《人工智能及应用》

第八讲：强化学习

授课人：张 鑫 zhangxin@uestc.edu.cn

专　业：机器人工程

提纲

1、概述

2、Q-learning

3、深度强化学习DQN

1.1基本概念

1 人与环境的交互模型

生活中常见的学习过程：

1）人通过观察环境景色，获得环境的状态； (观察迷宫路线)

2）经过一系列思考后，决策出一个动作； (走一步)

3）动作与环境作用，出现新的环境景色；并反馈奖励 (走出迷宫or No)

4）更新大脑中对环境的认识，及更新决策方法；

返回第一步；

1、概述

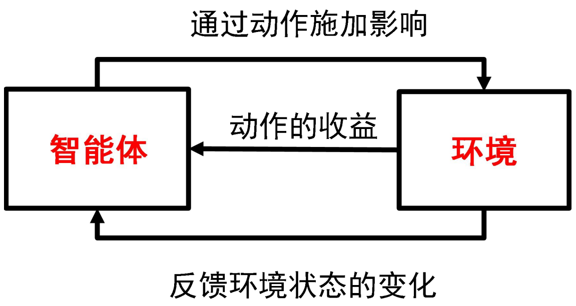
1.1基本概念

1 人与环境的交互模型

生活中常见的学习过程：

1）人通过观察环境景色，获得环境的状态； (向前走一步)
2）经过一系列思考后，决策出一个动作； (撞到了树上)
3）动作与环境作用，出现新的环境景色；并反馈奖励 (疼痛)
4）更新大脑中对环境的认识，及更新决策方法； (避开障碍)
返回第一步；

强化学习模仿了这个过程，在智能体与环境的交互中，学习能最大化收益的行动模式



1、概述

1.1基本概念

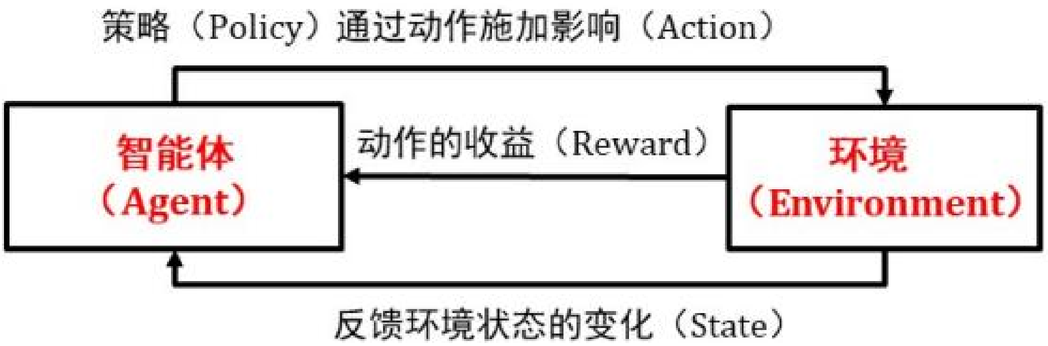
1 人与环境的交互模型

智能体（agent）

- 按照某种策略（policy），根据当前的状态（state）选择合适的动作（action）
- 状态指的是智能体对环境的一种解释
- 动作反映了智能体对环境主观能动的影响，动作带来的收益称为奖励（reward）
- 智能体可能知道也可能不知道环境变化的规律

环境（environment）

- 系统中智能体以外的部分
- 向智能体反馈状态和奖励
- 按照一定的规律发生变化



1 人与环境的交互模型

状态（state）

- 可以理解为智能体对环境的一种理解和编码，包含了对智能体采取决策产生影响的信息。

动作（action）

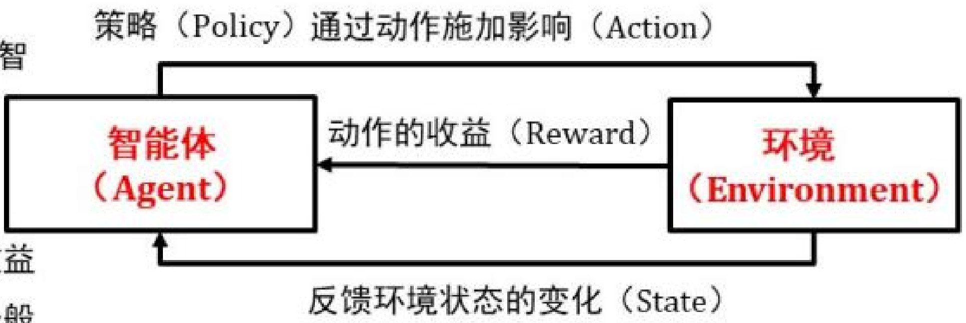
- 动作反映了智能体对环境主观能动的影响，动作带来的收益称为奖励（reward）
- 智能体可能知道也可能不知道环境变化的规律

策略（policy）

- 智能体在所处状态下去执行某个动作的依据，智能体可根据一个策略来旋转应该采取的动作。

奖励（reward）

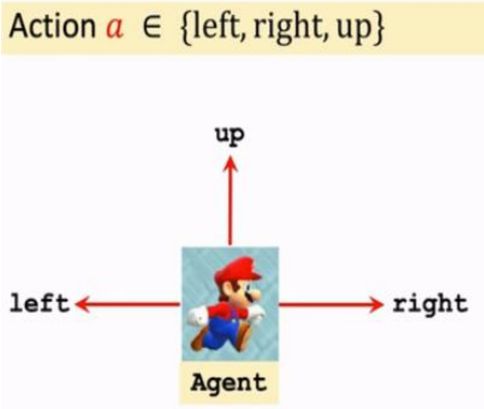
- 智能体序贯式采取一系列动作后从环境获得收益
- 注意奖励概念是显示中奖励和惩罚的统合，一般用正值来代表实际的惩罚

