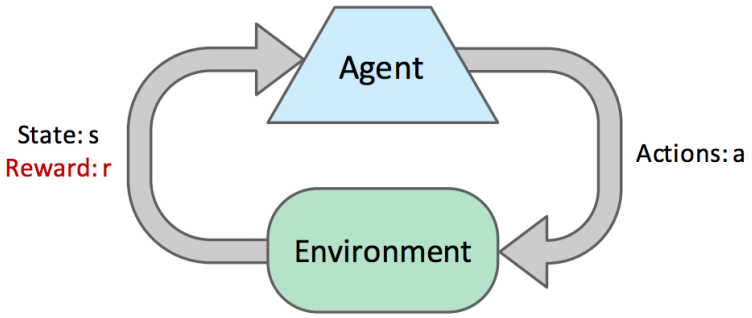
计算机科学188：人工智能导论
2024年春季 Note 24

作者（其他所有注释）：尼基尔·夏尔马
作者（贝叶斯网络注释）：乔希·胡格和杰基·梁，由王蕾吉娜编辑
作者（逻辑注释）：亨利·朱，由考佩林编辑
学分（机器学习和逻辑注释）：部分章节改编自教材《人工智能：一种现代方法》。

最后更新时间：2023年8月26日

强化学习

在上一篇笔记中，我们讨论了马尔可夫决策过程，我们使用诸如值迭代和策略迭代等技术来求解，以计算状态的最优值并提取最优策略。求解马尔可夫决策过程是离线规划的一个例子，在离线规划中，智能体对转移函数和奖励函数都有完整的了解，这些是它们在由MDP编码的世界中预先计算最优动作所需的所有信息，而无需实际采取任何动作。在本笔记中，我们将讨论在线规划，在此期间，智能体对世界中的奖励或转移没有先验知识（仍表示为MDP）。在在线规划中，智能体必须尝试探索，在此过程中它执行动作，并以到达的后继状态和获得的相应奖励的形式接收反馈。智能体通过一个称为强化学习的过程使用此反馈来估计最优策略，然后再使用此估计策略进行利用或奖励最大化。



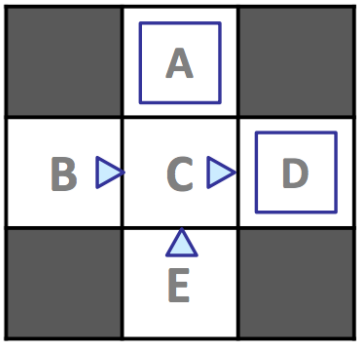
让我们从一些基本术语开始。在在线规划的每个时间步，智能体从状态 s 开始，然后采取动作 a 并最终到达后继状态 s'，获得一些奖励 r。每个 (s, a, s', r) 元组被称为一个样本。通常，智能体会持续相继采取动作并收集样本，直到到达终止状态。这样一组样本被称为一个情节。在探索过程中，智能体通常会经历许多情节，以便收集学习所需的足够数据。

强化学习有两种类型，基于模型的学习和无模型学习。基于模型的学习尝试利用在探索过程中获得的样本估计转移函数和奖励函数，然后使用这些估计值通过值迭代或策略迭代来正常求解马尔可夫决策过程。另一方面，无模型学习尝试直接估计状态的值或Q值，而无需.....

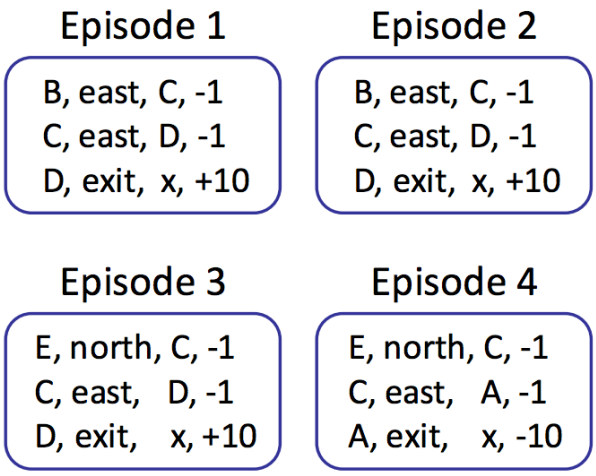
使用任何内存来构建马尔可夫决策过程（MDP）中奖励和转移的模型。

基于模型的学习

在基于模型的学习中，智能体通过统计在进入每个Q状态(s,a)后到达每个状态 s' 的次数，生成转移函数 $\widehat{T}(s,a,s')$ 的近似值。然后，智能体可以根据请求，通过对收集到的计数进行归一化来生成近似转移函数 $\widehat{T}$ —— 将每个观察到的元组 (s,a,s') 的计数除以智能体处于Q状态(s,a)的所有实例的计数总和。计数的归一化对它们进行缩放，使其总和为1，从而可以将它们解释为概率。考虑以下具有状态 $S=\{A,B,C,D,E,x\}$ 的示例MDP，其中 x 表示终端状态，折扣因子为 $\gamma=1$：



