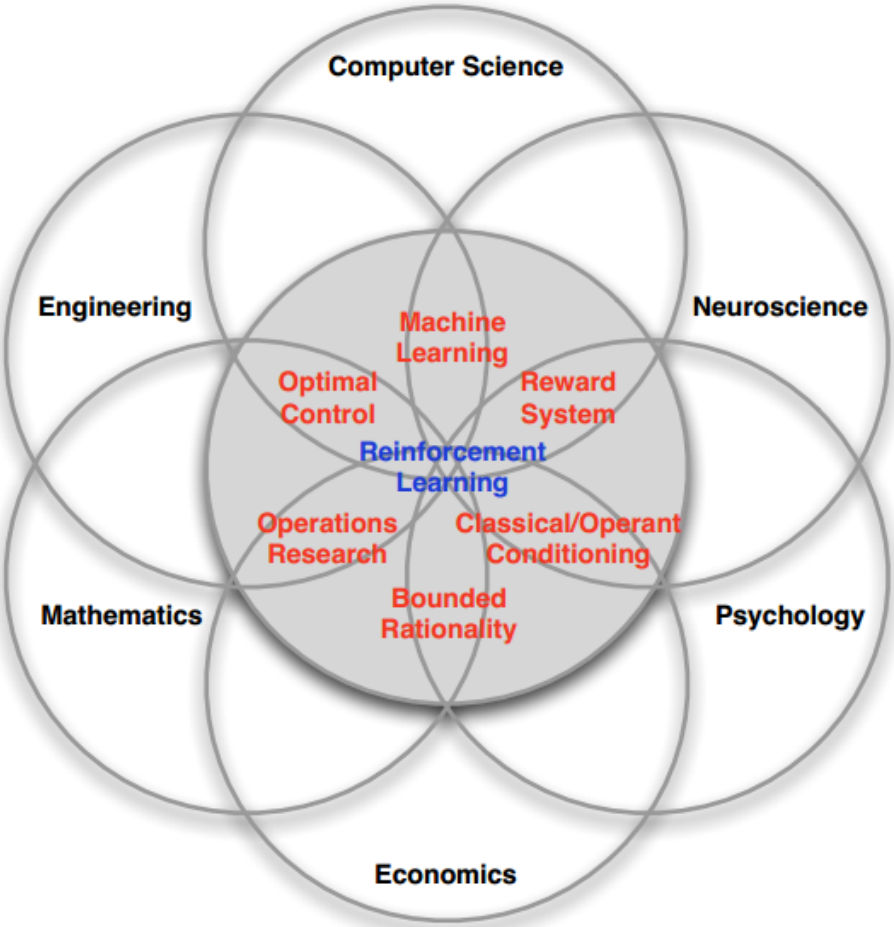
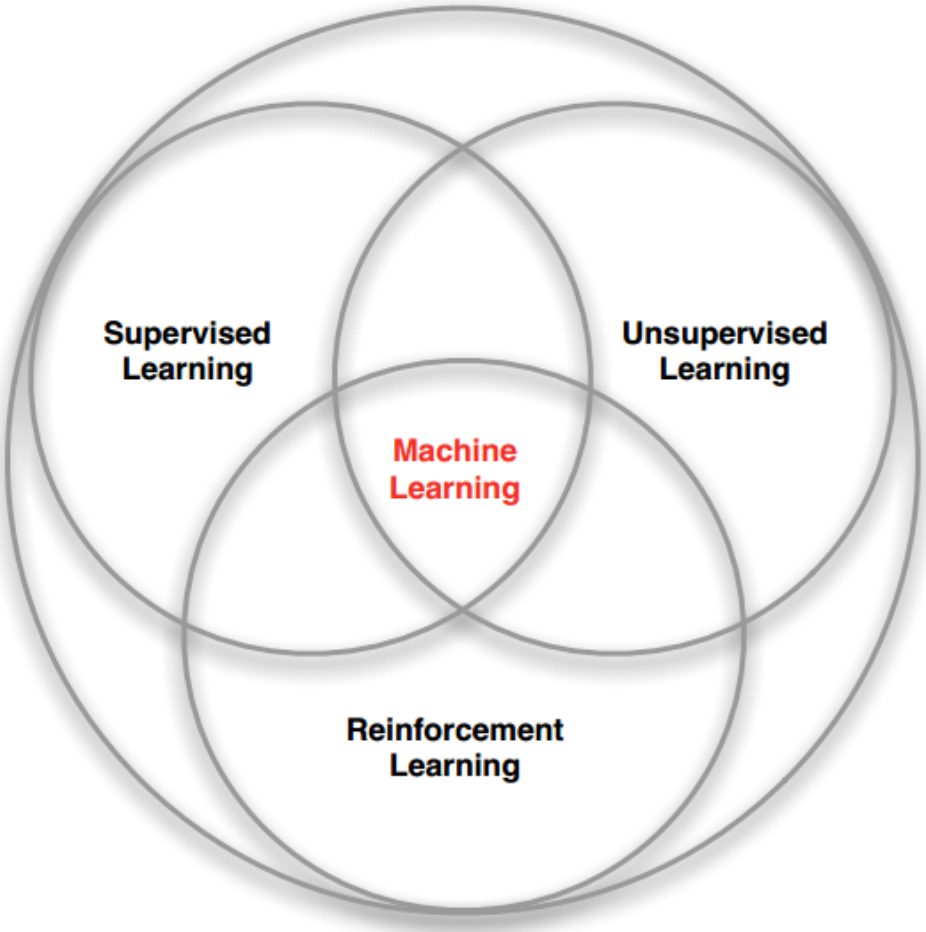
Lecture 1：Introduction to Reinforcement Learning

1. About RL

- 强化学习在不同领域有不同的表现形式：神经科学、心理学、计算机科学、工程领域、数学、经济学等有不同的称呼。



- 强化学习是机器学习的一个分支：监督学习、无监督学习、强化学习



- RL就是学习如何将状态映射到动作，使得到的奖励最大化。它主要包含四个元素：**agent，环境状态，行动，奖励。**
- 强化学习研究的是智能体agent与环境之间交互的任务，也就是让agent像人类一样通过试错，不断地学习在不同的环境下做出最优的动作，而不是有监督地直接告诉agent在什么环境下应该做出什么动作。在这里我们需要引入回报（reward）这个概念，回报是执行一个动作或一系列动作后得到的奖励，比如在游戏超级玛丽中，向上跳可以获得一个金币，也就是回报值为1，而不跳时回报就是0。回报又分为立即回报和长期回报，立即回报指的是执行当前动作后能立刻获得的奖励，但很多时候我们执行一个动作后并不能立即得到回报，而是在游戏结束时才能返回一个回报值，这就是长期回报。强化学习唯一的准则就是学习通过一序列的最优动作，获得最大的长期回报。比较有挑战性的是，任一状态下做出的动作不仅影响当前状态的立即回报，而且也会影响到下一个状态，因此也就会影响整个执行过程的回报
- 强化学习的特点：
    - 没有监督数据、只有奖励信号
    - 奖励信号不一定是实时的，而很可能是延后的，有时甚至延后很多。
    - 时间（序列）是一个重要因素
    - 当前的行为影响后续接收到的数据