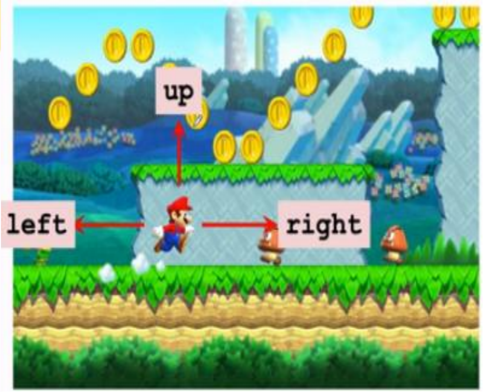


1.1基本概念

1 人与环境的交互模型

状态（state）、智能体（agent）、动作（action）、策略（policy）奖励（reward）





reward R

- Collect a coin: $R = +1$
- Win the game: $R = +10000$
- Touch a Goomba: $R = -10000$ (game over).
- Nothing happens: $R = 0$

1 人与环境的交互模型

状态转移（state transtion） 具有随机性

state transition

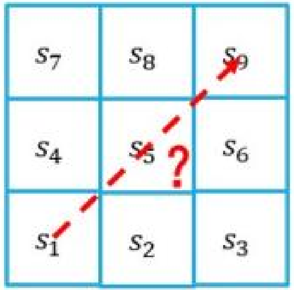
old state $\xrightarrow{\text{action}}$ new state

- E.g., “up” action leads to a new state.
- State transition can be random.
- Randomness is from the environment.
- $p(s'|s,a) = \mathbb{P}(S' = s'|S = s, A = a)$.

1 人与环境的交互模型

（序列优化）问题：

- 在下图网格中，假设有一个机器人位于s_1，其每一步只能向上或向右移动一格，跃出方格会被惩罚（且游戏停止）
- 如何使用强化学习找到一种策略，使机器人从s_1到达s_9？



刻画解该问题的因素

智能主体	迷宫机器人
环境	3×3方格
状态	机器人当前时刻所处方格
动作	每次移动一个方格
奖励	到达s_9时给予奖励；越界时给予惩罚

1、概述

1.1基本概念

强化学习的关键阶段

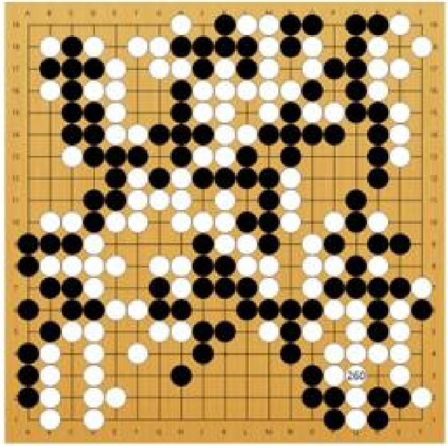
- 1998年，Richard S.Sutton出版了强化学习导论第一版，Reinforcement Learning：An Introduction
- 2013年DeepMind提出DQN（Deep Q Network），将深度网络与强化学习算法结合形成深度强化学习
- 2016年和2017年，谷歌的AlphaGo连续两年击败世界围棋冠军，更是将深度强化学习推到了风口浪尖之上。

强化学习的分类

- 根据强化学习算法是否依赖模型可以分为基于模型的强化学习算法和无模型的强化学习算法。
- 根据策略的更新和学习方法，强化学习算法可分为基于值函数的强化学习算法、基于直接策略搜索的强化学习算法以及AC的方法
- 根据环境返回的回报函数是否已知，强化学习算法可以分为正向强化学习和逆向强化学习

1、概述

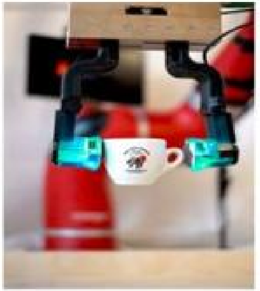
1.1基本概念



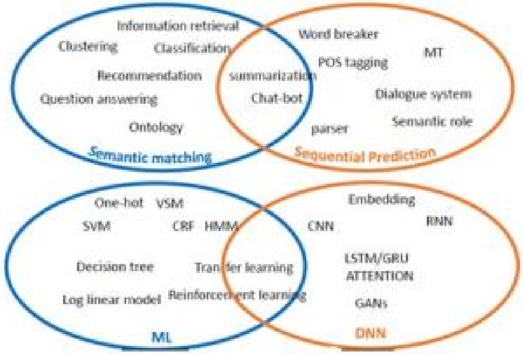
围棋游戏

注：AlphaGo的三大法宝：

- 深度学习(感知棋面)
- 强化学习（自我博弈）
- 蒙特卡洛树搜索（采样学习）



机器人运动：learning to learn



自然语言理解

2 马尔可夫决策过程

- **随机过程：**不断地进行随机试验（可以是离散或者连续的），不知道每次随机试验时结果可能服从的分布情况，每个时间点对应的结果的分布是未知的，即X(t)未知。但是，如果从开始实验到某个固定的结束时间点，都可以得到一组随机变量 X(t)，即每个时间点t对应一个随机变量。那么，这一系列的t对应的一族（无限多个）随机变量成为随机过程，记为 $\{X(t),\ t\in T\}$

- **马尔科夫性质：**也被称为“无记忆性”或“无后效性”，即下一状态的概率分布只由当前状态决定，与过去的事件无关。
- **马尔科夫过程：**具有马尔科夫性质的随机过程。

$$P(x_{t+1}|x_t)=P(x_{t+1}|x_0,x_1,\ldots x_t)$$

t+1时刻状态仅与t时刻状态相关，
将来的状态与t时刻之前的状态已经没有关系。

1、概述

1. 1基本概念

2 马尔可夫决策过程

- **马尔可夫决策过程：**在马尔科夫过程的基础上增加了奖励R 和衰减系数γ。
- 马尔可夫决策过程可用以下模型描述：MDP={S, A, P, R, γ}

状态及状态集：随机变量序列{St }(t=0,1,2,···)：St表示第t时刻的状态，每个随机变量St的取值范围为S={s1,s2,··· }，一般是**有限**的。

