

南京大学俞扬博士万字演讲全文：强化学习前沿（上）

本文作者：奕欣

2017-05-09 16:49

“导语：本文根据俞扬博士在中国人工智能学会AIDL第二期人工智能前沿讲习班“机器学习前沿”所作报告《强化学习前沿》编辑整理而来，雷锋网在未改变原意的基础上略作了删减。

雷锋网(公众号：雷锋网) [AI科技评论] 按：本文根据俞扬博士在中国人工智能学会AIDL第二期人工智能前沿讲习班“机器学习前沿”所作报告《强化学习前沿》编辑整理而来，雷锋网在未改变原意的基础上略作了删减，经俞扬博士指正确认，特此感谢。全文分为上下两篇，本文为上篇。



俞扬博士、副教授，主要研究领域为人工智能、机器学习、演化计算。分别于2004年和2011年获得南京大学计算机科学与技术系学士学位和博士学位。

2011年8月加入南京大学计算机科学与技术系、机器学习与数据挖掘研究所（LAMDA）从事教学与科研工作。曾获2013年全国优秀博士学位论文奖、2011年中国计算机学会优秀博士学位论文奖。发表论文40余篇，包括多篇Artificial Intelligence、IJCAI、AAAI、NIPS、KDD等国际一流期刊和会议上，研究成果



奕欣
初心者



职业开脑洞 脱线段子手。邮箱：
guoyixin@leiphone.com



发私信

当月热门文章

清华大学朱军博士：可扩展的贝叶斯方法与深度生成模型

发现未来连接创新，CCF-GAIR 2017将于7月7日再度起航！| GAIR 2017

南京大学俞扬博士万字演讲全文：强化学习前沿（上）

阿里云 iDST 总监初敏博士：AI技术与商业化之路 | GMIC 2017

谷歌大脑撰文解析 AutoML：神经网络如何自行设计神经架构？ | Google I/O 2017

最新文章

获得IDEAL'16、GECCO'11、PAKDD'08最佳论文奖，以及PAKDD’ 06数据挖掘竞赛冠军等。

任《Frontiers of Computer Science》青年副编辑，任人工智能领域国际顶级会议IJCAI’ 15/17高级程序委员、IJCAI'16/17 Publicity Chair、ICDM'16 Publicity Chair、ACML'16 Workshop Chair。指导的学生获天猫“双十一”推荐大赛百万大奖、Google奖学金等。

在此列出俞扬老师讲课目录，以供读者参考：

- 一、介绍 (Introduction)
- 二、马尔可夫决策过程 (Markov Decision Process)
- 三、从马尔可夫决策过程到强化学习 (from Markov Decision Process to Reinforce Learning)
- 四、值函数估计 (Value function approximation)
- 五、策略搜索 (Policy Search)
- 六、游戏中的强化学习 (Reinforcement Learning in Games)
- 七、强化学习总结
- 八、强化学习资源推荐

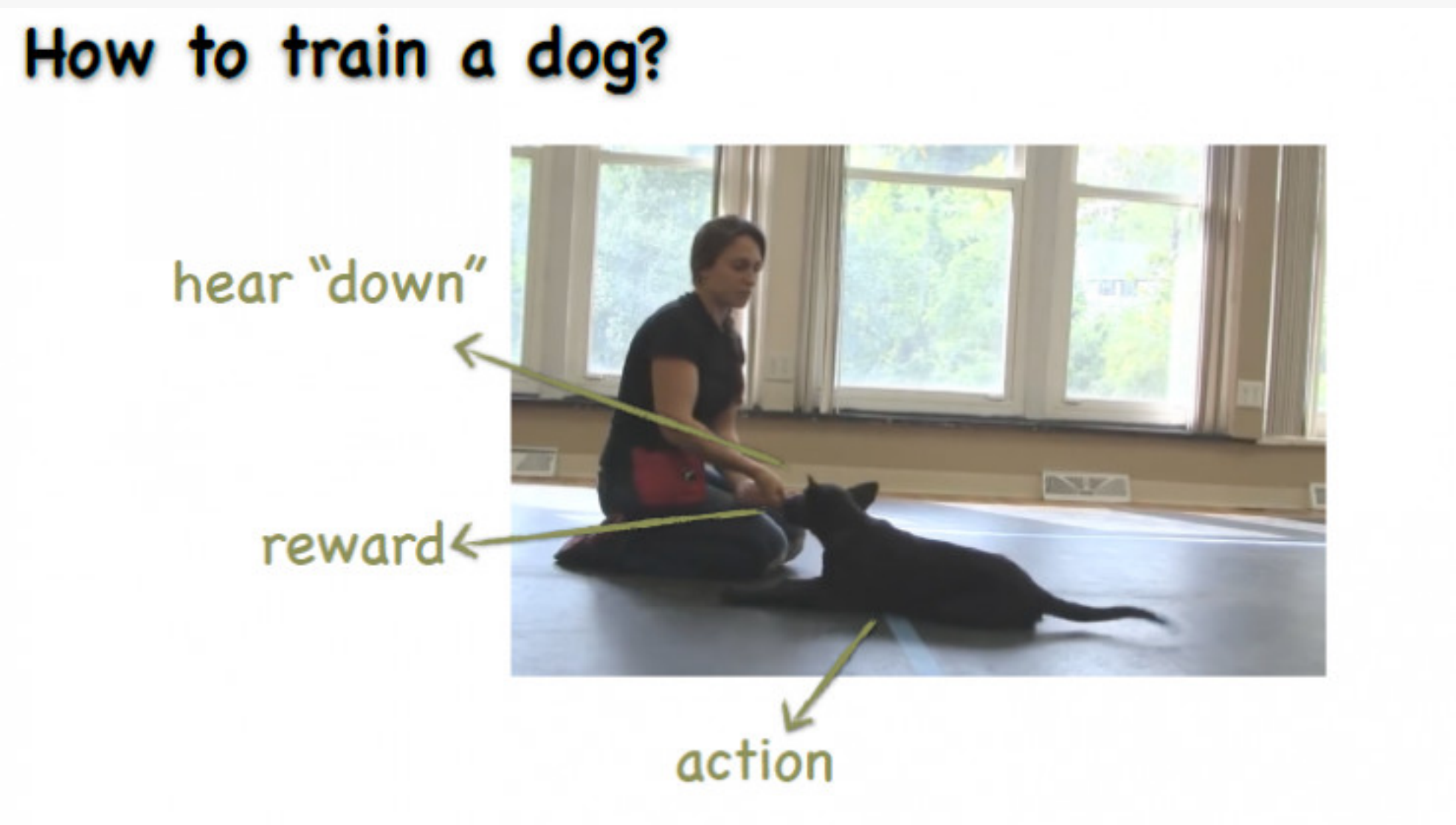
以下为俞扬博士的演讲正文：

大家好,我会尽量通过直观的方式简单的介绍一下强化学习的三个问题。由于水平有限，所以难免会有一些不足或者不到位的地方，请大家指正。

- 第一，强化学习到底是什么？
- 第二，强化学习有哪几类算法？这几类算法的思路是什么？
- 第三，强化学习能用在什么地方？应用时会遇到什么限制？

一、介绍 (Introduction)

从动物的学习过程说起



现在大家都在说人工智能，虽然可能难以精确的说清楚到底什么叫做智能，但我们知道拥有智能会有一些聪明的表现。例如像犬等一些动物，我们可能会认为是有一定智能的，我们可以训练一只幼犬听懂饲养员的指令。训练方法是：饲养员手里拿着食物，然后说“坐下”，当幼犬做出符合要求的动作时，就把食物

清华大学舒继武教授：基于非易失存储器的存储系统软件层优化 | CCF-ADL 火热报名中

AI时代的产品经理，应该注意什么？

端午福利！ AI慕课学院请你来听课（适用专属优惠码）

谷歌输入法背后的机器智能：思你所思，想你所想！

清华大学博士生涂锋斌：设计神经网络硬件架构时，我们在思考些什么？（下）| 硬创公开课总结

一键调用Siri、Alexa、Cortana和Google Assistant，一款蓝牙耳机居然做到了

热门搜索

- 乐视
- 汽车
- 增强现实
- 宅客
- Kindle Fire
- BlackBerry
- iOS 7
- 支付
- Firefox
- Google Now
- iPhone 6 Plus