2. The RL problem

2.1 Reward 奖励

2.1.1 Rewards

R_t是一个标量的信号反馈，它反映个体在t时刻做得怎么样。个体的工作就是最大化累计奖励。强化学习就是基于这样的"奖励假设"：所有问题解决的目标都可以被描述成最大化累积奖励。

Definition (Reward Hypothesis)

All goals can be described by the maximisation of expected cumulative reward

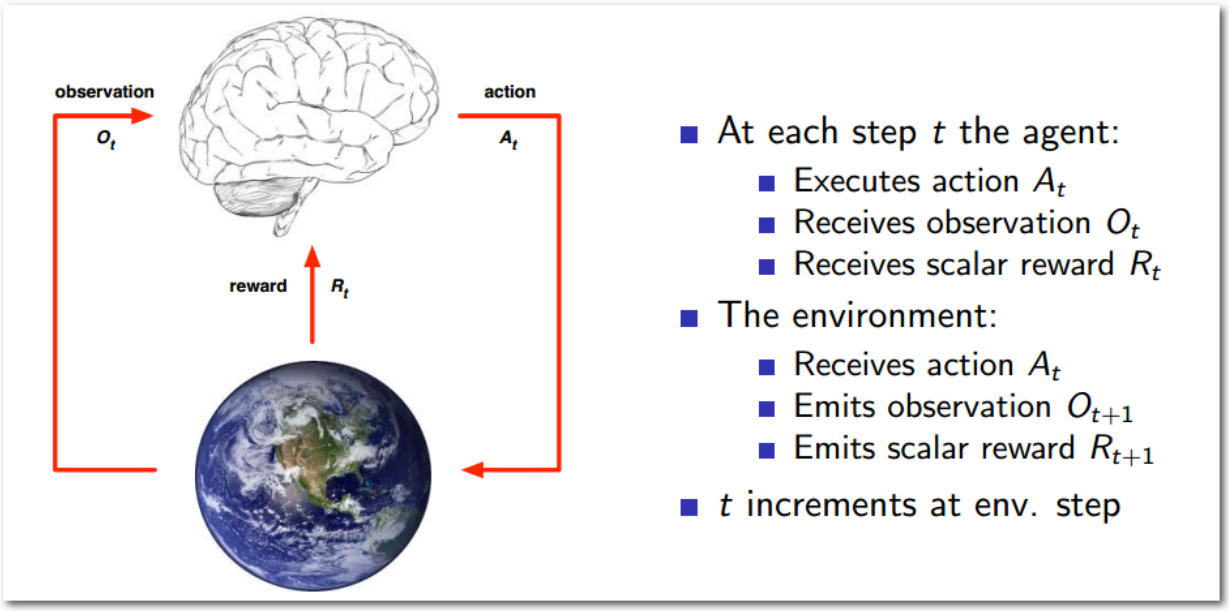
2.1.2 Sequential Decision Making 序贯决策

强化学习解决的是序贯决策问题，序贯决策（Sequential Decision）是用于随机性或不确定性动态系统最优化的决策方法。序贯决策的过程是：从初始状态开始，每个时刻作出最优决策后，接着观察下一步实际出现的状态，即收集新的信息，然后再作出新的最优决策，反复进行直至最后。

强化学习的目标是选择一定的行为系列以求最大化未来的总体奖励；这些行为可能是一个长期的序列；奖励可能而且通常是延迟的；有时候为了获取更多的长期奖励，宁愿牺牲即时（短期）的奖励。

2.2 Environments

2.2.1 Agent and Environment 个体和环境



在图片中，把大脑比作agent,把地球比作环境。
在 每一步行动中，agent：

- 做出一个行为 A_t，
- 外部世界观测的输入 O_t ，
- 从环境得到一个奖励信号 R_t 。

Environment：

- 接收个体的动作 A_t ，
- 更新环境信息，同时使得个体可以得到下一个观测O_{t+1}，
- 给个体一个奖励信号 R_{t+1}。

通过agent和环境进行不断的交互，我们发现增强学习是基于观察、奖励、行动措施的时间序列。时间序列代表着agent的经验，这个经验就是我们用于增强学习的数据。因此增强学习的问题就是聚焦这个数据来源 ， 也就是这个数据流,即图中的观察信息，采取的行动，以及奖励信号。

