

Brain Inspired Artificial Intelligence

3: Model-Based and Model-Free Reinforcement Learning

Eiji Uchibe

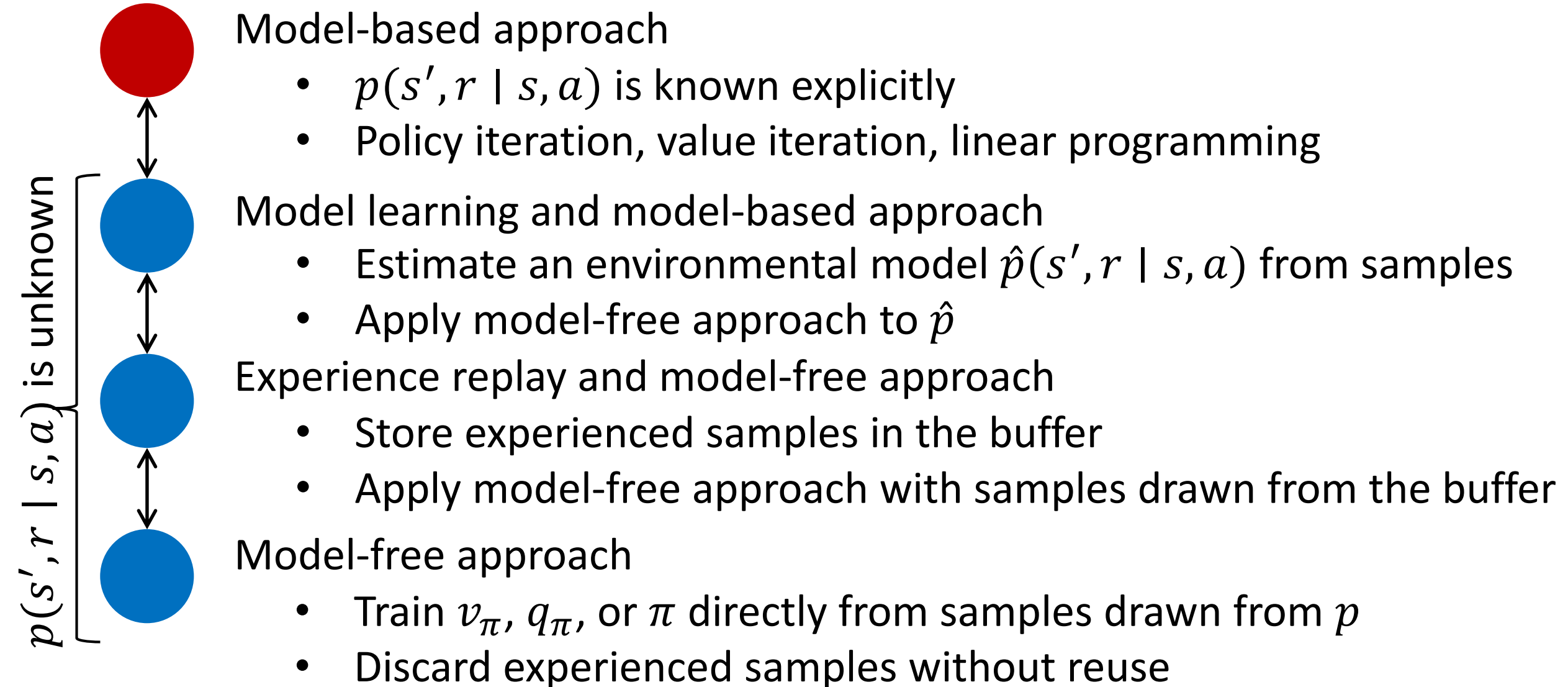
Dept. of Brain Robot Interface

ATR Computational Neuroscience Labs.

Outline

- We have studied Markov Decision Processes (MDPs)
 - Bellman expectation equation and Bellman optimality equation
- We will study model-based and model-free approaches to solve MDP problems
- Model-based methods
 - Policy iteration (= policy evaluation + policy improvement)
 - Value iteration
- Model-free methods
 - TD-learning
 - SARSA
 - Q-learning

Model-based vs. Model-free Methods



Behavior Policy

- Through interactions with the environment, a set of experienced transitions is collected:

for $i = 1$ to N

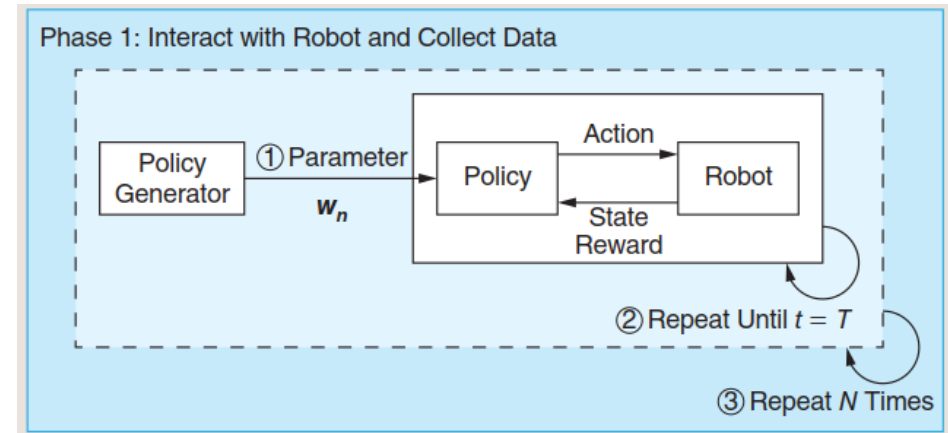
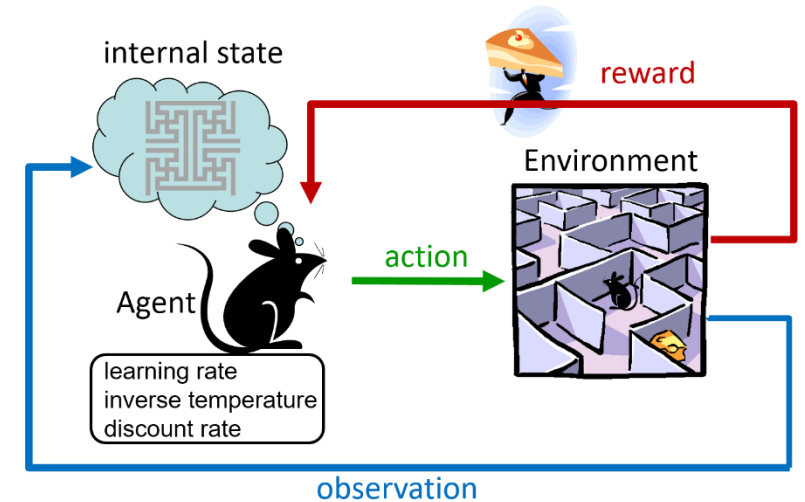
$$S_0^i \sim p_0(s)$$