动作集合：A={a_1、a_2......}，智能体与环境交互时所有可选动作(**有限**)

状态转移概率：P(S(t+1) |St)满足马尔可夫性

奖励函数：在St状态执行动作at后，所获得奖励rt+1=R(St,at)

衰退系数：当前从后续状态奖励中的提成，γ∈[0, 1]

	-1		
0	0	1#	
0	0	0	-1
0*	0	0	

$$S_0 \xrightarrow[a_0]{r_1} S_1 \xrightarrow[a_1]{r_2} S_2 \xrightarrow[a_2]{r_3 = R(s_2,a_2)} S_3 \quad a_3 \quad \cdots$$

2 马尔可夫决策过程

$$G_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots$$

回报Gt(未来累计奖励)

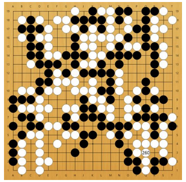
因为reward会随着时间的变化而衰减，所以我们引入了折扣(γ)的概念。

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=1}^{T-t} \gamma^{k-1} \mathrm{r}_{t+k}$$

未来所有的奖励总和

状态值函数(Value Function)

$V_\pi(s)=E_\pi[Gt|St]$



当前状态下，未来所有的奖励总和

动作值函数(Action-Value Function)

$q_\pi(s,a)= E_\pi[Gt|st,at]$

当前状态和动作下，未来所有的奖励总和

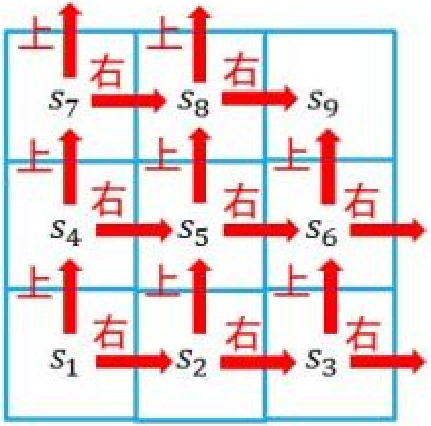
1、概述

1.1基本概念

2 马尔可夫决策过程

- 马尔可夫决策过程可用以下模型描述：MDP={S,A,Pr,R,γ}

$$S_0 \xrightarrow[a_0]{r_1} S_1 \xrightarrow[a_1]{r_2} S_2 \xrightarrow[a_2]{r_3 = R(s_2, a_2)} S_3 \quad a_3 \quad \cdots$$



- 马尔可夫过程中产生的状态序列称为轨迹(trajectory)

(1) 在t=0时，环境随机给出一个初始状态s_0~P(t=0);

(2) for t=0 : end

--智能体(Agent)根据策略π(s,a)=P(At=a|st=s)，选择一个动作at；

--环境对智能体的决策给出奖励：rt+1~R(.|st,at);

--动作at与环境发生作用,环境转移到下一个状态：st+1~P(.|st,at);

--智能体获得奖励rt以及环境的下一个状态st+1;

(S0,a0,r1,S1,a1,r2,⋯ ,ST)

有终止状态的问题叫做分段的（即存在回合的）（episodic）。例如下围棋

3 强化学习的优化目标与分类

- 在马尔可夫决策过程中，强化学习要做什么？

1）已知或部分已知的是：**奖励函数、状态转移函数**

$$R(s,a)=E(R|S_t=s,A_t=a,s_{t+1}=s')$$

$$P_{ss'}^{a}=P(S_{t+1}=s'|S_t=s,A_t=a)$$

2）需要学习的是：**策略函数(或值函数)**

$$\pi(s,a)=P(A_t=a|S_t=s)$$

3）强化学习的优化目标是：**极大化累积奖励**

$$G_t=r_{t+1}+\gamma r_{t+2}+\gamma^2 r_{t+3}+\cdots \quad =\sum_{k=1}^{T-t}\gamma^{k-1}\mathrm{r}_{t+k}$$

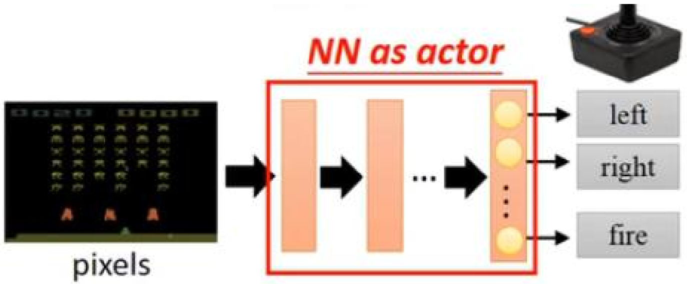
3 强化学习的优化目标与分类

- 在马尔可夫决策过程中，强化学习学到了什么？

  1）学习得到策略函数(policy based)

  策略函数π:S×A↦ [0, 1]，其中π(s,a)的值表示在状态s下采取动作a的概率。

  策略函数的输出也可以是确定的，即给定s情况下只有一个动作，记为a=π(s)。



3 强化学习的优化目标与分类

- 在马尔可夫决策过程中，强化学习学到了什么？

2）学习得到评价函数(Value based)