2.3 State

2.3.1 History and State 历史和状态

history是observation、reward、action的序列

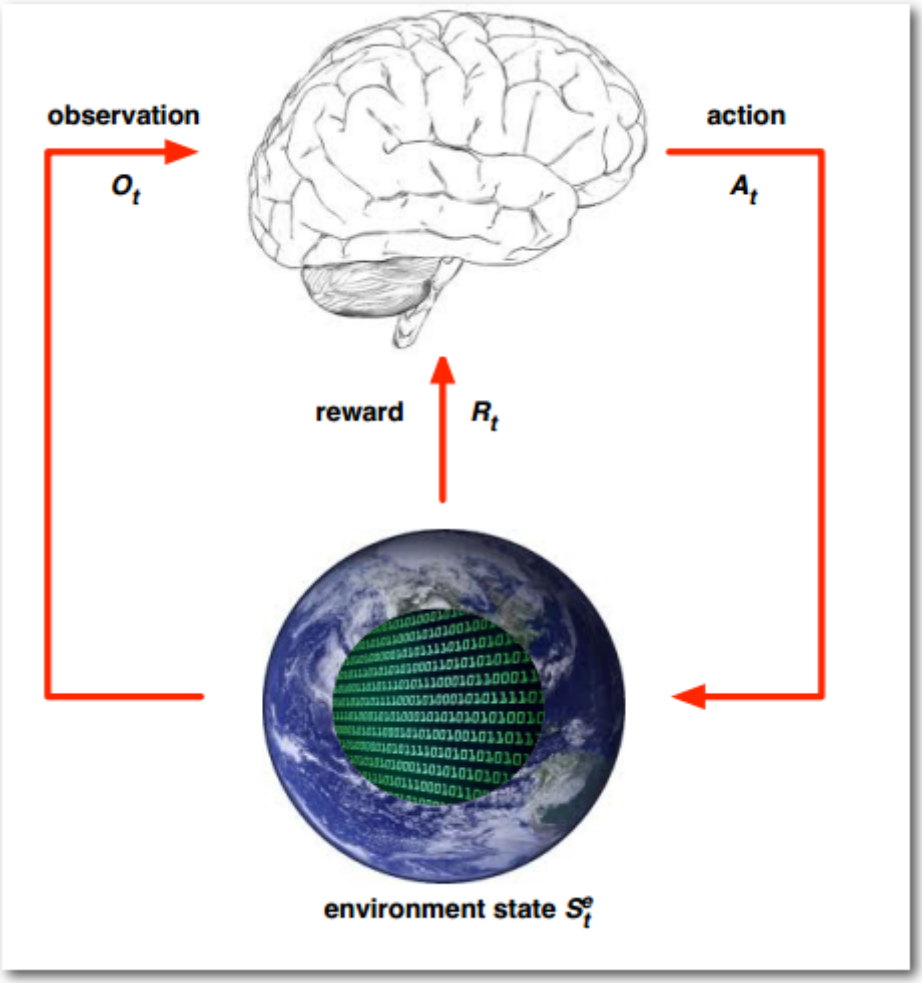
$$H_t = O_1, R_1, A_1, \ldots, A_{t-1}, O_t, R_t$$

- history能够决定action接下来采取什么行动；实际上，agent做的就是从history到action的映射
- 对于环境而言，history和产生的action交给environment，从而得到对应的observation和reward
- state是对history的总结，state决定了下一步的action。即state包含了我们需要的所有信息，通过这些信息，我们就可以采取下一步的action。state是关于history的函数：

$$S_t = f(H_t)$$

2.3.2 Environment State

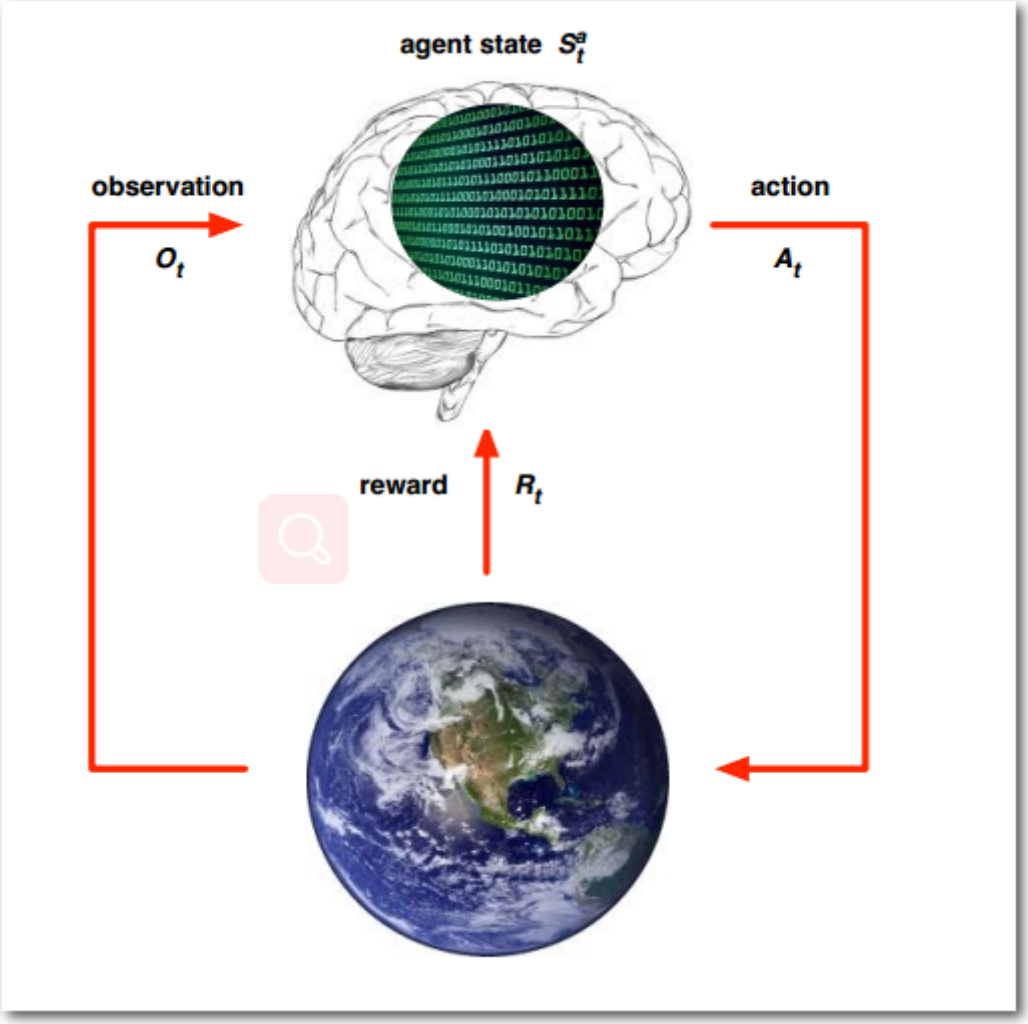
S_t^e是环境的私有呈现，包括环境用来决定下一个观测/奖励的所有数据，通常对个体并不完全可见，也就是个体有时候并不知道环境状态的所有细节,即使有时候环境状态对个体可以是完全可见的，这些信息也可能包含着一些无关信息。所以说agent不直接依赖于environment state，agent根据的是environment state产生的observation和reward来得到对应的输出



2.3.3 Agent State

S_t^a是个体的内部呈现，包括个体可以使用的、决定未来动作的所有信息，可以直接用于做出action的决策。个体状态是强化学习算法可以利用的信息。这个state可以是任何关于history的函数：

$$S_t^a = f(H_t)$$



2.3.4 Information State

Information State 也被称作Markov state，他包含了history中所有有用的信息，其定义如下

An information state (a.k.a. Markov state) contains all useful information from the history.

