

# 强化学习及其应用

Reinforcement Learning and Its Applications

## 第七章 模型与规划

**Model and Planning**

授课人：周晓飞

zhouxiaofei@iie.ac.cn

2018-6-28

# 第七章 模型与规划

## 7.1 模型学习

## 7.2 模型与规划

## 7.3 算法

# 第七章 模型与规划

## 7.1 模型学习

## 7.2 模型与规划

## 7.3 算法

## 问题描述

- *Last lecture:* learn **policy** directly from experience
- *Previous lectures:* learn **value function** directly from experience
- *This lecture:* learn **model** directly from experience
- and use **planning** to construct a value function or model
- Integrate learning and planning into a single architecture

## 问题描述

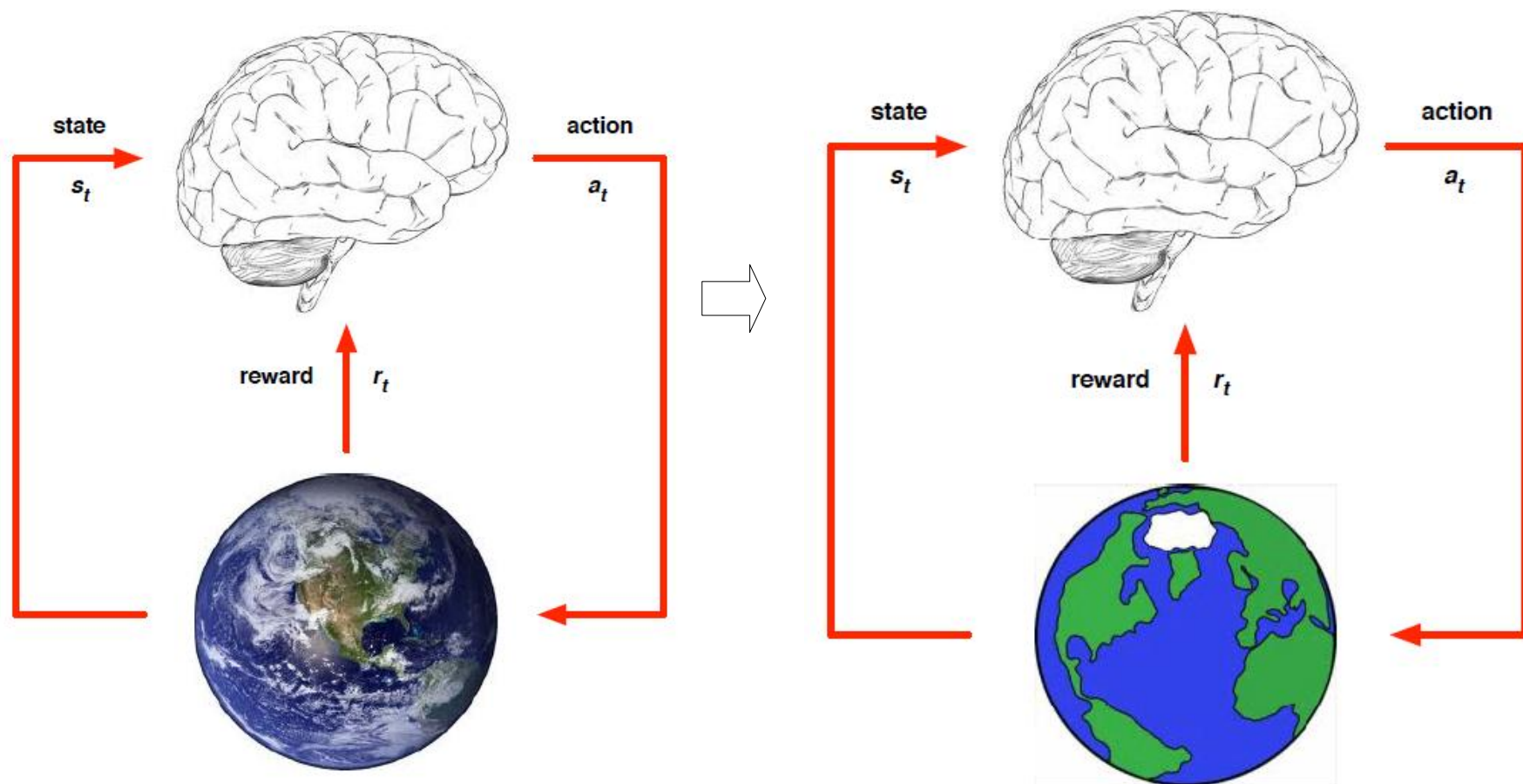
- *Last lecture*: learn **policy** directly from experience
- *Previous lectures*: learn **value function** directly from experience
- *This lecture*: learn **model** directly from experience
- and use **planning** to construct a value function or model
- Integrate learning and planning into a single architecture
- Model-Free RL
  - No model
  - **Learn** value function (and/or policy) from experience
- Model-Based RL
  - Learn a model from experience
  - **Plan** value function (and/or policy) from model