给它。反复进行训练，大概半小时的时间，它就学会听见“坐下”的命令就坐下。这个过程就是动物的学习过程，它的智能就表现在它能在一定时间内发现如何适应环境，获得食物奖赏。

在很早之前，就有许多学者在想能不能让计算机也做到相同的事情，自动发现如何适应环境，这也就是我们今天说的强化学习。有这么一种说法，说“强化学习是真正的人工智能”。我们现在不评价这句话讲的合适不合适，至少，强化目的是希望机器能和动物一样，有较好的适应环境的能力。

从动物的学习类比机器的强化学习

- 强化学习名字的由来

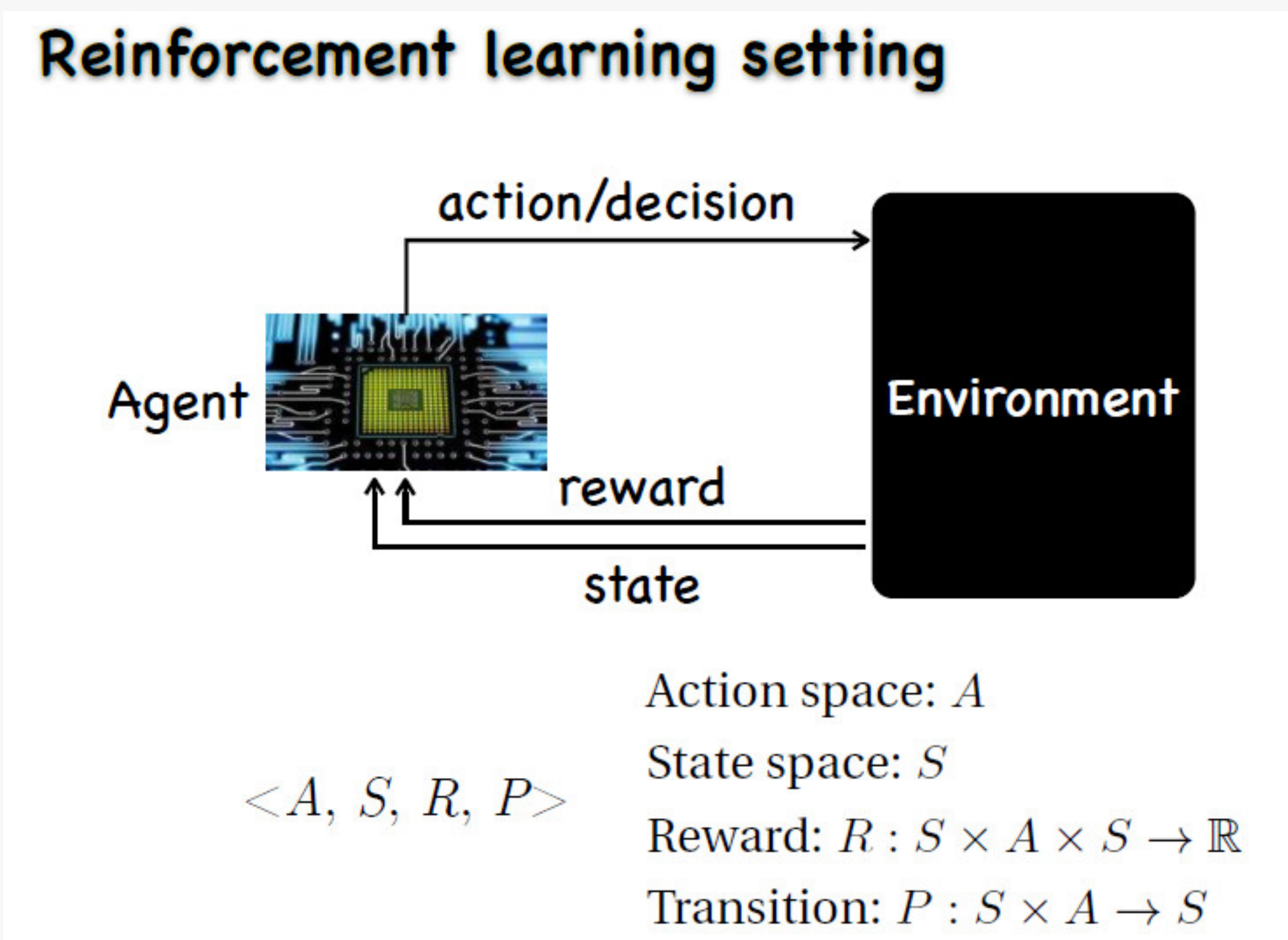
这里先解释一下强化学习这个名字。为什么叫强化学习呢？因为这个过程是不断的重复、不断强化认知，英文Reinforcement Learning 中的 Reinforcement 更准确的中文翻译也是“强化”。

- 类比强化学习和动物学习

训练幼犬的过程有两个要素：

- 饲养员需要对幼犬发出指令，比如让它“坐着”，
- 饲养员手中有动物非常想要的东西，即奖赏。对狗来说，奖赏就是食物。

对于智能体（Agent，即计算机）来说，我们希望通过类似的方法能够训练智能体，我们把其中的要素抽象出来，可以用下面这个图来表示：



现在智能体处于一个很暗的环境之中，意思是它并不知道这个环境里面到底是什么，这也是我们希望计算机通过强化学习能做到的事——把它扔到一个未知的环境里面，它能够通过和环境打交道来适应这个环境，学习到做什么动作才是最好的。

Agent能够从环境里面观测到的东西有两个：

- 状态。它能够观测到的环境和它自己的状态；
- 奖赏。当它做出一定动作以后，这个环境可能会给它一个奖赏。

它根据观察到的状态做出的行动，叫做动作或决策；这个动作放到环境里以后，会在环境里被执行；执行以后，环境会发生变化。

总体来说，如果按照刚才的要素把它刻画出来，它对应的变量有：动作的集合、状态的集合，奖赏函数，以及做完一个动作以后，决定环境会发生什么变化的转移函数。

对于Agent来说，自身具备的选择决策的能力，叫做策略。这个策略意思就是，观测到了环境现在处于什么状态，而选择做出什么动作出来。这里的策略和监督学习里的模型其实是一回事。

初识强化学习

- 从智能体的视角来看它所处的环境，以及它所做的动作

- 刚睁开眼睛的时候，它看到的环境是一个初始状态。
- 根据这个状态，智能体做了一个动作。我们把策略写成 π ， π 会根据当前的状态选择一个动作，然后到环境中去具体执行。
- 执行了以后，这个环境会发生状态转移（Transition），变到下一个状态。同时，也会反馈给智能体一个回报或者奖赏（Reward）。
- 最后，智能体继续根据新的状态来决定它下面做什么样的动作。

所以从智能体的视角来看，即看到什么状态，然后决定做一个相应的动作并会收到回报，然后又在下一个状态做一个动作并收到一个回报，这样一直下去。

Reinforcement learning setting

$\langle A, S, R, P \rangle$

Action space: A

State space: S

Reward: $R : S \times A \times S \rightarrow \mathbb{R}$

Transition: $P : S \times A \rightarrow S$

Agent: Policy: $\pi : S \times A \rightarrow \mathbb{R}, \sum_{a \in A} \pi(a|s) = 1$

Policy (deterministic): $\pi : S \rightarrow A$

Agent's goal:

learn a policy to maximize long-term total reward

T-step: $\sum_{t=1}^T r_t$ discounted: $\sum_{t=1}^{\infty} \gamma^t r_t$

all RL tasks can be defined by maximizing total reward

```
graph LR; Agent[Agent] -- "action/decision" --> Environment[Environment]; Environment -- "reward" --> Agent; Environment -- "state" --> Agent;
```

所以大家可以很明确地看到：

- 第一，这个智能体不是做一次决策就完成了学习的过程。实际上，它要做的是一个序列的决策。
- 第二，我们怎么评判智能体策略的好坏呢？一般评判的形式就是，它能拿到的奖赏会有多大。每一步都可能有奖赏，所以评判的形式是把总的奖赏加起来看看它有多大。