Definition

A state S_t is Markov if and only if

$$\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} \mid S_1, ..., S_t]$$

Markov性质是指下一时刻的状态仅有当前当前状态决定，与过去的状态无太大的关系：

$$H_{1:t} \longrightarrow S_t \longrightarrow H_{t+1:\infty}$$

举个例子，当我们正在训练一个直升机让他去做飞行时，你不需要知道他10min之前的速度，方向，位置等信息，这些信息对于你的决策没啥意义。你只需要知道他当前的位置状态，就足以做出正确的决策，这些信息组成的状态就是具有Markov 性质的状态。相反的，假设你现在不知道飞机的速度信息，那么这个状态就不是Markov的状态，你需要查看历史记录的其他状态，来确定当前的速度，这样，这个状态就不是markov的状态。

2.3.5 Fully Observable Environments 完全可观测的环境

Fully Observable: agengt可以直接观察到环境状态，即：

$$O_t = S_t^a = S_t^e$$
$$agent\ \ state = environment\ \ state = information\ \ state$$

正式地说，这种问题是一个马尔可夫决策过程（Markov Decision Process， MDP）

2.3.6 Partially Observable Environments 部分可观测的环境

Partial observability :agent不能直接观测环境，是间接的观测环境，所以它不能看到所有的环境信息。举几个例子：

- 一个带有摄像头的机器人并不确切的知道自己在哪里，只能通过摄像头来辨别出它在房间的什么位置，它需要自己进行定位
- 贸易agent只能观测到最新的价格，她不知道价格的趋势，不知道这些来自哪里，它要做的是基于已有的信息采取下一步行动。
- 一个扑克牌玩家只能观测到那些已经打出的牌，其他牌都是面向桌面的。

从上面的例子我们可以看出：

$$agent\ \ state \neq envirment\ \ state$$

这种问题是一个部分可观测马儿可夫决策过程POMDP(partially observable Markov decision preocess)，要解决这个问题，个体必须构建自己的状态表示S_t^a,我们有好多种方法创建S_t^a，：

- Complete history:$S_t^a = H_t$,就是记住所有的历史状态，记住目前为止的每一次观测、动作、奖励。
- Beliefs of environment state: 使用贝叶斯概率，你不相信每一步都是正确的。你不知道环境发生了变化，你要对你身处的环境，引进概率分布：

$$S_t^a = (P[S_t^e = s^1], \dots, P[S_t^e = s^n])$$

这个概率向量决定了状态，我们通过他来决定下一步的行动，所以我们要保存所有状态的概率值

- Recurrent neural network：使用循环神经网络，不需要使用概率,只需要线性组合的形式，将最近的agents的状态与最近的观测结合起来，就能得到最新的状态：

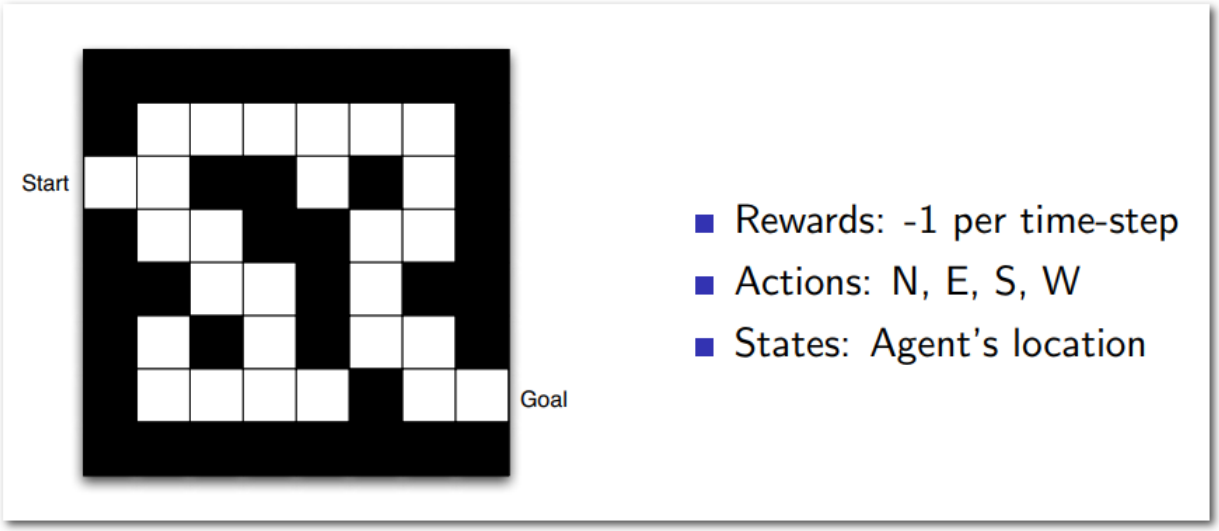
$$S_t^a = \sigma(S_{t-1}^a W_s + O_t W_o)$$

3. Inside An RL Agnet 强化学习个体的主要组成部分

agent 包含三部分，这三个部分是agent的主要内容，并不是唯一。

- Policy: 策略，Policy表明了agent能够采取的 行动，它是一个行为函数，该函数以它的状态作为输入，以它的下一步的行动决策作为输出。
- Value function: 这个函数用来评估agent在某种特殊状态下的好坏，采取某种行动之后的好坏
- Model: agent用来感知环境是如何变化的，这里并不是真实的环境，而是在agent眼里的环境。model就是用来判断环境变化的。