## 问题描述

- *Last lecture*: learn **policy** directly from experience
- *Previous lectures*: learn **value function** directly from experience
- *This lecture*: learn **model** directly from experience
- and use **planning** to construct a value function or model
- Integrate learning and planning into a single architecture
- Model-Free RL
  - No model
  - **Learn** value function (and/or policy) from experience
- Model-Based RL
  - Learn a model from experience
  - **Plan** value function (and/or policy) from model

# 模型学习

## 问题描述



## 问题描述

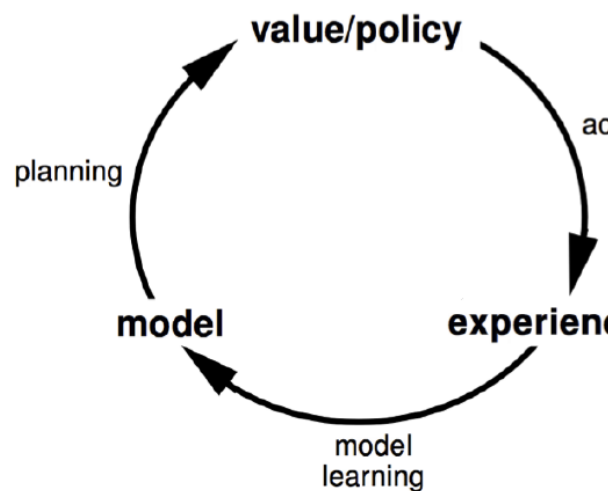
**基本流程**：源于已知 episodes from experiences, 学习 Model, 产生 episodes from Model, 由产生的 episodes, 学习 Policy 和 Value。

Advantages:

- Can efficiently learn model by supervised learning methods
- Can reason about model uncertainty

Disadvantages:

- First learn a model, then construct a value function  
⇒ two sources of approximation error





## 模型学习

### ■ 什么是模型？

- A *model*  $\mathcal{M}$  is a representation of an MDP  $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$ , parametrized by  $\eta$
- We will assume state space  $\mathcal{S}$  and action space  $\mathcal{A}$  are known
- So a model  $\mathcal{M} = \langle \mathcal{P}_\eta, \mathcal{R}_\eta \rangle$  represents state transitions  $\mathcal{P}_\eta \approx \mathcal{P}$  and rewards  $\mathcal{R}_\eta \approx \mathcal{R}$

$$s_{t+1} \sim \mathcal{P}_\eta(s_{t+1} \mid s_t, a_t)$$

$$r_{t+1} = \mathcal{R}_\eta(r_{t+1} \mid s_t, a_t)$$

- Can assume conditional independence between state transitions and rewards

$$\mathbb{P}[s_{t+1}, r_{t+1} \mid s_t, a_t] = \mathbb{P}[s_{t+1} \mid s_t, a_t] \mathbb{P}[r_{t+1} \mid s_t, a_t]$$

## 模型学习

### ■ 模型学习过程

- Goal: estimate model  $\mathcal{M}_\eta$  from experience  $\{s_1, a_1, r_2, \dots, s_T\}$
- This is a supervised learning problem

$$s_1, a_1 \rightarrow r_2, s_2$$

$$s_2, a_2 \rightarrow r_3, s_3$$

$$\vdots$$

$$s_{T-1}, a_{T-1} \rightarrow r_T, s_T$$

- Learning  $s, a \rightarrow r$  is a *regression* problem
- Learning  $s, a \rightarrow s'$  is a *density estimation* problem
- Pick loss function, e.g. mean-squared error, KL divergence, ...
- Find parameters  $\eta$  that minimise empirical loss

## 模型学习

### ■ 模型函数种类

- Table Lookup Model
- Linear Expectation Model
- Linear Gaussian Model
- Gaussian Process Model
- Deep Belief Network Model
- ...

