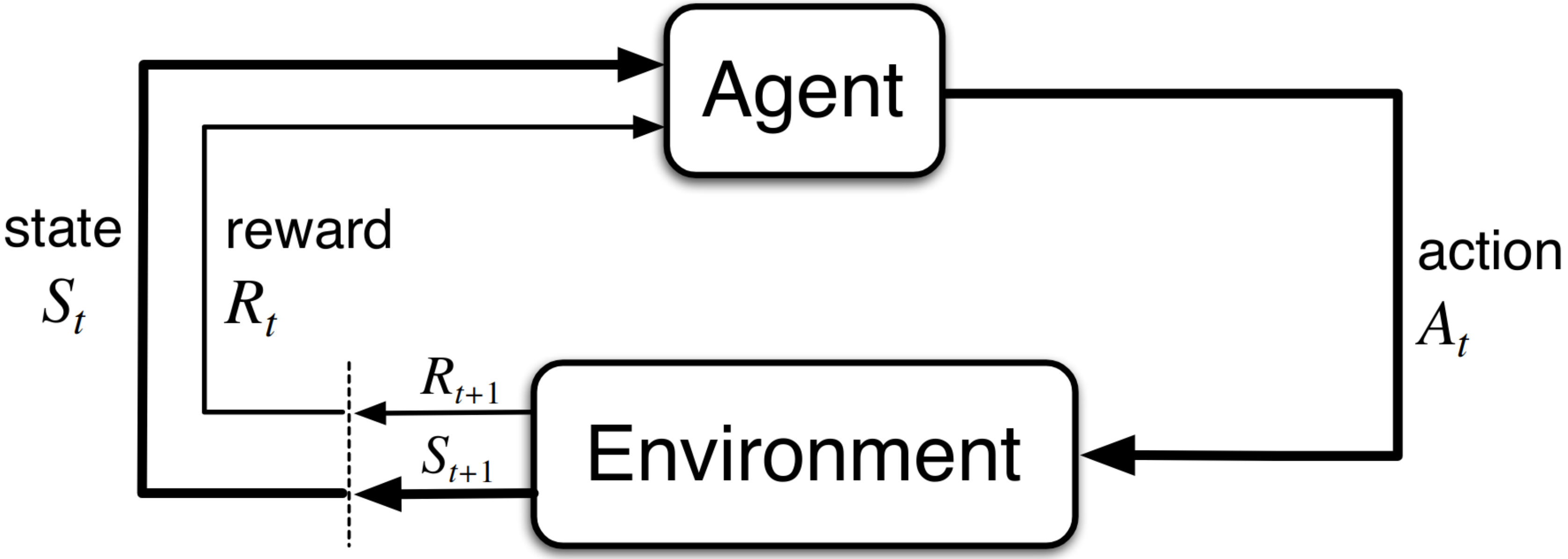
Reinforcement Learning

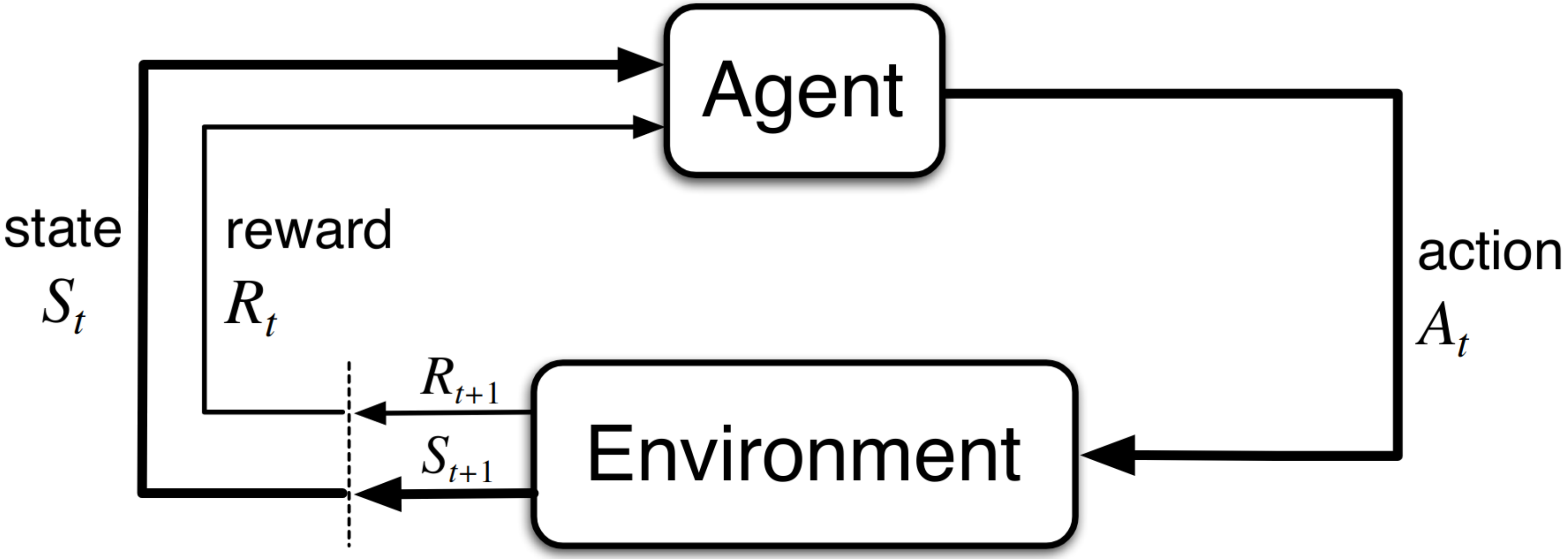
Timo Klein | 11.01.2023

Agent-Environment Framework



- Agent selects **action** a_t based on **state** s_t
- Observes **next state** s_{t+1} and **reward** r_{t+1}

Markov Decision Process (MDP)



- Formally: 5-tuple $\mathcal{M}=\langle \mathcal{S},\ \mathcal{A},\ P,\ R,\ \gamma \rangle$.
- *Goal*: $\max \mathbb{E}_\pi[\sum_{t=1}^{T} \gamma^t r(s_t, a_t)]$ by finding an optimal policy $\pi^\star$.

Sources: Sutton & Barto (2018)

Notation & Terms

Name	Notation
Transition function	$s' \sim P(s' \mid s, a)$
Reward function	$r \sim R(s, a)$
Policy	$a \sim \pi(a \mid s)$
Value function	$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t \mid s_0 = s, \pi\right]$
Q-function (Action-Value function)	$Q^{\pi}(s, a) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t \mid s_0 = s, a_0 = a, \pi\right]$
Bellman equation	$Q^{\pi}(s, a) = \mathbb{E}_{\pi}[r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) \mid s_t = s, a_t = a]$

Sources: Sutton & Barto (2018)