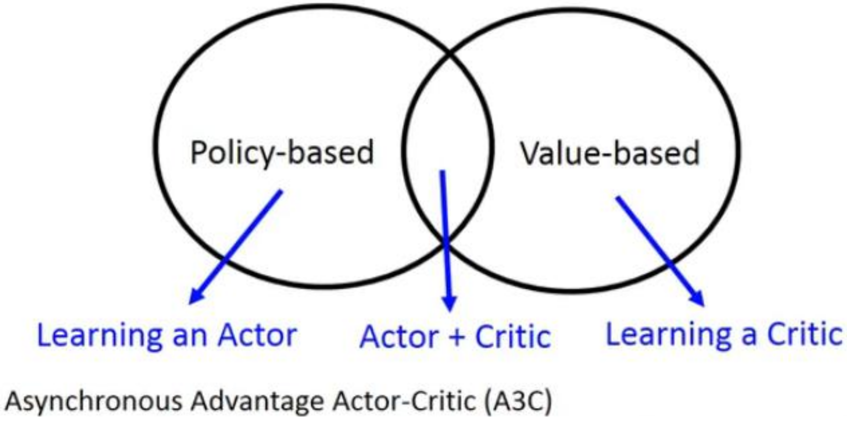
状态价值函数（Value Function）V:S↦ R，其中$V_\pi(s)=E_\pi[G_t|S_t]$，即在第t步状态为s时，按照策略π行动后，在未来所获得反馈值的期望。

动作价值函数(Action-Value Function) q:S×A↦ R，其中$q_\pi(s,a)= E_\pi[G_t|s_t,a_t]$,表示在第t步状态为s时，按照策略π采取动作a后，在未来所获得反馈值的期望。

实时估计玩家的胜率

3 强化学习的分类

- 在马尔可夫决策过程中，强化学习学到了什么？

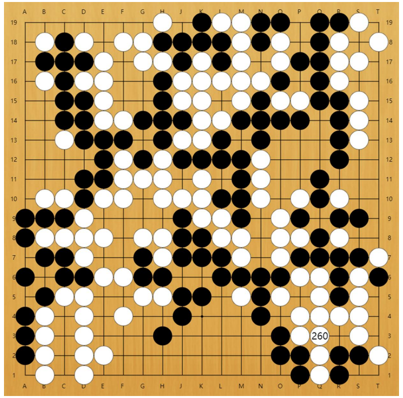


- 基于价值（Value-based）的方法
  - 对价值函数进行建模和估计，以此为依据制订策略
- 基于策略（Policy-based）的方法
  - 对策略函数直接进行建模和估计，优化策略函数使反馈最大化
- 基于模型（Model-based）的方法
  - 对环境的运作机制建模，然后进行规划（planning）等

1、概述

1.1基本概念

4 强化学习的应用

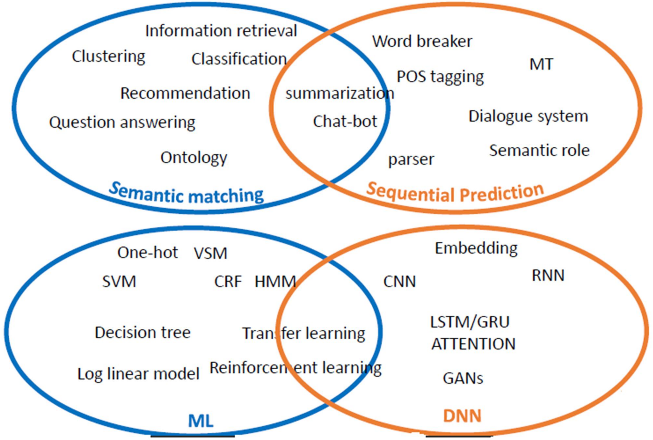


围棋游戏AlphaGo的三大法宝

深度学习(感知棋面)

强化学习（自我博弈）

蒙特卡洛树搜索（采样学习）



自然语言理解

5 与其它机器学习方法的差异

	有监督学习	无监督学习	强化学习
学习依据	基于监督信息	基于对数据结构的假设	基于评价（evaluative）
数据来源	一次性给定	一次性给定	在交互中产生（interactive）
决策过程	单步（one-shot）	无	序列（sequential）
学习目标	样本到语义标签的映射	同一类数据的分布模式	选择能够获取最大收益的的状态到动作的映射

强化学习的特点

- 基于评估：强化学习利用环境评估当前策略，以此为依据进行优化
- 交互性：强化学习的数据在与环境的交互中产生
- 序列决策过程：智能主体在与环境的交互中需要作出一系列的决策，这些决策往往是前后关联的；
- 现实中常见的强化学习问题往往还具有奖励滞后；

提纲

1、概述

2、Q-learning

3、深度强化学习DQN

2、Q-learning

2.1 Q函数

- Q-learning是一种value-based强化学习
- **状态价值函数**（Value Function）V:S↦ R，其中V_π (s)=E_π [Gt |St]，即在第t步状态为s时，按照策略π行动后，在未来所获得反馈值的期望。

$$V_\pi(s) = E_\pi[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | s_t = s] = \mathrm{E}[\sum_{k=0}^{T-t} \gamma^k r_{t+k+1}] = \mathrm{E}[\sum_{k=0}^{T-t} \gamma^k R(s_{t+k}, a_{t+k})]$$

$$= \sum_{a \in A} \pi(s,a) q_\pi(s,a)$$

- **动作价值函数** (Action-Value Function) q:S×A↦ R，其中q_π (s,a)= E_π [Gt |st,at],表示在第t步状态为s时，按照策略π采取动作a后，在未来所获得反馈值的期望。

$$q_\pi(s,a) = E_\pi[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | s_t = s, a_t = a]$$

$$= \sum_{s' \in S} P(s'|s,a)[R(s,a,s') + \gamma V_\pi(s')]$$

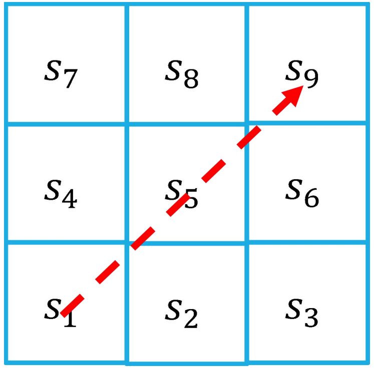
2、Q-learning

2.1 Q函数

- 状态价值函数（Value Function） $V_\pi(s)=E_\pi[r_{t+1}+\gamma r_{t+2}+\gamma^2 r_{t+3}+\cdots | s_t=s] \quad =\sum_{a\in A}\pi(s,a)q_\pi(s,a)$

