Reinforcement Learning: An Introduction notebook

黎雷蕾

2017 年 12 月 6 日

目录

Chapter 8

Planning and Learning with Tabular Methods

8.1 Models and Planning

- distribution models: 所有可能性;
  - 优点: 可以生成所有可能的过程, 以及各个过程的概率分布.
  - 缺点: 难以构建, 需要对环境足够的先验知识, 并且工程量浩大, 一旦模型出现错误无法更改.
- sample models: 所有可能性中的抽样（蒙特卡洛）;
  - 优点: 可以通过 agent 与环境交互的经验构建模型, 构建模型的方式更为简单, 并且可以更改模型中的错误.
  - 缺点: 只能生成一种可能的过程, 并且在概率估计上可能出现偏差.
- State-space planning: 通过状态空间搜索一个最优的政策或一个目标的最佳路径;
- plan-space planning: 动作导致状态到状态之间的转化，并且 V 值通过状态来进行计算;

Random-sample one-step tabular Q-planning

无限重复如下步骤：

1. 随机选择一个状态 S 和一个动作 A;
2. 将 S, A 发送给一个抽样模型，获得下一个状态 S' 和下一个 Reward(R);
3. $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',A) - Q(S,A)]$;

8.2 Dyna: Integrating Planning, Acting, and Learning

真实经验 (real experience) 的作用:

- 提升模型的性能 (model-learning)；
- 提升价值函数或策略 (direct reinforcement learning)；

它们的关系如下：

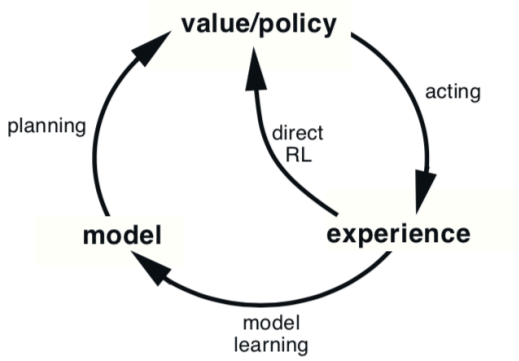

Figure 8.1: Relationships among learning, planning, and acting.

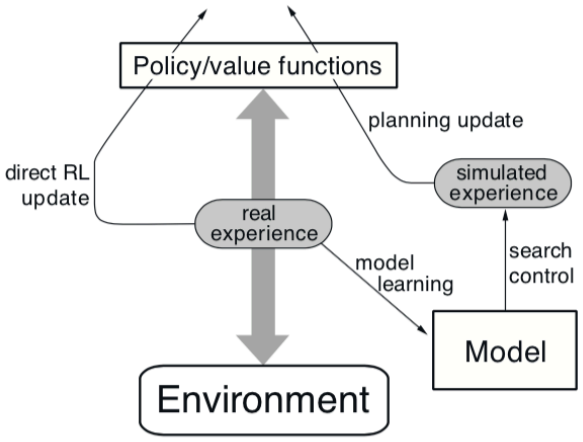

Figure 8.2: The general Dyna Architecture. Real experience, passing back and forth between the environment and the policy, affects policy and value functions in much the same way as does simulated experience generated by the model of the environment.

Tabular Dyna-Q

1. 初始化 $Q(s,a)$ 和 $Model(s,a)$;
2. 一直循环:
   (a) $S \leftarrow$ 当前非中断状态;
   (b) $A \leftarrow \epsilon - greedy(S,Q)$;
   (c) 执行动作 A，观察由此产生的 R, S';
   (d) $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',A) - Q(S,A)]$;
   (e) $Model(S,A) \leftarrow R, S'$;
   (f) 以下步骤重复 n 次:
      i. $S \leftarrow$ 先前随机观测到的状态;
      ii. $A \leftarrow$ 在 S 状态下采用的 A;
      iii. $R, S' \leftarrow Model(S,A)$;
      iv. $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',A) - Q(S,A)]$;

8.3 When the Model Is Wrong

如果模型存在误差，那么计算出来的可能就不是最优策略了。

一般来说，Dyna-Q+ 相比于 Dyna-Q 来说具有一定的启发搜索能力，对于时间 τ 内未被尝试的搜索状态转移，其回报是 $r + \kappa\sqrt{\tau}$.

8.4 Prioritized Sweeping

如果我们采用均匀采样的话，会有许多价值函数的取值是 0，那么过多的 0 会导致 backup 操作无效而被浪费。我们从目标状态向前 (backward) 搜索 (回溯) 可能会是一个有效的策略. 我们希望该方法能够用于生成奖励函数，对于目标状态，仅仅是其中的一个特例而已。

prioritized sweeping 的思想是：在随机的环境中，变量状态转移的概率也对变化量的大小产生影响，可以根据 backup 的优先级进行排序，首先进行优先级别高的进行 backup：

Prioritized sweeping for a deterministic environment

1. 初始化 $Q(s,a)$,$Model(s,a)$，一个空队列 $PQueue$；
2. 一直重复:
   (a) $S \leftarrow$ 当前的非中断状态;
   (b) $A \leftarrow policy(S,Q)$;
   (c) 执行动作 A，观察由此产生的 R, S';
   (d) $Model(S,A) \leftarrow R, S'$;
   (e) $P \leftarrow |R + \gamma \max_a Q(S',a) - Q(S,A)|$;
   (f) 如果 $P > \theta$，以优先级 P 将 S, A 插入 $PQueue$;
   (g) 如果 $PQueue$ 不为空，那么重复下面步骤 n 次：
      i. $S, A \leftarrow first(PQueue)$;
      ii. $R, S' \leftarrow Model(S,A)$;
      iii. $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',A) - Q(S,A)]$;

iv. 对于所有的 $\overline{S}, \overline{A}$ 预测值为 S 的，重复如下步骤:

