# 强化学习基本原理及编程实现：概论

郭宪

2019.09.08

人工智能学院

College of Artificial Intelligence

南開大學

Nankai University

# 关于本门课

1. 强化学习概述及多臂赌博机(09.08)

第一次实验：井字游戏和多臂赌博机（09.12）

2. 马尔科夫决策过程及数学基础（09.22）

3. 从动态规划到强化学习（09.29）

第二次实验：MDP、策略迭代、值迭代（10.10）

4. 表格型强化学习：蒙特卡洛和时间差分（10.13）

第三次实验：蒙特卡洛方法及时间差分方法（10.17）

5. 函数逼近方法及深度学习框架介绍（10.20）

第四次实验：ModelArts 实验（10.24）

6. 基于函数逼近的强化学习方法（10.27）

第五次实验：深度强化学习DQN（10.31）

7. 基于策略梯度的强化学习方法（11.03）

第六次实验：基于策略梯度的方法（11.07）

8. 基于置信域的强化学习方法（11.10）

9. 基于确定性策略的强化学习方法（11.17）

# 关于本门课

10. 智能芯片介绍（11.24）

第七次实验：智能硬件上的应用（11.28）

11. 多智能体强化学习(12.01)

第八次实验：多智能体强化学习实验（12.05）

12. 基于模型的强化学习方法（12.08）

13. 逆向强化学习（12.15）

14. 强化学习前沿：分层强化学习（12.22）

15. 强化学习前沿：元强化学习(12.29)

16. 强化学习前沿：贝叶斯强化学习（01.05）

17. 强化学习前沿：终身学习（01.12）

参考材料：

R.S Sutton, A.G Barto, Reinforcement Learning: An Introduction

https://github.com/ShangtongZhang/reinforcement-learning-an-introduction

郭宪, 方勇纯，深入浅出强化学习：原理入门

郭宪, 宋俊潇，方勇纯，深入浅出强化学习：编程实战

（待出版）https://github.com/Teacher-Guo/RL_code

各种公开发表的论文

# 关于本门课

理论基础：

高等数学（微积分）

概率与数理统计（概率分布，期望，方差）

最优控制（可选，传统控制方法）

编程基础：

数据结构，C（C++），基本算法，Tensor Flow /pytorch

机器学习基础：

神经网络，优化算法

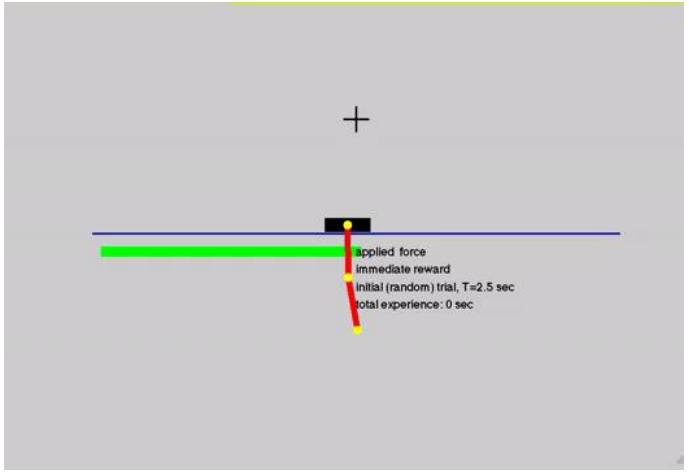
**持久的学习热情和韧劲！**

平时作业50%，　最后考试50%

# 强化学习概述

1. 强化学习能解决的问题
2. 强化学习与其他学科的联系
3. 强化学习如何去解决问题
4. 强化学习算法历史
5. 强化学习的分类
6. 强化学习的发展趋势
7. 强化学习课程路线图

