机器学习笔记

聂芃

2022 年 10 月 24 日

目录

第一章　绪论

本节介绍一些基础的概念，但我懒得复制粘贴了，并且也不是非常重要。仅作简单摘录并提供参考资料，大家可以自行阅读学习。

- 什么是机器学习，表示学习和深度学习：见“神经网络与深度学习”的 1.2-1.4，以及 2.1-2.2；
- 机器学习的三种分类：即监督学习，无监督学习和强化学习。详见“神经网络与深度学习”的 2.5，其中表 2.1 对三种机器学习类型进行了比较。

我们要做的蜘蛛纸牌 AI 的项目应该属于三种类型中的强化学习。与强化学习的相关内容如下：

- 什么是强化学习：见 EasyRL 的 1.1，RLbook 的 1.1-1.2。
- 强化学习系统的要素：即智能体和环境。其中前者包含策略，奖励信号，价值函数。对环境的模型是可选的。详见 EasyRL 的 1.2，1.4，“神经网络与深度学习”的 14.1.2 以及 RLbook 的 1.3.
- 典型的例子：见 RLbook 的 1.2，1.5，“神经网络与深度学习”的 14.1.1

第二章　Markov 决策过程

本章讲解的是强化学习中重要的数学模型：Markov 链，或称 Markov 过程。尽管大家可能对此并不熟悉，但是其中运用到的数学工具并不复杂。

尽管本章看起来十分“数学”，但事实上只涉及到了随机过程以及概率论中的一小部分内容，度过对各种符号与定义的陌生后，会发现本章中各种定义的动机都是自然的，几乎所有的定理也都是符合直观的（也许是几何直观，也许是对概率的一种朴素认识）。

与此同时，本章的主要目的也只是让大家了解诸如策略、预测、控制等概念，为之后介绍强化学习中的方法打下基础。即便第一次看的时候不能完全理解本章在讲什么也是完全没有问题的，在后面的学习中发现了不清楚的概念再回过来看本章就可以了。

需要注意的一点是，本章中为了控制篇幅 + 我懒得抄书了，所以没有添加任何例子，一定程度上会让大家感到引入各种定义的动机不明。如果遇到了这样的情况，请查阅参考资料。当然，如果大家的屏幕够大的话，一边看我的笔记，一边看参考资料是最好的，因为具体的例子必然会帮助大家理解这些抽象的概念。

本章的学习目标是：

- 了解随机过程中的数学模型：Markov 链。
- 了解 Markov 奖励过程；理解奖励函数、折扣因子 γ 的作用，回报、状态价值函数的定义及由来（所谓由来，就是说我们如此定义的动机是什么，如此定义的合理性，以及如此定义的目的是表示什么、解决什么）；感性认识 Markov 奖励过程的 Bellman equation，以及两种求解状态价值函数的算法。
- 了解 Markov 决策过程；理解策略的定义，以及策略对 Markov 过程的影响；理解 Q 函数（动作价值函数）的定义及由来；感性认识 Markov 决策过程的 Bellman expectation equation；认识备份图，并且运用备份图（从一种几何直观的角度）理解前文的数学公式。
- 理解策略评估（价值预测）、预测、控制的定义和由来；理解预测和控制的区别；理解最佳价值函数、最佳策略的定义及由来。
- 理解策略迭代、价值迭代的定义、由来及操作方式；理解策略迭代与价值迭代的区别。
- 在学完本章所有知识之后，能够认识到：Markov 链是什么，为什么我们又定义了奖励过程和决策过程（也就是说，引入它们的必要性），如何对一个策略进行评估，以及如何改进策略。

2.1　（离散时间）Markov 过程

本节可参考 https://zhuanlan.zhihu.com/p/343462109 的 1.1，1.2

定义 2.1.1. 设 $s_1, s_2, \dots$ 都是状态，记 $S = s_1, s_2, \dots$，设 $\{X_n | X_n \in S\}_{n=0}$ 是状态空间 S 上的离散随机变量序列，那么称一个过程是 *Markov* 过程，或者说满足 *Markov* 性质，如果状态转移与过去是无关的，

只取决于现在，即

$$P\left(X_{n+1}=s_j|X_n=s_i,X_{n-1},X_{n-2},\dots\right)=P\left(X_{n+1}=s_j|X_n=s_i\right)$$