- 动作价值函数 (Action-Value Function)

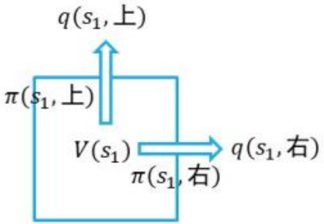
$$q_\pi(s,a)=E_\pi[r_{t+1}+\gamma r_{t+2}+\gamma^2 r_{t+3}+\cdots | s_t=s, a_t=a] \quad =\sum_{s'\in S}P(s'|s,a)[R(s,a,s')+\gamma V_\pi(s')]$$



走迷宫的例子

$$V_\pi(s)=\sum_{a\in A}\pi(s,a)q_\pi(s,a)$$

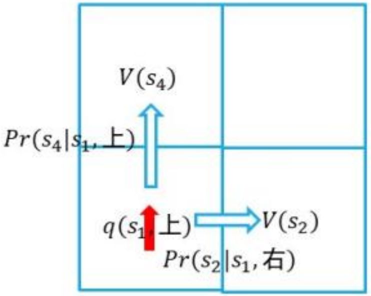
$$V_\pi(s_1)=\pi(s_1,上)q_\pi(s_1,上)+\pi(s_1,右)q_\pi(s_1,右)$$



不同动作下的反馈累加

$$q_\pi(s,a)=\sum_{s'\in S}P(s'|s,a)[R(s,a,s')+\gamma V_\pi(s')]$$

$$q_\pi(s_1,上)=P(s_4|s_1,上)[R(s_1,上,s_4)+\gamma V_\pi(s_4)]$$



动作确定时状态转移后的反馈结果

2、Q-learning

2.2 贝尔曼方程

状态价值函数

$$V_\pi(s)=\sum_{a\in A}\pi(s,a)q_\pi(s,a)$$

动作价值函数

$$q_\pi(s,a)=\sum_{s'\in S}P(s'|s,a)[R(s,a,s')+\gamma V_\pi(s')]$$

$$V_\pi(s)=\sum_{a\in A}\pi(s,a)q_\pi(s,a)$$
$$=\sum_{a\in A}\pi(s,a)\sum_{s'\in S}P(s'|s,a)[R(s,a,s')+\gamma V_\pi(s')]$$

$$q_\pi(s,a)=\sum_{s'\in S}P(s'|s,a)[R(s,a,s')+\gamma V_\pi(s')]$$
$$=\sum_{s'\in S}P(s'|s,a)[R(s,a,s')+\gamma\sum_{a'\in A}\pi(s',a')q_\pi(s',a')]$$

上述方程亦称为**贝尔曼方程**，它刻画了状态价值函数和动作价值函数自身以及两者相互之间的递推关系；

2、Q-learning

2.2 贝尔曼方程--例子

- 状态价值函数（Value Function） $V_\pi(s)=\sum_{a\in A}\pi(s,a)\sum_{s'\in S}P(s'|s,a)[R(s,a,s')+\gamma V_\pi(s')]$
- 动作价值函数(Action-Value Function) $q_\pi(s,a)=\sum_{s'\in S}P(s'|s,a)[R(s,a,s')+\gamma\sum_{a'\in A}\pi(s',a')q_\pi(s',a')]$

例，如右图的迷宫，计算状态值与动作值

R函数：到达终点时奖励100,其余地方0

折扣因子：0.9

S_{11}		终点
起点		S_{23}

→	→	
→	→	↑

状态动作序列： s_{21}，右，s_{22}，右，s_{23}，上，s_{13}

$$V_\pi(s_{23})=R(s_{23},a_{23})=100$$

$$V_\pi(s_{22})=R(s_{22},a_{22})+\gamma V_\pi(s_{23})=0+0.9*100=90$$

$$V_\pi(s_{21})=R(s_{21},a_{21})+\gamma V_\pi(s_{22})=0+0.9*90=81$$

1 蒙特卡洛算法

如右图的迷宫，怎么求解？

S_{11}		终点
起点		S_{23}

?	?	
?	?	?

一种简单思路就是，把右图的迷宫**反复走N遍**，记录下过程中各状态动作对(s,a)的回报值，并作**平均**，由此得到对应的动作值函数$q_\pi(s,a)$的估计值。

——**蒙特卡洛算法**

- 状态价值函数（Value Function）

$$V_\pi(s)=\sum_{a\in A}\pi(s,a)\sum_{s'\in S}P(s'|s,a)[R(s,a,s')+\gamma V_\pi(s')]$$

- 动作价值函数(Action-Value Function)

$$q_\pi(s,a)=\sum_{s'\in S}P(s'|s,a)[R(s,a,s')+\gamma\sum_{a'\in A}\pi(s',a')q_\pi(s',a')]$$

算法：$$q_\pi(s,a)=\frac{1}{k+1}\sum_{i=1}^{k+1}q_\pi^i(s,a)$$

缺点：需要执行大量完整的抽样过程，比较耗时间(效率低)。

2、Q-learning

2.3 贝尔曼方程求解

2 时序差分算法

s_{11}		终点
起点		s_{23}

- 对右图的迷宫：假设已经走了k次，各状态动作对的反馈为 $q_\pi^i(s,a) \quad i=1,2,\cdots,k$
- 现在又走了第k+1次，动作状态值函数如下更新：