for $t = 1$ to T

$$A_t^i \sim b(a | S_t^i)$$

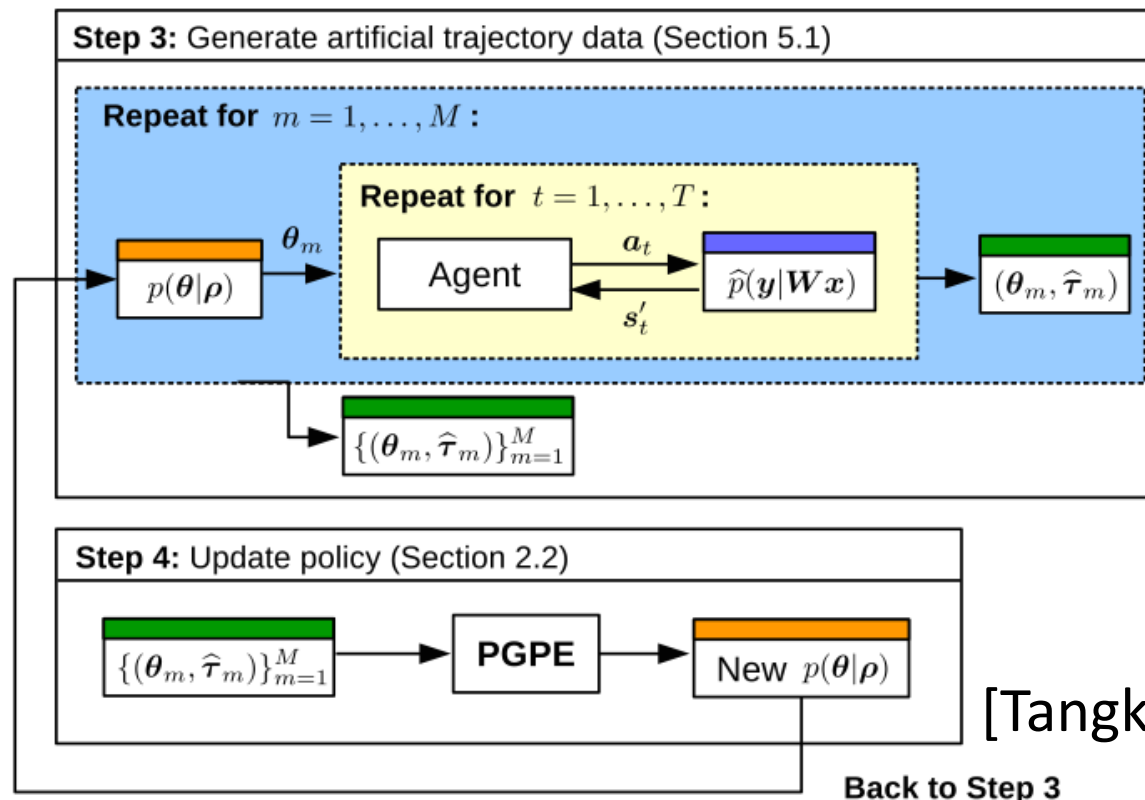
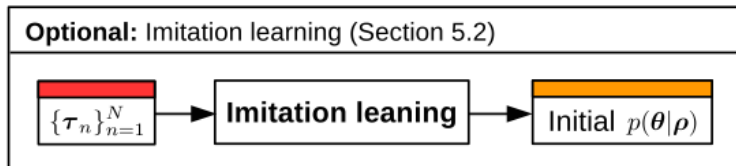
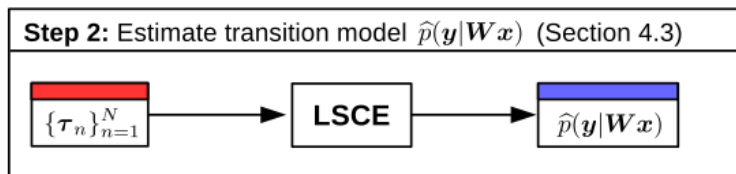
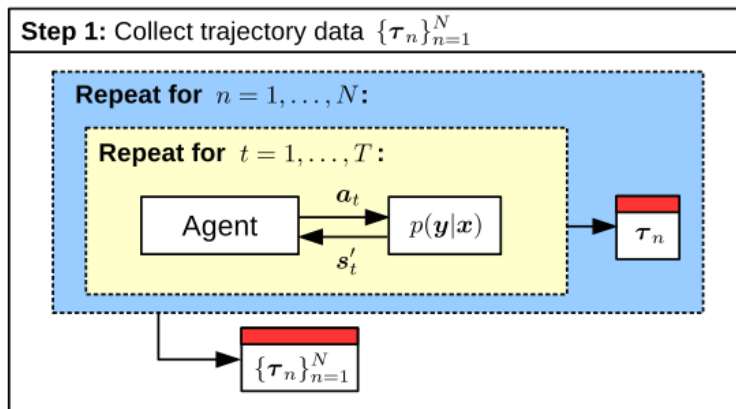
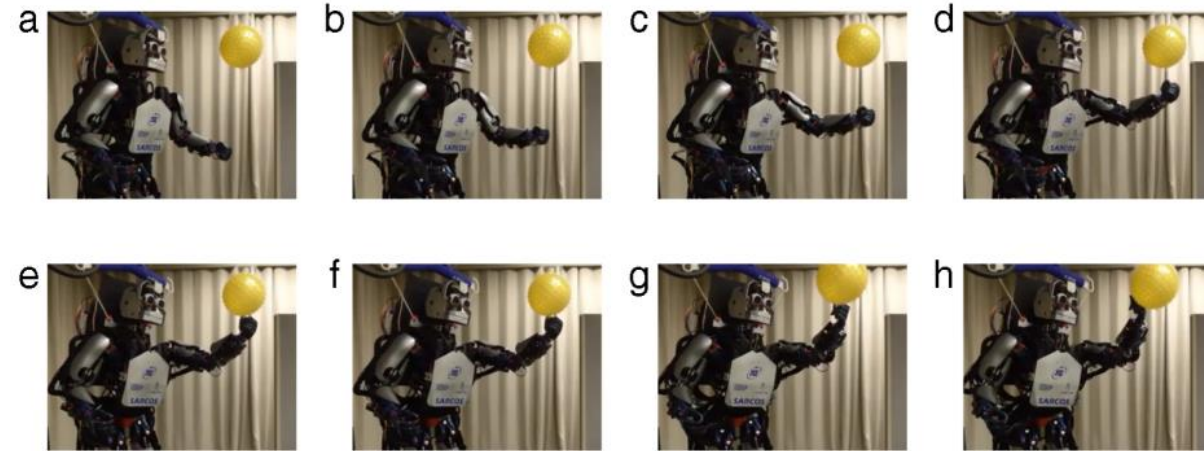
$$S_{t+1}^i, R_{t+1}^i \sim p(s', r | S_t^i, A_t^i)$$

- $p_0(s)$ and $p(s', r | s, a)$ are unknown in model-free approaches
- $b(a | s)$ is called the behavior policy, which may be different from learning policies



Model-Learning and Model-Free RL approaches

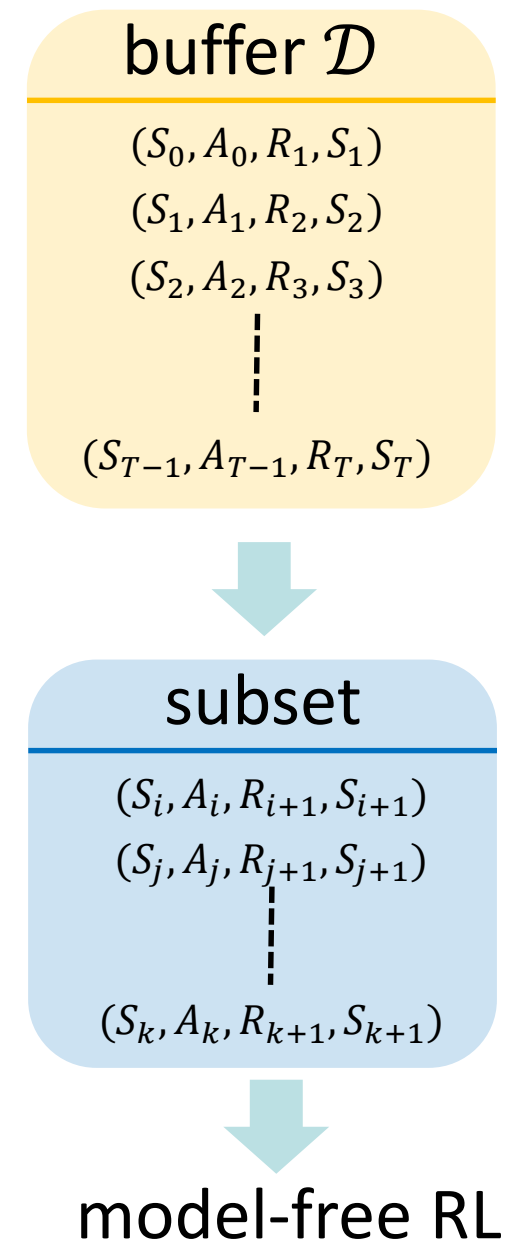
- Estimate $\hat{p}(s', r | s, a)$ from a set of experienced trajectories