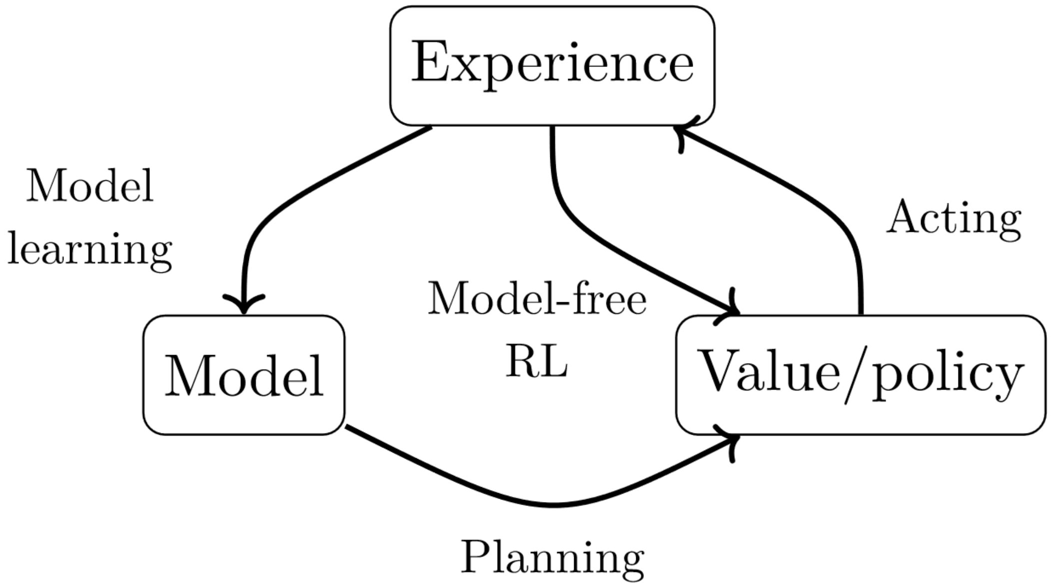
Notation & Terms

Name	Notation
Transition function	$s' \sim P(s' \mid s, a)$
Reward function	$r \sim R(s, a)$
Policy	$a \sim \pi(a \mid s)$
Value function	$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t \mid s_0 = s, \pi\right]$
Q-function (Action-Value function)	$Q^{\pi}(s, a) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t \mid s_0 = s, a_0 = a, \pi\right]$
Bellman equation	$Q^{\pi}(s, a) = \mathbb{E}_{\pi}[r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) \mid s_t = s, a_t = a]$

Use NNs for this
►deep RL

Sources: Sutton & Barto (2018)

Model-Free vs Model-based



- **Model-free RL**
  Estimate directly $\pi, Q^{\pi}(s,a)$ from samples
- **Model-based RL**
  1. Try to estimate $\hat{P}(s' \mid s,a)$ (and) $\hat{R}(s,a)$
  2. Use these to train $\pi, Q^{\pi}(s,a)$

Model-Free: (Deep) Q-learning

- Remember the Bellman equation

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}[r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) \mid s_t = s, a_t = a]$$

universität wien

Model-Free: (Deep) Q-learning

- Remember the Bellman equation

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) \mid s_t = s, a_t = a]$$

- Here we use the Bellman *optimality* equation

$$Q^{\pi}_{tar}(s,a) = \mathbb{E}_{\pi}[r_{t+1} + \gamma \max_{a'} Q^{\pi}(s', a')]$$

Model-Free: (Deep) Q-learning

- Remember the Bellman equation

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) \mid s_t = s, a_t = a]$$

- Here we use the Bellman *optimality* equation

$$Q^{\pi}_{tar}(s,a) = \mathbb{E}_{\pi}[r_{t+1} + \gamma \max_{a'} Q^{\pi}(s', a')]$$

- Idea: *Directly approximate* Q^{π} using Q^{π}_{tar}

MODEL-FREE: (DEEP) Q-LEARNING

- Remember the Bellman equation

$$Q^\pi(s,a) = \mathbb{E}_\pi[r_{t+1} + \gamma Q^\pi(s_{t+1}, a_{t+1}) \mid s_t = s, a_t = a]$$

- Here we use the Bellman *optimality* equation

$$Q^\pi_{tar}(s,a) = \mathbb{E}_\pi[r_{t+1} + \gamma \max_{a'} Q^\pi(s', a')]$$

- Idea: *Directly approximate* Q^π using Q^π_{tar}
- **Objective function**

$$L(\theta) = \frac{1}{N} \left(Q^\pi_\theta(s,a) - Q^\pi_{tar}(s,a)\right)^2$$

Exploration-Exploitation in Q-Learning

- Naïve Q-learning policy $\max_{\mathrm{a}} Q(s,a)$ never explores

Exploration-Exploitation in Q-Learning

- Naïve Q-learning policy $\max_{\text{a}} Q(s, a)$ never explores
- Solution: Use ϵ-greedy action selection

Exploration-Exploitation in Q-Learning

- Naïve Q-learning policy $\max_{a} Q(s, a)$ never explores
- Solution: Use ϵ-greedy action selection
  Select optimal action with probability $1 - \epsilon$
  Select random action with probability ϵ