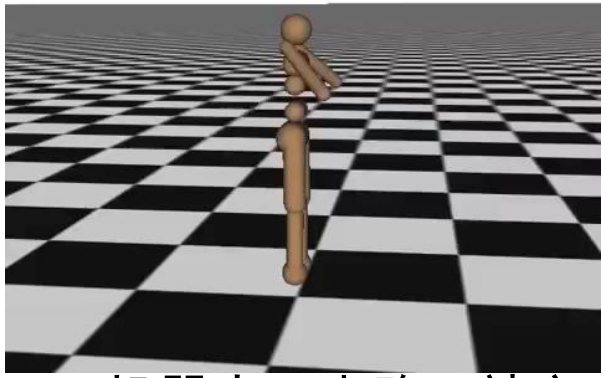
# 1. 强化学习能解决的问题




非线性控制

Iteration 0 视频游戏

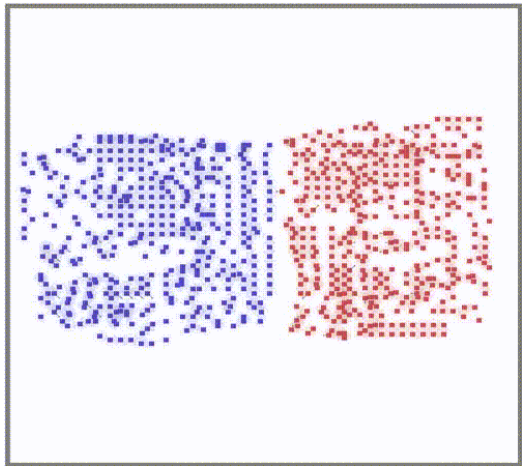

AlphaGo



机器人，走路，站立

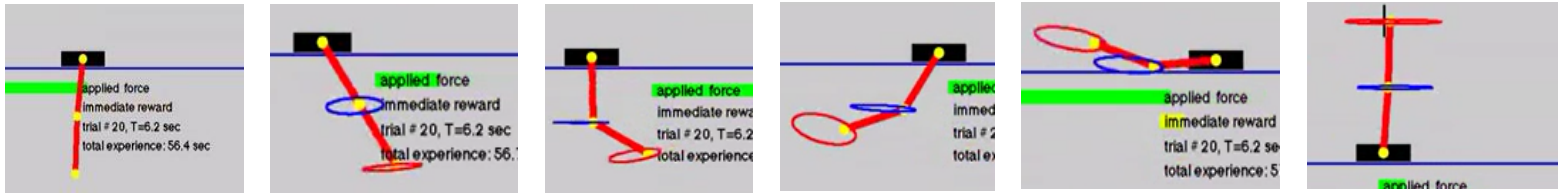
# 1. 强化学习能解决的问题


启元世界，多智能体协作（星际）



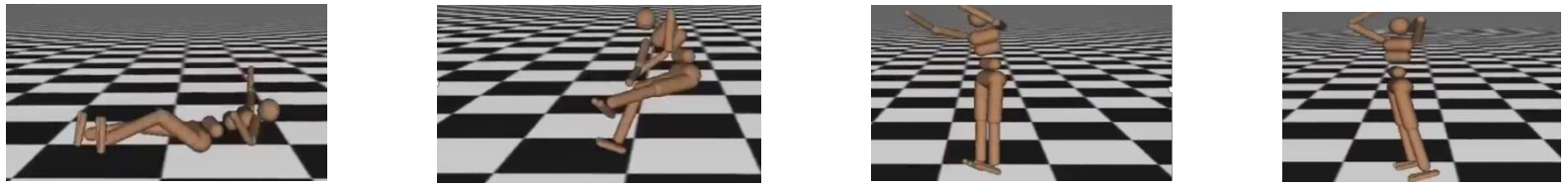
海量智能体（上海交通大学）

# 1. 强化学习能解决的问题




图A 非线性系统二级倒立摆


图B AlphaGo与柯洁第二番棋



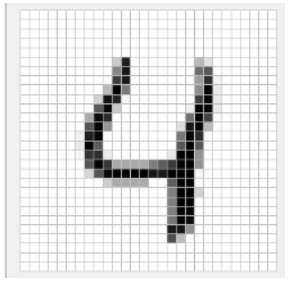
图C 机器人学习站立

# 1. 强化学习能解决的问题

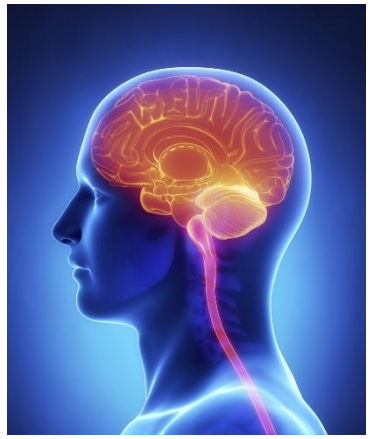
除了非线性控制、视频游戏、下棋、机器人，强化学习还可用于人机对话、无人驾驶、机器翻译、文本序列预测等领域

