第十五讲 强化学习

王石平，教授/博导/旗山学者
数学与计算机科学学院
福州大学 FUZHOU UNIVERSITY

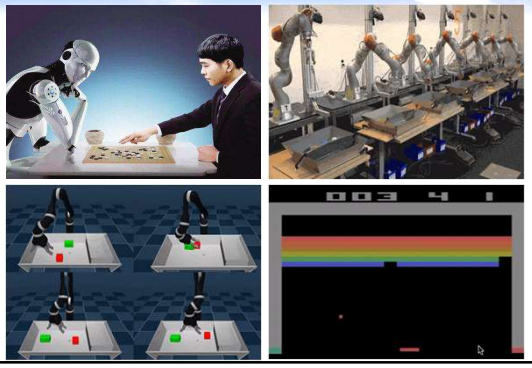
§ 纲要

- 理论动机
- 理论架构
- 实验结果

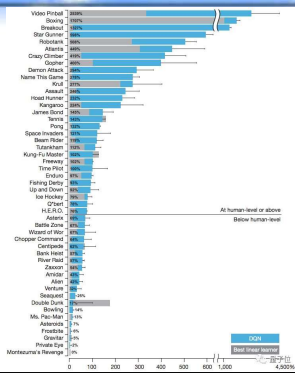
§ 纲要

- 理论动机
- 理论架构
- 实验结果

§ 理论动机



§ 理论动机



§ 纲要

- 理论动机
- 理论架构
- 实验结果

§ 强化学习vs传统机器学习

机器学习可以分为三类，分别是supervised learning，unsupervised learning和reinforcement learning。而强化学习与其他机器学习不同之处为：

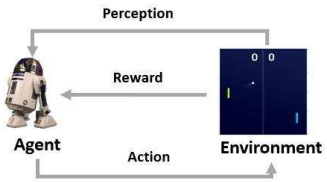
1.没有教师信号，也没有label。只有reward，其实reward就相当于label。

2.反馈有延时，不是能立即返回。

3.相当于输入数据是序列数据。

4.agent执行的动作会影响之后的数据。

§ 理论框架



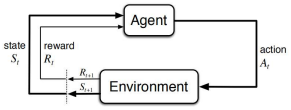
§ 马尔可夫决策过程

马尔科夫决策过程是基于马尔科夫论的随机动态系统的最优决策过程。

马尔可夫定义：下一个状态的产生只和当前的状态有关，即：

$$P[S_{t+1}|S_t] = P[S_{t+1}|S_1, \cdots, S_t]$$

马尔可夫决策过程简单说就是一个智能体（Agent）采取行动（action）从而改变自身状态（state）获得奖励（reward）与环境（Environment）发生交互的循环过程。



智能体与环境交互产生的序列：

$$S_0, A_0, R_1, S_1, A_1, R_2, \cdots$$

称为序列决策过程，而马尔科夫决策过程就是该序列决策过程的一种公式化。

§ 强化学习

强化学习是机器学习的一个重要分支，是多学科多领域交叉的一个产物，它的本质是解决decision making问题，即自动进行决策，并且可以做连续决策。

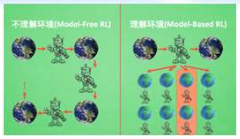
它主要包含四个元素：agent，**环境状态**，**行动**，**奖励**，

强化学习的目标：获得最多的累计奖励。

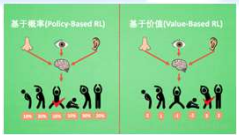
让我们以小孩学习走路来做个形象的例子：
小孩想要走路，但在这之前，他需要先站起来，站起来之后还要保持平衡，接下来还要先迈出一条腿，是左腿还是右腿，迈出一步后还要迈出下一步。

小孩就是agent，他试图通过采取行动（即行走）来操纵环境（行走的表面），并且从一个状态转变到另一个状态（即他走的每一步），当他完成任务的子任务（即走了几步）时，孩子得到奖励（给巧克力吃），并且当他不能走路时，就不会给巧克力。