## 例子 : Table Lookup Model

- Model is an explicit MDP,  $\hat{\mathcal{P}}, \hat{\mathcal{R}}$
- Count visits  $N(s, a)$  to each state action pair

$$\hat{\mathcal{P}}_{s,s'}^a = \frac{1}{N(s, a)} \sum_{t=1}^T \mathbf{1}(s_t, a_t, s_{t+1} = s, a, s')$$

$$\hat{\mathcal{R}}_s^a = \frac{1}{N(s, a)} \sum_{t=1}^T \mathbf{1}(s_t, a_t = s, a) r_t$$

- Alternatively
  - At each time-step  $t$ , record experience tuple  $\langle s_t, a_t, r_{t+1}, s_{t+1} \rangle$
  - To sample model, randomly pick tuple matching  $\langle s, a, \cdot, \cdot \rangle$

## 例子 : Table Lookup Model

### ■ AB Example

Two states  $A, B$ ; no discounting; 8 episodes of experience

$A, 0, B, 0$

$B, 1$

$B, 1$

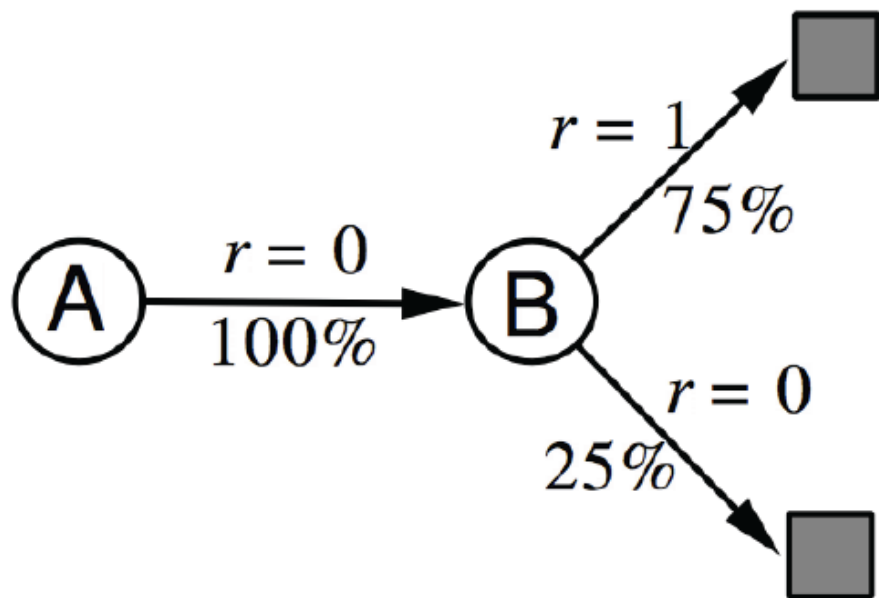
$B, 1$

$B, 1$

$B, 1$

$B, 1$

$B, 0$



We have constructed a **table lookup model** from the experience

# 第七章 模型与规划

7.1 模型学习

7.2 模型与规划

7.3 算法

## Planning with a Model

- A simple but powerful approach to planning
- Use the model **only** to generate samples
- **Sample** experience from model

$$s_{t+1} \sim \mathcal{P}_\eta(s_{t+1} \mid s_t, a_t)$$

$$r_{t+1} = \mathcal{R}_\eta(r_{t+1} \mid s_t, a_t)$$

- Apply **model-free** RL to samples, e.g.:
  - Monte-Carlo control
  - Sarsa
  - Q-learning
- Sample-based planning methods are often more efficient

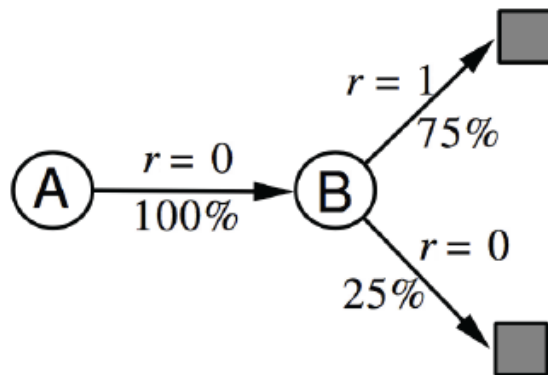
## Planning with a Model

### ■ AB Example

- Construct a table-lookup model from real experience
- Apply model-free RL to sampled experience