强化学习针对的是**智能决策问题**

深度学习针对的是**智能感知问题**



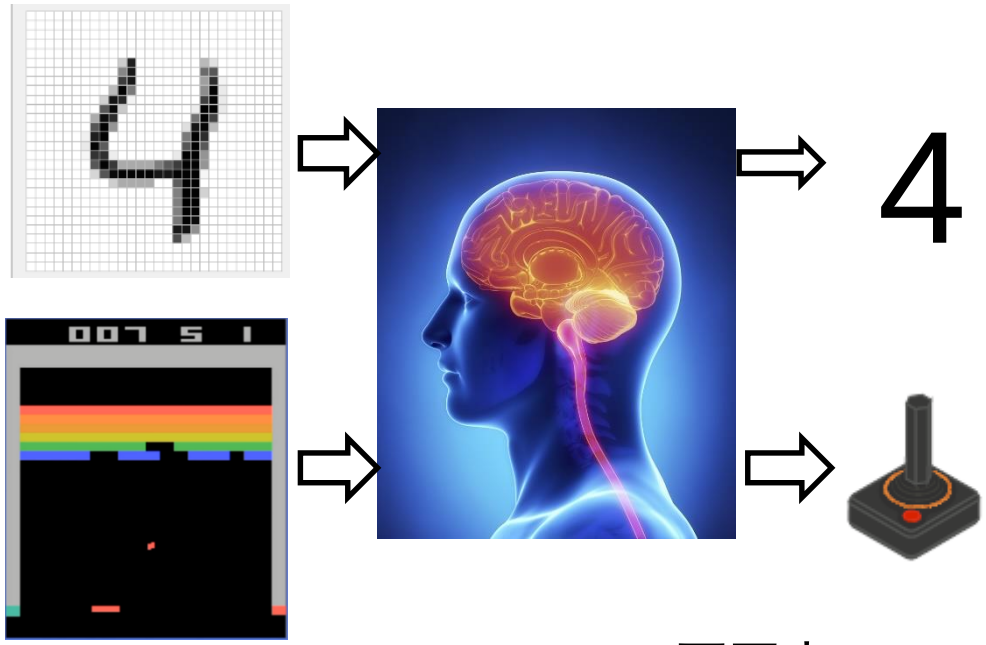
智能感知



4


智能决策

# 2. 强化学习与其他机器学习的联系和区别




相同点：

都是从数据中学习，逐步改善性能。

不同点：

1. 需要的数据类型不同。监督学习需要标签数据，强化学习需要交互数据。

2. 优化目标不同

# 2. 强化学习与其他优化方法

序贯最优决策的方法：

当状态空间很小时，可利用线性规划，如资源分配，调度等 ⟶ 通过凸优化

当模型已知，回报函数解析时，可用动态规划（最优控制方法） ⟶ 解HJB方程

状态方程：

$$\dot{X}=f\left(t,X,U\right)\quad X\left(t_0\right)=X_0$$

性能指标函数：

$$J\left[x\left(t_0\right),t_0\right]=\phi\left[X\left(t_f\right),t_f\right]+\int_{t_0}^{t_f}L\left(x(t),u(t),t\right)$$

最优控制问题：

$$V\left(X,t\right)=\min_{u\in\Omega}\left\{\phi\left[X\left(t_f\right),t_f\right]+\int_{t_0}^{t_f}L\left(x(t),u(t),t\right)\right\}$$

$$V\left(X,t\right)=\min_{u\in\Omega}\left\{\phi\left[X\left(t_f\right),t_f\right]+\int_{t_0}^{t_f}L\left(x(t),u(t),t\right)\right\}$$

$$=\min_{u\in\Omega}\left\{\int_{t_0}^{t_0+dt}L\left(x(\tau),u(\tau),\tau\right)d\tau+V\left(X+\Delta X,t+dt\right)\right\}$$

$$=\min_{u\in\Omega}\left\{L\left(x(t),u(t),t\right)dt+V\left(X,t\right)+\frac{\partial V}{\partial X}fdt+\frac{\partial V}{\partial t}dt+\varepsilon\right\}$$

⇩

$$-\frac{\partial V}{\partial t}=\min_{u(t)\in U}\left\{L\left(x(t),u(t),t\right)+\frac{\partial V}{\partial X^T}f\left[x(t),u(t),t\right]\right\}$$

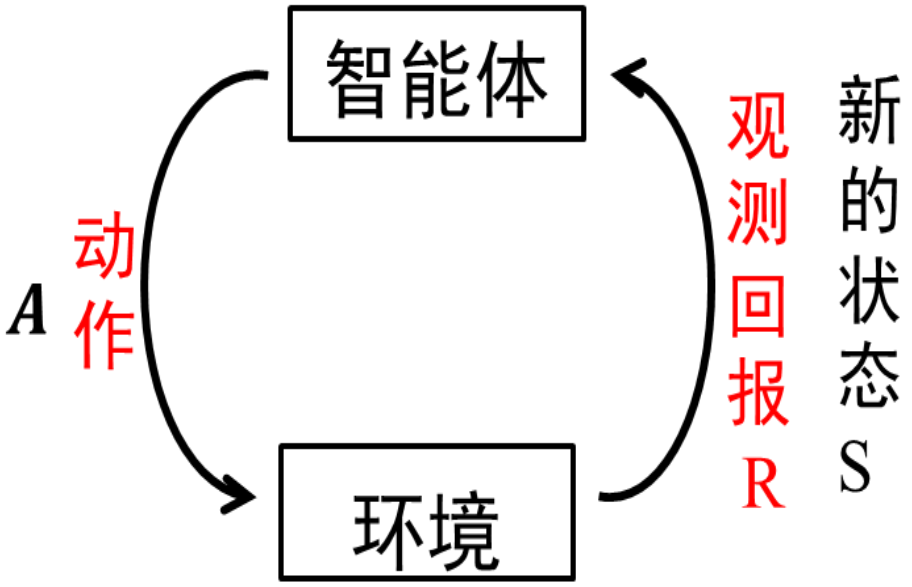
# 2. 强化学习与其他优化方法

当控制策略简单时，进化算法。 ⟶ 遗传算法

当模型未知或非常复杂，回报函数未知，强化学习算法。

强化学习是基于采样（与环境交互）的方法，学习最优策略 ⟶ 通过交互的方法

# 3. 强化学习如何解决问题




状态转移概率 $P(S_{t+1}|S_t, a)$

学习目标： $\pi(\cdot|s): s \mapsto a$

使得长期累积回报（值函数）最大

与环境交互，产生数据流：

$$s_0 \xrightarrow{a_1} (r_1, s_1) \xrightarrow{a_2} (r_2, s_2) \rightarrow \cdots$$