假设我们允许智能体按照上述描述的策略 π_{explore} 对马尔可夫决策过程进行四轮探索（一个定向三角形表示朝三角形所指方向移动，蓝色方块表示选择退出作为行动），并得到以下结果：



我们现在共有12个样本，每轮有3个，计数如下：

s	a	s′	count
A	$exit$	x	1
B	$east$	C	2
C	$east$	A	1
C	$east$	D	3
D	$exit$	x	3
E	$north$	C	2

回顾 $T\left(s,a,s'\right)=P\left(s' \mid a,s\right)$，我们可以通过将每个元组 $\left(s,a,s'\right)$ 的计数除以处于Q状态 (s,a) 的总次数来估计转移函数，并直接从探索期间获得的奖励中得出奖励函数：

- **Transition Function**: $\hat{T}(s,a,s')$
  - $\hat{T}(A,exit,x)=\frac{\#(A,exit,x)}{\#(A,exit)}=\frac{1}{1}=1$
  - $\hat{T}(B,east,C)=\frac{\#(B,east,C)}{\#(B,east)}=\frac{2}{2}=1$
  - $\hat{T}(C,east,A)=\frac{\#(C,east,A)}{\#(C,east)}=\frac{1}{4}=0.25$
  - $\hat{T}(C,east,D)=\frac{\#(C,east,D)}{\#(C,east)}=\frac{3}{4}=0.75$
  - $\hat{T}(D,exit,x)=\frac{\#(D,exit,x)}{\#(D,exit)}=\frac{3}{3}=1$
  - $\hat{T}(E,north,C)=\frac{\#(E,north,C)}{\#(E,north)}=\frac{2}{2}=1$

•奖励函数：$\widehat{R}(s,a,s')$

- $\widehat{R}(A,\text{ exit },x)=-10$
- $\widehat{R}(B,\text{ east},C)=-1$
- $\widehat{R}(C,\text{ east},A)=-1$
  - $\hat{R}(C,east,D)=-1$
- $\widehat{R}(D,\text{ exit},x)=+10$
  - $\hat{R}(E,north,C)=-1$

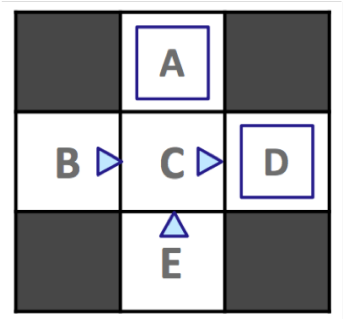
根据大数定律，随着我们让智能体经历更多的情节来收集越来越多的样本，我们对 $\widehat{T}$ 和 $\widehat{R}$ 的模型将会得到改进，其中 $\widehat{T}$ 会趋向于收敛到 T，并且当我们发现新的 $\left(s,a,s'\right)$ 元组时，$\widehat{R}$ 会获取到此前未被发现的奖励的知识。每当我们认为合适时，我们可以通过使用当前的 $\widehat{T}$ 和 $\widehat{R}$ 模型运行值迭代或策略迭代来结束智能体的训练，以生成一个策略 π_{exploit}，并使用 π_{exploit} 进行利用，让我们的智能体遍历马尔可夫决策过程，采取行动以寻求奖励最大化而非寻求学习。我们很快会讨论如何有效地在探索和利用之间分配时间的方法。基于模型的学习非常简单直观但却非常有效，仅仅通过计数和归一化就能生成 $\widehat{T}$ 和 $\widehat{R}$。然而，对每个看到的 $\left(s,a,s'\right)$ 元组都维护计数可能会很昂贵，所以在下一节关于无模型学习的内容中，我们将开发完全绕过维护计数的方法，并避免基于模型的学习所需的内存开销。

