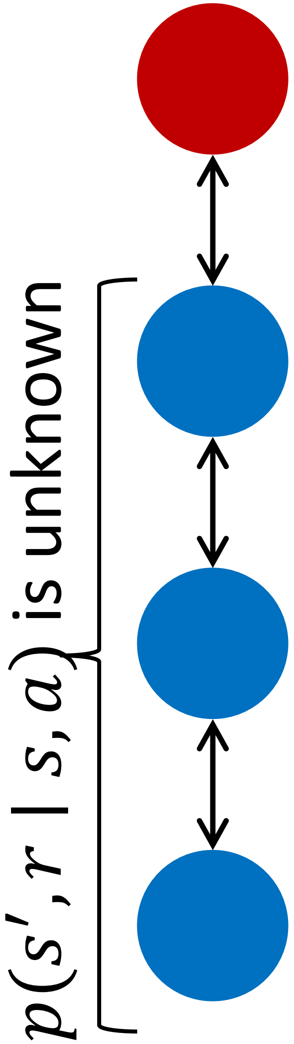
# Brain-Inspired Artificial Intelligence
# 3: Model-Based Reinforcement Learning

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 approaches to solve MDP problems
- Model-based methods
  - Policy iteration (= policy evaluation + policy improvement)
  - Value iteration

# 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-free approach

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

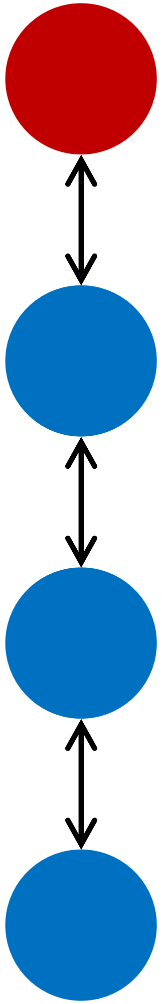
## Experience replay and model-free approach

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

## Model-free approach

- Train $V^{\pi}$, $Q^{\pi}$, or $\pi$ directly from samples drawn from $p$
- Discard experienced samples without reuse

# 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-free 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 $V^{\pi}$, $Q^{\pi}$, or $\pi$ directly from samples drawn from $p$
- Discard experienced samples without reuse

# Two Model-based Approaches

- Policy iteration: $\pi_0 \xrightarrow{\text{E}} V^{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} V^{\pi^1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi^* \xrightarrow{\text{E}} V^*$
  - Policy evaluation and policy improvement
  - 1. Initialize the policy. 2. Evaluate the corresponding value function. 3. Improve the policy.
- Value iteration: $V_0 \rightarrow V_1 \rightarrow V_2 \rightarrow \cdots \rightarrow V_\infty \rightarrow \pi^*$
  - 1. Initialize the value function. 2. Update the value function. 3. Retrieve the optimal policy.

# Policy Evaluation

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

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

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

$$= \sum_a \sum_{s',r} r\pi(\, a \mid s\,)p(s',r \mid s,a) + \gamma \sum_a \sum_{s',r} V^\pi(s')\pi(\, a \mid s\,)p(s',r \mid s,a)$$

$$= \bar{r}(s) + \gamma \sum_{s'} V^\pi(s')p(s' \mid s)$$

# Closed Solution of Bellman Expectation Equation

- Reminder: Bellman expectation equation

$$V^{\pi}(s) = \bar{r}(s) + \gamma \sum_{s'} V^{\pi}(s') p(s' \mid s) \qquad \text{for all } s$$

- 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

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

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

- If $\gamma < 1$, $\boldsymbol{v}$ is uniquely determined by $\boldsymbol{V} = (\boldsymbol{I} - \gamma \boldsymbol{P})^{-1} \boldsymbol{r}$ where $\boldsymbol{I}$ 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 $B_\pi$

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

  - $B_\pi$ is linear, but dependent on $\pi$

$$V_k \xrightarrow{B_\pi} V_{k+1} \xrightarrow{B_\pi} V_{k+2}$$

$$V_{k+1} = B_\pi V_k$$

$$V_{k+2} = B_\pi V_{k+1} = (B_\pi)^2\, V_k$$

- Initialize $V(s)$ arbitrary (except that the terminal state must be 0)

- Compute $V_{k+1}(s)$ by the following rule

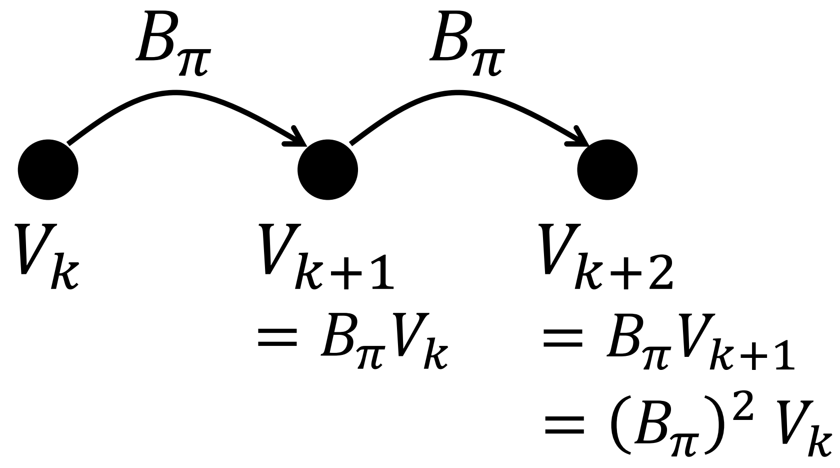
$$V_{k+1}(s) = B_\pi V_k(s) \quad \text{for all } s$$