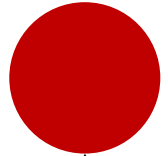
[Tangkaratt et al., 2016]

Experience Replay

- Experienced transitions are stored in the buffer \mathcal{D}
- Some transitions are repeatedly sampled from the buffer and applied to Model-free RL
- Uniform sampling is used
 - If there are N experienced transitions, the probability to select i -th transition is $1/N$
- There are no priority to select samples
- One of widely used techniques in modern RL studies
- Any **off-policy** RL algorithms are applicable

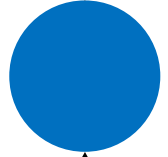


Model-based vs. Model-free Methods



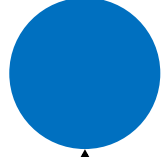
Model-based approach

- $p(s', r \mid s, a)$ is known explicitly
- Policy iteration, value iteration, linear programming



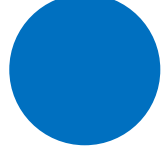
Model learning and model-based approach

- Estimate an environmental model $\hat{p}(s', r \mid s, a)$ from samples
- Apply model-free approach to \hat{p}



Experience replay and model-free approach

- Store experienced samples in the buffer
- Apply model-free approach with samples drawn from the buffer



Model-free approach

- Train v_π , q_π , or π directly from samples drawn from p
- Discard experienced samples without reuse

Two Model-based Approaches

- Policy iteration:
 - Policy evaluation and policy improvement
 - 1. Initialize the policy. 2. Evaluate the corresponding value function. 3. Improve the policy.

$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \cdots \xrightarrow{I} \pi_* \xrightarrow{E} v_*$$

- Value iteration:
 - 1. Initialize the value function. 2. Update the value function. 3. Retrieve the optimal policy.

$$v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_* \rightarrow \pi_*$$

Policy Evaluation

- For a given **fixed** policy, policy evaluation tries to compute $v_\pi(s)$ that satisfies Bellman expectation equation

$$\begin{aligned} v_\pi(s) &= \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) [r + \gamma v_\pi(s')] \\ &= \sum_a \sum_{s',r} \pi(a|s) p(s',r|s,a) [r + \gamma v_\pi(s')] \\ &= \sum_a \sum_{s',r} r \pi(a|s) p(s',r|s,a) + \gamma \sum_a \sum_{s',r} v_\pi(s') \pi(a|s) p(s',r|s,a) \\ &= \bar{r}(s) + \gamma \sum_{s'} v_\pi(s') p(s'|s) \end{aligned}$$

Closed Solution of Bellman Expectation Equation

- Reminder: Bellman expectation equation

$$v_{\pi}(s) = \bar{r}(s) + \gamma \sum_{s'} v_{\pi}(s') p(s' | s) \quad \text{for all } s$$

- matrix-vector representation

$$\begin{bmatrix} v_{\pi}(s_1) \\ \vdots \\ v_{\pi}(s_{|\mathcal{S}|}) \end{bmatrix} = \begin{bmatrix} \bar{r}(s_1) \\ \vdots \\ \bar{r}(s_{|\mathcal{S}|}) \end{bmatrix} + \gamma \begin{bmatrix} p(s_1|s_1) & \cdots & p(s_{|\mathcal{S}|}|s_1) \\ \vdots & \ddots & \vdots \\ p(s_1|s_{|\mathcal{S}|}) & \cdots & p(s_{|\mathcal{S}|}|s_{|\mathcal{S}|}) \end{bmatrix} \begin{bmatrix} v_{\pi}(s_1) \\ \vdots \\ v_{\pi}(s_{|\mathcal{S}|}) \end{bmatrix}$$

- $v_{\pi}(s)$ is obtained by solving a system of $|\mathcal{S}|$ simultaneous **linear** equations

Closed Solution of Bellman Expectation Equation

- Matrix/Vector representation


$$\mathbf{v} = \mathbf{r} + \gamma \mathbf{P} \mathbf{v}$$

$$\mathbf{v} = \begin{bmatrix} v_{\pi}(s_1) \\ \vdots \\ v_{\pi}(s_{|\mathcal{S}|}) \end{bmatrix}, \quad \mathbf{r} = \begin{bmatrix} \bar{r}(s_1) \\ \vdots \\ \bar{r}(s_{|\mathcal{S}|}) \end{bmatrix}, \quad \mathbf{P} = \begin{bmatrix} P(s_1|s_1) & \cdots & P(s_{|\mathcal{S}|}|s_1) \\ \vdots & \ddots & \vdots \\ P(s_1|s_{|\mathcal{S}|}) & \cdots & P(s_{|\mathcal{S}|}|s_{|\mathcal{S}|}) \end{bmatrix}$$

- If $\gamma < 1$, \mathbf{v} is uniquely determined by

$$\mathbf{v} = (\mathbf{I} - \gamma \mathbf{P})^{-1} \mathbf{r}$$

where \mathbf{I} is an identity matrix

- When $|\mathcal{S}|$ is too large, the matrix inversion is not tractable
 iteration method

Iterative Policy Evaluation

- Introduce Bellman expectation operator B_π

$$B_\pi v(s) \triangleq \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) [r + \gamma v(s')]$$

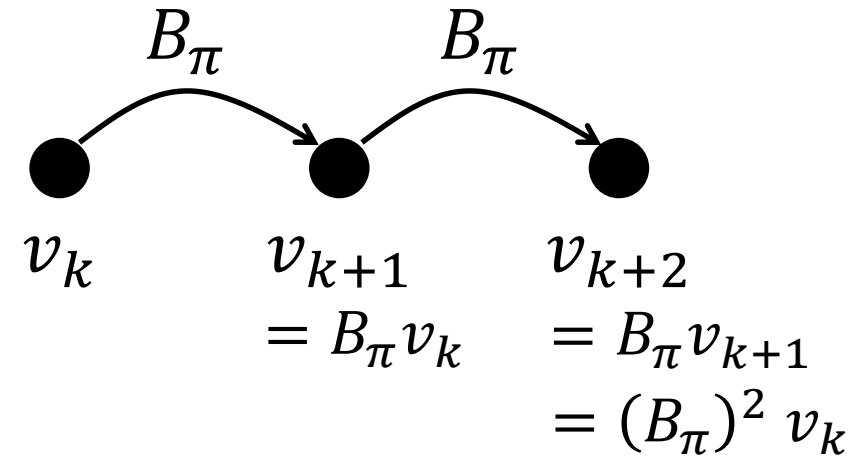
– B_* is linear, but dependent on π

- Initialize v_0 arbitrary (except that the terminal state must be 0)
- Compute v_{k+1} by the following rule

$$v_{k+1} = B_\pi v_k \triangleq r + \gamma P v_k$$

- v_k converges to the value function that satisfies the Bellman equation

$$\lim_{k \rightarrow \infty} v_k(s) = v_\pi(s) \quad \text{for all } s$$



Policy Evaluation

Input π , the policy to be evaluated

Algorithm parameter: a small threshold $\theta > 0$
determining accuracy of estimation

Initialize $V(s)$, for all s , arbitrarily except that
 $V(\text{terminal}) = 0$

Loop:

$\Delta \leftarrow 0$

Loop for each $s \in \mathcal{S}$:

$v \leftarrow V(s)$

$V(s) \leftarrow \sum_a \pi(a | s) \sum_{s', r} p(s', r | s, a) [r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$ (a small positive number)