长期累积奖赏有好几种计数法，比如我们可以把总的T步将上全部加起来，或者用折扣（discounted）的方法，可以让它走无穷多步，但是不是按照原数值加起来，而是要考虑权重，这个权重会因时间的流逝而产生折扣。

- 在算总奖赏的时候，为什么要考虑权重？

一方面，是因为在数学上比较好处理；另外一方面，是说在很多真实的应用里边，我们对未来的价值的估计有可能是折扣的。举例说明：

如果你今天能够赚到100块，或者下个月能够赚到200块，在这两个决策里面你要选一个，你可能会选择今天就拿这100块钱，将来对你来说可能会比较稳妥，下个月发生的事情还不知道会怎么回事，所以在有的应用里边会考虑折扣，但需要在不同的应用中考虑具体的需求。

智能体要做的事，就是找一个能够带来最大的长期累积奖赏的策略。

通过设置奖赏，我们可以定义智能体，让它去做不同的事情。就像刚才训练这只幼犬一样，我们通过给它一个吃的，可以让它做趴下的动作，也可以让它做站起来的动作。

寻找最优策略的两个例子

实际上强化学习的框架只是一个一般的框架，这个框架可以包含很多很多问题在里面。下面举两个例子。

- **第一个例子：寻找最短路径的问题。**

条件如下图所示，我们要找一条从s到t的最短路径。这是很简单的问题，这里演示把最短路径变成强化学习的问题去解决，我们可以这样做：

Reward examples

shortest path:

- every node is a state, an action is an edge out
- reward function = the negative edge weight
- optimal policy leads to the shortest path

先定义状态和动作：

1. 把每个节点当成是一个状态；
2. 把每个节点上面连着的边，当作这个状态下面可以做的动作。

定义好了状态和动作，我们就要找最短路径，也就是要找到路径的权重和加权最小。

通常强化学习是让奖赏最大化，因此这里把路径上的权重先取一个负的值，让它最大化这个负的值。稍微做一点变化的是，t会指出来一个单独的绕自己循环的节点。

接下来，就开始找最优策略。我们先假设能够找到最优的策略。最优的策略是什么呢？就是从s开始，我们选择每一步从哪一条边走出去，能使总的奖赏最大。我们看到这里有一个100，这是很大的，所以一定能走到t去，除掉100这个意外情况，上图加粗线所示路径的奖赏应该是最大的。

对于最优的策略来说，它们对应的就是一个最优的路径，我们这里先不管最优的策略怎么求解。

- **第二个例子：最大化任意函数**

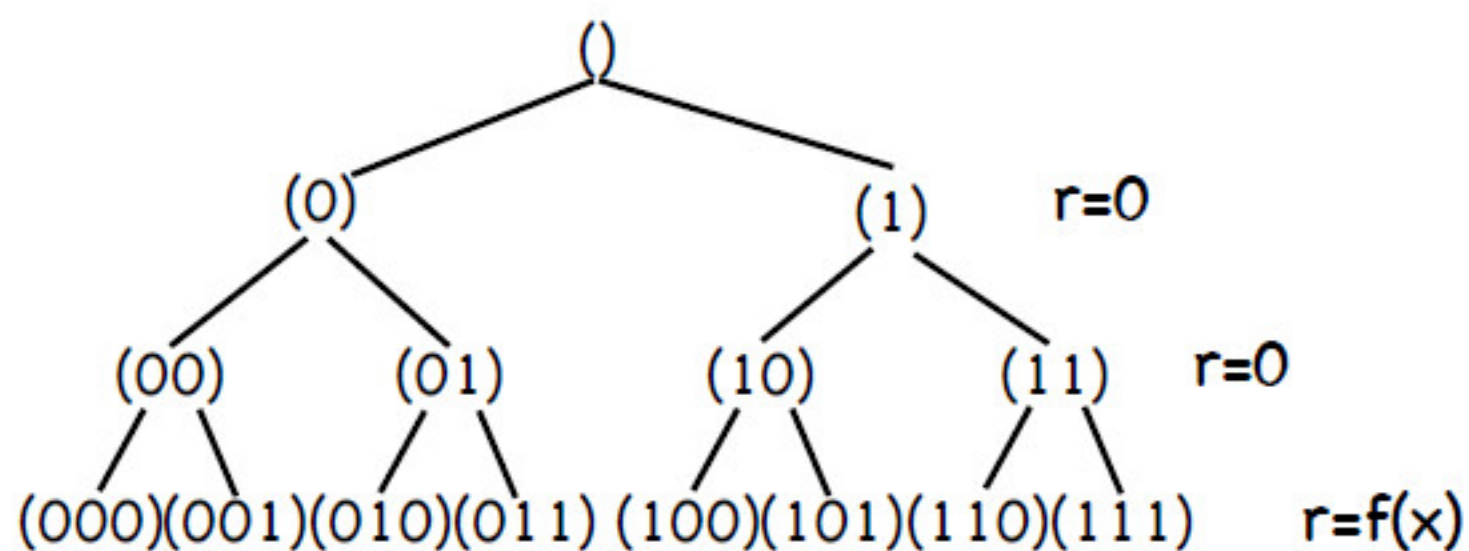
我们展示了怎么用强化学习来解决最短路径这个问题；除此之外，强化学习还可以包容很多其他问题。

比如，我们要在0、1的N维空间里面最大化一个函数f。这不是一个容易解决的问题，特别是没有规定这

个f是什么，换句话说这个f是什么都可以。

Reward examples

general binary space problem $\max_{x \in \{0,1\}^n} f(x)$



solving the optimal policy is NP-hard!

这个问题也可以变成一个强化学习的问题，怎么变呢？

我设定初始的状态里边是空集；这个时候有两个动作，往左走是加一个0，往右走是加一个1，0再往左走再加一个0，再往右走再加一个1；走出N层以后最上面这层就是0、1空间里面所有的组合，对应所有可能的解。

我们还要设定一个奖赏——中间每一层奖赏都是0，只有最后一层的奖赏是F。这就会使得，如果有一个最优的强化学习的策略，能够找到最优的路径到达节点，那么它就能使得这个奖赏最大、F最大。

通过这个例子，我想表达一个观点——如果我们面对的这个学习问题比较简单，就没必要用强化学习。不能因为它自己在市面上比较火，而把以往的一些问题换成用强化学习的方法来解决。

强化学习（RL）和规划（Planning）的不同

总结一下，强化学习和规划哪里不同？

- 第一，强化学习看到的世界一个黑箱子，而对于规划而言，这个世界却是很清楚的。比如我们的最短路径，所有的节点、便点、权重点都是已知的；而对于强化学习，状态如何转移、边的权制是多少、甚至有哪些状态都需要自己探索、发现。
- 第二，规划的问题可能就是一个解、一个路径；而强化学习的解是一个模型。和监督学习一样，只要输入任意一个状态，强化学习都会告诉你应该做什么决策。因此，除了给出最优路径上的每一个状态、每一个节点应该往哪边走以外，实际上任何一个节点都能告诉我从这个节点到目标去应该怎么走。