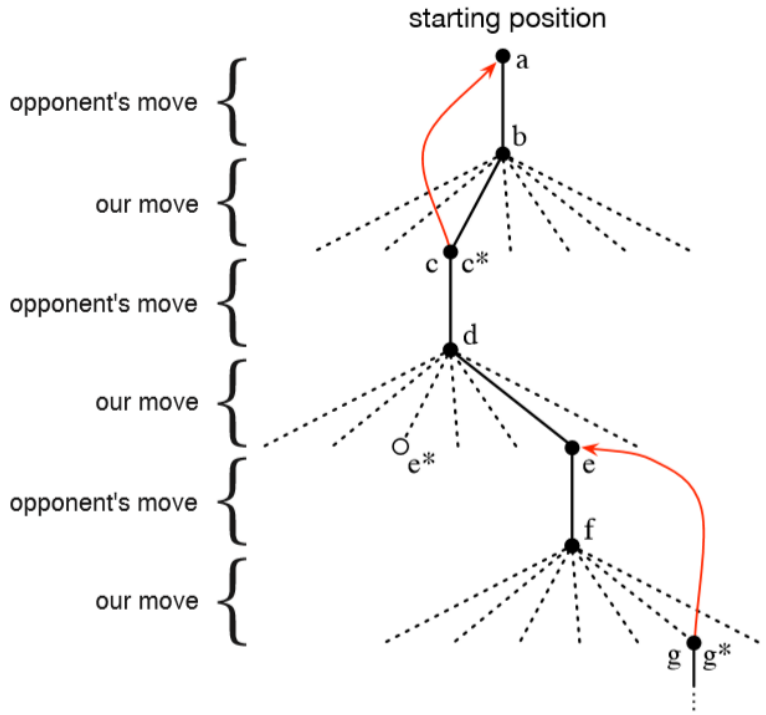
智能体利用数据进行学习，优化自身行为。

强化学习的特征：

1. 试错探索
2. 延迟回报

学习过程可以纳入马尔科夫决策过程的框架中

# 3. 强化学习如何解决问题




采样过程

| | | |
|---|---|---|
| X | O | O |
| O | X | X |
| | | X |

井字游戏

更新过程： $V(s_t) \leftarrow V(S_t) + \alpha[V(s_{t+1}) - V(s_t)]$

# 4. 强化学习算法历史

**1. 1998年前，基本理论框架形成。**

回报函数 r

Sutton 的书 《Reinforcement Learning An Introduction》

基于值函数的方法

**2. 1998年到2013年，各种直接策略搜索的方法出现**

基于策略梯度的RL
基于EM的RL
基于路径积分（Path Integral）的RL
基于回归的RL
基于模型的RL

**3. 2013年到现在，深度强化学习火热**

深度学习强大的表示能力+强化学习强大的决策能力= 强强联合超越人类

DQN，DNC，AlphaGo等

# 5. 强化学习的分类

根据是否依赖模型分为：基于模型的强化学习和无模型的强化学习

基于模型的强化学习算法：用数据先学习系统模型，然后基于模型得到最优策略

无模型的强化学习算法：直接通过交互数据得到最优策略

根据策略更新方法：基于值函数的强化学习， 基于直接策略搜索的强化学习，Actor-Critic的方法。

基于值函数的方法：求出最优值函数，然后重构出最优策略

基于直接策略搜索的方法：直接在策略空间进行搜索

Actor-Critic方法：同时逼近值函数和最优策略，类似于策略迭代

# 5. 强化学习的分类

根据回报函数是否已知分为：正向强化学习和逆向强化学习

正向强化学习：从回报学到最优策略

逆向强化学习：从专家示例中学到回报函数