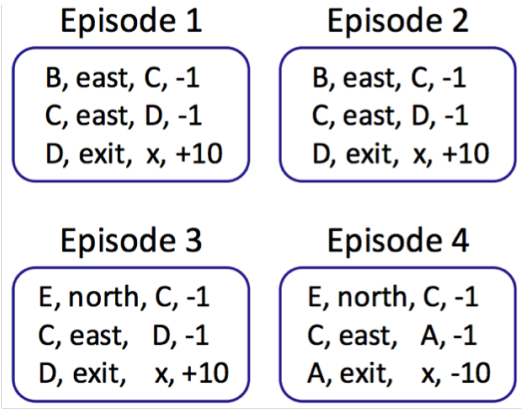
无模型学习

迈向无模型学习！有几种无模型学习算法，我们将介绍其中三种：直接评估、时间差分学习和Q学习。直接评估和时间差分学习属于一类称为被动强化学习的算法。在被动强化学习中，给智能体一个要遵循的策略，并在其经历各个情节时学习该策略下状态的值，这正是当 T 和 R 已知时MDP的策略评估所做的事情。Q学习属于另一类无模型学习算法，称为主动强化学习，在此期间，学习智能体可以使用接收到的反馈在学习时迭代更新其策略，直到经过充分探索最终确定最优策略。

直接评估

我们要介绍的第一种被动强化学习技术称为直接评估，这是一种像其名字听起来那样枯燥且简单的方法。直接评估所做的就是固定某个策略 π 并让智能体在遵循 π 的同时经历多个回合。当智能体通过这些回合收集样本时，它会记录从每个状态获得的总效用以及访问每个状态的次数。在任何时候，我们都可以通过将从 s 获得的总效用除以访问 s 的次数来计算任何状态 s 的估计值。让我们对之前的示例运行直接评估，回想一下 $\gamma = 1$ 。

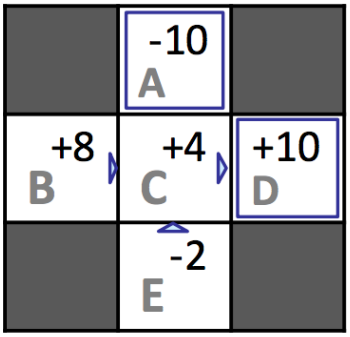




回顾第一集，我们可以看到，从状态 D 到终止状态，我们总共获得了10的奖励，从状态 C 我们总共获得了 $(-1) + 10 = 9$ 的奖励，从状态 B 我们总共获得了 $(-1) + (-1) + 10 = 8$ 的奖励。完成这个过程会得出每个状态在各集中的总奖励以及由此得到的估计值，如下所示：

s	Total Reward	Times Visited	$V^{\pi}(s)$
A	-10	1	-10
B	16	2	8
C	16	4	4
D	30	3	10
E	-4	2	-2

虽然直接评估最终会学习每个状态的状态值，但它通常收敛得不必要地慢，因为它浪费了关于状态之间转换的信息。



在我们的示例中，我们计算了 $V^{\pi}(E) = -2$ 和 $V^{\pi}(B) = 8$，不过根据我们收到的反馈，这两个状态都只有 C 作为后继状态，并且在转移到 C 时会产生相同的 -1 奖励。根据贝尔曼方程，这意味着在 π 下 B 和 E 应该具有相同的值。然而，在我们的智能体处于状态 C 的4次中，它转移到了 D 并三次获得了10的奖励，转移到 A 并一次获得了 -10 的奖励。它单次获得 -10 奖励时恰好从状态 E 开始而不是 B，但这严重扭曲了 E 的估计值。经过足够多的回合， B 和 E 的值将收敛到它们的真实值，但这样的情况会导致这个过程比我们希望的要花费更长的时间。通过选择使用我们的第二种被动强化学习算法，即时间差分学习，可以缓解这个问题。

时序差分学习

时序差分学习（TD学习）采用从每次经验中学习的理念，而不是像直接评估那样简单地记录总奖励和状态访问次数并在最后进行学习。在策略评估中，我们使用由固定策略和贝尔曼方程生成的方程组来确定该策略下状态的值（或者像值迭代那样使用迭代更新）。

$$V^{\pi}(s) = \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V^{\pi}(s')]$$

这些方程中的每一个都将一个状态的值等同于该状态后续状态的折扣值的加权平均值，再加上转移到这些后续状态时获得的奖励。时序差分学习试图回答如何在不知道权重的情况下计算这个加权平均值的问题，它巧妙地通过指数移动平均值来做到这一点。我们首先初始化 $\forall s, V^{\pi}(s) = 0$。在每个时间步，智能体从状态 s 采取行动 $\pi(s)$，转移到状态 s'，并获得奖励 $R(s, \pi(s), s')$。我们可以通过将接收到的奖励与在 π 下 s' 的折扣当前值相加来获得一个样本值：

$$\text{sample} = R(s, \pi(s), s') + \gamma V^{\pi}(s')$$

此样本是对 $V^{\pi}(s)$ 的新估计值。下一步是使用指数移动平均线将此抽样估计值纳入我们现有的 $V^{\pi}(s)$ 模型，该模型遵循以下更新规则：

