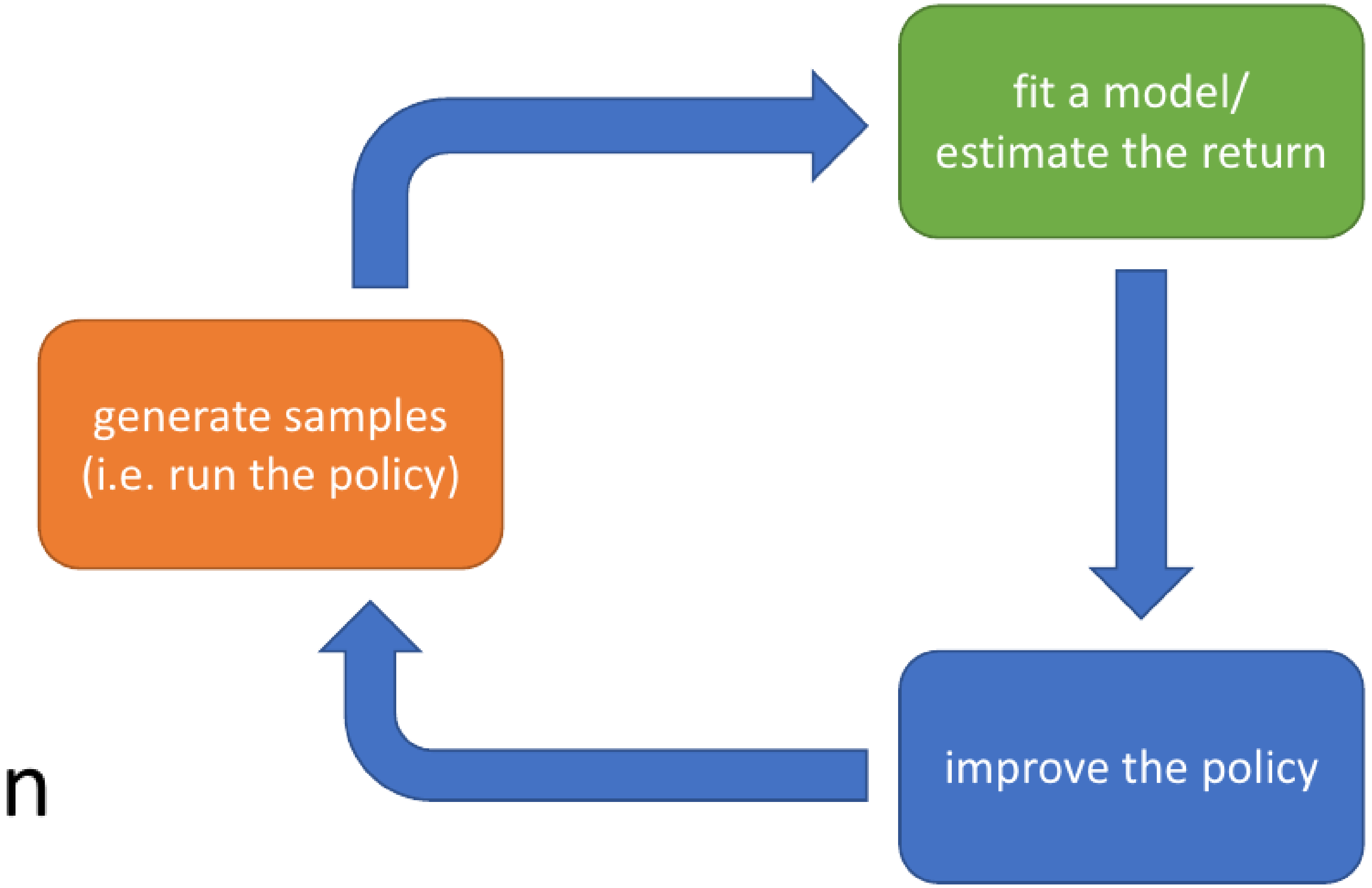
fit $V(\mathbf{s})$ or $Q(\mathbf{s}, \mathbf{a})$

$$\theta \leftarrow \theta + \alpha \nabla_\theta E[Q(\mathbf{s}_t, \mathbf{a}_t)]$$

Types of Reinforcement Learning Algorithms

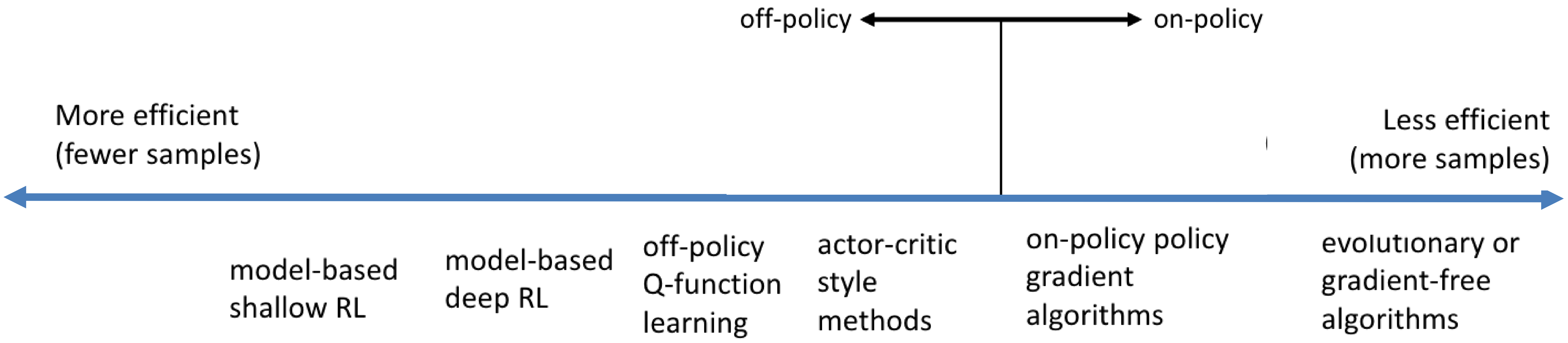
❖ Why so many RL algorithms?

- Different tradeoffs
  - Sample efficiency
  - Stability & ease of use
- Different assumptions
  - Stochastic or deterministic?
  - Continuous or discrete?
  - Episodic or infinite horizon?
- Different things are easy or hard in different settings
  - Easier to represent the policy?
  - Easier to represent the model?



Types of Reinforcement Learning Algorithms

❖ Comparison : sample efficiency



Why would we use a *less* efficient algorithm?

Wall clock time is not the same as efficiency!

Types of Reinforcement Learning Algorithms

❖ Comparison : Assumption

- Common assumption #1: full observability
  - Generally assumed by value function fitting methods
  - Can be mitigated by adding recurrence

- Common assumption #2: episodic learning
  - Often assumed by pure policy gradient methods
  - Assumed by some model-based RL methods

- Common assumption #3: continuity or smoothness
  - Assumed by some continuous value function learning methods
  - Often assumed by some model-based RL methods