其中 $P(X_{n+1}|X_n)$ 称为状态转移概率。在其它参考资料中，选取了另一种记法 $P(s_{t+1}|s_t)$，但是因为有可能引起歧义（把状态和某个时刻的随机变量用同一个符号表示了，尽管事实上在选取合适的角标的情况下，并不会造成过大歧义），所以我先采用了比较繁琐但是精确的写法。之后可能会偶尔采用简单的写法（在尽量不引起歧义的情况下），因为这种写法在符号上更简洁。

定义 2.1.2. 如果转移概率进一步满足与时间无关，即

$$P\left(X_{n+1}=s_j|X_n=s_i\right)=P\left(X_1=s_j|X_0=s_i\right)=p_{ij}$$

那么称这一过程是时间齐次的离散时间 *Markov* 链

对于这样的过程，我们可以构造转移概率矩阵：

$$P=\begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ p_{n1} & \cdots & \cdots & p_{nn} \end{bmatrix}$$

定义 2.1.3. 定义瞬态分布为给定时刻下系统处在状态空间里的各个状态的概率，记为

$$a^{(t)}=\left[a_1^{(t)},\dots,a_n^{(t)}\right]^T$$

记初始分布为

$$a^{(0)}=\left[a_1^{(0)},\dots,a_n^{(0)}\right]^T$$

定义 2.1.4. 定义 n 步转移概率

$$p_{ij}^{(n)}=P\left(X_n=j|X_0=i\right)$$

定义 n 步转移概率矩阵

$$P^{(n)}=\begin{bmatrix} p_{11}^{(n)} & \cdots & p_{1N}^{(n)} \\ \vdots & \vdots & \vdots \\ p_{N1}^{(n)} & \cdots & p_{NN}^{(n)} \end{bmatrix}$$

定理 2.1.5.

$$P^{(n)}=P^n$$

2.2 Markov 奖励过程

本节见 EasyRL 2.2

Markov 奖励过程是 Markov 链再加上了一个奖励函数 R。奖励函数 R 是一个期望，就是说当你到达某一个状态的时候，可以获得多大的奖励。

定义 2.2.1. 回报（*return*）是指把奖励进行折扣后所获得的收益。回报可以定义为奖励的逐步叠加，如下式所示。

$$G_t=\sum_{i=1}\gamma^{i-1}R_{t+i}$$

这里有一个折扣因子 $\gamma\in[0,1]$，越往后得到的奖励，折扣得越多。这说明我们其实更希望得到现有的奖励，未来的奖励就要把它打折扣。

定义 2.2.2. 当我们有了回报过后,就可以定义一个状态的价值了,就是状态价值函数(*state-value function*)。对于 *Markov* 奖励过程，状态价值函数被定义成是回报的期望，如下式所示（t 时刻，状态 s)。

$$V_t(s) = \mathbb{E}(G_t | s_t = s)$$

关于折扣因子的作用，见 EasyRL 2.2.2

定理 2.2.3 (全期望公式)**.**

$$\mathbb{E}(X) = \sum_i \mathbb{E}(X|A_i) P(A_i)$$

定理 2.2.4 (Bellman equation)**.**

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

其中 $R(s)$ 是即时奖励，$\gamma \sum_{s' \in S} P(s'|s) V(s')$ 是未来奖励的折扣总和

直观上它说得对，具体证明过程见 EasyRL 2.2.3.1 的式 2.8 与 2.2.3.2 的式 2.9

我们可以将 Bellman Equation 写成矩阵的形式

定理 2.2.5.

$$\begin{bmatrix} V(s_1) \\ V(s_2) \\ \vdots \\ V(s_n) \end{bmatrix} = \begin{bmatrix} R(s_1) \\ R(s_2) \\ \vdots \\ R(s_n) \end{bmatrix} + \gamma \begin{bmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & \vdots & \vdots \\ p_{n1} & \cdots & p_{nn} \end{bmatrix} \begin{bmatrix} V(s_1) \\ V(s_2) \\ \vdots \\ V(s_n) \end{bmatrix}$$