蒙特卡洛算法：$q'_\pi(s,a)=\dfrac{1}{k+1}\sum_{i=1}^{k+1} q_\pi^i(s,a)$

时序差分：

$$q'_\pi(s,a)=q_\pi(s,a)+\alpha[q_\pi^{k+1}(s,a)-q_\pi(s,a)]$$

$$=q_\pi(s,a)+\alpha[R(s,a)+\gamma q_\pi(s',a')-q_\pi(s,a)]$$

s' 状态s下一个时序的状态

a' 状态s'时采取的动作

2、Q-learning

2.3 贝尔曼方程求解

3 基于时序差分的Q-learning

- 对右图的迷宫：假设已经走了k次，各状态动作对的反馈为 $q_\pi^i(s,a) \quad i=1,2,\cdots,k$

s_{11}		终点
起点		s_{23}

- 现在又走了第k+1次，动作状态值函数如下更新：

s' 状态s下一个时序的状态

a' 状态s'时采取的动作

蒙特卡洛算法：$$q'_\pi(s,a) = \frac{1}{k+1}\sum_{i=1}^{k+1} q_\pi^i(s,a)$$

时序差分：$$q'_\pi(s,a) = q_\pi(s,a) + \alpha[q_\pi^{k+1}(s,a) - q_\pi(s,a)]$$

$$= q_\pi(s,a) + \alpha[R(s,a) + \gamma q_\pi(s',a') - q_\pi(s,a)]$$

Q-learning: $$q'_\pi(s,a) = q_\pi(s,a) + \alpha[R(s,a) + \gamma \max_{a^*} q_\pi(s',a^*) - q_\pi(s,a)]$$

2、Q-learning

2.4 Q-learning算法

1 Q-learning算法伪码

$$q'_\pi(s,a) = q_\pi(s,a) + \alpha[R(s,a) + \gamma \max_{a*} q_\pi(s',a*) - q_\pi(s,a)]$$

初始化q_π函数
循环
    初始化s
    循环
        $a = \mathrm{argmax}_{a'} q_\pi(s,a')$
        执行动作a，观察奖励R和下一个状态s'
        更新$q_\pi(s,a) \leftarrow q_\pi(s,a) + \alpha\left[R + \gamma \max_{a'} q_\pi(s',a') - q_\pi(s,a)\right]$
        $s \leftarrow s'$
    直到s是终止状态
直到q_π收敛

2、Q-learning

2.4 Q-learning伪码

2 Q-learning算法的例子

s_d

s_7	s_8	s_9
s_4	s_5	s_6
s_1	s_2	s_3

初始化q_π函数
循环
  初始化s
  循环
    $a = \mathrm{argmax}_{a'} q_\pi(s, a')$
    执行动作a，观察奖励R和下一个状态s'
    更新$q_\pi(s,a) \leftarrow q_\pi(s,a) + \alpha\left[R + \gamma \max_{a'} q_\pi(s',a') - q_\pi(s,a)\right]$
    $s \leftarrow s'$
  直到s是终止状态
直到q_π收敛

起始点s1，终止点s9
出界，奖励为-1
折扣因子r=0.99
影响系数a=0.5

2、Q-learning

2.4 Q-learning伪码

2 Q-learning算法的例子

s_d	s_7	s_8	s_9
	s_4	s_5	s_6
	s_1	s_2	s_3

初始化q_π函数
循环
　　初始化s为初始状态
　　循环
　　　　$a = \mathrm{argmax}_{a'} q_\pi(s, a')$
　　　　执行动作a，观察奖励R和下一个状态s'
　　　　更新$q_\pi(s,a) \leftarrow q_\pi(s,a) + \alpha\left[R + \gamma \max\limits_{a'} q_\pi(s',a') - q_\pi(s,a)\right]$
　　　　$s \leftarrow s'$
　　直到s是终止状态
直到q_π收敛

初始化q_π 函数
在右图中，a/b表示q_π(s,上)=a, q_π(s,右)=b
状态可随机初始化，此处设0.2/0

初始化s，s的值在右图中用黑框框出

0/0	0.2/0	0.2/0	0/0
	0.2/0	0.2/0	0.2/0
	$\boxed{0.2/0}$	0.2/0	0.2/0

2、Q-learning

2.4 Q-learning伪码

2 Q-learning算法的例子

s_d	s_7	s_8	s_9
	s_4	s_5	s_6
	s_1	s_2	s_3

初始化q_π函数
循环
  初始化s为初始状态
  循环
    $a = \text{argmax}_{a'} q_\pi(s, a')$
    执行动作a，观察奖励R和下一个状态s'
    更新$q_\pi(s,a) \leftarrow q_\pi(s,a) + \alpha\left[R + \gamma \max_{a'} q_\pi(s',a') - q_\pi(s,a)\right]$
    $s \leftarrow s'$
  直到s是终止状态
直到q_π收敛

$a = \text{argmax}_{a'} q_\pi(s_1, a') = \text{上}$

$R = 0$，$s' = s_4$

$$q_\pi(s_1, \text{上}) \leftarrow 0.2 + 0.5 \times [0 + 0.99 \times \max\{0, 0.2\} - 0.2] = 0.199$$

$s \leftarrow s_4$

$^0/_0$	$^{0.2}/_0$	$^{0.2}/_0$	$^0/_0$
	$\boxed{^{0.2}/_0}$	$^{0.2}/_0$	$^{0.2}/_0$
	$^{0.199}/_0$	$^{0.2}/_0$	$^{0.2}/_0$

2、Q-learning

2.4 Q-learning伪码

2 Q-learning算法的例子

s_d	s_7	s_8	s_9
	s_4	s_5	s_6
	s_1	s_2	s_3

初始化q_π函数
循环
    初始化s为初始状态
    循环
        $a = \text{argmax}_{a'} q_\pi(s, a')$
        执行动作a，观察奖励R和下一个状态s'
        更新$q_\pi(s, a) \leftarrow q_\pi(s, a) + \alpha\left[R + \gamma \max_{a'} q_\pi(s', a') - q_\pi(s, a)\right]$
        $s \leftarrow s'$
    直到s是终止状态
直到q_π收敛