强化学习（RL）和监督学习（SL）的不同

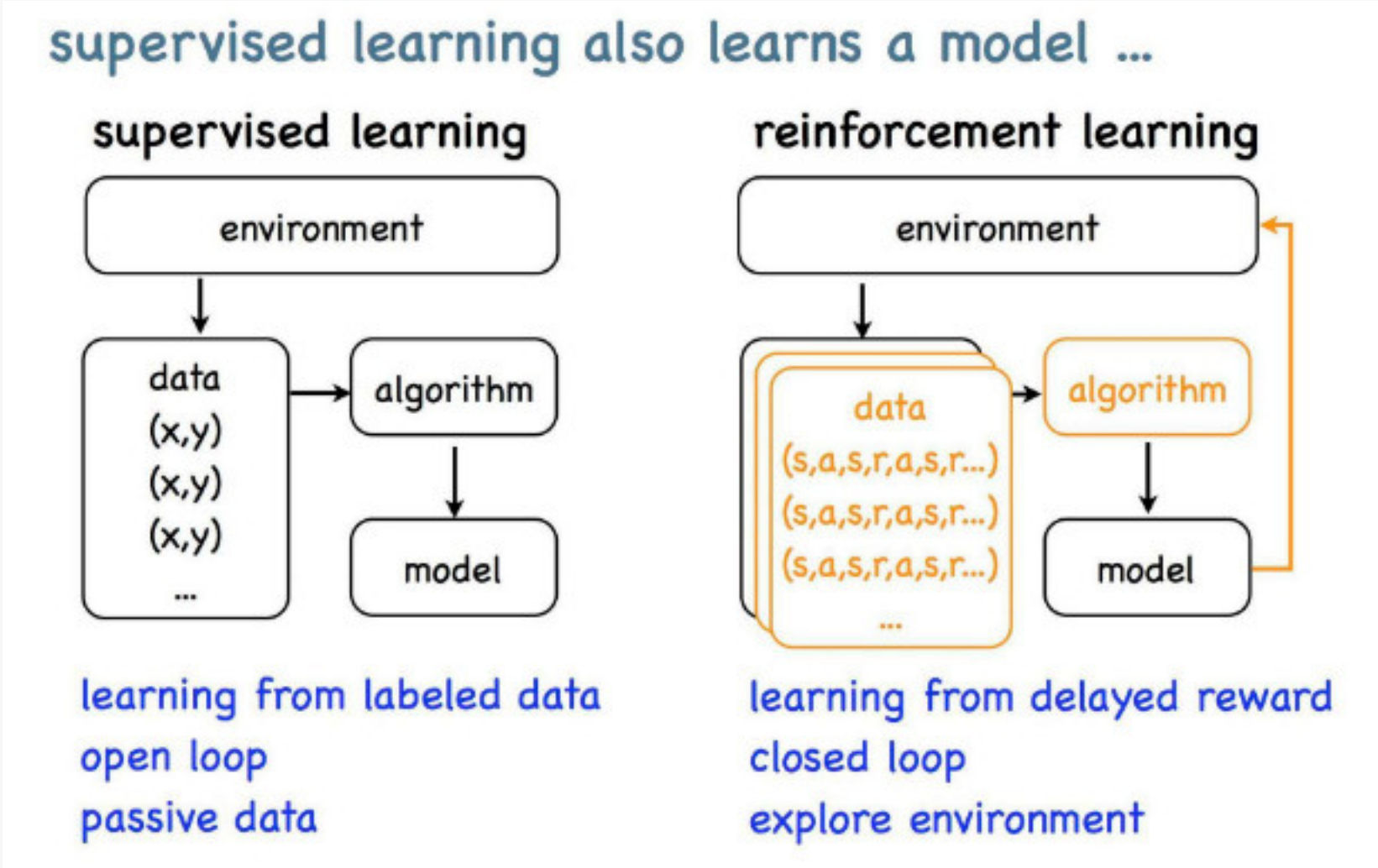
刚才说到强化学习和监督学习有很多相似的地方，比如说模型实际上是一样的。那它们之间有何差异呢？

- 监督学习总的来说是一个开环的学习。

1. 通常，监督学习任务会从环境中收集一批数据；
2. 接着我们用监督学习算法从数据中产生模型；
3. 最后就可以用这个模型来做预测了。

- 但是对于强化学习来说，它面对的是一个闭环的学习。

- 首先，也是从环境中产生数据；
- 用强化学习的算法从数据中产生模型；
- 还要把模型放回到环境中运行，接着又会产生新的数据出来，再重复以上步骤。



因此从大体上看，两者的主要区别，一个是开环学习，一个是闭环学习。这点不一样就带来了很多具体区别：

首先，在监督学习里，数据是分成观测的特征值和一个标记。这个标记的含义是，看到这样一个观测的值、特征以后，应该做出什么样的预测。

但是在强化学习里面，这个数据首先是一个序列，做了一个动作以后下面又到了什么状态，有一个反馈值，并且有了新的状态。这个序列里面虽然有反馈奖赏，但这个奖赏并不能告诉我们应该做什么样的动作，而只是对现在的策略有一个评估值，我们把所有奖赏加起来作为当前策略的一个评估，可以得知策略做的有多好，但并不知道应该做什么样的动作是最好的，这个也是数据上的两个差别。

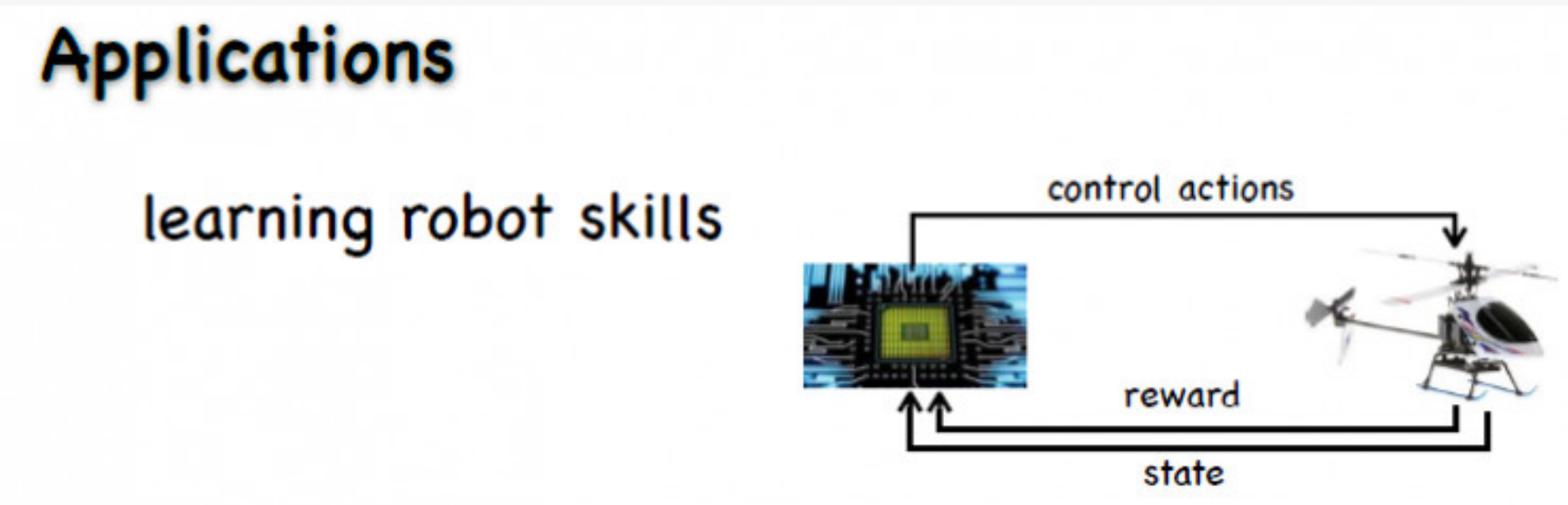
另外，强化学习的算法和监督学习的算法也是不一样的。两者的模型可能是一样的。监督学习里面可以是一个决策树，也可以是一个神经网络，也可以是一个模型，在强化学习里也是一样。

总结起来，两者最核心的区别，在于强化学习需考虑自身对环境的影响。

强化学习的应用

- 经典应用

由于强化学习做的是序列的预测和序列的学习，所以它以往主要的一个应用领域，是做机器控制，比如说直升机的操控。



在直升机的应用里面，智能体就是直升机，环境是其飞行空域，状态可以是直升机的高度、速度、姿态等等，采取的决策是操纵杆指令。我们希望直升机能够做出我们想要的轨迹，但是又不会掉下来。这些目标

可以作为直升机的奖赏，让它来学习一个策略，以实时控制直升机的运动。

- 更多的应用

有不少真实世界的应用，其背后面临的问题都符合强化学习的问题设定。比如说股市预测和商品推荐。



1、股市预测

首先这是一个序列决策，要做出很多的决策，每做一个决策动作都要看当前的股市的状态如何，动作可以是买、卖，和观望。

那为什么这个问题是强化学习问题呢？也有很多序列决策有可能并不是强化学习的问题，我们靠什么判断序列决策到底是不是强化学习呢？关键因素在于：决策放到环境里面执行以后，是否会改变这个环境。