或者简单记为

$$V = R + \gamma P V$$

容易得到，如果 $(I - \gamma P)$ 可逆，那么有

$$V = (I - \gamma P)^{-1} R$$

矩阵的求逆的复杂度是 $o(n^3)$，当处理大矩阵时，计算 Markov 奖励过程价值可以利用迭代的方法，例如蒙特卡罗算法，动态规划和时序差分学习等。在 EasyRL 2.2.5 中提供了关于蒙特卡罗和动态规划的伪代码以及解释。简单地说，蒙特卡罗算法就是通过采样计算结果，动态规划是通过反复迭代 Bellman Equation 直至结果收敛。关于这三种算法的更加详细的说明，会放在之后的章节中。

2.3 Markov 决策过程

见 EasyRL 2.3

Markov 决策过程简单地说就是 Markov 奖励过程加上一个决策，并且状态转移和价值函数都增加了一个条件“决策（或者说，动作）”。更详细的说明在 2.3 的第一段中（我懒得抄了）

2.3.1 策略

Markov 决策过程与 Markov 过程一个极大的区别在于：Markov 过程中，转移的概率是已经决定了的，到达了一个状态 s 的同时，下一步转移到各个状态的概率就确定了；然而在 Markov 决策过程中多了“动作 a”，而这一动作的存在，就改变了转移的概率，或者说，每种动作对应了它自己的一种转移概率，动作

的不同就会导致转移概率发生变化。这里所谓的“动作”几乎就是其字面意思（事实上，大部分未加数学定义的词语都几乎是其字面意思，我们日常生活中提到它们时使用的含义就可以当作在文章中提及它们时使用的含义），不加额外定义。

动作的引入对状态转移的改变体现如下：假设在 t 时刻位于状态 s_i，并且采取了动作 a，那么

$$p_{ij} = P\left(X_{t+1} = s_j | X_t = s_i, a_t = a\right) = P\left(s' | s, a\right)$$

自然的，奖励函数和状态价值函数也发生了相应的改变，我们记在在状态 s 采取了动作 a 的奖励函数为 $R\left(s, a\right)$。

定义 2.3.1. *所谓“策略”，就是定义了在某一个状态应该如何采取动作。这里的动作并不要求是唯一的，可以是遵循某种概率从多个动作中选取一个。后者和我们日常生活中对“策略”一词的使用是相似的。*

我们将策略记为 π，并且将在 t 时刻、状态 s 采取动作 a 的概率记作

$$\pi\left(a | s\right) = P\left(a_t = a | s_t = s\right)$$

当我们已知策略函数的情况下，就可以将状态转移函数 $P\left(s' | s, a\right)$ 中的 a 去掉。记 A 是动作的集合，则

$$P^{\pi}\left(s' | s\right) = \sum_{a \in A} \pi\left(a | s\right) P\left(s' | s, a\right)$$

同样的，对于奖励函数进行改写：

$$R^{\pi}\left(s\right) = \sum_{a \in A} \pi\left(a | s\right) R\left(s, a\right)$$

2.3.2 价值函数

在 Markov 决策过程中，状态价值函数的定义与先前是类似的，只不过要在记号中引入策略：

$$v^{\pi}\left(s\right) = \mathbb{E}_{\pi}\left(G_t | s_t = s\right)$$

除此之外，再引入动作价值函数，或者说，Q 函数。

定义 2.3.2. *Q 函数定义了在某个状态采取某个动作得到的回报的期望。*

$$q^{\pi}\left(s, a\right) = \mathbb{E}_{\pi}\left(G_t | s_t = s, a_t = a\right)$$

自然的，有

$$v^{\pi}\left(s\right) = \sum_{a \in A} \pi\left(a | s\right) q^{\pi}\left(s, a\right)$$

2.3.3 Bellman 期望方程

仿照 Markov 奖励过程中的 Bellman equation，我们可以得到决策过程中关于状态价值函数和 Q 函数的 Bellman expectation equation。

定理 2.3.3.

$$v^{\pi}\left(s\right) = \sum_{a \in A} \pi\left(a | s\right) \left(R\left(s, a\right) + \gamma \sum_{s' \in S} P\left(s' | s, a\right) v^{\pi}\left(s'\right) \right)$$

定理 2.3.4.

$$q^{\pi}\left(s, a\right) = R\left(s, a\right) + \gamma \sum_{s' \in S} P\left(s' | s, a\right) \sum_{a' \in A} \pi\left(a' | s'\right) q^{\pi}\left(s', a'\right)$$