# Iterative Policy Evaluation

- $V_k$ converges to the value function that satisfies the Bellman equation

$$\lim_{k\to\infty} V_k(s) = V^{\pi}(s) \qquad \text{for all } s$$

- In particular, $\pi$ need not be stochastic



# Policy Improvement

- After $V^{\pi}$ is computed by policy evaluation, we wanto find an improved policy $\pi'$ based on $V^{\pi}$

$$\begin{aligned}\pi'(s) &= \arg\max_a Q^{\pi}(s,a) \\ &= \arg\max_a \mathbb{E}[R_{t+1} + \gamma V^{\pi}(s_{t+1}) \mid s_t = s] \\ &= \arg\max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V^{\pi}(s')]\end{aligned}$$

- We do not need to maintain a stochastic policy

- After policy improvement, it is proved that

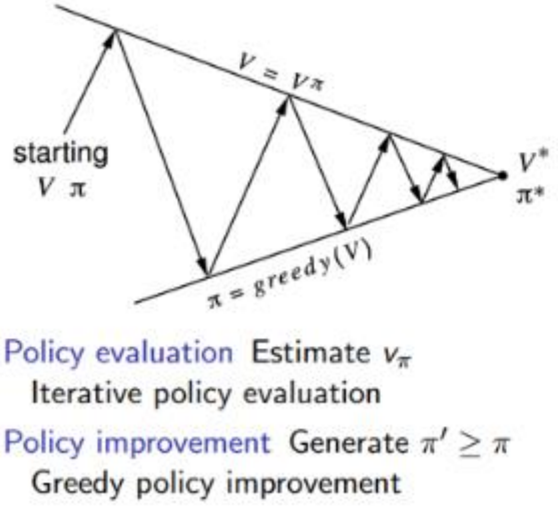
$$V^{\pi'}(s) \geq V^{\pi}(s)$$

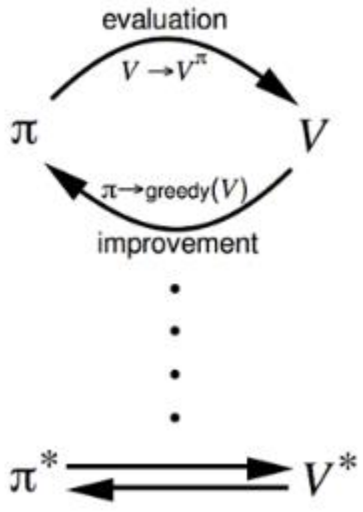
# Policy Iteration

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

$$\pi_0 \xrightarrow{\text{E}} V^{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} V^{\pi^1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi^* \xrightarrow{\text{E}} V^*$$

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



**Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$**

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

2. Policy Evaluation
   Loop:
      $\Delta \leftarrow 0$
      Loop for each $s \in \mathcal{S}$:
         $v \leftarrow V(s)$
         $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s))\big[r + \gamma V(s')\big]$
         $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   *policy-stable* $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
      *old-action* $\leftarrow \pi(s)$
      $\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$
      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

R. Sutton and A.G. Barto. (2018). Reinforcement Learning: An Introduction. (2nd edition). MIT Press.

# 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

$$V_0 \rightarrow V_1 \rightarrow V_2 \rightarrow \cdots \rightarrow V_\infty \rightarrow \pi^*$$

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

# Value Iteration

- 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 $\pi$

$$V_k \xrightarrow{B_*} V_{k+1} \xrightarrow{B_*} V_{k+2}$$

$$V_{k+1} = B_*V_k$$

$$V_{k+2} = B_*V_{k+1} = (B_*)^2\, V_k$$

- Initialize $V_0(s)$ arbitrary (except that the terminal state must be 0)

- Compute $V_{k+1}$ by the following rule

$$V_{k+1}(s) = B_*V_k(s) \quad \text{for all } s$$

# Value Iteration

- $V_k$ converges to the optimal value function that satisfies the Bellman optimality equation

$$\lim_{k \to \infty} V_k(s) = V^*(s) \qquad \text{for all } s$$

$$V_k \xrightarrow{B_*} V_{k+1} \xrightarrow{B_*} V_{k+2}$$

$$V_{k+1} = B_* V_k$$

$$V_{k+2} = B_* V_{k+1} = (B_*)^2 \, V_k$$

**Value Iteration, for estimating $\pi \approx \pi_*$**

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $v \leftarrow V(s)$
| $V(s) \leftarrow \max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
| $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \arg\max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$

R. Sutton and A.G. Barto. (2018). Reinforcement Learning: An Introduction. (2nd edition). MIT Press.

# Simulation: FrozenLake task

Open in Colab

- Discrete-state and discrete-action MDP task provided by OpenAI gym
  - Simplify the task by considering deterministic MDP
- Positive reward (+1) for entering the goal from left
- 0 otherwise
- Episode ends when entering a hole

# Summary

- Which is faster?
  - It depends on the problem
- VI takes more iterations than PI, but PI requires more time on each iteration
- PI must perform policy evaluation on each iteration which involves iteration
- VI is easier to implement since it does not require the policy evaluation step