根据任务大小和多少分为：分层强化学习、元强化学习、多智能体强化学习、迁移学习等

# 6. 强化学习的发展趋势

1. 强化学习算法与深度学习的结合会更加紧密。

DQN的成功离不开CNN---值函数表示

Alpha Zero的成功离不开CNN—策略网络表示

2. 强化学习算法与专业知识结合得将更加紧密。

Alpha Zero 的成功离不开围棋中发展很好的蒙特卡罗树搜索

3. 强化学习算法理论分析会更强，算法会更稳定和高效。

新的探索策略，数据效率的提升。来自统计学的理论知识将会持续发力

4. 强化学习算法跟脑科学、认知神经科学和记忆会更紧密。

大脑的记忆机理，模型和无模型联合机理的利用将飞速提升AI智能

# 6. 强化学习的发展趋势

1. 贝叶斯强化学习

融合推理能力，可解决POMDP问题

2. 分层强化学习

解决大规模学习问题

3. 元强化学习

解决对任务学习

4. 多智能体强化学习

博弈：合作，竞争，混合

# 7. 强化学习课程的路线图

1. 搞清楚**马尔科夫决策过程**的概念

2. 抓住强化学习的基本迭代过程：策略评估和策略改善

3. 掌握强化学习最常用的两种方法：基于值函数的方法和基于直接策略搜索的方法

4. 强化学习的其他方法：AC框架，基于模型的强化学习，基于记忆的强化学习等等

# 第一个强化学习的例子：多臂赌博机

目的：理解强化学习中动作的评估

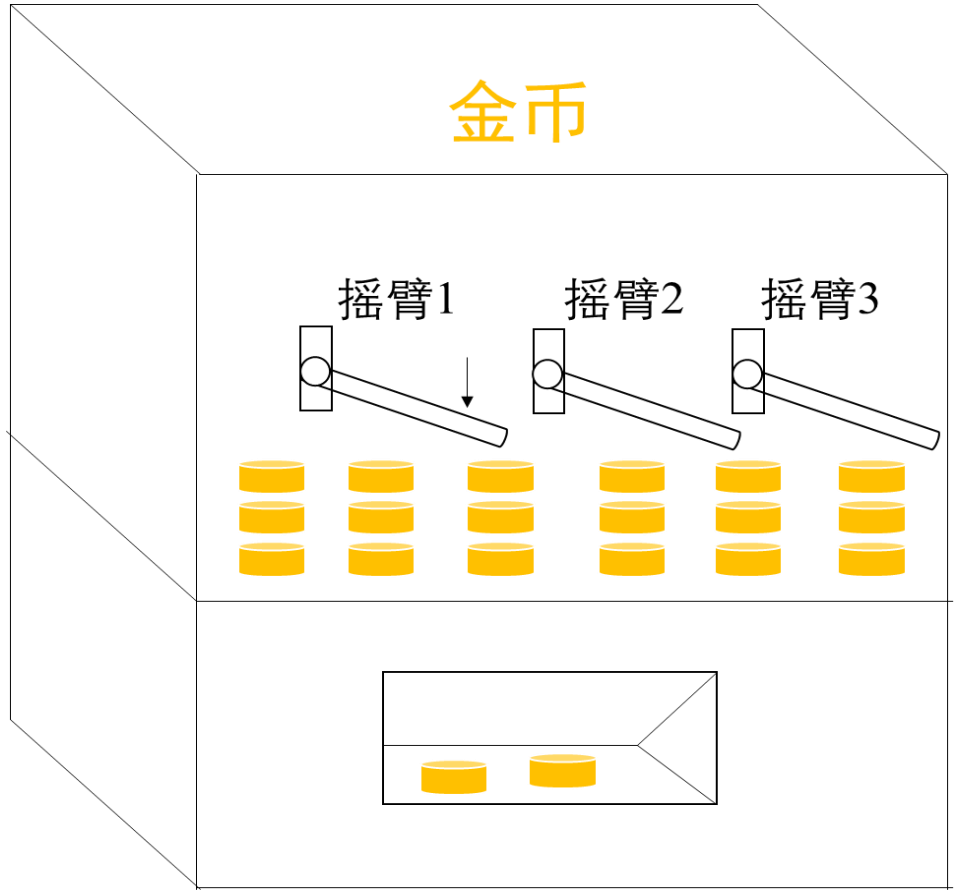
人工智能学院

College of Artificial Intelligence

南開大學

Nankai University

# 强化学习极其简单的例子：多臂赌博机




多臂赌博机：

K个臂

摇动每个臂时，得到不同概略分布的回报

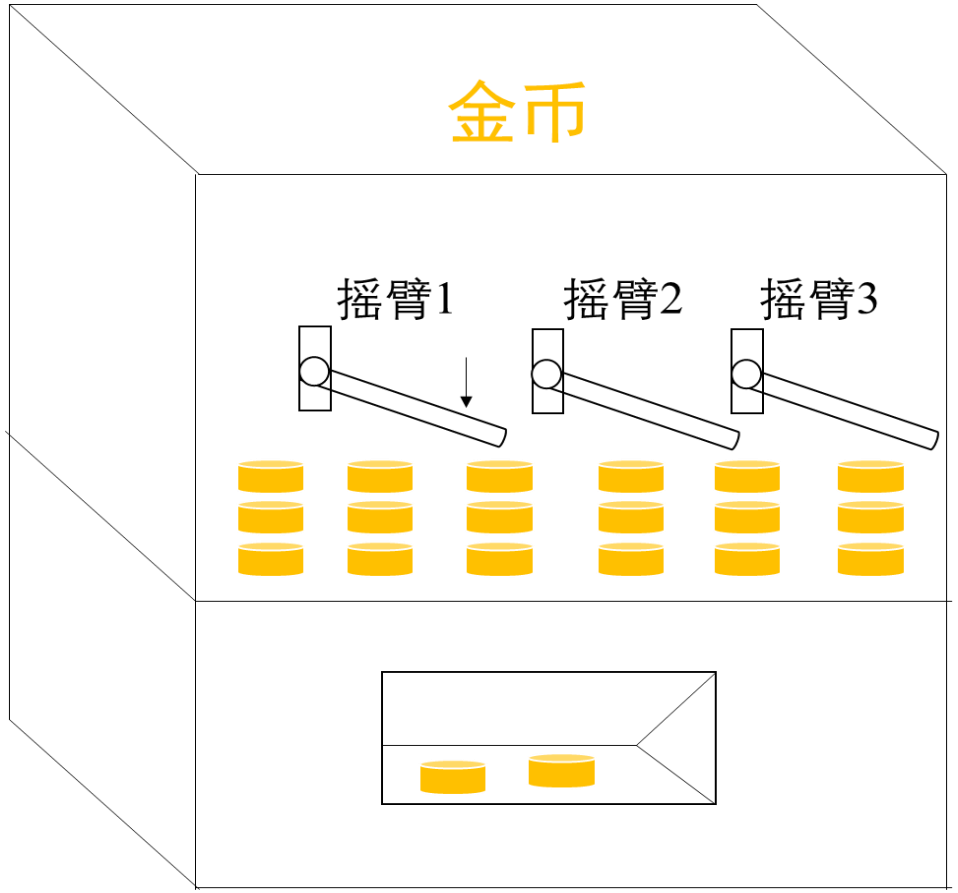
目的：

如何摇动N次得到最高的回报

问题：

如何通过学习知道，摇哪个臂能得到最好的回报

# 强化学习极其简单的例子：多臂赌博机




动作：a = [0, 1, 2]