$$V^{\pi}(s) \leftarrow (1 - \alpha) V^{\pi}(s) + \alpha \cdot \text{sample}$$

在上式中， α 是一个受 $0 \leq \alpha \leq 1$ 约束的参数，称为学习率，它指定了我们要赋予现有 $V^{\pi}(s), 1 - \alpha$ 模型的权重，以及我们要赋予新抽样估计值 α 的权重。通常从学习率 $\alpha = 1$ 开始，相应地将 $V^{\pi}(s)$ 赋给第一个样本的值，然后逐渐将其缩小至 0，此时所有后续样本将被清零，不再影响我们的 $V^{\pi}(s)$ 模型。

让我们停下来分析一下更新规则。通过分别将 $V_k^{\pi}(s)$ 和 sample_k 定义为经过 k^{th} 更新和 k^{th} 采样后状态 s 的估计值，来注释我们模型在不同时间点的状态，我们可以重新表述我们的更新规则：

$$V_k^{\pi}(s) \leftarrow (1 - \alpha) V_{k-1}^{\pi}(s) + \alpha \cdot \text{sample}_k$$

对 $V_k^\pi(s)$ 的这种递归定义展开后恰好非常有趣：

$$\begin{aligned}
V_k^\pi(s) &\leftarrow (1-\alpha)V_{k-1}^\pi(s) + \alpha \cdot \text{sample}_k \\
V_k^\pi(s) &\leftarrow (1-\alpha)[(1-\alpha)V_{k-2}^\pi(s) + \alpha \cdot \text{sample}_{k-1}] + \alpha \cdot \text{sample}_k \\
V_k^\pi(s) &\leftarrow (1-\alpha)^2 V_{k-2}^\pi(s) + (1-\alpha)\cdot\alpha \cdot \text{sample}_{k-1} + \alpha \cdot \text{sample}_k \\
&\vdots \\
V_k^\pi(s) &\leftarrow (1-\alpha)^k V_0^\pi(s) + \alpha \cdot [(1-\alpha)^{k-1} \cdot \text{sample}_1 + \ldots + (1-\alpha)\cdot \text{sample}e_{k-1} + \text{sample}_k] \\
V_k^\pi(s) &\leftarrow \alpha \cdot [(1-\alpha)^{k-1} \cdot \text{sample}_1 + \ldots + (1-\alpha)\cdot \text{sample}_{k-1} + \text{sample}_k]
\end{aligned}$$

因为 $0 \leq (1-\alpha) \leq 1$，当我们将数量 $(1-\alpha)$ 提升到越来越大的幂次时，它会越来越接近0。根据我们推导的更新规则展开，这意味着较旧的样本被赋予的权重呈指数级减少，这正是我们想要的，因为这些较旧的样本是使用我们针对 $V^\pi(s)$ 的较旧（因而更差）版本的模型计算出来的！这就是时序差分学习的美妙之处——通过一个简单直接的更新规则，我们能够：

- 在每个时间步进行学习，因此在获取状态转移信息时就使用它们，因为我们在样本中使用的是 $V^\pi(s')$ 的迭代更新版本，而不是等到最后才进行任何计算。
- 对较旧的、可能不太准确的样本赋予指数级更少的权重。
- 与直接评估相比，通过更少的回合更快地收敛到学习真实状态值。

Q学习

直接评估和时序差分学习最终都会在它们所遵循的策略下学习到所有状态的真实值。然而，它们都有一个主要的固有问题——我们想要为我们的智能体找到一个最优策略，这需要知道状态的Q值。为了从我们拥有的值中计算Q值，我们需要如贝尔曼方程所规定的转移函数和奖励函数。

$$Q^*(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

因此，时序差分学习或直接评估通常与一些基于模型的学习一起使用，以获取对 T 和 R 的估计，以便有效地更新学习智能体所遵循的策略。通过一种被称为Q学习的革命性新思想，这变得可以避免，Q学习直接提出学习状态的Q值，从而无需知道任何值、转移函数或奖励函数。因此，Q学习是完全无模型的。Q学习使用以下更新规则来执行所谓的Q值迭代：

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_{a'} Q_k(s',a')]$$

请注意，此更新只是对值迭代更新规则的轻微修改。实际上，唯一真正的区别在于，由于我们在处于某个状态时在进行转移之前选择一个动作，但在处于Q状态时在选择新动作之前进行转移，所以max运算符在动作上的位置发生了变化。