Note

$v_\pi(s)$: function of state

$V(s)$: array estimate of $v_\pi(s)$

For example, $V(s)$ is implemented
by an $|\mathcal{S}|$ -dimensional vector

Policy Improvement

- After v_π is computed by policy evaluation, we want to find an improved policy π' based on v_π

$$\begin{aligned}\pi'(s) &= \arg \max_a q_\pi(s, a) \\ &= \arg \max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \\ &= \arg \max_a \sum_{s', r} P(s', r \mid s, a) [r + \gamma v_\pi(s')]\end{aligned}$$

- We do not need to maintain a stochastic policy.
- After policy improvement, it is proved that

$$v_{\pi'}(s) \geq v_\pi(s)$$

Policy Iteration

- Policy iteration is the algorithm that utilizes policy evaluation and policy improvement

$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \cdots \xrightarrow{I} \pi^* \xrightarrow{E} v_*$$

- It is proved that the value function converges to the optimal value function by policy iteration

Policy Iteration

1. Initialization

$V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation

3. Policy improvement

policy-stable \leftarrow true

For each $s \in \mathcal{S}$:

old-action $\leftarrow \pi(s)$

$\pi(s) \leftarrow \arg \max_a \sum_{s',r} p(s',r | s, a)[r + \gamma V(s')]$

If old-action $\neq \pi(s)$, then policy-stable \leftarrow false

If policy-stable, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Value Iteration

- One drawback to policy iteration is that each of its iteration involves policy evaluation which is computationally expensive
- What happen if we **truncate** policy evaluation?
- Value iteration directly updates the value function

$$v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_* \rightarrow \pi^*$$

- It is proved that the value function converges to the optimal value function by value iteration.

Value Iteration Algorithm

- Introduce Bellman optimality operator B_*

$$B_* v(s) \triangleq \max_a \sum_{s', r} p(s', r | s, a) [r + \gamma v(s')]$$

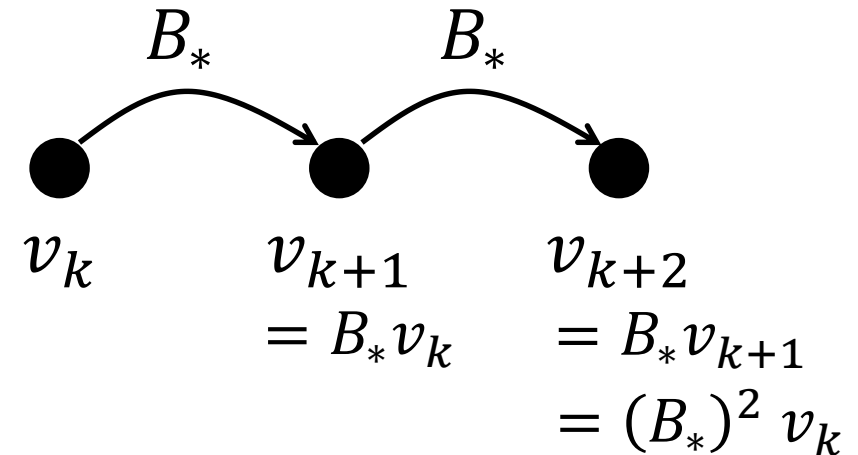
– B_* is nonlinear, but independent of π

- Initialize v_0 arbitrary (except that the terminal state must be 0)
- Compute v_{k+1} by the following rule

$$v_{k+1}(s) = B_* v_k(s)$$

- v_k converges to the value function that satisfies the Bellman optimality equation

$$\lim_{k \rightarrow \infty} v_k(s) = v_*(s) \quad \text{for all } s$$



Value Iteration Algorithm

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation

Initialize $V(s)$, for all s , arbitrarily except that $V(\text{terminal}) = 0$

Loop:

$\Delta \leftarrow 0$

Loop for each $s \in \mathcal{S}$:

$v \leftarrow V(s)$

$V(s) \leftarrow \max_a \sum_{s',r} p(s',r | s,a)[r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that

$\pi(s) = \arg \max_a \sum_{s',r} p(s',r | s,a)[r + \gamma V(s')]$

Compute the optimal
state value function v_*

Compute the optimal
policy π^* from v_*

Model-Free Reinforcement Learning

- Model based RL needs $p(s', r|s, a)$
- Model free RL uses samples $\{S_t, A_t, R_{t+1}, S_{t+1}\}_{t=0}^T$
- Algorithms
 - TD learning
 - SARSA
 - Q-learning
 - Expected SARSA

Model-Free Policy Evaluation

- The goal is to estimate v_π when π is given, but we do not know $p(s', r \mid s, a)$

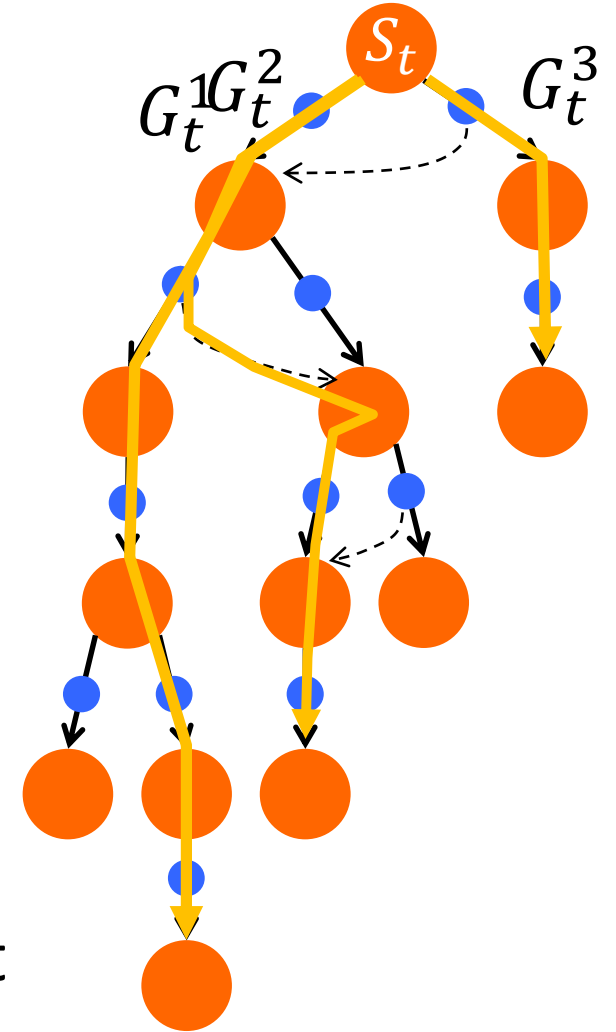
- Reminder: Definition of $v_\pi(s)$

$$v_\pi(s) = \mathbb{E}_\pi\{G_t \mid S_t = s\}, \quad G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- A naïve way is to collect many returns $\{G_t^i\}_{i=1}^N$ for every state and compute its empirical average

$$V(S_t) = \frac{1}{N} \sum_{i=1}^N G_t^i$$

- The above equation gives an unbiased estimate, but it usually has a large variance



Model-Free Policy Evaluation

- To derive an online update rule, we consider the sample-approximate Bellman expectation operator

$$B_{\pi}v(s) = \sum_{a,s',r} p(a,s',r|s)[r + \gamma v(s')]$$



$$\hat{B}_{\pi}v(S_t) \triangleq R_{t+1} + \gamma v(S_{t+1})$$

- It is proved that \hat{B}_{π} converges to B_{π} when we have $\{S_t^i, A_t^i, R_{t+1}^i, S_{t+1}^i\}_{i=1}^N$ as $N \rightarrow \infty$

Note

$p(a, s', r | s)$ depends on $\pi(a | s)$
because

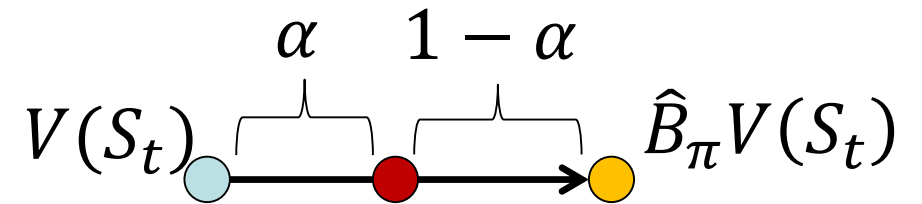
$$\begin{aligned} p(a, s', r | s) \\ = p(s', r | s, a) \pi(a | s) \end{aligned}$$