立即回报：r，服从三个不同的回报

状态：s

目标：在状态s， 最优的动作

定义动作的值：$q_*(a) = \mathbb{E}[R_t \mid A_t = a]$

强化学习：利用回报r学习最优的动作，学习动作的评估

$$s \xrightarrow[r=0.2]{a=0} s_T$$

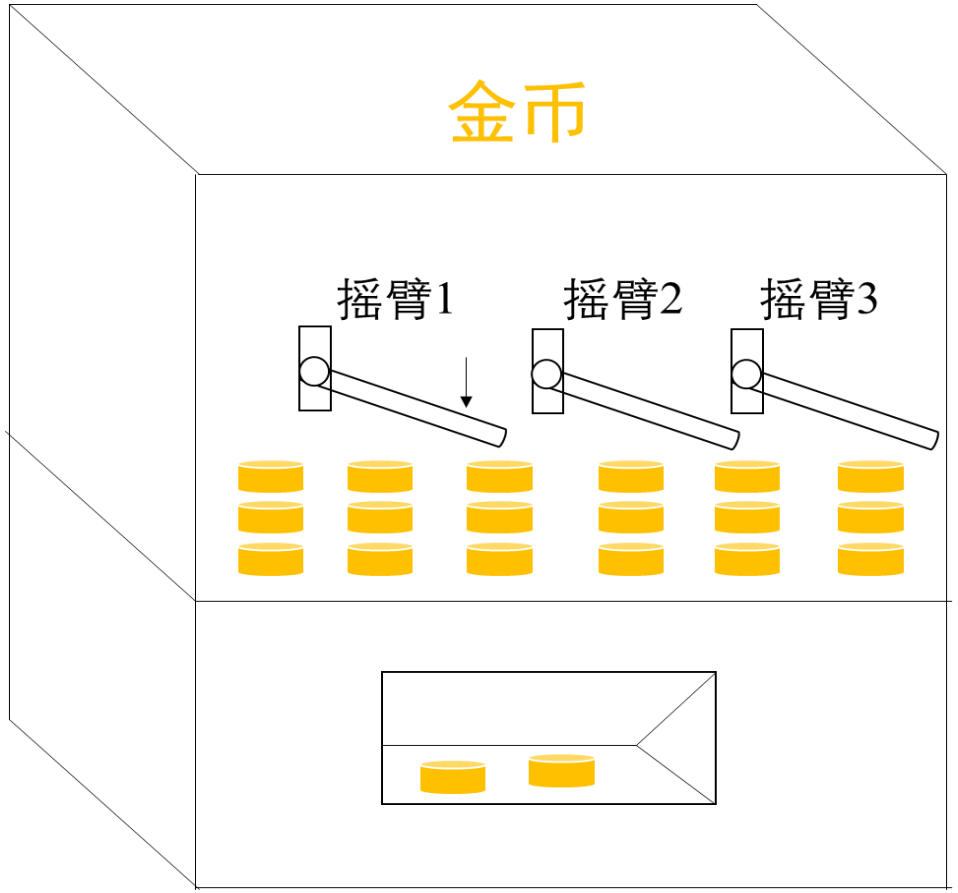
$$s \xrightarrow[r=0.3]{a=1} s_T$$

$$s \xrightarrow[r=0.6]{a=2} s_T$$

定义每个动作的估计值：

$$Q(a) = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathrm{II}_{A_i=a}}{\sum_{i=1}^{t-1} \mathrm{II}_{A_i=a}}$$

# 强化学习极其简单的例子：多臂赌博机




$$s \xrightarrow[r=0.2]{a=0} s_T,\ q[0]=0.2$$

$$s \xrightarrow[r=0.3]{a=1} s_T,\ q[1]=0.3$$

$$s \xrightarrow[r=0.6]{a=2} s_T, q[2]=0.6$$

在每个杆都试过一次后，你如何选下一个杆？

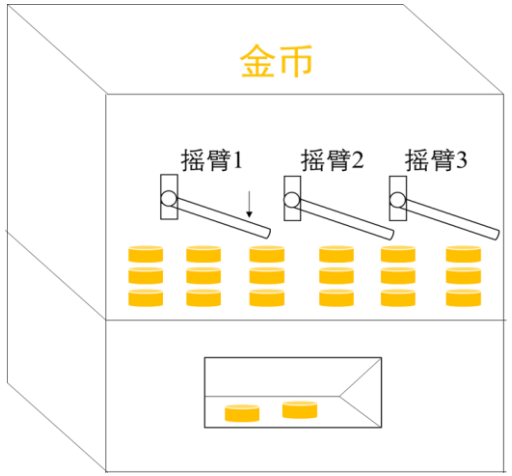
贪婪策略：$\mathrm{a}=\arg\max_a q(a)$ exploitation

选择非贪婪动作 exploration

探索-利用平衡策略与环境进行交互：

$$\varepsilon-greedy: a=\begin{cases} arg\max_a q(a) & with\ probability\ 1-\varepsilon \\ a\ random\ action & with\ probability\ \varepsilon \end{cases}$$

# 强化学习极其简单的例子：多臂赌博机




**A simple bandit algorithm**

Initialize, for $a = 1$ to $k$:
$Q(a) \leftarrow 0$
$N(a) \leftarrow 0$

Loop forever:
$A \leftarrow \begin{cases} \text{argmax}_a Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$
$R \leftarrow bandit(A)$
$N(A) \leftarrow N(A) + 1$
$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}[R - Q(A)]$

增量式计算：

$$
\begin{aligned}
Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\
&= \frac{1}{n}\left( R_n + \sum_{i=1}^{n-1} R_i \right) \\
&= \frac{1}{n}\left( R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i \right) \\
&= \frac{1}{n}\left( R_n + (n-1)Q_n \right) \\
&= \frac{1}{n}\left( R_n + nQ_n - Q_n \right) \\
&= Q_n + \frac{1}{n}[R_n - Q_n]
\end{aligned}
$$