我们将用下面这个迷宫的例子来解释这个问题：



在迷宫的例子中，我们的目标是用最短的路径达到终点。Reward，Actions，States如图所示

3.1 Policy 策略

policy是从state到action的一种映射。policy可以是确定性的，也可以是不确定性的。
在某种state下我们拥有确定的policy ，比如

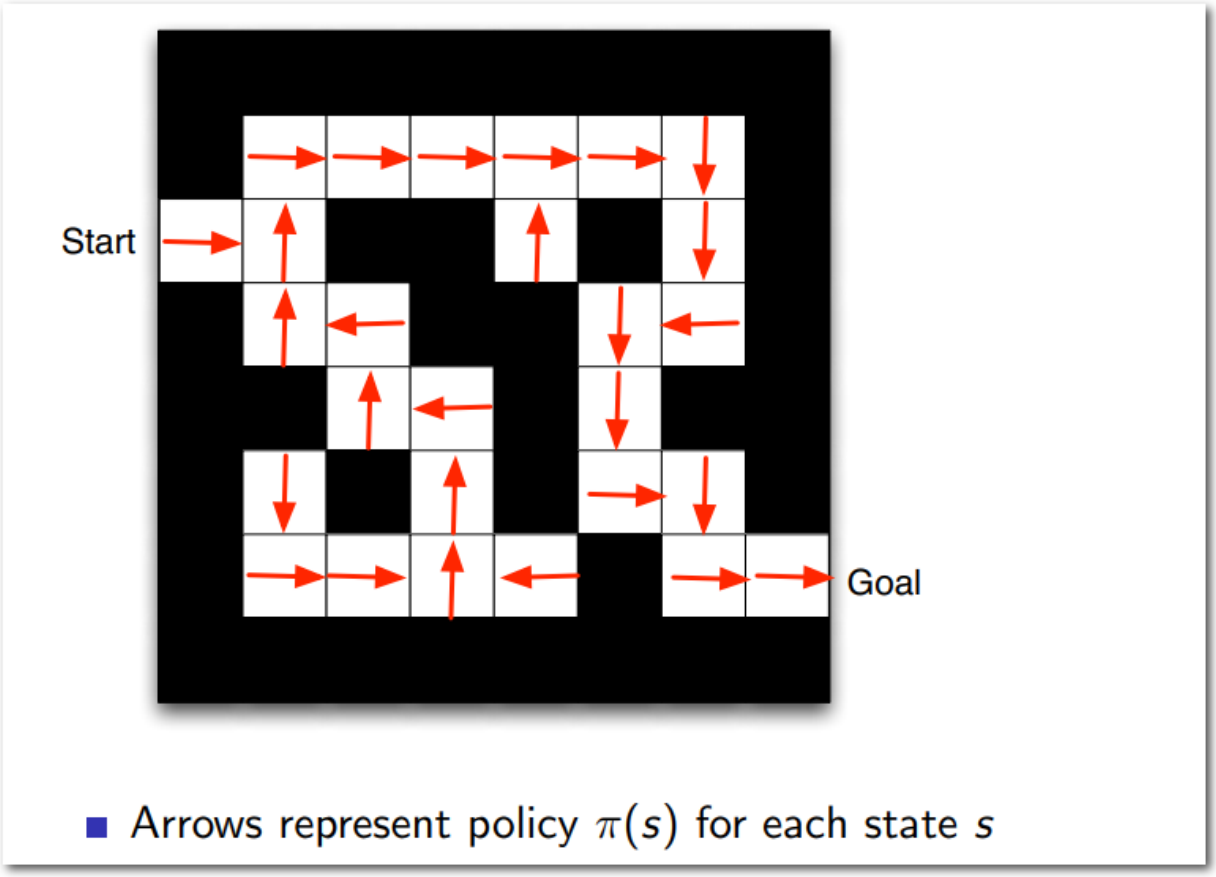
$$a = \pi(s)$$

这里的函数π告诉我们如何从某个情形的状态s变成某个行动a

此外，我们也可以使用随机的policy，我们的policy不一定是确定的函数，它们也可能具有随机性。这种随机函数可以帮助我们随机的指定决策，这样我们拥有更多的状态空间。例如：

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

该公式表示在该状态下，某个动作action发生的概率。



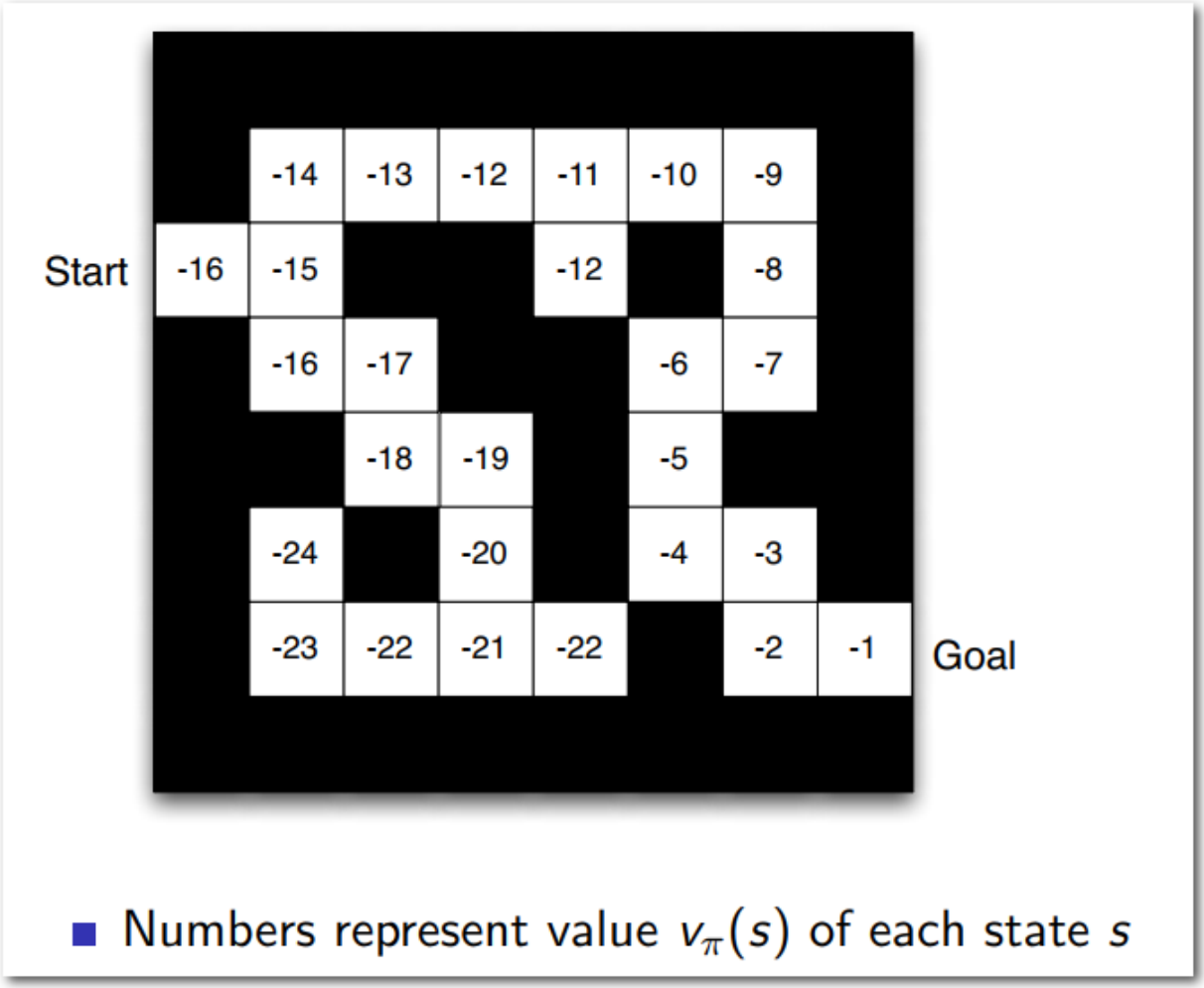
图中我们用箭头表示每一个状态的策略，对于Deterministic policy而言，它的输出是一个action，而对于Stochastic policy而言，它的输出则是一个动作发生的概率。