TD Learning

- Weighted average between the current estimate $V(S_t)$ and $\hat{B}_\pi V(S_t)$

$$V(S_t) \leftarrow (1 - \alpha)V(S_t) + \alpha\hat{B}_\pi V(S_t)$$

where $\alpha \in (0, 1]$ is called the learning rate



- Re-write the right hand side of the equation

$$V(S_t) \leftarrow V(S_t) + \alpha\delta_t$$

$$\delta_t \triangleq \underbrace{R_{t+1} + \gamma V(S_{t+1})}_{\text{target value}} - V(S_t)$$

- This is called TD learning, where δ_t is a temporal difference (TD) error

Tabular TD for Estimating v_π

Input: the policy π to be evaluated

Algorithm parameter: step size $\alpha \in (0, 1]$

Initialize $V(s)$, for all s , arbitrarily except that $V(\text{terminal}) = 0$

Loop for each episode:

 Initialize S

$A \leftarrow$ action given by π for S

 Take action A , observe R, S'

$V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$

$S \leftarrow S'$

until S is terminal

TD Approach for Estimating q_π

- TD-learning is a method of policy evaluation when $p(s', r | s, a)$ is unknown
- However, we cannot perform policy improvement in the model-free settings:

$$\pi(s) \leftarrow \arg \max_a \sum_{r, s'} p(s', r | s, a) [r + \gamma v_\pi(s')]$$

- So, we need Bellman expectation operator for q_π and its sample-approximator

Bellman Expectation Operator for q_π

- Reminder: Bellman expectation operator for v_π

$$B_\pi v(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) [r + \gamma v(s')]$$

- Bellman expectation operator for q_π

$$\begin{aligned} T_\pi q(s, a) &\triangleq \sum_{s',r} p(s',r | s, a) [r + \gamma v(s')] \\ &= \sum_{s',r} p(s',r | s, a) \left[r + \gamma \sum_{a'} \pi(a' | s') q(s, a) \right] \end{aligned}$$

Sample-Approximate Bellman Expectation Operator for q_π

- Similarly, we obtain the sample-approximate Bellman expectation operator

$$T_\pi q(s, a) \triangleq \sum_{r, s', a'} p(r, s', a' | s, a) [r + \gamma q(s', a')]$$



$$\hat{T}_\pi q(S_t, A_t) = R_{t+1} + \gamma q(S_{t+1}, A_{t+1})$$

Note

$$\begin{aligned} p(r, s', a' | s, a) \\ = \pi(a' | s') p(s', r | s, a) \end{aligned}$$

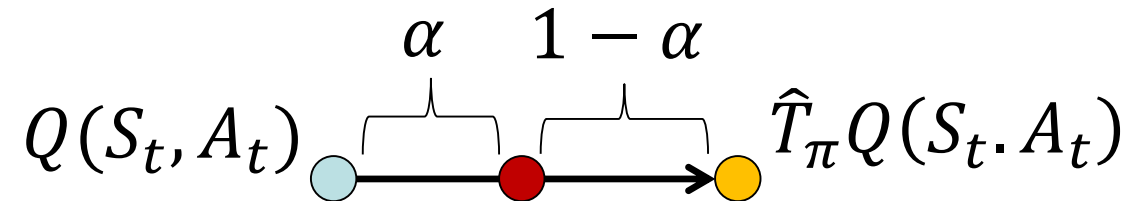
- It is proved that \hat{T}_π converges to T_π when we have $\{S_t^i, A_t^i, R_{t+1}^i, S_{t+1}^i, A_{t+1}^i\}_{i=1}^N$ as $N \rightarrow \infty$

SARSA: On-Policy TD Control

- Weighted average between the current estimate $Q(S_t, A_t)$ and $\hat{T}_\pi Q(S_t, A_t)$

$$Q(S_t, A_t) \leftarrow (1 - \alpha)Q(S_t, A_t) + \alpha \hat{T}_\pi Q(S_t, A_t)$$

where $\alpha \in (0, 1]$ is called the learning rate



- Re-write the right hand side of the equation

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \delta_t^{\text{SARSA}}$$

$$\delta_t^{\text{SARSA}} \triangleq \underbrace{R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})}_{\text{target value}} - Q(S_t, A_t)$$

- This is called SARSA

SARSA (On-Policy TD Control)

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$

Initialize $Q(s, a)$, for all s and a , arbitrarily except that $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

 Initialize S

 Choose A at S using policy derived from Q (e.g., ε -greedy)

 Loop for each step of episode:

 Take action A , observe R, S'

 Choose A' at S' using policy derived from Q (e.g., ε -greedy)

$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma Q(S', A') - Q(S, A)]$

$S \leftarrow S'; A \leftarrow A';$

 until S is terminal

Note on TD and SARSA

- The operators depend on the current learning policy explicitly

$$B_{\pi}v(s) = \sum_{a,s',r} p(s',r|s,a) \pi(a | s) [r + \gamma v(s')] \approx R_{t+1} + \gamma v(S_{t+1})$$

$$\begin{aligned} T_{\pi}q(s, a) &\triangleq \sum_{r,s',a'} \pi(a' | s') p(r, s'|s, a) [r + \gamma q(s', a')] \\ &\approx R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) \end{aligned}$$

- Therefore, samples to approximate the operators should be collected by the learning policy $\pi(a | s)$

 on-policy learning

From On-Policy Learning to Off-Policy Learning

- On-policy learning is stable but sample-inefficient because it cannot use samples $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$ generated by a behavior policy $b(a | s)$ that is different from $\pi(a | s)$

- Consider Bellman optimality operator

$$\begin{aligned} B_* v(s) &= \max_a \sum_{s', r} p(s', r | s, a) [r + \gamma v(s')] \\ &= \max_a \mathbb{E}[r + \gamma v(s')] \end{aligned}$$

- Since the max operator is outside of the expectation, it is not trivial to derive the sample-approximate Bellman optimality operator for v_*

$$\hat{B}_* v(s) = \max_a [R_{t+1} + \gamma V(S_{t+1})]$$

Bellman Optimality Equation for q_*

- Reminder: Bellman optimality equation for v_*

$$v_*(s) = \max_a \sum_{s', r} p(s', r | s, a) [r + \gamma v_*(s')]$$

- Bellman optimality equation for q_*

$$\begin{aligned} q_*(s, a) &\triangleq \sum_{s', r} p(s', r | s, a) [r + \gamma v_*(s')] \\ &= \sum_{s', r} p(s', r | s, a) \left[r + \gamma \max_{a'} q_*(s', a') \right] \end{aligned}$$

Bellman Optimality Operator for q_* and its Sample-Approximation

- Define Bellman optimality operation for q_*

$$T_* q(s, a) \triangleq \sum_{s', r} p(s', r \mid s, a) \left[r + \gamma \max_{a'} q(s', a') \right]$$

– Expected value of $r + \gamma \max_{a'} q(s', a')$

- Sample-approximated Bellman optimality equation for q_*

$$\hat{T}_* q(s, a) \triangleq R_{t+1} + \gamma \max_{a'} q(S_{t+1}, a')$$

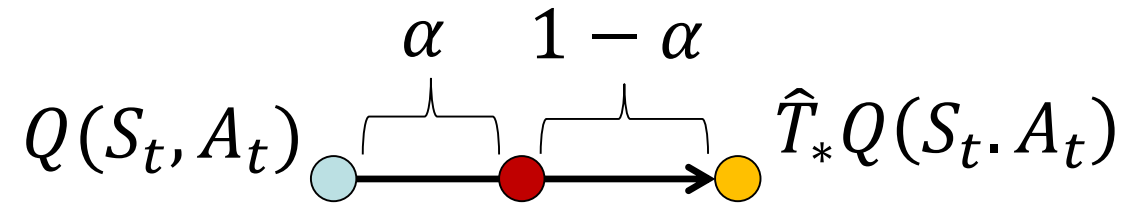
- T_* and \hat{T}_* do not depend on the learning policy π