有了这个新的更新规则，Q学习的推导方式与TD学习基本相同，即通过获取 $\mathbf{Q}$ 值样本：

$$\text{sample} = R(s,a,s') + \gamma \max_{a'} Q(s',a')$$

并将它们纳入指数移动平均线。

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha \cdot \text{sample}$$

只要我们花足够的时间进行探索，并以适当的速度降低学习率 α，Q学习就能为每个Q状态学习到最优的Q值。这就是Q学习如此具有革命性的原因——虽然时序差分学习和直接评估是通过在确定策略最优性之前遵循策略来学习策略下状态的值，但Q学习即使采取次优或随机行动也能直接学习到最优策略。这被称为离策略学习（与直接评估和时序差分学习相反，它们是在线策略学习的例子）。

近似Q学习

Q学习是一种令人难以置信的学习技术，一直处于强化学习领域发展的核心位置。然而，它仍有一些改进的空间。目前，Q学习只是以表格形式存储所有状态的Q值，考虑到强化学习的大多数应用都有数千甚至数百万个状态，这并不是特别有效。这意味着我们在训练期间无法访问所有状态，即使有足够的内存也无法存储所有的Q值。

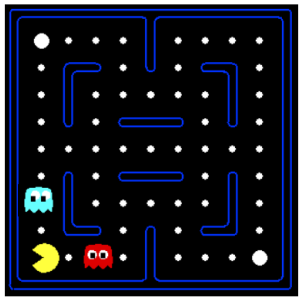


Figure 1

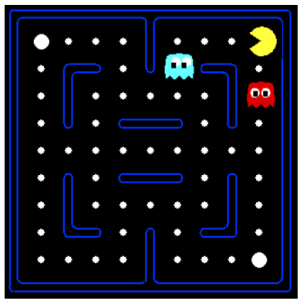


Figure 2

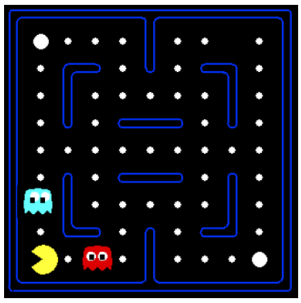


Figure 3

如上所述，如果吃豆人在运行普通Q学习后得知图1是不利的，它仍然不会知道图2甚至图3也是不利的。近似Q学习试图通过学习一些一般情况并推断到许多类似情况来解决这个问题。泛化学习经验的关键是基于特征的状态表示，它将每个状态表示为一个称为特征向量的向量。例如，吃豆人的特征向量可能编码

- 到最近幽灵的距离。
- 到最近食物颗粒的距离。
- 幽灵的数量。
- 吃豆人被困住了吗？0或1

有了特征向量，我们可以将状态值和Q状态值视为线性值函数：

$$\begin{aligned} V(s) &= w_1 \cdot f_1(s) + w_2 \cdot f_2(s) + \ldots + w_n \cdot f_n(s) = \vec{w} \cdot \vec{f}(s) \\ Q(s,a) &= w_1 \cdot f_1(s,a) + w_2 \cdot f_2(s,a) + \ldots + w_n \cdot f_n(s,a) = \vec{w} \cdot \vec{f}(s,a) \end{aligned}$$

其中 $\overrightarrow{f}(s) = [f_1(s) \quad f_2(s) \quad \ldots \quad f_n(s)]^T$ 和 $\overrightarrow{f}(s,a) = [f_1(s,a) \quad f_2(s,a) \quad \ldots \quad f_n(s,a)]^T$ 表示

分别对应状态 s 和Q状态(s,a)的特征向量，并且 $\overrightarrow{w} = \begin{bmatrix} w_1 & w_2 & \dots & w_n \end{bmatrix}$ 表示一个权重向量。将差异定义为

$$\text{difference} = [R(s,a,s') + \gamma \max_{a'} Q(s',a')] - Q(s,a)$$

近似Q学习与Q学习的工作方式几乎相同，使用以下更新规则：

$$w_i \leftarrow w_i + \alpha \cdot \text{difference} \cdot f_i(s,a)$$

对于近似Q学习，我们无需为每个状态存储Q值，而只需存储单个权重向量，并可根据需要按需计算Q值。因此，这不仅为我们提供了一个更通用的Q学习版本，而且显著提高了内存效率。
关于Q学习的最后一点说明，我们可以用差异重新表示精确Q学习的更新规则如下：

$$Q(s,a) \leftarrow Q(s,a) + \alpha \cdot \text{difference}$$

第二种表示法为我们提供了对更新的一种略有不同但同样有价值的解释：它计算采样估计值与当前 $Q(s,a)$ 模型之间的差异，并将模型朝着估计值的方向移动，移动的幅度与差异的大小成正比。