3.2 Value Function 价值函数

Value Function是对未来奖励的预测，用来评价当前状态的好坏程度。当面对两个不同的状态时，个体可以用一个Value值来评估这两个状态可能获得的最终奖励，继而指导选择不同的行为，即制定不同的策略。同时，一个价值函数是基于某一个特定策略的，不同的策略下同一状态的价值并不相同。某一策略下的价值函数用下式表示：

$$v_\pi(s) = E_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s]$$

γ表示折扣因子,表示随着step增加，该step的reward变得越来越不重要



在迷宫中，我们可以得到每一个状态的value，在每个状态过程中，通过评估此时action所到达的状态的最大值，来确定我们的policy。

3.3 Model 模型

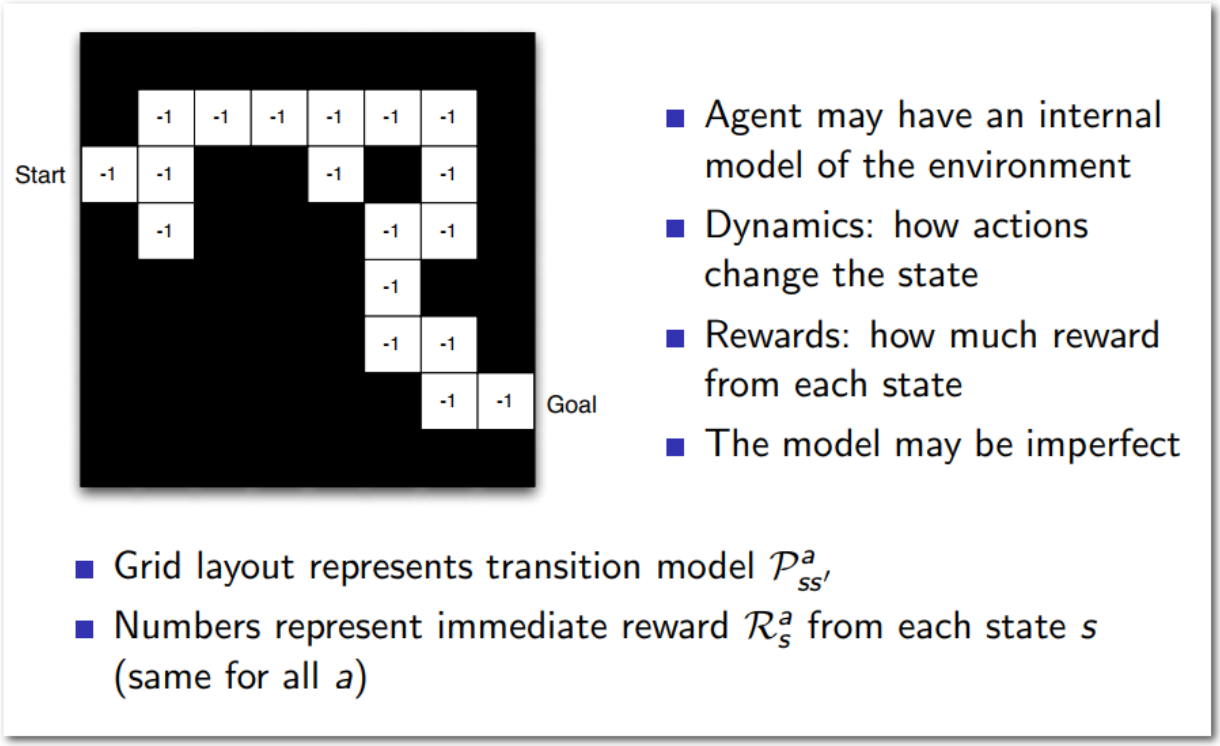
model 并不是环境本身,主要是用来预测下一步环境的变化，以此来确定下一步我们的行动应该是什么。model主要包含两部分：transition model与reward model。
transition model($\boldsymbol{P}$)用来预测下个状态，预测环境的动态变化。

> 状态转换模型：$\mathcal{P}_{ss'}^{a} = P[S' = s' | S = s, A = a]$
> 这个模型告诉我们根据先前的的状态以及动作、所处的环境预测下一个状态的概率。

reward model($\boldsymbol{R}$)用来估计我们得到的奖励。公式表示如下：

> Reward模型：$\mathcal{R}_{s}^{a} = E[R | S = s, A = a]$
> 这个模型告诉我们预期的奖励是基于先前以及当下的状态

Model对环境的动态特性(Dynamics)进行建模，判断某一个 S_t将会产生什么样的 S_{t+1} 和对应的 A_{t+1} 。构建环境并不总是需要model这个环节。



图中的-1表示每一步就减去一个奖励。model就是创建自己对于环境的映射，model尝试找到环境的变化，这个映射代表了model目前对于环境动态特性的理解。当你移动的时候，理解这个映射，也就是理解状态之间的装换。

3.4 Categorizing RL agents

我们可以对强化学习进行分类，分类的依据是根据agent 是否包含policy、value、model这三个关键元素。 根据这个依据，可以分为分为三类：

1. **Value Based** 基于价值函数 (我们使用Value function 来计算Policy，Policy是隐含的。不需要直接定义)
    - No Policy (Implicit)
    - Value Function
2. **Policy Based** 基于策略函数(每一个状态都映射到一个action，直接使用这个action做决策)
    - Policy
    - No Value Function
3. **Actor Critic** (综合使用Policy 和 value function。使用Policy，同时保留每次所得到的value function)
    - Policy
    - Value Function

此外，在强化学习中，根据agent是否包含model分为两类:Model Free与Model Based

- **Model Free**
    - Policy and/or Value Function
    - No Model
- **Model Based**
    - Policy and/or Value Function