在股市交易时，成交的那一刻会决定股价是多少，这相当于决策改变了环境。有时可能很少的交易，也会引起其他投资人对股市的预期，从而影响股市的走势。

2、另一个例子是商品推荐

为什么推荐问题也是可以看作它是一个强化学习问题呢？推荐系统会在网页上放置推荐展品，而用户的购买行为和推荐行为是有关系的。对于推荐的展品，即使比较普通也可以收到很多客户浏览，而优秀的商品如果没有被推荐出来则可能无人问津。总的来说，决策会影响整个系统。

3、近期的应用

在处理结构化数据时，比如做自然语言处理、把离散结构的知识库用到学习系统，会面临一个问题，即我们面对的语言或者知识库难以融入可微分模型中。一些研究者最近就想出来一些办法，把一个句子输出的词或知识库里面的操作，作为强化学习的动作，这样通过强化学习一些方法的可微分性纳入整个可微分学习系统中来。按照深度学习中比较流行的端到端训练的说法，强化学习的框架纳入进来以后，可把整个系统变成端到端的学习。

二、马尔科夫决策过程（ Markov Decision Process ）

强化学习基本数学模型——马尔科夫过程（ Markov Process ）

大家可能听到了很多词，包括MDP，Q-Learning、还有很多算法的名字，我在报告里就简单介绍一下强化学习发展的过程，以及里面会碰到什么问题。

强化学习的历史非常悠久，其中，早期的强化学习和它的一个数学模型MDP有很大关系，我先直观介绍一下MDP。

- 对MDP的直观介绍

MDP (Markov Decision Process) 里面有三个词，其中过程 “Process” 是代表时间变动的变量，马尔科夫 “Markov” 说明这个变动是没有记忆效应的，下一步往哪儿走只取决于当前状态。马尔科夫过程可以用图来描述，这个图上的每个点就是这一个状态。这上面有很多边，表示它可以做的动作。对于每一个状态来说，出边的概率和为1。这是从它的状态和转移角度来看的。

Markov Process

(finite) state space S , transition matrix P

a process s_0, s_1, \dots is Markov if has no memory

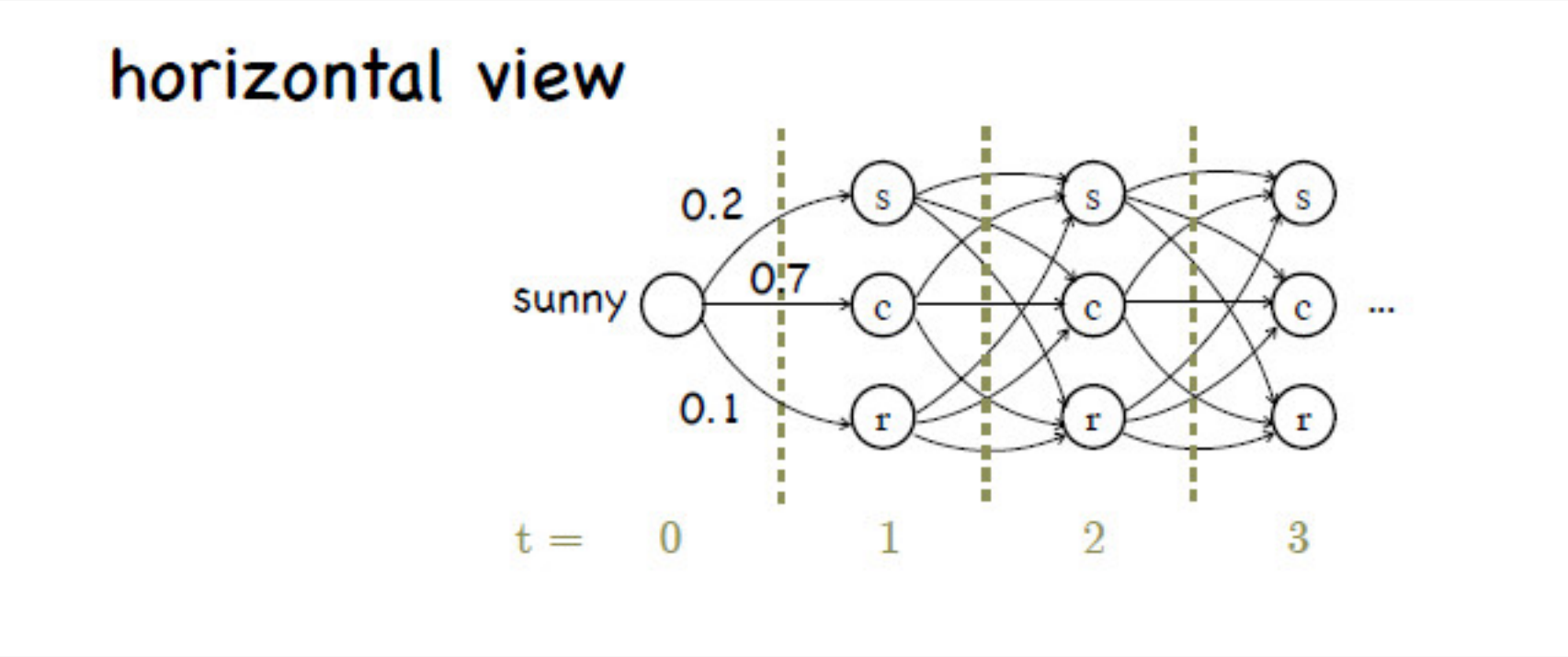
$P(s_{t+1} \mid s_t, \dots, s_0) = P(s_{t+1} \mid s_t)$ discrete $S \rightarrow$ Markov chain

$P =$

	s	c	r
sunny	0.2	0.7	0.1
cloudy	0.3	0.3	0.4
rainy	0.2	0.5	0.3

$s_{t+1} = s_t P = s_0 P^{t+1}$

我们还可以从时间的角度来看。比如说现在在某个状态，而到下一时刻，它会根据不同的转移概率转移到不同的状态去。随着时间的变化而转移到下一个时刻的状态去，我们把它称之为水平（horizon）视角。



稳态分布 (Stationary Distribution)

- 什么是稳态分布？

大部分马尔科夫的过程都会有一个稳态分布，意为当时间很长甚至无穷远的时候，大部分马尔科夫都会收敛到一个均衡的分布上，不再随时间变化。

比如说天气，今天的天气是出太阳，确定了出太阳、多云和下雨的转移概率以后，可能到30天以后它出太阳、下雨还是多云的概率和今天是不是出太阳已经没有关系了，它会收敛到一个确定的概率分布上面去。

- 马尔科夫回报过程 (Markov Reward Process) ？

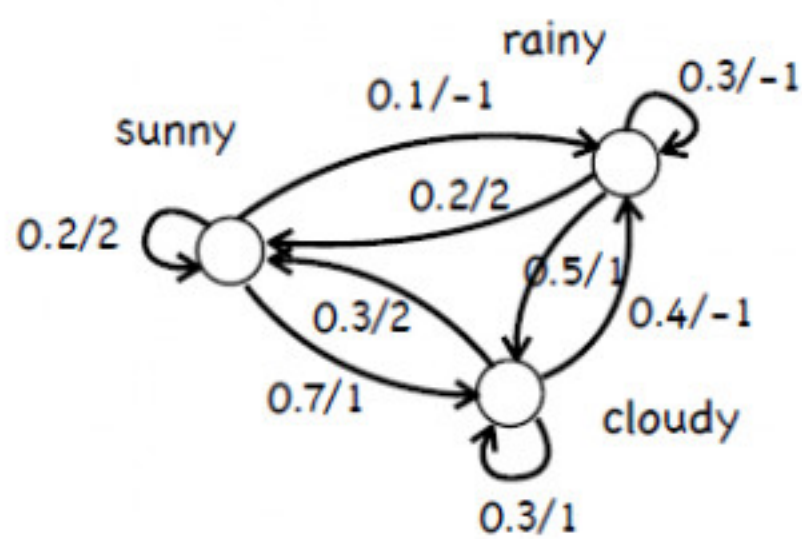
马尔科夫回报过程是当状态出现转移的时候，除了用刚才的转移概率描述以外，还存在一个奖赏。

假设天气一直是出太阳的状态，这样运行下去以后，我能拿到的总回报是多少。这个总的回报可以用一个符号V来表示。根据之前我们的描述，我们可以有不同的计算方式，比如说全部加起来或者打个折再相

加。

Markov Reward Process

introduce reward function R



how to calculate the long-term total reward?