Q-Learning: Off-Policy TD Control

- Weighted average between the current estimate $Q(S_t, A_t)$ and $\hat{T}_* Q(S_t, A_t)$

$$Q(S_t, A_t) \leftarrow (1 - \alpha)Q(S_t, A_t) + \alpha \hat{T}_* Q(S_t, A_t)$$

where $\alpha \in (0, 1]$ is called the learning rate



- Re-write the right hand side of the equation

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \delta_t^Q$$

$$\delta_t^Q \triangleq \underbrace{R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a')}_{\text{target value}} - Q(S_t, A_t)$$

- This is called Q-learning

Q-Learning (Off-Policy TD Control)

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$

Initialize $Q(s, a)$, for all s and a , arbitrarily except that $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

 Initialize S

 Loop for each step of episode:

 Choose A at S using a behavior policy

 Take action A , observe R, S'

$$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$$


$$S \leftarrow S'$$

 until S is terminal

Note on Q-Learning

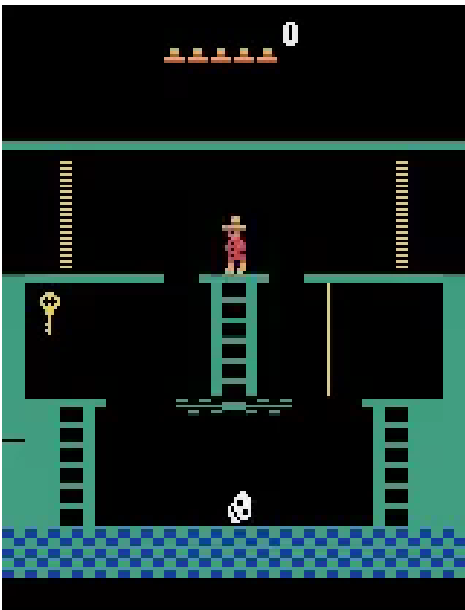
- The operator does not depend on the current learning policy

$$T_*q(s, a) \triangleq \sum_{r, s'} p(r, s' | s, a) \left[r + \gamma \max_{a'} q(s', a') \right]$$
$$\approx R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a)$$

- Therefore, the policy $b(a | s)$ that is different from the learning policy $\pi(a | s)$ can be used to collect samples to approximate the operator
 off-policy learning
- The off-policy property is critical for sample efficiency
- However, off-policy learning is usually unstable as compared with on-policy learning if reinforcement learning is combined with deep learning

Incorporating demonstrations

- Demonstrations are usually generated by the demonstrator's policy that are different from the learning policy
- To use demonstrations, learning algorithms should be off-policy



Game	DQfD	Prev. Best	Algorithm
Alien	4745.9	4461.4	Dueling DQN (Wang et al. 2016)
Asteroids	3796.4	2869.3	PopArt (van Hasselt et al. 2016)
Atlantis	920213.9	395762.0	Prior. Dueling DQN (Wang et al. 2016)
Battle Zone	41971.7	37150.0	Dueling DQN (Wang et al. 2016)
Gravitar	1693.2	859.1	DQN+PixelCNN (Ostrovski et al. 2017)
Hero	105929.4	23037.7	Prioritized DQN (Schaul et al. 2016)
Montezuma Revenge	4739.6	3705.5	DQN+CTS (Ostrovski et al. 2017)
Pitfall	50.8	0.0	Prior. Dueling DQN (Wang et al. 2016)
Private Eye	40908.2	15806.5	DQN+PixelCNN (Ostrovski et al. 2017)
Q-Bert	21792.7	19220.3	Dueling DQN (Wang et al. 2016)
Up N Down	82555.0	44939.6	Dueling DQN (Wang et al. 2016)

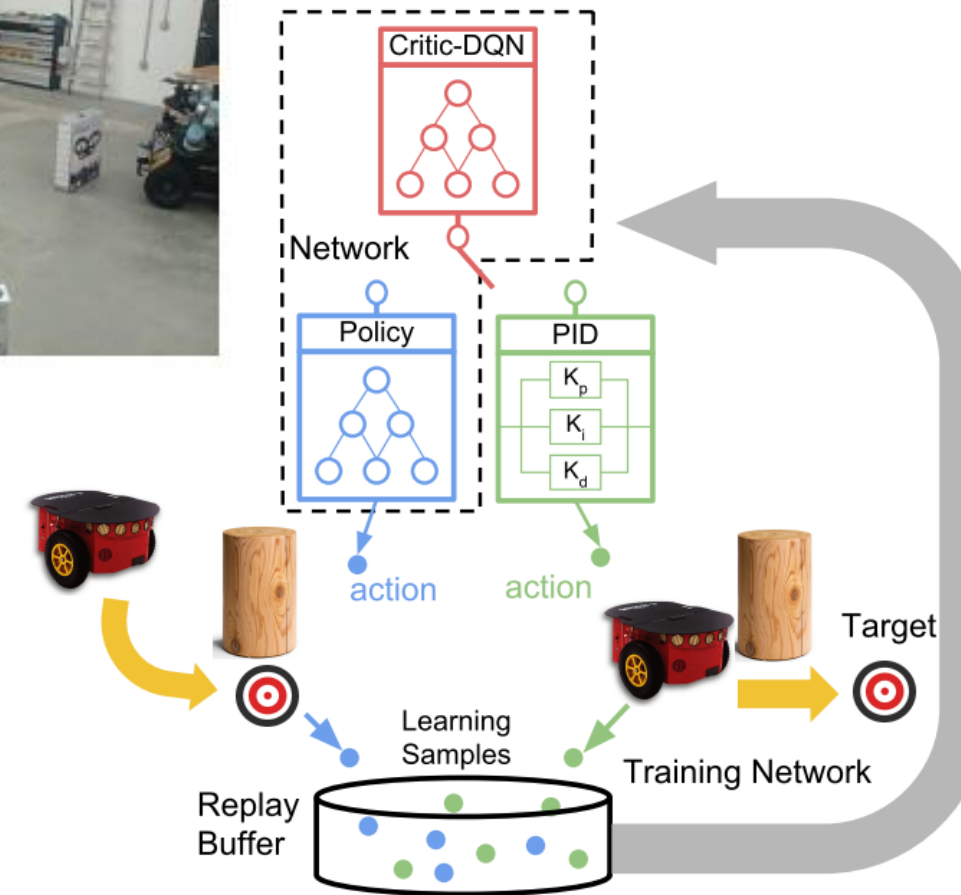
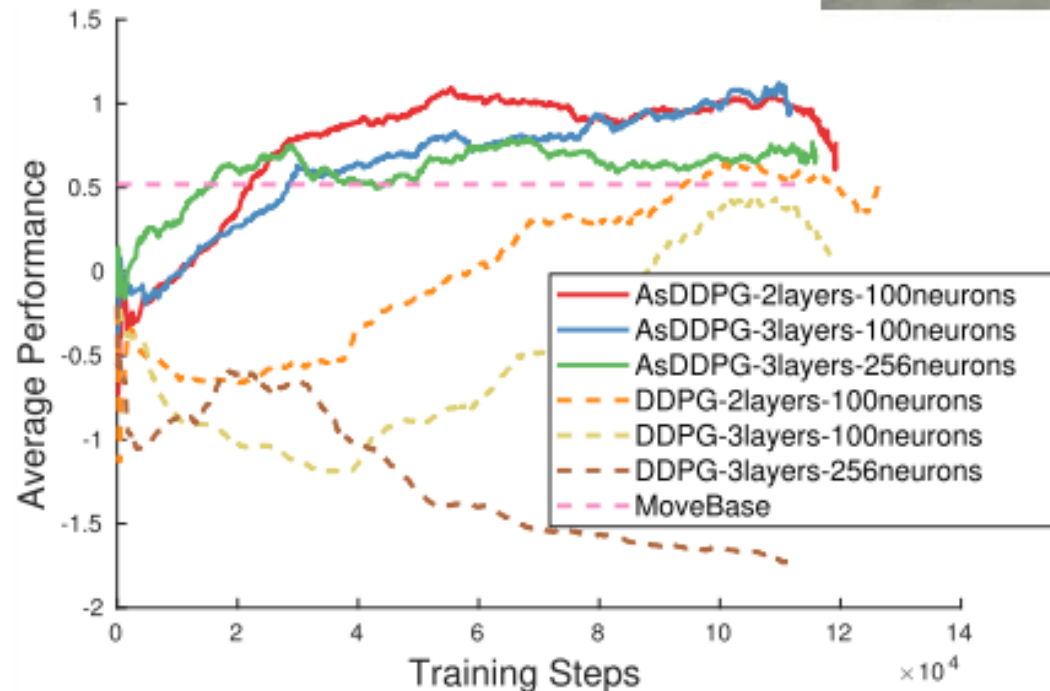