Real experience

A, 0, B, 0  
B, 1  
B, 1  
B, 1  
B, 1  
B, 1  
B, 1  
B, 1  
B, 0



Sampled experience

B, 1  
B, 0  
B, 1  
A, 0, B, 1  
B, 1  
A, 0, B, 1  
B, 1  
B, 0

e.g. Monte-Carlo learning:  $V(A) = 1$ ,  $V(B) = 0.75$



# 第七章 模型与规划

7.1 模型学习

7.2 模型与规划

**7.3 算法**

## Dyna

We consider two sources of experience

Real experience Sampled from environment (true MDP)

$$s' \sim \mathcal{P}_{s,s'}^a$$

$$r = \mathcal{R}_s^a$$

Simulated experience Sampled from model (approximate MDP)

$$s' \sim \mathcal{P}_\eta(s' \mid s, a)$$

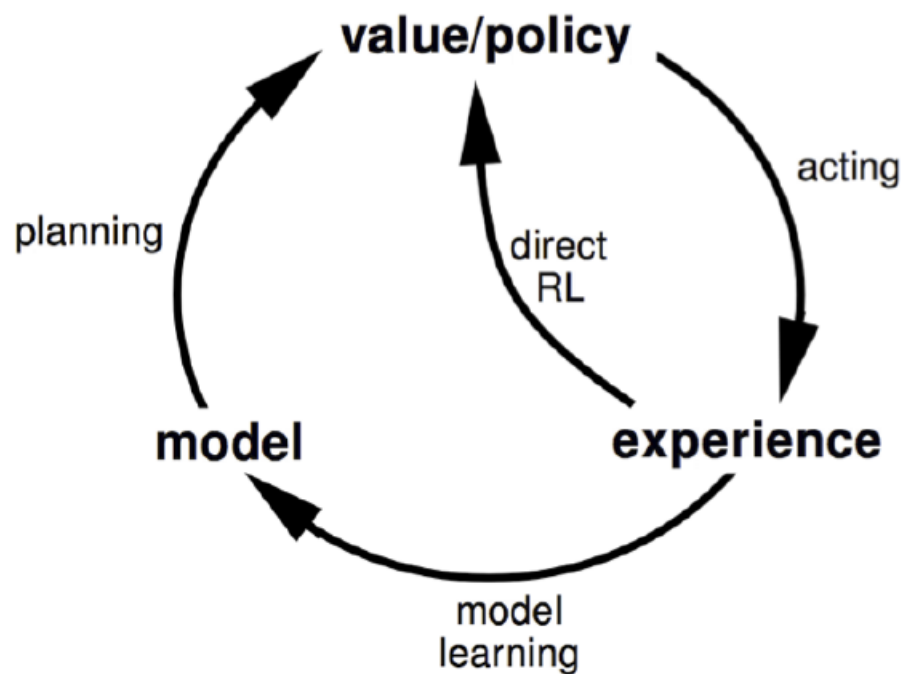
$$r = \mathcal{R}_\eta(r \mid s, a)$$

## Dyna

- Model-Free RL
  - No model
  - **Learn** value function (and/or policy) from real experience
- Model-Based RL (using Sample-Based Planning)
  - Learn a model from real experience
  - **Plan** value function (and/or policy) from simulated experience
- Dyna
  - Learn a model from real experience
  - **Learn and plan** value function (and/or policy) from real and simulated experience

**Dyna 方法综合了 Model-Free 和 Model-Based.**

## Dyna



## Dyna-Q Algorithm

Initialize  $Q(s, a)$  and  $Model(s, a)$  for all  $s \in \mathcal{S}$  and  $a \in \mathcal{A}(s)$

Do forever:

- (a)  $s \leftarrow$  current (nonterminal) state
  - (b)  $a \leftarrow \varepsilon$ -greedy( $s, Q$ )
  - (c) Execute action  $a$ ; observe resultant state,  $s'$ , and reward,  $r$
  - (d)  $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
  - (e)  $Model(s, a) \leftarrow s', r$  (assuming deterministic environment)
  - (f) Repeat  $N$  times:
    - $s \leftarrow$  random previously observed state
    - $a \leftarrow$  random action previously taken in  $s$
    - $s', r \leftarrow Model(s, a)$
    - $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
-

## Dyna-Q Algorithm

Initialize  $Q(s, a)$  and  $Model(s, a)$  for all  $s \in \mathcal{S}$  and  $a \in \mathcal{A}(s)$

Do forever:

- (a)  $s \leftarrow$  current (nonterminal) state
- (b)  $a \leftarrow \varepsilon$ -greedy( $s, Q$ )
- (c) Execute action  $a$ ; observe resultant state,  $s'$ , and reward,  $r$
- (d)  $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
- (e)  $Model(s, a) \leftarrow s', r$  (assuming deterministic environment)

Model-Free

- (f) Repeat  $N$  times:

- $s \leftarrow$  random previously observed state
  - $a \leftarrow$  random action previously taken in  $s$
  - $s', r \leftarrow Model(s, a)$
  - $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$

Model-Based

Simulated experience & Planning

# 本讲参考文献

1. Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. (Second edition, in progress, draft).
2. David Silver, Slides@ 《Reinforcement Learning: An Introduction》, 2016.