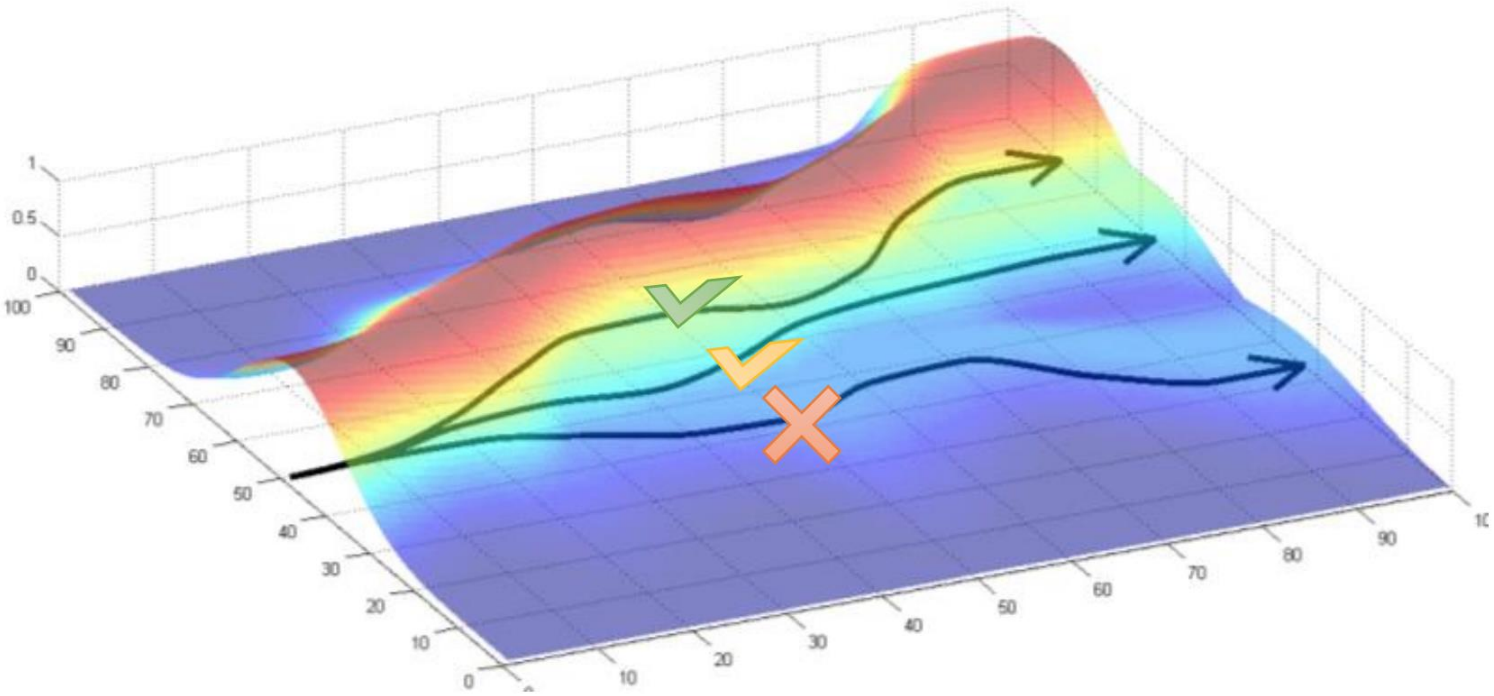
Exploration-Exploitation in Q-Learning

- Naïve Q-learning policy $\max_{a} Q(s, a)$ never explores
- Solution: Use ϵ-greedy action selection
  Select optimal action with probability $1 - \epsilon$
  Select random action with probability ϵ
- This makes Q-Learning an *off-policy* algorithm

Exploration-Exploitation in Q-Learning

- Naïve Q-learning policy $\max_{\mathrm{a}} Q(s, a)$ never explores
- Solution: Use ϵ-greedy action selection
  Select optimal action with probability $1 - \epsilon$
  Select random action with probability ϵ
- This makes Q-Learning an *off-policy* algorithm
- Data comes from ϵ-greedy policy
  But we're learning the greedy policy $Q_{tar}^{\pi}(s, a) = \mathbb{E}_{\pi}[r_{t+1} + \gamma \max_{a'} Q^{\pi}(s', a')]$

MODEL-FREE: POLICY GRADIENT (INTUITION)



Sources: Berkeley RL (2022)

Model-Free: Policy Gradient

- RL goal: Find policy that maximizes reward $\max_{\theta} J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)]$

MODEL-FREE: POLICY GRADIENT

- RL goal: Find policy that maximizes reward $\max_{\theta} J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)]$

- Why can't we do gradient descent on the objective? $\nabla_\theta J(\pi_\theta) = \nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)]$

MODEL-FREE: POLICY GRADIENT

- RL goal: Find policy that maximizes reward $$\max_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)]$$

- Why can't we do gradient descent on the objective? $$\nabla_\theta J(\pi_\theta) = \nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)]$$

- Because R is non-differentiable!