两个定理的证明几乎就是根据定义以及

$$v^{\pi}(s)=\sum_{a\in A}\pi(a|s)\,q^{\pi}(s,a)$$

$$q^{\pi}(s,a)=R(s,a)+\gamma\sum_{s'\in S}P(s'|s,a)\,v^{\pi}(s')$$

代入。具体过程可见 EasyRL 2.3.4

2.3.4 备份图

备份图可以理解为利用空心圆圈代表状态，利用实心圆圈代表动作，绘制成树状的图形。它是帮助从直观上理解前述公式的有效工具。具体的内容见 EasyRL 2.3.5

2.3.5 预测与控制

预测，就是评估采用某个策略能够得到多大的奖励。计算状态价值函数的过程实际上就是在做策略评估。控制，是指寻找一个最佳的策略，得到最佳价值函数。关于两者的区别与联系，见 EasyRL 2.3.7。有关策略评估，见 EasyRL 2.3.6，2.3.8。

定义 2.3.5. 定义最佳价值函数：

$$v^{*}(s)=\max_{\pi}v^{\pi}(s)$$

定义最佳策略：

$$\pi^{*}(s)=\arg\max_{\pi}v^{\pi}(s)$$

最佳价值函数是说，我们去搜索一种策略来让每个状态的价值最大。在这种情况上面，我们得到的策略就可以说它是最佳策略。

当取得最佳的价值函数过后，我们可以通过对 Q 函数进行极大化来得到最佳策略，只需要贪婪地选取价值最大的动作。

2.3.6 策略迭代、价值迭代

策略迭代和价值迭代是解决控制问题的两种算法。详细的内容可见 EasyRL 2.3.11-2.3.13，有具体的例子。

策略迭代分为两步：策略评估和策略改进。先通过策略评估得到状态价值函数 v，再推算出 Q 函数，并且通过贪心法在 Q 函数上极大化得到新的策略，即策略改进。再对新的策略进行策略评估、策略改进，依此类推，反复迭代得到一个最佳的策略。在 EasyRL 2.3.11.1 中利用 Bellman 最优方程论证了这种方法的合理性（虽然其实和没说差不多，这本身就是一个朴素的结论）。

定理 2.3.6 (Bellman 最优方程)**.**

$$v^{*}(s)=\max_{a}q^{*}(s,a)=\max_{a}\left(R(s,a)+\gamma\sum_{s'\in S}P(s'|s,a)\,v^{*}(s)\right)$$

价值迭代就是利用 Bellman 最优方程进行迭代

$$v(s)\leftarrow\max_{a\in A}\left(R(s,a)+\gamma\sum_{s'\in S}P(s'|s,a)\,v(s')\right)$$

EasyRL 2.3.12.3 中提供了一个伪代码

两种方法的区别在于：策略迭代利用 Bellman 期望方程迭代，并且分两步；价值迭代利用 Bellman 最优方程，只有一步。

第三章　表格型方法

本章讲解的是使用基于价值的表格型的方法去求解强化学习。本章的编写几乎是按照 EasyRL 的结构进行的，你总可以在 EasyRL 的对应部分找到我所写的内容。因为我比较懒，所以在整理的过程中几乎只保留了最核心的内容，那么必然不够具体，并且不容易完全吸收。我推荐的本章阅读方法为：按小节进行，先简单阅读我的笔记，对核心内容有所了解，再去阅读 EasyRL，完成整章的阅读后，可以再过一遍本笔记，结合 EasyRL 的关键词部分梳理本章结构。

与此同时，RLbook 用了更大的篇幅来讲解表格型方法：第二章作为引入，第三章介绍 Markov 决策过程，第四章介绍动态规划，第五、六、七章分别介绍了蒙特卡罗、时序差分、n 步自举。其内容相较于 EasyRL 更详细、更广且更深。但是一方面其内容有点过于深（至少我是这么认为的），一方面其翻译与排版不尽人意，因此并没有采用 RLbook 作为本章的主要参考。如果想要更加深入了解表格型方法，可以大致阅读 RLbook 的第一部分（尽管第二到四章对应的是本笔记上一章的内容），以及其每章后的相关书目。