https://www.youtube.com/watch?v=bb_acE1yzDo

[Hester et al., 2018; Nair et al., 2018]

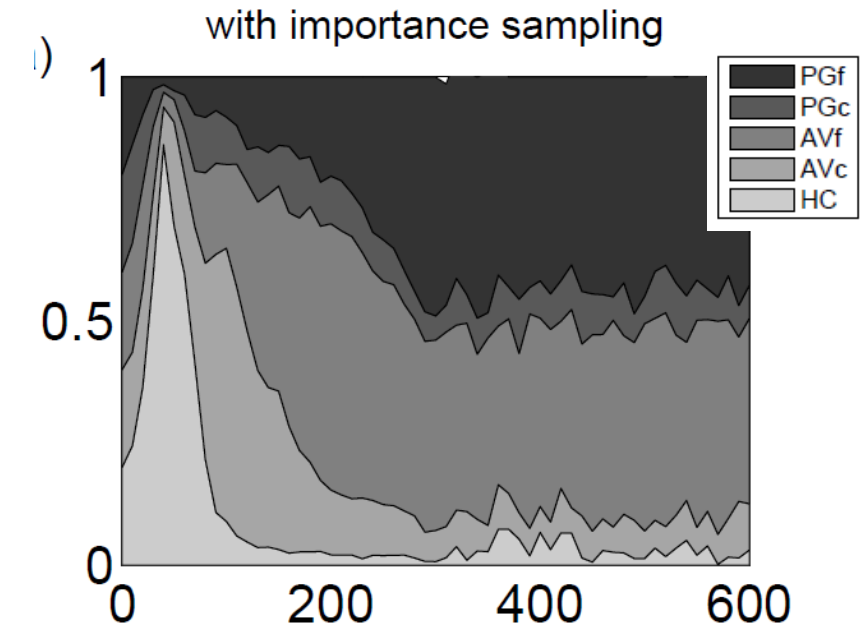
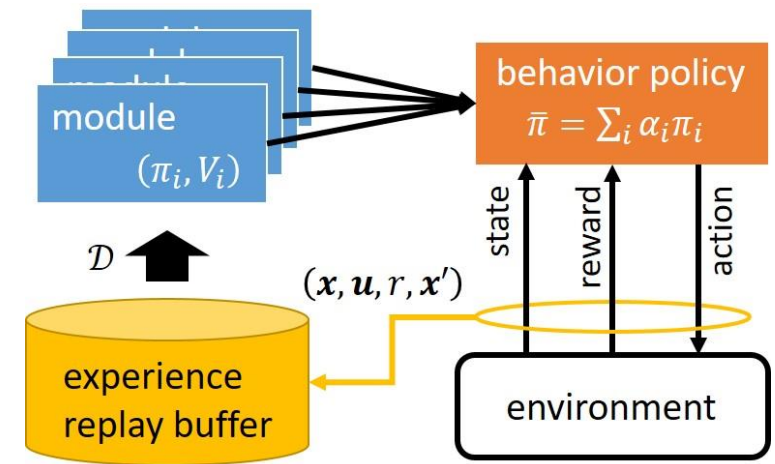
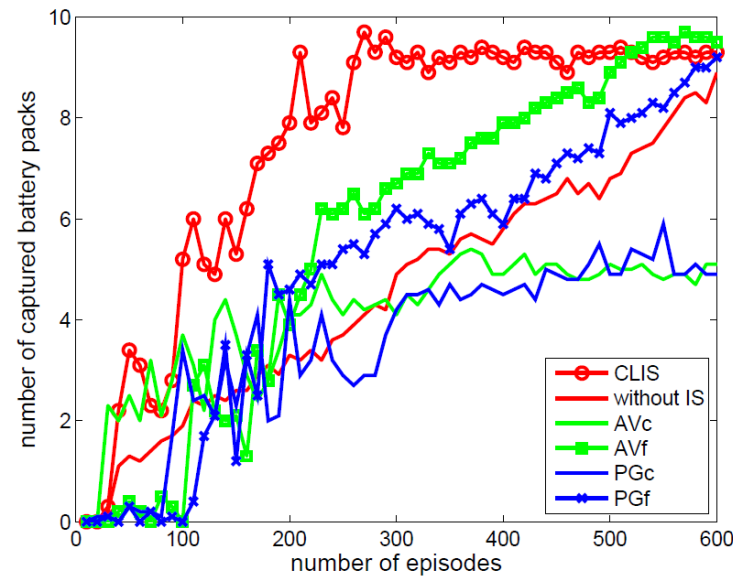
Using a hand-made controller to collect data

- Introduce a simple feedback (PID) controller is integrated
- The PID controller collects samples at the early stage of learning



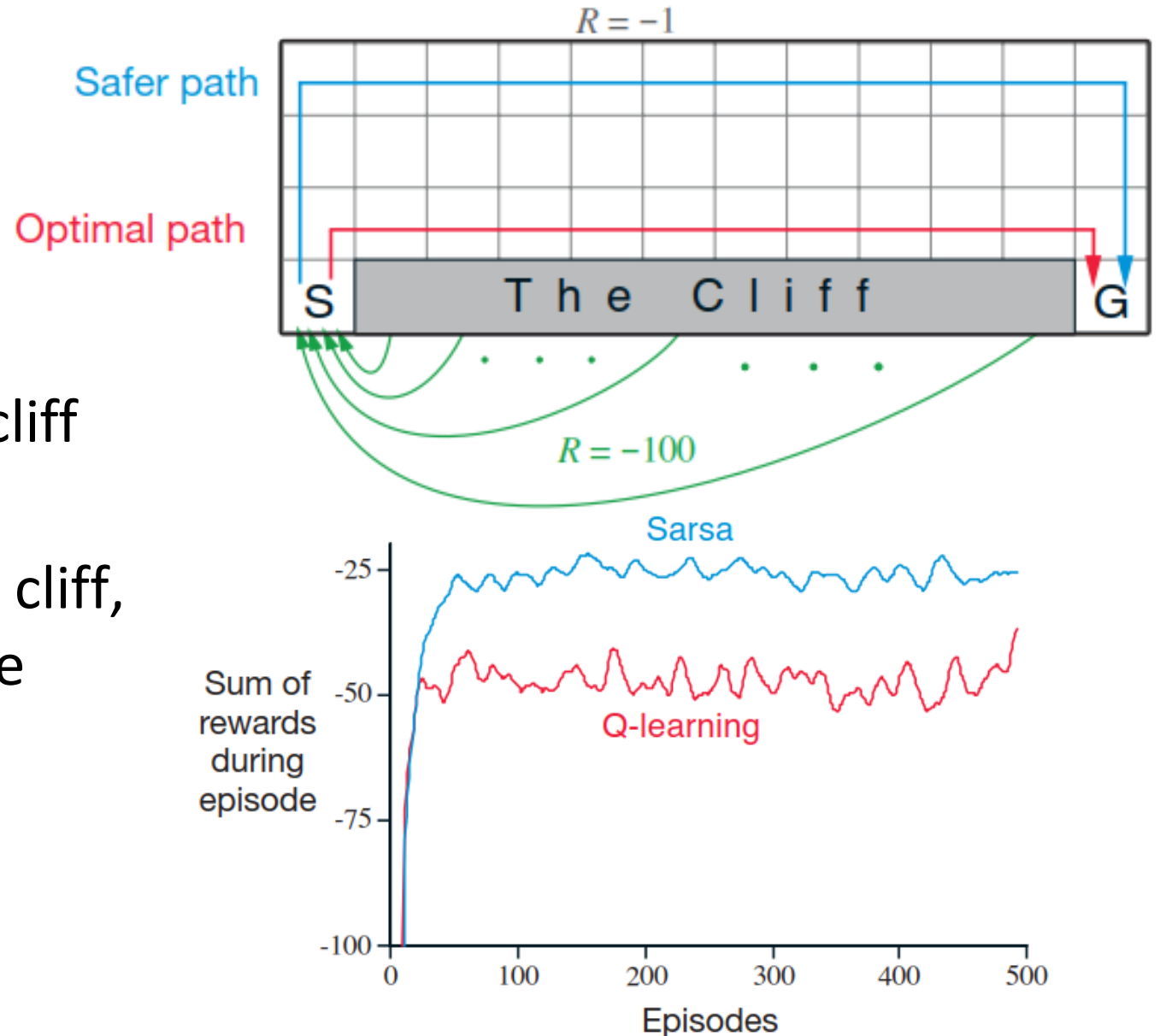
Multiple Heterogeneous Reinforcement Learning