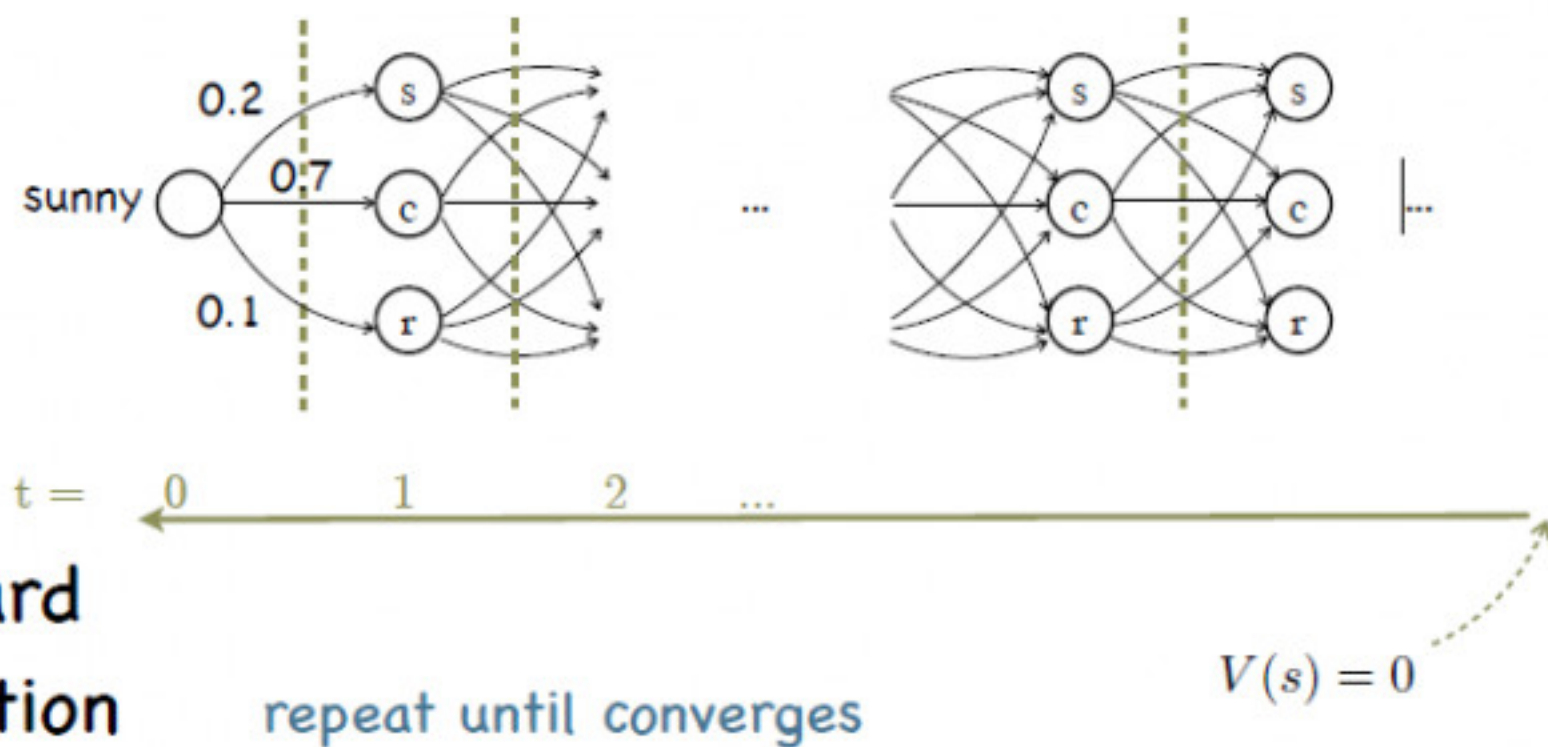
$$V(\text{sunny}) = E[\sum_{t=1}^T r_t | s_0 = \text{sunny}]$$
$$V(\text{sunny}) = E[\sum_{t=1}^{\infty} \gamma^t r_t | s_0 = \text{sunny}]$$

value function

怎么算长期回报？我们从初始状态开始，按照0.2、0.7、0.1分别转移到不同的状态之后，按新的概率，把这个状态以下总的回报值加起来，就得到这个状态回报的值。相当于这一步展开以后再部加起来。这就变成一个递归式，也就是第0步变成第1步要计算的步骤，第1步又变成第2步要算的步骤。

Markov Reward Process

horizontal view: consider discounted infinite steps



$$V(s) = \sum_{s'} P(s'|s) (R(s') + \gamma V(s'))$$

算法里有一个加速计算的方式，叫动态规划，是倒过来算的。

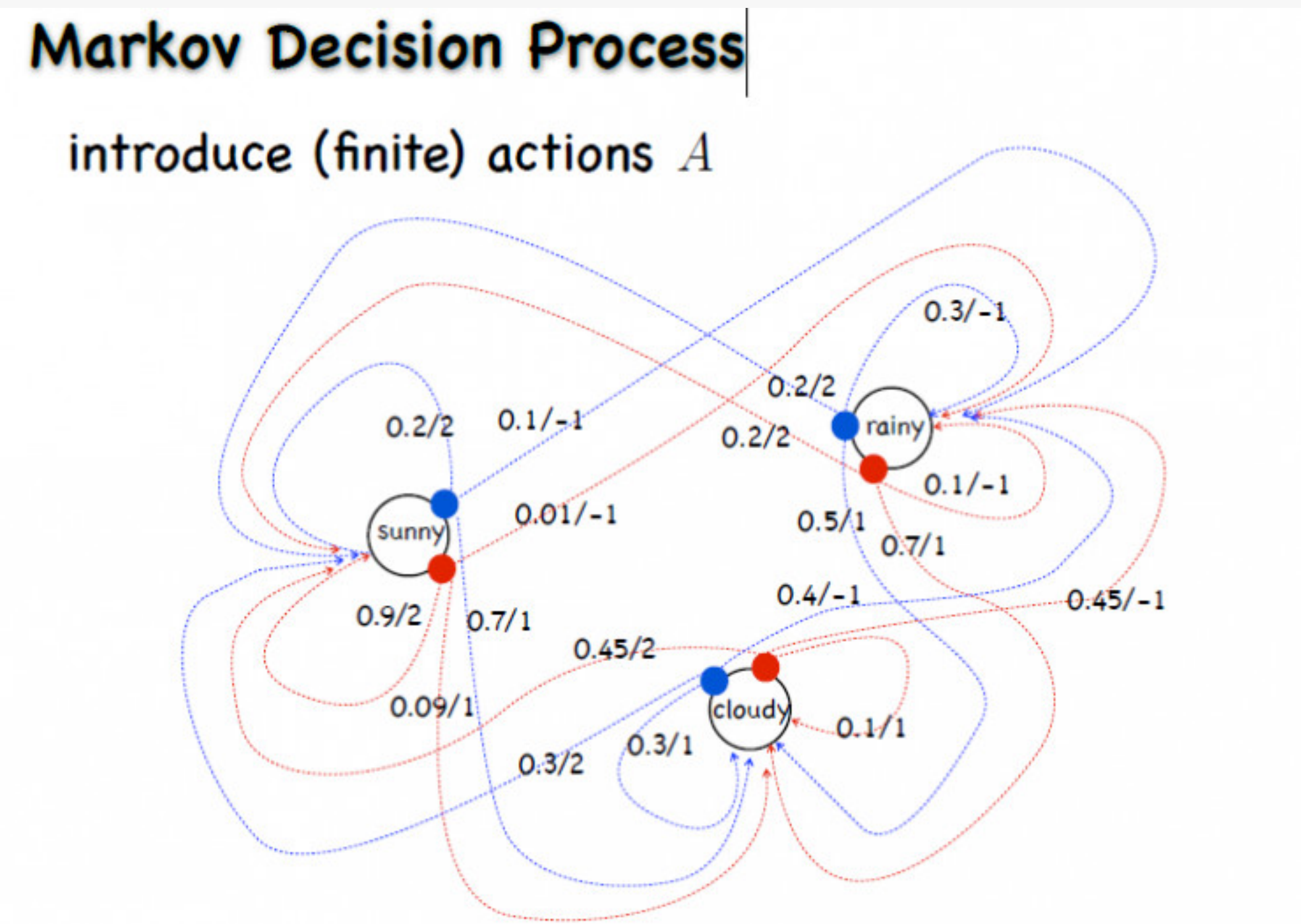
可以理解为，首先设置最后一层（第T层）的V值是0，倒过来算T-1层的V层是多少，再倒过来算T-2的.....把这个式子重复T次。