本章的学习目标是：

- 理解有模型与免模型的含义与区别。
- 理解两种免模型预测的方法。
- 理解两种免模型控制的方法。
- 了解同策略、异策略的含义与区别。

3.1　有模型、免模型

EasyRL 3.1 中有关于本节的详细例子（人与熊）。

模型的有无是针对环境来讲的：如果整个 Markov 决策过程是已知的，即状态转移函数以及奖励函数是已知的，我们就称为“有模型”，反之则称为“免模型”。在免模型的学习中，智能体通过与环境多次交互，采集轨迹（状态组成的序列）数据，从轨迹中获取信息来改进策略。

很多强化学习的经典算法是免模型的。这出于两方面原因：一方面问题中的模型可能是未知的，一方面问题中的模型可能太大了，无法进行迭代的运算。上一章中讲到的利用 Bellman 方程进行迭代的方法，即动态规划，就是有模型的。其基本思想是“自举”：基于后继状态值的估计来更新状态值的估计（具体内容见 EasyRL 2.2.5）。

3.2　免模型预测

常用的免模型预测算法有两个：蒙特卡罗与时序差分。

蒙特卡罗简单来讲就是从某个状态 s 走到终止状态（就是 Markov 过程的结束状态）得到一条轨迹，计算出回报，作为一次采样；多次采样后取平均值作为状态 s 的价值函数，根据大数定律我们相信当采样的次数足够多时，此平均值总会收敛到真实值。蒙特卡罗有如下几个特点：

- 蒙特卡罗不但是免模型的方法（区别于动态规划），它事实上也可以应用在非 Markov 过程的问题中（区别于时序差分），因为在采样和计算平均值的过程中，并没有用到任何关于 Markov 过程的性质。
- 蒙特卡罗的轨迹必须以终止状态结尾，而时序差分可以从不完整的序列中学习。

时序差分是介于蒙特卡罗与动态规划之间的算法，它同时运用了采样与自举的想法。最简单的时序差分算法是 TD(0)，一步时序差分：每走一步就自举，利用 $R_{t+1}+\gamma v\left(S_{t+1}\right)$ 更新 $v\left(S_{t+1}\right)$。同时，我们也可以选择走多步后再更新，得到 n 步时序差分；当 n 等于无穷时，就变成了蒙特卡罗。

EasyRL 3.3.3 中对比了动态规划、蒙特卡罗以及时序差分。

3.3 免模型控制

免模型控制与有模型控制的区别在于使用了广义策略迭代：使用蒙特卡罗或者时序差分而非动态规划来得到动作价值函数，再利用贪心算法改进策略。

有两种常用的算法：Sarsa 与 Q 学习。其中前者是同策略时序差分控制，后者是异策略时序差分控制。下面先解释这两种算法，再解释什么是同策略、异策略。

Sarsa 算法简单地说就是把预测过程中时序差分更新状态价值函数的过程变成了更新动作价值函数，同样分为一步 Sarsa 与 n 步 Sarsa。Q 学习中引入了两个策略：目标策略与行为策略。后者负责探索，得到 Q 表格（即 $q(s,a)$ 组成的表格），目标策略负责利用 Q 表格学习。

现在我们就可以来讨论同策略与异策略了。Sarsa 是典型的同策略的学习方法：它既使用策略 π 来学习，还使用策略 π 与环境交互；Q 学习是典型的异策略的学习方法，探索与学习交由两个不同的策略完成。相比较而言，同策略的学习方法为了兼顾探索与利用，在训练的过程中会显得较为胆小与保守，而异策略的学习方法由于目标策略采用贪心算法，会十分激进。

最后我们来介绍一种探索的方法：ε-贪心探索。如果我们严格遵循贪心算法进行探索，很可能无法到达所有的状态与动作，这时候便引入了 ε-贪心探索：有 $1-\varepsilon$ 的几率按照 Q 函数进行贪心的探索，有 ε 的几率随机选择一个动作（注意，这里的随机选择并没有将贪心的动作排除在外）。在实际的过程中，通常会设置 ε 随时间递减，以实现在最开始广泛探索，在一段时间后基本确定较好的 Q 函数后，减少探索。

EasyRL 中提供了一个实例：使用 Q 学习解决悬崖寻路问题。其中介绍了强化学习的一些基本接口，提供了 Q 学习的代码，可以阅读一下。RLbook 的例 6.5 与练习 6.9，6.10 提供了网格世界三种变体（只是题目，没有代码），有兴趣可以自己实践一下。