$a = \text{argmax}_{a'} q_\pi(s_4, a') =$ 上

$R = 0$，$s' = s_7$

$$q_\pi(s_4, \text{上}) \leftarrow 0.2 + 0.5 \times [0 + 0.99 \times \max\{0,0.2\} - 0.2] = 0.199$$

$s \leftarrow s_7$

$0/0$	$\boxed{0.2/0}$	$0.2/0$	$0/0$
	$0.199/0$	$0.2/0$	$0.2/0$
	$0.199/0$	$0.2/0$	$0.2/0$

2 Q-learning算法的例子

s_d

s_7	s_8	s_9
s_4	s_5	s_6
s_1	s_2	s_3

初始化q_π函数
循环
　　初始化s为初始状态
　　循环
　　　　$a = \text{argmax}_{a'} q_\pi(s,a')$
　　　　执行动作a，观察奖励R和下一个状态s'
　　　　更新$q_\pi(s,a) \leftarrow q_\pi(s,a) + \alpha\left[R + \gamma \max_{a'} q_\pi(s',a') - q_\pi(s,a)\right]$
　　　　$s \leftarrow s'$
　　直到s是终止状态
直到q_π收敛

$$a = \text{argmax}_{a'} q_\pi(s_7,a') = 上$$
$$R = -1，\ s' = s_d$$
$$q_\pi(s_7, 上) \leftarrow 0.2 + 0.5 \times [-1 + 0.99 \times \max\{0,0\} - 0.2] = -0.4$$
$$s \leftarrow s_d$$

因为sd是终止状态，因此一个片段（episode）结束

$\boxed{0/0}$

$-0.4/0$	$0.2/0$	$0/0$
$0.199/0$	$0.2/0$	$0.2/0$
$0.199/0$	$0.2/0$	$0.2/0$

2、Q-learning

2.4 Q-learning伪码

2 Q-learning算法的例子

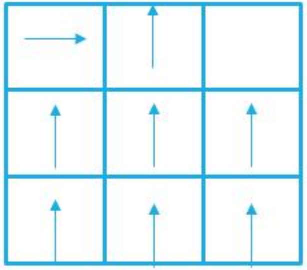
s_d

s_7	s_8	s_9
s_4	s_5	s_6
s_1	s_2	s_3

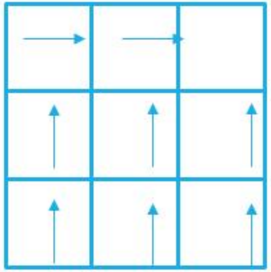
${}^{-0.4}/_{0}$	${}^{0.2}/_{0}$	${}^{0}/_{0}$
${}^{0.199}/_{0}$	${}^{0.2}/_{0}$	${}^{0.2}/_{0}$
${}^{0.199}/_{0}$	${}^{0.2}/_{0}$	${}^{0.2}/_{0}$

${}^{-0.4}/_{0.099}$	${}^{-0.4}/_{0}$	${}^{0}/_{0}$
${}^{0.100}/_{0}$	${}^{0.2}/_{0}$	${}^{0.2}/_{0}$
${}^{0.198}/_{0}$	${}^{0.2}/_{0}$	${}^{0.2}/_{0}$

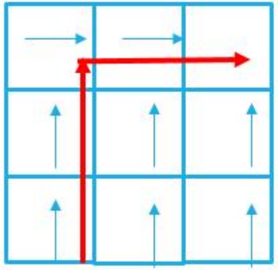
${}^{-0.4}/_{0.050}$	${}^{-0.4}/_{0.5}$	${}^{0}/_{0}$
${}^{0.099}/_{0}$	${}^{0.2}/_{0}$	${}^{0.2}/_{0}$
${}^{0.148}/_{0}$	${}^{0.2}/_{0}$	${}^{0.2}/_{0}$



第一个片段后



第二个片段后



第三个片段后

3 使用贪心策略的Q-learning算法

Q-learning: $q'_\pi(s,a)=q_\pi(s,a)+\alpha[R(s,a)+\boxed{\gamma\max_{a^*}q_\pi(s',a^*)}-q_\pi(s,a)]$

初始化q_π函数
循环
  初始化s为初始状态
  循环
    采样$a\sim\boxed{\epsilon-greedy_\pi(s)}$
    执行动作a，观察奖励R和下一个状态s'
    更新$q_\pi(s,a)\leftarrow q_\pi(s,a)+\alpha\left[R+\gamma\boxed{\max_{a'}q_\pi(s',a')}-q_\pi(s,a)\right]$
    $s\leftarrow s'$
  直到s是终止状态
直到q_π收敛

采样策略与更新策略不同

ϵ贪心（ϵ-greedy）策略

$$\epsilon\text{-}greedy_\pi(s)=\begin{cases}\mathrm{argmax}_a q_\pi(s,a), & 以1-\epsilon的概率\\ 随机的a\in A\ , & 以\epsilon的概率\end{cases}$$