- Multiple learning module with different state representation learn in parallel
- Automatic selection of an appropriate module



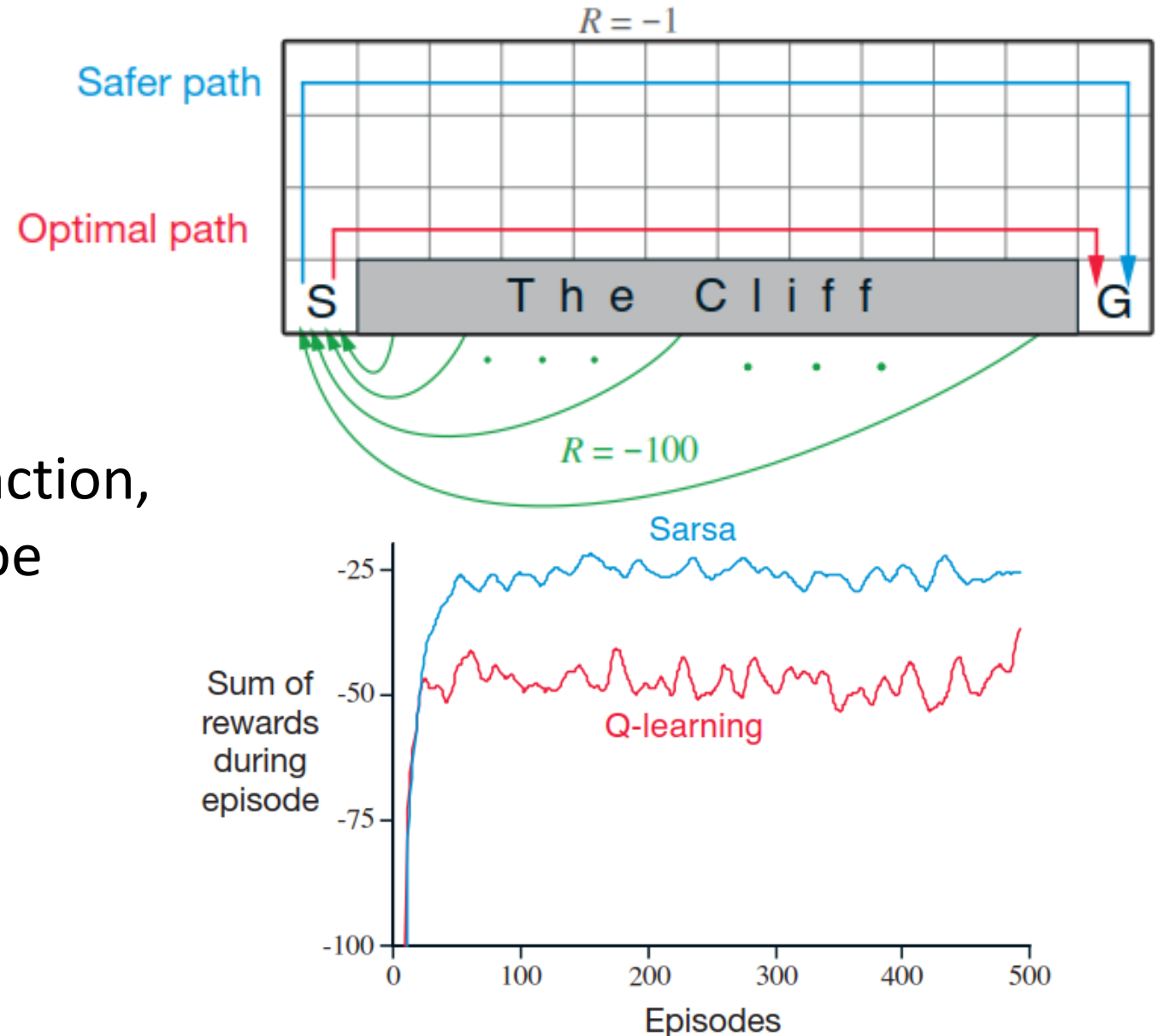
Difference between Q-learning and SARSA

- Deterministic state transition
- 4 actions (left, right, up, and down)
- $\gamma = 1$
- $r(s, a, s') = \begin{cases} 0 & \text{if } s' \in G \\ -100 & \text{if } s' \text{ is a cliff} \\ -1 & \text{otherwise} \end{cases}$
- When the agent arrives at the cliff, it is sent back to the start state
- ε -greedy with $\varepsilon = 0.1$



Difference between Q-learning and SARSA

- SARSA found the safer path
- The agent trained with Q-learning tended to go to the cliff during learning
- In fact, Q-learning found the optimal state-action value function, and the optimal policy could be derived by the greedy policy ($\epsilon = 0$)



References

- Hester, T., et al. Deep Q-learning from demonstrations. In Proc. of AAAI, 2018.
- Lin, L.-J. Programming robots using reinforcement learning. In Proc. of AAAI, 781-786, 1991.
- Mnih, V., et al. [Huma-level control through deep reinforcement learning](#). Nature, 518:529-533, 2015.
- Nair, A., McGrew, B., Andrychowicz, M., Zaremba, W., and Abbeel, P. Overcoming Exploration in Reinforcement Learning with Demonstrations. In Proc. of ICRA 2018.
- Tangkaratt, V., Morimoto, J., and Sugiyama, M. Model-based reinforcement learning with dimension reduction. Neural Networks 84, 1-16, 2016.
- Uchibe, E., and Doya, K. Competitive-cooperative-concurrent reinforcement learning with importance sampling. In Proc. of SAB, pp. 287–296, 2004.
- Uchibe, E. Cooperative and competitive reinforcement and imitation learning. In Proc. of ICDL-EpiRob, 2018.
- Xie, L., Wang, S., Rosa, S., Markham, A., and Trigoni, N. Learning with Training Wheels: Speeding up Training with a Simple Controller for Deep Reinforcement Learning. In Proc. of ICRA 2018.
- Watkins, C.J.C.H., and Dayan, P. Q-learning. Machine Learning, 8: 279-292, 1992.