# 强化学习极其简单的例子：多臂赌博机

**A simple bandit algorithm**

Initialize, for $a = 1$ to $k$:

$Q(a) \leftarrow 0$

$N(a) \leftarrow 0$

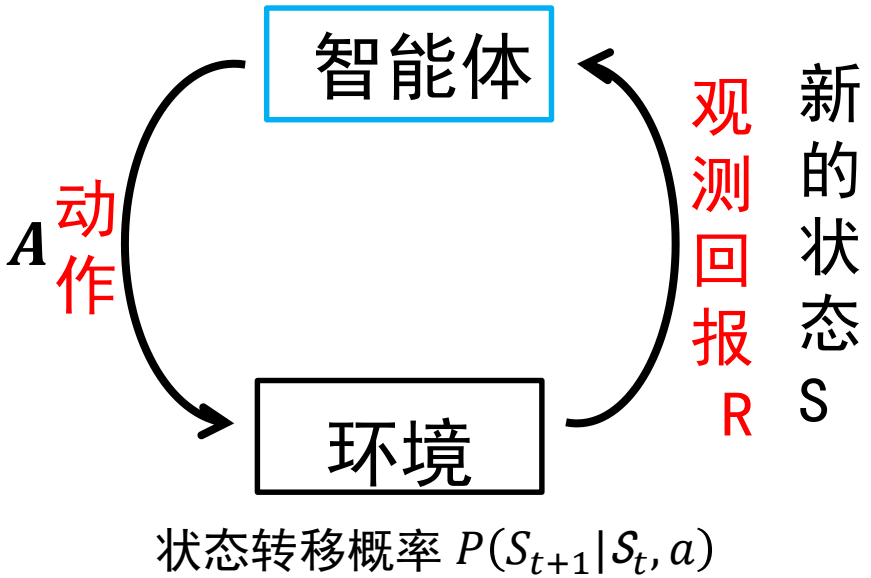
Loop forever:

$$A \leftarrow \begin{cases} \operatorname{argmax}_a Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$