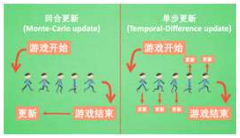
§ 强化学习分类



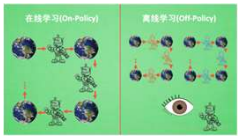
Model-free 与 Model-based



Policy-based RL 与 Value-based RL

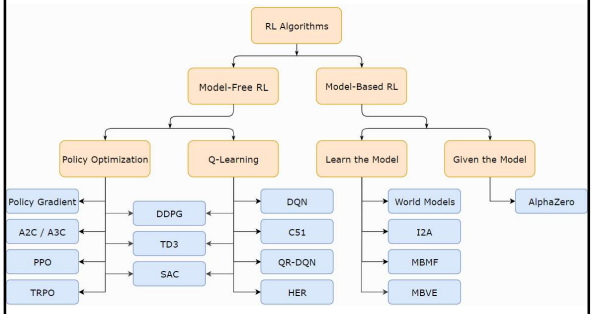


回合更新 与 单步更新



在线学习 与 离线学习

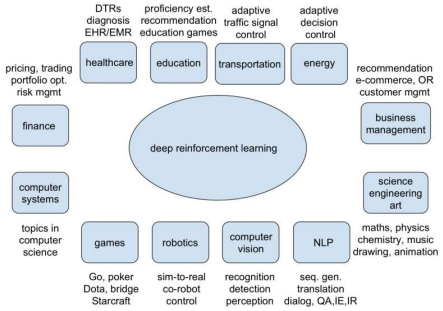
§ 强化学习算法



§ 相关资料

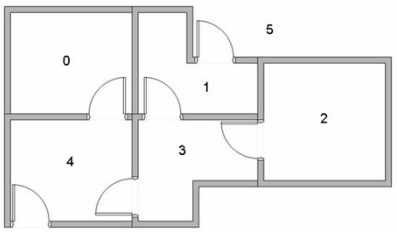
- DQN (Deep Q-Networks): Mnih et al, 2013
- TRPO (Trust Region Policy Optimization): Schulman et al, 2015
- DDPG (Deep Deterministic Policy Gradient): Lillicrap et al, 2015
- A2C / A3C (Asynchronous Advantage Actor-Critic): Mnih et al, 2016
- AlphaZero: Silver et al, 2017
- C51 (Categorical 51-Atom DQN): Bellemare et al, 2017
- I2A (Imagination-Augmented Agents): Weber et al, 2017
- QR-DQN (Quantile Regression DQN): Dabney et al, 2017
- PPO (Proximal Policy Optimization): Schulman et al, 2017
- HER (Hindsight Experience Replay): Andrychowicz et al, 2017
- MBMF (Model-Based RL with Model-Free Fine-Tuning):Nagabandi et al, 2017
- World Models: Ha and Schmidhuber, 2018
- SAC (Soft Actor-Critic): Haarnoja et al, 2018
- TD3 (Twin Delayed DDPG): Fujimoto et al, 2018
- MBVE (Model-Based Value Expansion): Feinberg et al, 2018

§ 应用场景



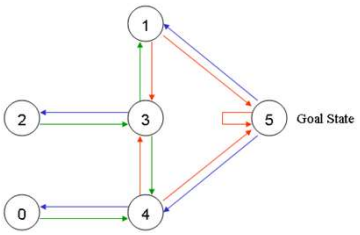
§ 举例

假设一栋建筑里面有5个房间，房间之间通过门相连（如图所示），我们将这5个房间按照0至4进行编号，然后将建筑的外围当做一个大房间，编号为5。我们的目标是让agent学会从任意一个房间开始，以最快的速度走到房间5。



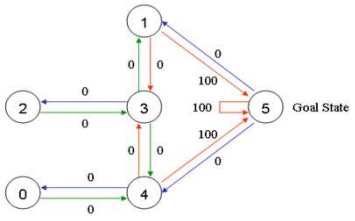
§ 模型构建

为了便于理解，我们将上面的房间用一个图来表示。
图中结点表示房间，边表示门。



§ 模型构建

将去往目标房间的边的reward值设置为100，其他边的reward值设置为0。
这样，在到达目标房间后，agent将获得100reward值，否则没有reward值。



§ 模型构建

根据图上边和结点之间的联系和奖励，得到reward值矩阵。
矩阵中的每一个元素表示在某个状态下选择某个可取动作可以获得的奖励。
其中-1代表两个结点之间没有边相连，即动作不可取。

Action

State	0	1	2	3	4	5
0	−1	−1	−1	−1	0	−1
1	−1	−1	−1	0	−1	100
2	−1	−1	−1	0	−1	−1
3	−1	0	0	−1	0	−1
4	0	−1	−1	0	−1	100
5	−1	0	−1	−1	0	100