这是走T步的，还有走无穷多步的。我们假设站在无穷大的最后一个点上，这个点照样每个状态上面的V都是0，然后算无穷大-1步是多少，无穷大-2步是多少，往后退无穷多步。但是算法无法实现这个过程，实际上用算法也不需要退无穷多步，因为存在折扣，即退一定步数以后，这个值就会固定不变。

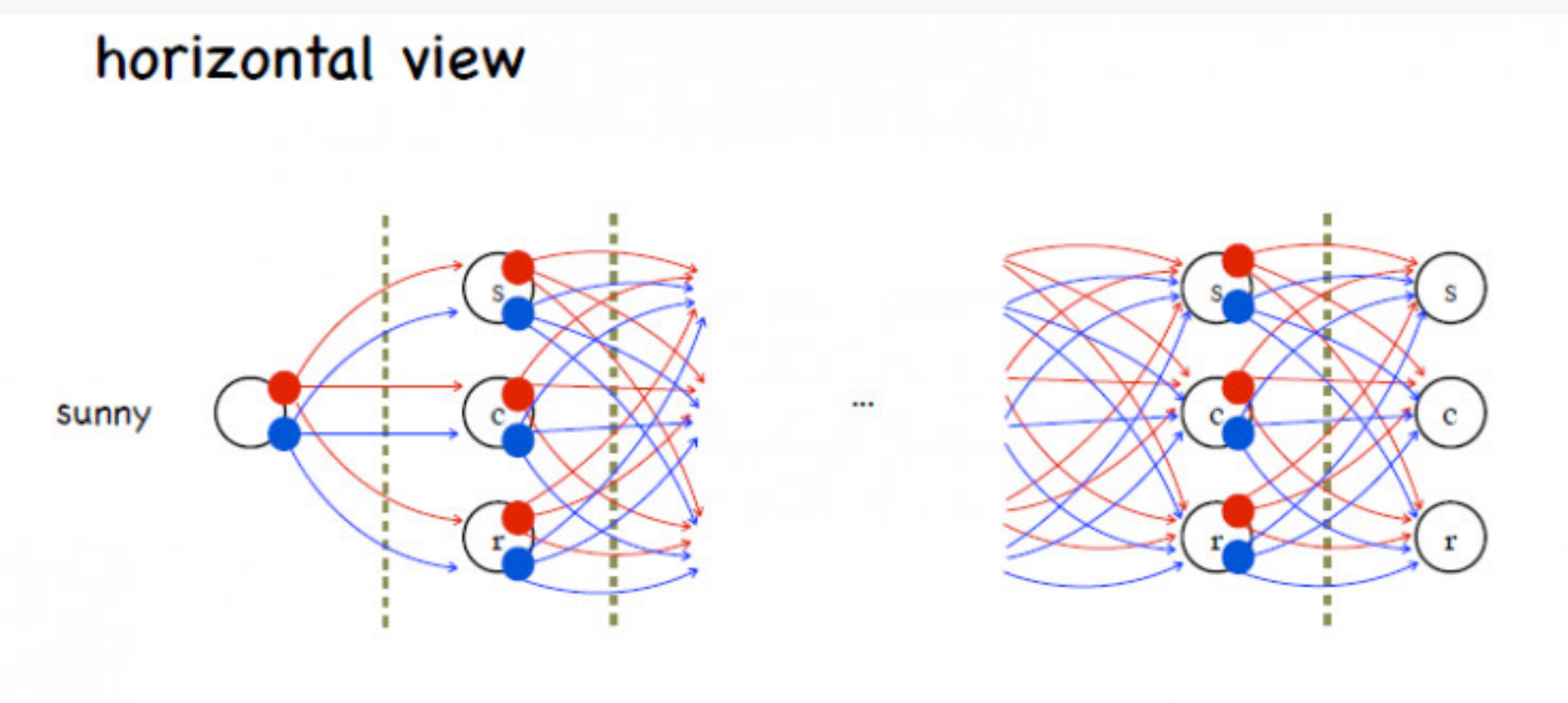
马尔科夫决策过程 (Markov Decision Process)

- 如何形成马尔科夫决策过程？

对于马尔科夫过程和马尔科夫决策过程，我们只能观察它运行下去的结果，而不能对它的运行过程加以干涉。加上一个决策以后就可以干涉了，这就是马尔科夫决策过程，不同的动作决定了转移的概率是不一样的，所以现在我们可以在每个状态上选择不同的动作。



再看马尔科夫决策过程的水平视角，由于每个状态可能做不同的动作，所以转移概率也不同。



总的来说，马尔科夫决策过程里有一个四元组，即状态、动作、奖赏、转移。

这个四元组和强化学习里面的四元组一样的，所以早期的强化学习是完全以MDP为数学基础的，对它来说也是要找一个策略，这个策略就是选择不同动作会有不同的概率，或者是确定性策略，在一个状态就输出一个动作。

早期强化学习的策略和其特点

早期的策略是用表格表示的，表格上记录了每个状态下的动作，是一个非常简单的模型。这个模型在强化学习里面很常用。在监督学习中，早期也用过这种模型，但由于在真实应用里面很少用得上，所以很快就被淘汰了。

它的特点是表达能力极强，不过前提是动作和状态都是离散的。为什么表达能力极强呢？比如说对于确定性策略，每个状态下面做什么动作可以直接修改，而不影响到其他状态的动作，所以它的表达很灵活。早期强化学习的很多理论都是建立在这种表达上，它虽然不实用，但是理论性质很好。

如何求解马尔科夫决策过程上的最优策略？

我们首先希望在马尔科夫决策过程上计算出给定策略的总回报。

这和前面讲的在马尔科夫回报过程上计算总回报是一样的，因为一旦给定策略以后，它的转移分布已经全部确定了。这就退化成一个马尔科夫回报过程，即给定一个策略以后我们计算回报方式跟前面一样。稍微不一样的一点是，它的转移是按照策略给出的动作的概率进行的。所以写V的时候，V右上角写了一个 π ，这个 π 就是表示我们当前固定的策略是什么，给出了不同的策略以后，我们要算的V值的结果是不一样的。这个V值表示的含义是，从s这个部分出发，走了很久以后看它的回报是多少。

但如果只是计算V值，从中导出策略不是那么方便，它表达的是总的回报，但我们想知道的是，在每个状态上做哪个动作比较好。如果只知道V值的话，是无法直接得知当前的状态上选择哪个动作好。只能每个动作尝试一下，走到下一个状态，看哪个动作导致的下一个状态的V值最好的，就用哪一个。这样比较麻烦。

为了避免麻烦，我们常用Q值函数。Q值函数比V函数多了一个动作输入，它要估计的是在状态s做了动作a以后，再跟着这个策略 π 一直做下去，它的回报是多少。有了Q值函，看到状态s后，把每个a带进去，看哪个a出来的Q值大，就用哪个a。所以这样就可以在当前状态直接决定用哪个动作了。