$R \leftarrow bandit(A)$

$N(A) \leftarrow N(A) + 1$

$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\left[R - Q(A)\right]$




状态转移概率 $P(S_{t+1}|S_t, a)$

所有强化学习算法包括且只包括两个过程：

采集数据：利用探索-利用平衡策略进行采集数据

学习：利用采集到的数据优化当前策略

# 强化学习极其简单的例子：多臂赌博机

Upper-Confidence-Bound 动作选择

$$A_t = \arg\max_a [Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}}]$$

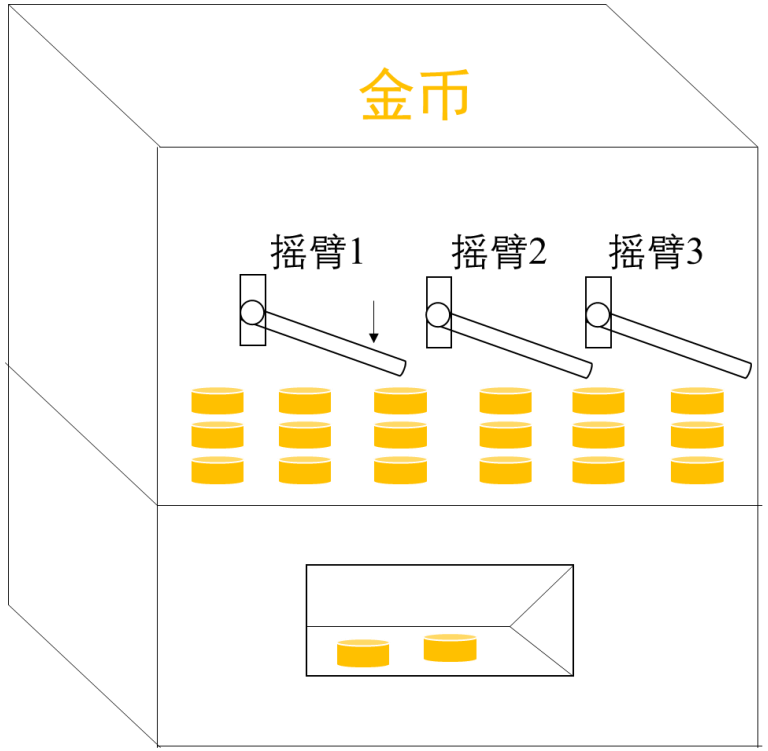
↓

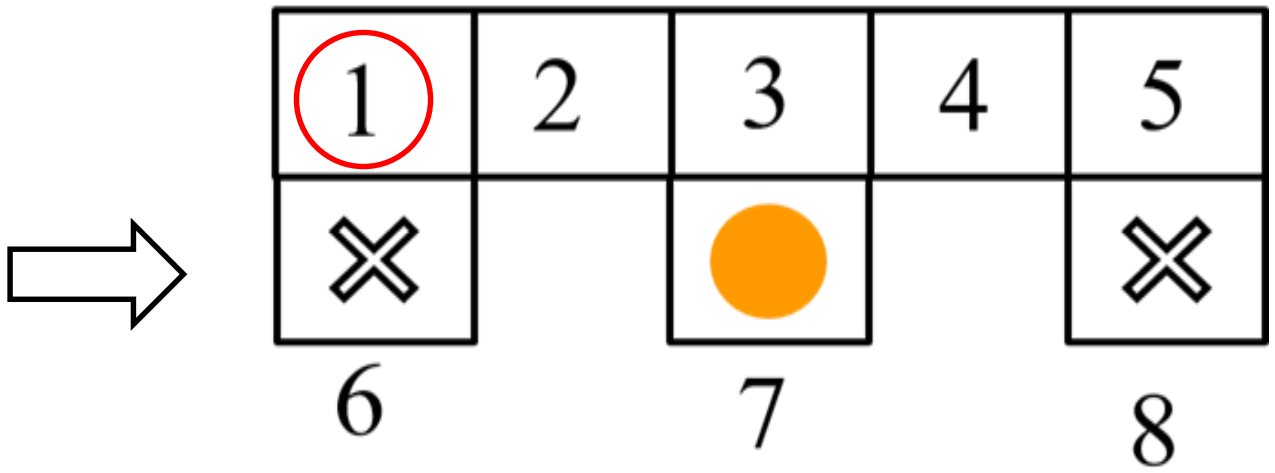
不确定性的度量

玻尔兹曼分布：动作选择

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

# 从多臂赌博机到马尔科夫决策过程




单状态 ⟹ 多状态 ⟹ 状态与状态之间符合马尔科夫性

# Python基础

## 1. print 语句: print()

```
print('hello reinforcement learning')
```

1.1 打印字符串：%s

```
print('My name is %s'%('bao zi xian er'))
```

1.2 打印整数：%d

```
print("I'm %d years old"%(31))
```

1.3 打印浮点数：%f

```
print("I'm %f meters in height"%(1.75))
```

1.4 打印浮点数,并保留两位有效数字：%f

```
print("I'm %.2f meters in height"%1.75)
```

1.5 当然也可以打印中文,中英混合

```
print("老师how萌傻！")
```

## 2. 条件语句: if…else…

```
score =700
if score >700:
    print("上清华或北大！")
else:
    print("复读")
```

```
score = 600
if score >700:
    print("上清华")
elif score>=650:
    print("上其他双一流大学")
elif score > 600 or score==600:
    print("上一本")
else:
    print("复读")
```

# Python基础

## 3. 循环语句：

### 3.1. for… in ， 依次将list或tuple中的每个元素迭代出来

```
a=[1,3,5,7,9]
for i in a:
    if i==1:
        print("10以内的奇数为\n%d"%i)
    else:
        print(i)
```

更多例子：

```
b=["天","地","玄","黄"]
for i in b:
    print(i)
for i in range(100):
    print(i)
```

### 3.2.While循环，只要条件满足，就一直循环下去

```
i=0
while i<100:
    print(i)
    i+=1
```

continue和break应用

```
while i<100:
    if i<50:
        i+=1
        continue
    print(i)
    i+=1
    if i>80:
        break
```

跳出本次循环，不往下继续执行

结束大循环

# Python基础

## 4. 函数定义

利用 def fun(x) 定义函数，其中fun为定义的函数名，x为参数名。

实例：

```
def step(s, a):
    s_next = s+a*0.01
    return s_next
if __name__=="__main__":
    print(step(2,3))
```

## 5. 类，面向对象，对象为程序的基本单元。类包括成员变量和成员函数

```
class maze:
    def __init__(self, dt):
        #成员变量用self
        self.dt = dt
    #成员函数
    def step(self,s,a):
        s_next = s+a*self.dt
        return s_next
if __name__=="__main__":
    maze1=maze(dt=0.01)
    s_next=maze1.step(2,3)
    print(s_next)
```

# Numpy 基础

1. 创建矩阵

```
import numpy as np
from numpy import *
A = np.array([[1,2,3],[2,4,6]])
```

2. numpy创建的矩阵的属性

维数：ndim

```
print("The dimision of A is %d"%A.ndim)
```

形状：shape

```
print("The shape of A is", A.shape)
```

大小：size

```
print("The size of A is %d"%A.size)
```

3. 创建数组

```
B = np.array([1,2,3])
```

将数组变为矩阵（加一个维度）：

```
B=B[np.newaxis,:]
```

4. 创建全零矩阵

```
C = np.zeros((3,3))
print(C)
```

5. 利用np.arange函数创建整数数组

```
D = np.arange(10)
print(D)
```

6. 利用reshape函数改变数据的形状

```
E= D.reshape(2,5)
print(E)
```

# Numpy 基础

6. 矩阵乘法：

对应元素相乘： `print(A*B)`

矩阵相乘： `print(np.dot(A,b))`

7. 矩阵转置：

```
print(np.transpose(A))
```

8. 矩阵元素的访问

```
print(A[1,1])
print(a1[0,:])
print(a1[0,1:2])
print(a1[0,-1])
```

9. 矩阵的合并

行合并：`print(np.hstack((a1,a2)))`

列合并：`print(np.vstack((a1,a2)))`

10. 常用的函数：sin, cos, exp()

```
print(np.sin(1))
print(np.cos(1))
print(np.exp(1))
```

# 本次作业

1. 阅读sutton书的前两章，并写不少于500字的读书笔记

2. 每四人组队，并阅读第1章和第2章的代码

3. 在原代码基础上进行修改，每个小组训练4*4的四子游戏，训练时间限制在2个小时内，并进行小组间对抗，根据对抗结果获得本次作业的分数。

作业发到邮箱：2120180394@mail.nankai.edu.cn

第一次实验：井字游戏和多臂赌博机（09.12）