Model-Free: Policy Gradient

- RL goal: Find policy that maximizes reward $\max_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)]$

- Why can't we do gradient descent on the objective? $\nabla_\theta J(\pi_\theta) = \nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)]$

- Because R is non-differentiable!

- Solution 1: Use the likelihood-ratio estimator $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} R_t(\tau) \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]$

Model-Free: Policy Gradient

- RL goal: Find policy that maximizes reward $\max_{\theta} J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)]$

- Why can't we do gradient descent on the objective? $\nabla_\theta J(\pi_\theta) = \nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)]$

- Because R is non-differentiable!

- Solution 1: Use the likelihood-ratio estimator $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} R_t(\tau) \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]$

- Solution 2: Approximate expectation with MC $\nabla_\theta J(\pi_\theta) \approx \sum_{t=0}^{T} R_t(\tau) \nabla_\theta \log \pi_\theta(a_t \mid s_t)$

Model-free: Actor-Critic

- Policy gradient works, but there are some problems

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} R_t(\tau) \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]$$

Model-free: Actor-Critic

- Policy gradient works, but there are some problems

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} R_t(\tau) \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]$$

Unbiased, but high variance
Does not capture relative quality of action

MODEL-FREE: ACTOR-CRITIC

- Policy gradient works, but there are some problems

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} R_t(\tau) \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]$$

Unbiased, but high variance
Does not capture relative quality of action

- Solution 1: Combine Policy Gradient with TD Learning

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} Q^\pi(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]$$

Model-free: Actor-Critic

- Policy gradient works, but there are some problems

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} R_t(\tau)\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]$$

Unbiased, but high variance
Does not capture relative quality of action

- Solution 1: Combine Policy Gradient with TD Learning

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} Q^\pi(s_t, a_t)\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]$$

- Solution 2: Subtract a state-dependent baseline

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} (Q^\pi(s_t, a_t) - V^\pi(s_t))\,\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]$$

Model-Free: The Advantage Function

- The previous slide used a quantity called advantage function

$$A^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)$$

Model-Free: The Advantage Function

- The previous slide used a quantity called advantage function

$$A^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)$$

- Must estimate two functions Q^{π} and V^{π}
  ►Error prone and expensive

Model-Free: The Advantage Function

- The previous slide used a quantity called advantage function
$$A^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)$$
- Must estimate two functions Q^{π} and V^{π}
  ►Error prone and expensive
- Can we do better?

Model-Free: The Advantage Function

- The previous slide used a quantity called advantage function

$$A^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)$$

- Must estimate two functions Q^{π} and V^{π}
  ►Error prone and expensive
- Can we do better?
- YES! Estimate advantage using *only* V^{π}

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{\pi}[r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1})]$$
$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{\pi}[r_{t+1}] + \mathbb{E}_{\pi}[\gamma Q^{\pi}(s_{t+1}, a_{t+1})]$$
$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{\pi}[r_{t+1}] + \gamma V^{\pi}(s_{t+1})$$
$$A^{\pi}(s_t, a_t) \approx r_{t+1} + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$$

CONCLUSION

- RL ports dynamic programming to large problems with sampling
- Deep RL approximates quantities with NNs
- There is model-free and model-based RL
- Three families of model-free deep RL algorithms

Name	Type	Approximated functions	Data used	Action space
Q-Learning	Value-based	$Q(s,a)$	Off-policy	Discrete
Policy gradient/REINFORCE	Policy-based	$\pi(a \mid s)$	On-policy	Continuous + Discrete
Actor-Critic	Combined	Combinations of Q, π, V	Depends	Continuous + Discrete

- Most strong current algorithms (PPO, SAC, TD3) are Actor-Critic algorithms

REFERENCES

- Sutton & Barto book (2018)
- Foundations of deep Reinforcement Learning Theory and Practice in python
- Berkeley CS 285 (Policy Gradient image)