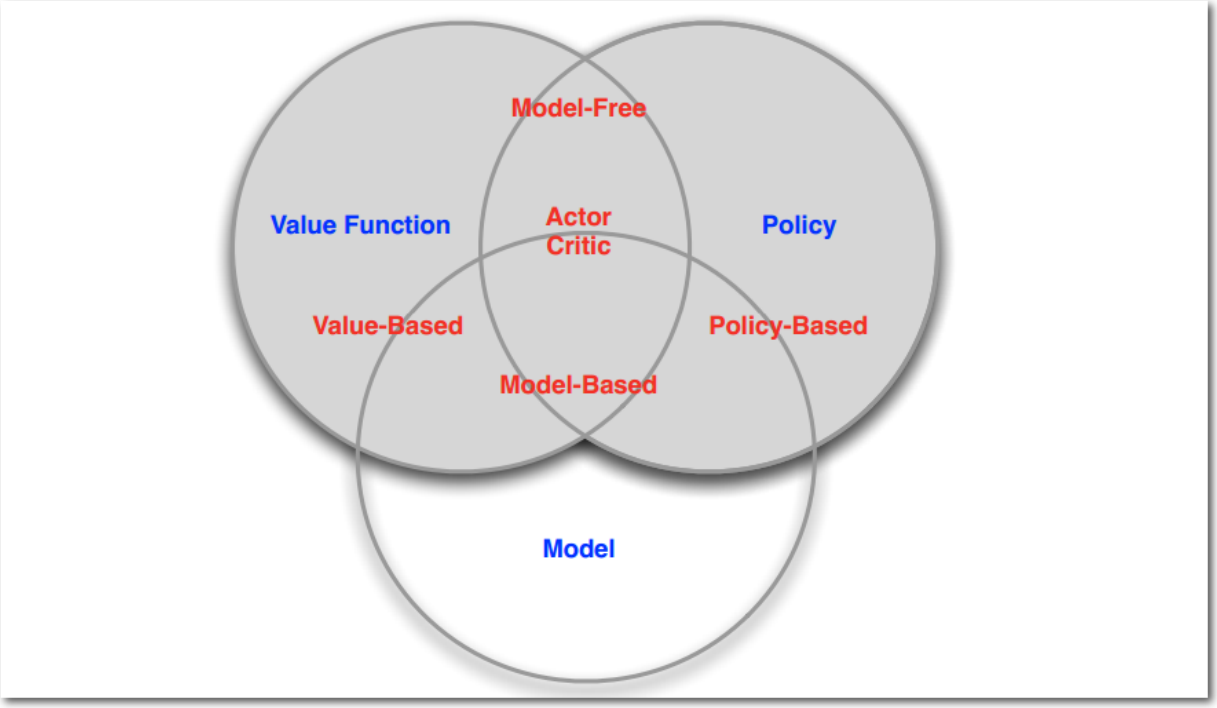
- Model

Model Free意味着我们不会创建一个动态模型，来预测直升飞机接下来会如何运动，只是直接根据value function或者policy来直接得出结论，不需要在意环境会如何改变。

Model Based意味着我们需要建立一个model来表示环境的工作原理，来预测环境会根据我们的决策如何改变。比如建立一个关于直升飞机的动态model,通过这个model，我们能知道接下来会发生什么，并找到最优的行动方式。