$R=$

§ 模型构建

类似的，构建关于状态-动作对的矩阵Q。
其中的Q值表示在某个状态下采取某个动作所能获得的累积奖励。
之后，根据Q-learning算法迭代更新Q矩阵。

$$Q=\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

§ Q-learning

Q-learning 是一个基于值的强化学习算法，利用 Q 函数寻找最优的「动作-选择」策略。

它根据动作值函数评估应该选择哪个动作，这个函数决定了处于某一个特定状态以及在该状态下采取特定动作的奖励期望值。

目的：最大化 Q 函数的值（给定一个状态和动作时的未来奖励期望）。

Q-table 帮助我们找到对于每个状态来说的最佳动作。

通过选择所有可能的动作中最佳的一个来最大化期望奖励。

Q 作为某一特定状态下采取某一特定动作的质量的度量。

函数 Q（state，action）→返回在当前状态下采取该动作的未来奖励期望。

这个函数可以通过 Q-learning 算法来估计，使用 Bellman 方程迭代地更新 Q(s，a)

Q-table 给出相同的任意的设定值→ 随着对环境的持续探索→Q 给出越来越好的近似。

§ Q-learning理论

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

$$Q(S,A) \leftarrow (1-\alpha)Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a)\right]$$

Q估计值 **Q现实值**

Q(S, A)为动作值函数，表示在状态S下采取动作A所能获得的累积奖励expected return

S’ 表示在状态S下采取动作A进入的下一个状态

α表示学习效率，0<α<1，α越大，学习效率越高，更新幅度越大

γ 为折扣因子，γ 越接近0，说明agent越在乎眼前利益，越接近1，则越在乎长远利益。

§ Q-learning算法

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(terminal\text{-}state,\cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    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

§ Q-learning解决过程

折扣因子γ=0.8，Q初始化为一个零矩阵。初始化状态为1，为了简化公式，我们考虑α = 1的特殊情况。即新的Q值完全等于Q现实值。

$$Q=\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

$$Q(S,A) \leftarrow R + \gamma \max_a Q(S',a)$$

§ Q-learning算法

$$R=\begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{array}{c} \begin{array}{cccccc} \multicolumn{6}{c}{\text{Action}} \\ 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

观察reward值矩阵的第二行，对应状态1，由状态1可以选择的动作有：转移到状态3或状态5，假设我们随机的选择了动作转移到状态5.

§ Q-learning 一次迭代

$$Q(S,A) \leftarrow R + \gamma \max_a Q(S',a)$$

$$Q(1,5) = R(1,5) + 0.8 * max\{Q(5,1), Q(5,4), Q(5,5)\}$$
$$= 100 + 0.8 * max\{0,0,0\} = 100$$

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

§ Q-learning二次迭代

选取状态3作为初始状态，由状态3可以选择的动作有：转移到状态1,2,或4，假设我们随机的选择了动作转移到状态1，

$$Q(S,A) \leftarrow R + \gamma \max_a Q(S',a)$$

$$Q(3,1) = R(3,1) + 0.8 * max\{Q(1,3), Q(1,5)\}$$
$$= 0 + 0.8 * max\{0,100\} = 80$$

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

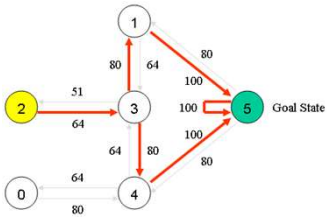
§ Q-learning稳定Q值

当我们继续进行更多回合，矩阵Q最终将会收敛成左边这样。然后对其进行规范化，每个非零元素都除以矩阵Q的最大元素（这里为500），可以得到右边这样一个矩阵：（这里省略了百分号）

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 320 & 0 & 500 \\ 0 & 0 & 0 & 320 & 0 & 0 \\ 0 & 400 & 256 & 0 & 400 & 0 \\ 320 & 0 & 0 & 320 & 0 & 500 \\ 0 & 400 & 0 & 0 & 400 & 500 \end{bmatrix} \end{array} \Rightarrow Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 80 & 0 \\ 0 & 0 & 0 & 64 & 0 & 100 \\ 0 & 0 & 0 & 64 & 0 & 0 \\ 0 & 80 & 51 & 0 & 80 & 0 \\ 64 & 0 & 0 & 64 & 0 & 100 \\ 0 & 80 & 0 & 0 & 80 & 100 \end{bmatrix} \end{array}$$

§ Q-learning 图表示

然后我们的agent便学习到了转移至目标状态的最佳路径，对于每一个状态S，我们选择Q矩阵中对应状态S的Q值最大的动作，即可得到最佳路径。



§ 纲要

- 理论动机
- 理论架构
- **实验结果**

§ 实验