Q和V是有直接的对应关系的，如果按照策略来选择动作，平均的Q值就是V值。

计算最优策略

- 最优策略是否存在？

我们考虑最优策略的时候会想，是否会有一个策略在所有状态上表现都是最好的，还是只能找到在绝大部分时候表现都最好、但在个别状态上面值要差一点的策略。实际上前者是存在的，这个结论依赖于一个假设，即策略需要用表格来表示。因为用表格来表示的话，它的表达能力足够强。

Optimality

there exists an optimal policy π^*

$$\forall \pi, \forall s, V^{\pi^*}(s) \geq V^{\pi}(s)$$

optimal value function

$$\forall s, V^*(s) = V^{\pi^*}(s)$$
$$\forall s, \forall a, Q^*(s, a) = Q^{\pi^*}(s, a)$$

s	0	0.3
	1	0.7
c	0	0.6
	1	0.4
r	0	0.1
	1	0.9

最优策略对应的V值就是最优V值，对应的Q值就是最优Q值，怎么样求取最优的策略呢？由于这个V和Q之间是有一定关系的，所以我这里先直接给出两个等式，一个是通过Q值来算Q值的，一个是通过V值来算V值的。只要把最优Q和V的关系带到一般Q和V的关系中就直接可得。



Bellman optimality equations

s	0	0.3
	1	0.7
c	0	0.6
	1	0.4
r	0	0.1
	1	0.9

$$V^*(s) = \max_a Q^*(s, a)$$

from the relation between V and Q

$$Q^*(s, a) = \sum_{s'} P(s'|s, a) (R(s, a, s') + \gamma V^*(s'))$$

we have

$$Q^*(s, a) = \sum_{s'} P(s'|s, a) (R(s, a, s') + \gamma \max_a Q^*(s', a))$$

$$V^*(s) = \max_a \sum_{s'} P(s'|s, a) (R(s, a, s') + \gamma V^*(s'))$$

the unique fixed point is the optimal value function

- 最优策略的两种算法

有这两个等式以后，就可以来求取最优策略。

Policy iteration algorithm:

loop until converges

policy evaluation: calculate V

policy improvement: choose the action greedily

$$\pi_{t+1}(s) = \arg \max_a Q^{\pi_t}(s, a)$$

converges: $V^{\pi_{t+1}}(s) = V^{\pi_t}(s)$

$$Q^{\pi_{t+1}}(s, a) = \sum_{s'} P(s'|s, a) (R(s, a, s') + \gamma \max_a Q^{\pi_t}(s', a))$$

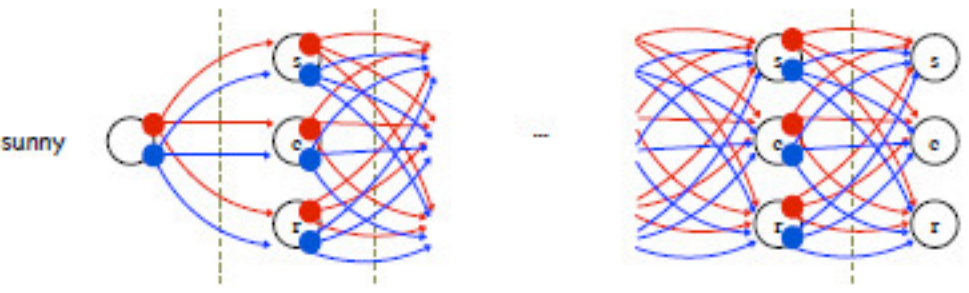
recall the optimal value function about Q

- 第一种方法：首先评估给定一个策略以后，这个策略有多好，然后找一个方向来提高这个策略。


这个算法的意思是，先计算你给出的这个策略的V值，然后用这种等式来更新这个策略，更新完以后又去计算这个V值是多少，又来更新这个策略，这样就可以保证这个策略最后收敛到最优策略。当然前提是你使用的是这个表格状态表示，有穷多个的状态和有穷多个的动作，这个方式对应的等式就是刚才的第一个等式。这个算法可能效率比较低，因为它需要不断的评估更新后的策略。这一方法称为策略迭代。

- 第二种方法：直接通过V值来更新V值，这一方法称为值迭代。

$$Q^{\pi_{t+1}}(s,a) = \sum_{s'} P(s'|s,a) (R(s,a,s') + \gamma \max_a Q^{\pi_t}(s',a))$$
$$V_{t+1}(s) = \max_a \sum_{s'} P(s'|s,a) (R(s,a,s') + \gamma V_t(s'))$$



Dynamic programming



R. E. Bellman
1920-1984

根据这两个等式就可以有两种计算最优策略的方法。在这里纪念一下提出者Bellman，实际上动态规划就是他的发明。

- **最优策略的复杂度是多少？**

另外，我们看到这样一个求解最优策略的过程，它的复杂度是多少呢？它的复杂度是状态数量乘以动作数量 $O(|S|*|A|)$ ，这已经是在一个很简单的MDP上（确定性 MDP），这个复杂度从状态数量和动作数量上看，好像是一个线性的复杂度，复杂度并不高。前面我们说了强化学习求解最优策略是NP难的问题，那么这个差别在什么地方呢？差别就在于，通常在度量一个问题的复杂度时，并不是根据它有多少状态来度量的，而是用状态空间的维度来度量。因此Bellman发明了一个词叫“维度灾难”。如果我们用维度来度量的话，这个复杂度就是一个非常高的复杂度，比如说对于围棋来说，它的维度是 19×19 ，但是它的有效状态数量超过了10的170次方。

这里简单的介绍了一下在MDP、马尔科夫决策上，怎么去求得一个策略。但是MDP并不是强化学习，因为它的四元组都已给出，特别是奖赏和转移。你任给它一个状态和动作，都可以算出奖赏值；转移也是，输入一个状态、动作以后，它会告诉你转移出来的状态是什么。

本文为俞扬博士《强化学习前沿》的上篇，下篇敬请关注雷锋网的后续内容。

雷锋网原创文章，未经授权禁止转载。详情见[转载须知](#)。




CCF-GAIR
2017全球人工智能与机器人峰会

6折

限时抢购

★ 6人收藏

分享：



相关文章

马尔可夫决策

马尔可夫

强化学习

南京大学



南京大学俞扬博士：强化学习前沿（下）



南京大学俞扬博士：强化学习前沿（下）



One-Page AlphaGo --十分钟看懂 AlphaGo 的核心



美国罗德岛大学杨庆教授：如何把机器学习技术应用于








“大号的 iPhone 7 Plus ?”



文章点评：

我有话要说.....

☐ 同步到新浪微博

提交



热门关键字

热门标签 微信小程序平台 微信小程序在哪 CES 2017 CES 2016年最值得购买的智能硬件 2016 互联网 小程序 微信朋友圈 抢票软件 智能手机 智能家居 智能手环 智能机器人 智能电视 360智能硬件 智能摄像机 智能硬件产品 智能硬件发展 智能硬件创业 黑客 白帽子 大数据 云计算 新能源汽车 无人驾驶 无人机 大疆 小米无人机 特斯拉 VR游戏 VR电影 VR视频 VR眼镜 VR购物 AR 直播 扫地机器人 医疗机器人 工业机器人 类人机器人 聊天机器人 微信机器人 微信小程序 移动支付 支付宝 P2P 区块链 比特币 风控 高盛 人脸识别 指纹识别 黑科技 谷歌地图 谷歌 IBM 微软 乐视 百度 三星s8 腾讯 三星Note8 小米MIX 小米Note 华为 小米 阿里巴巴 苹果 MacBook Pro iPhone Facebook GAIR IROS 双创周 云栖大会 智能硬件公司 智能硬件 QQ红包 支付宝红包 敬业福 支付宝敬业福 支付宝集五福 Waymo 虚拟现实 深度学习 人工智能 中国银联 蚂蚁金服 WRC CNCC 残差网络 斯坦尼康 家庭影院 阿尔法机器人 神经网络 samsung pay 5ghz olly 联通小号 eatsa 刘强东 amazon echo 树莓派 项目 思科中国 google 开源 tensorflow 更多

联系我们 关于我们 加入我们 意见反馈 投稿