提纲

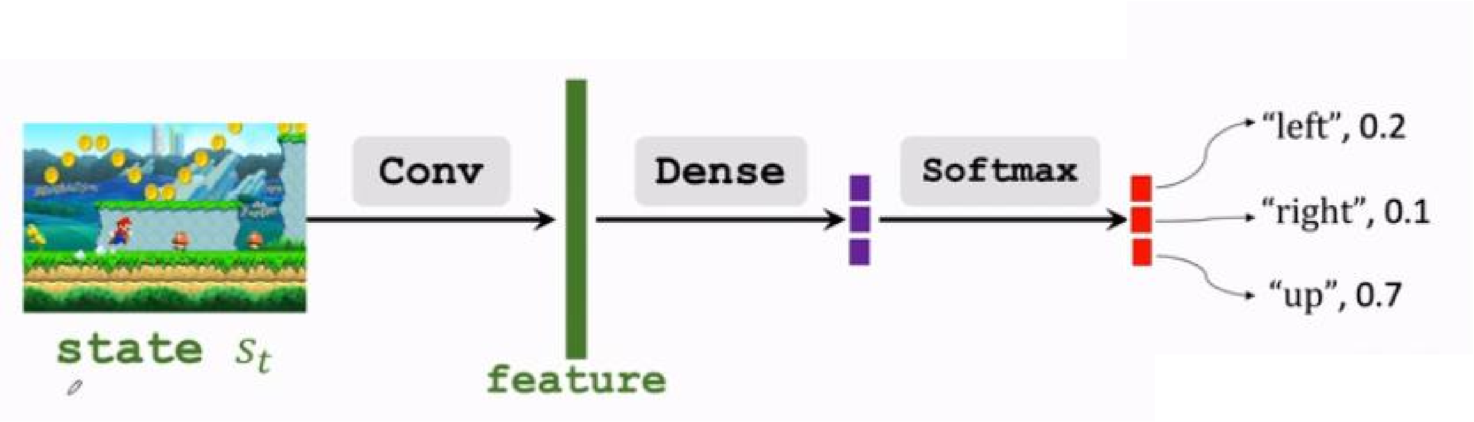
1、概述

2、Q-learning

3、深度强化学习DQN

1 动机

- 强化学习有较强决策能力，但缺乏感知能力；
- 深度网络有较强的图像识别能力，但多用于分类、识别等任务，决策能力不足。
- 因此，学者们尝试把深度网络和强化学习结合起来，用在围棋、游戏、智能驾驶、无人装备自主作战等任务。



3、深度强化学习DQN

3.1 概述

2 思路

- 强化学习中，**Q-learning**的主要任务就是要学习一个动作-价值函数；

Q-learning: $$q'_\pi(s,a) = q_\pi(s,a) + \alpha[R(s,a) + \gamma \max_{a^*} q_\pi(s',a^*) - q_\pi(s,a)]$$

- 深度强化学习中，主要思路就是用神经网络来模拟这个动作-价值函数q_π；

神经网络具有非常强的非线性拟合能力，因此该方法理论上可行，值得尝试。

$$q(s,a,\theta) \longrightarrow q_\pi(s,a)$$

- 标签:训练神经网络，关键是要给样本指定标签，观察**Q-learning**更新公式，将标签(训练目标)定义为:

$$Q^t = R(s,a) + \gamma \max_{a^*} q_\pi(s',a^*)$$

目标函数 $$J(\theta) = [Q^t - q(s,a,\theta)]^2 = [R(s,a) + \gamma \max_{a^*} q_\pi(s',a^*) - q(s,a,\theta)]^2$$

2 思路

- **标签**:训练神经网络，关键是要给样本指定标签，观察**Q-learning**更新公式，将标签(**训练目标**)定义为：

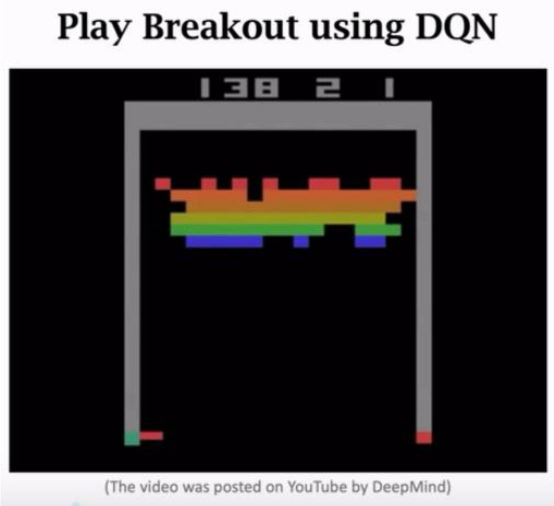
$$Q^t = R(s,a) + \gamma \max_{a^*} q(s', a^*)$$

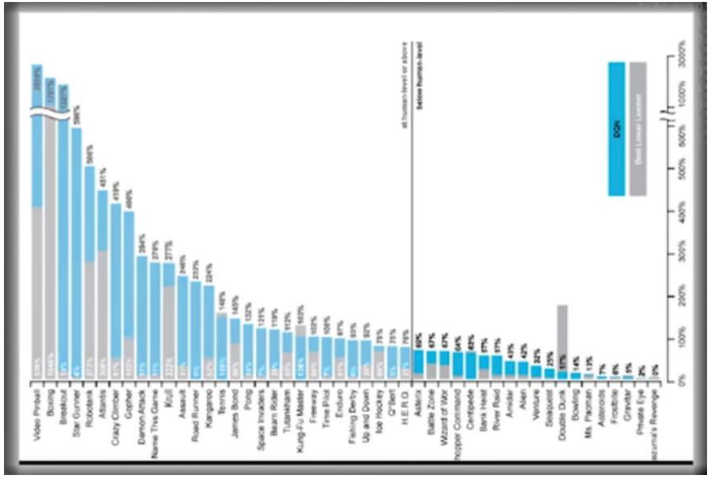
损失函数 $J(\theta) = [Q^t - q(s,a,\theta)]^2 = [R(s,a) + \gamma \max_{a^*} q(s', a^*) - q(s,a,\theta)]^2$

- 由于**训练目标**也是未知的，因此损失函数无法直接计算，因此研究者们采用两个网络，自己训练自己的方式(双手左右互博)

$$J(\theta) = [Q^t - q(s,a,\theta)]^2 \approx [q(s,a,\theta') - q(s,a,\theta)]^2$$

3 效果





在大部分游戏上，深度强化网络已比玩家要厉害

谢 谢