A. $\overline{R} \leftarrow \overline{S}, \overline{A}, S$ 的预测回报;

B. $P \leftarrow |\overline{R} + \gamma \max_a Q(S, a) - Q(\overline{S}, \overline{A})|$.

C. 如果 $P > \theta$，以优先级 P 将 $\overline{S}, \overline{A}$ 插入 $PQueue$;

8.5 Full vs. Sample Backups

Full backup 没有抽样误差，对全局的估计将会更好，但是开销很大；在实际中用处不大，一般都是采用 sampling backups。

full backup 的动作价值函数可以写为:

$$Q(s, a) \leftarrow \sum_{s', r} \hat{p}(s', r|s, a) \left[r + \gamma \max_{a'} Q(s', a') \right] \tag{8.1}$$

其中 $\hat{p}(s', r|s, a)$ 是一个动态的估计模型，示意图如下:

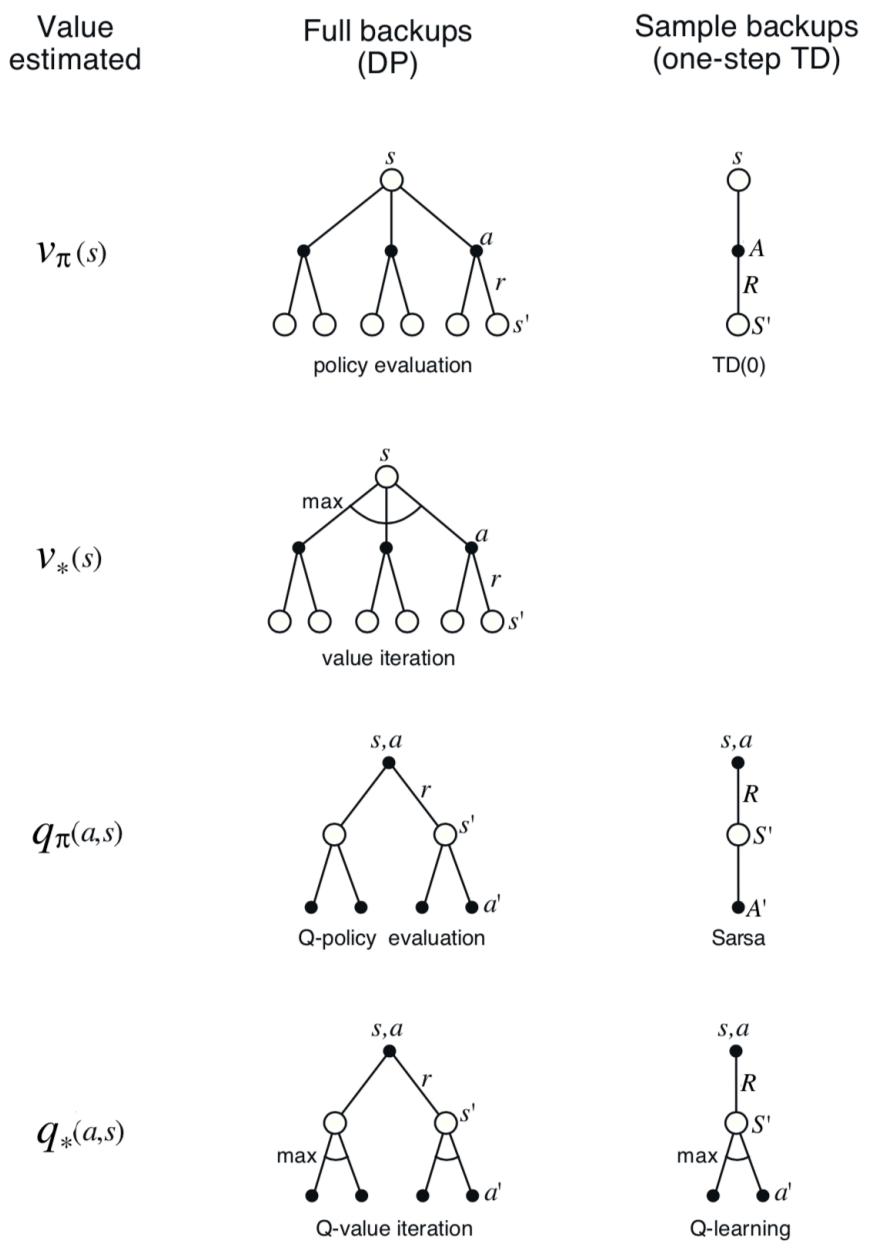

Figure 8.9: The one-step backups.

在这种条件下 Q-learning 的更新公式如下:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[R + \gamma \max_{a'} Q(S', a') - Q(s,a) \right] \tag{8.2}$$

这两种情况的不同的是环境的随机性，特别是给定一个状态和动作，许多可能情况都有许多可能性。如果有足够多的时间来进行一个 full backup，由于没有抽样误差，那么它的效果肯定会优于 sampling backups。

8.6 Trajectory Sampling

- The classical approach from dynamic programming: 通过对整个状态空间进行扫描，备份每一个状态-动作对。
- The second approach is to sample from the state or state–action space according to some distribution: 相对于第一种方式，第二种采用的是抽样代替全部扫描，这样的话可以加快扫描速度，避免一些极低概率的情况。

8.7 Real-time Dynamic Programming