3.5 RL Agent Taxonomy

下面通过一张图片来展示分类情况：

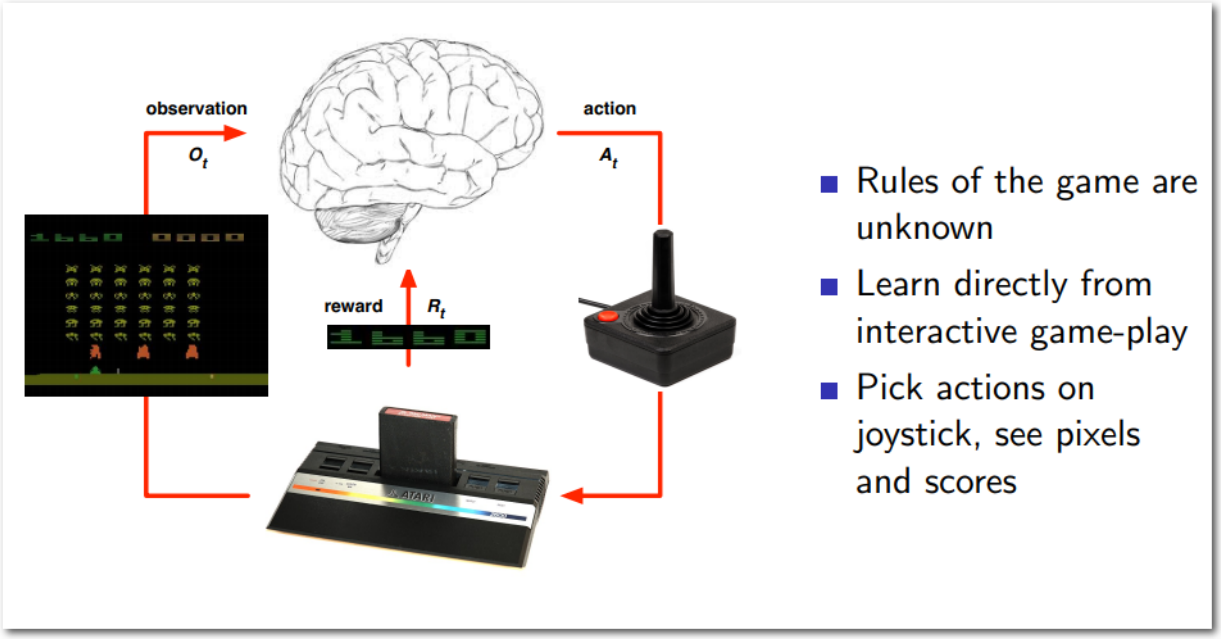


4.Problems within RL 强化学习的一些问题

4.1 Learning and Planning 学习和规划

4.1.1 Reinforcement Learning

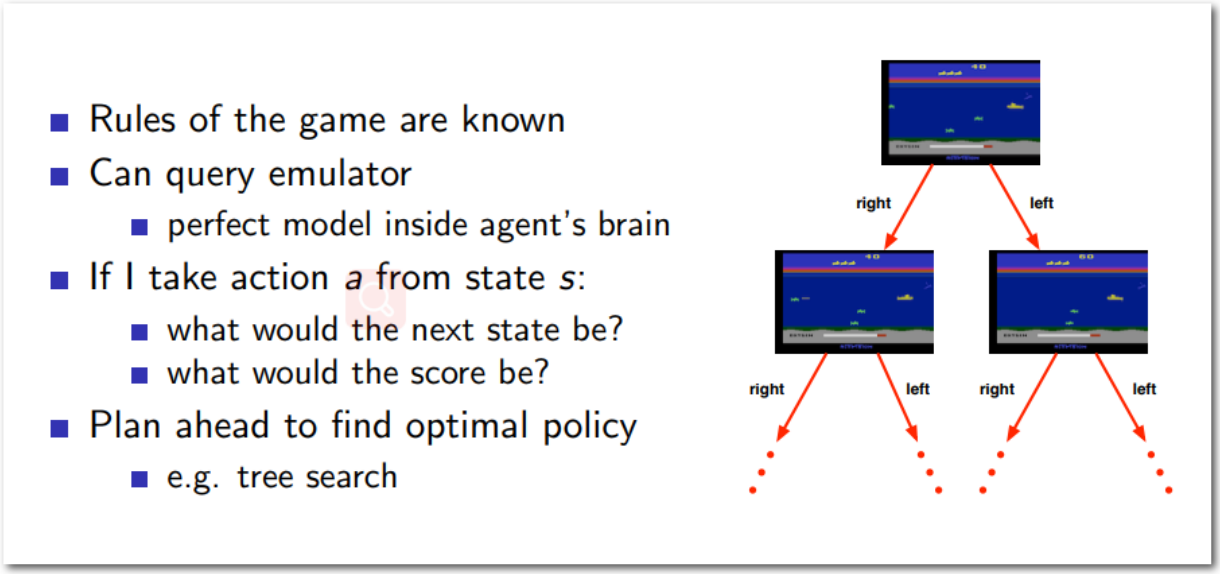
环境初始时是未知的。它不知道环境，它需要与环境进行交互，不断地进行尝试，不断地进行学习，逐渐改善其行为策略，最终在环境中找到最好的方式采取行动，并最大化奖励。



如上图所示，我们不知道游戏的规则，只是观察分数和屏幕，并控制操纵杆，通过不断的试错，找出游戏的规则。

4.1.2 Planning 规划

环境是已知的，agent 被告知了整个环境运作规则的详细信息,不需要与外部环境进行交互。根据已知的信息，agent能够计算出一个完美的模型,根据这个环境的模型和当前的状态进行计算，然后改善policy,知道如何采取行动，寻找最优解。



假如有人告诉我们模拟器是如何工作的，那么agent就能得到完美模拟器，我么可以将这个模拟器看作是一个完美的外部model,这样在不与外部环境进行交互的情况下，通过已知的变化规则，在内部进行模拟整个决策过程，进行提前规划。

一个常用的强化学习问题解决思路是，先学习环境如何工作，也就是了解环境工作的方式，即学习得到一个模型，然后利用这个模型进行规划。learning和planning 是相互关联的。

4.2 Exploration and Exploitation 探索和利用

强化学习是一种不断试错的模型，agent 需要从环境中获得一个最好的policy，这样在探索过程中就不会丢失太多的奖励。探索意味着发现更多的关于环境的信息，开发意味着开发利用你所找到的信息。我们需要平衡两者之间的关系，进而最大化我们的奖励。

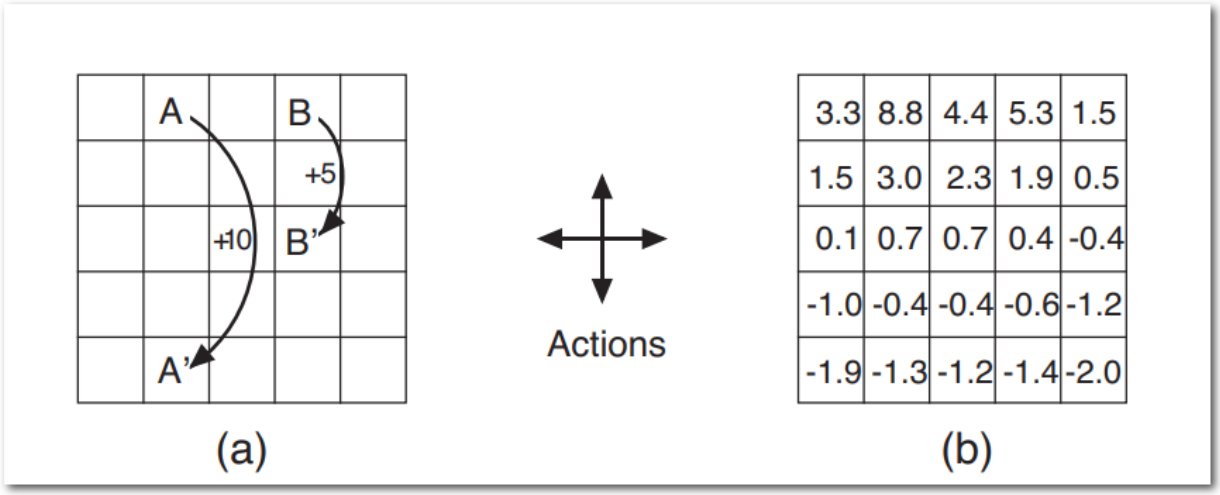
- **Exploration(探索)**：倾向于探索环境中新的信息
- **Exploitation(开发)**：倾向于开发使用我们已经探测得到的最大reward

举个例子：在网站上投放什么样的广告。通过一段时间的开发利用，我们知道投放什么样的广告人们的点击最多。但是，有的时候我们需要尝试投放一些不同的广告，这些广告还没有被人们看到过，也许人们会点击他们，也许这些广告更受欢迎，这样的话就会有更多的收入。

4.3 Prediction and Control 预测和控制

- **预测(Prediction)**:给定一个policy，这个policy能够得到多少reward。这是一个预估未来的过程
- **控制（Control）**：确定众多决策中，哪一个决策能够得到最多的奖励。

首先是预测问题：在下面的方格中，我们规定从A->A'，我们将得到+10的奖励，而B->B'可以得到+5的反馈，而其它步骤则是反馈-1。现在，我们给定一个policy：在任何state中，他的行为模式都是随机的，也就是上下左右的概率各25%，那么，预测问题要做的就是，在这种决策模式下，我们的value function是什么，也就是下图中(b)：



接着是控制问题：在控制问题中，我们的问题背景和上面一模一样，唯一的区别就是，我们不再限制policy，也就是说行为模式是未知的，我们要自己确定，所以我们通过解决控制问题，求得每一个state的最优的value function，如图b所示，也得到了最优的policy，如图c所示：