WK 6 算法小结

截至目前，我们已经学习 DP \ MC \ TD 以及 TD 衍生的 Q-Learning。我们先对每种算法进行总结归纳。

算法内容

DP

总览

动态规划是一个算法设计中就出现的名词，这里与算法中的DP类似，都是在给定完整环境的情况下计算最优策略的算法集合。强化学习中的环境就是MDP的情况，这样的假设可以大幅减少传统dp的计算量。DP的核心思想就是用价值函数去组织一种搜索，从而找到好的策略。这里的价值函数就是在有限马尔科夫决策过程中提到的价值函数。一旦找到最优价值函数v_*就可以获得最优策略，这符合Bellman方程：

$$
\begin{aligned}
v_*(s) &= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] \\
&= \max_a \sum_{s^{'},r} p(s^{'}, r | s, a)[r + \gamma v_*(s^{'})]
\end{aligned}
$$

策略评估

首先需要进行策略评估，即考虑给定任意策略π怎样计算值函数v_π。这里进行评估的方法基于Bellman方程。

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi[G_t | S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s^{'},r} p(s^{'}, r | s, a)[r + \gamma v_\pi(s^{'})]
\end{aligned}
$$

式中$\pi(a|s)$是在状态s时使用策略π采取动作a的概率。期望下标表示在策略π的条件下。如果$\gamma < 1$或所有状态在策略$\pi(a|s)$下都能达到最终状态，就能保证v_π唯一存在。其实上式就是一个同时存在$|\mathcal{S}|$个线性方程与$|\mathcal{S}|$个未知数（即$v_\pi(s)$）的系统。求解方法利用不动点法求逼近：

$$
v_{k+1}(s) = \mathbb{E}[R_{t+1} + \gamma v_k(S_{t+1}) | S_t = s]\$ = \sum_a \pi(a|s) \sum_{s^{'},r} p(s^{'}, r | s, a)[r + \gamma v_k(s^{'})]
$$

当$\max_{s\in\mathcal{S}} |v_{k+1}(s) - v_k(s)|$很小时就停止。

策略提升

计算价值函数是为了得到更好策略。假设已确定策略π与价值函数v_π，则对于某些s我们需要知道改变新状态会不会更好，就需要考虑在状态s下选择a动作，并遵从策略π。

$$
\begin{aligned}
q_\pi(s, a) &= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a] \\
&= \sum_{s^{'},r} p(s^{'}, r | s, a)[r + \gamma v_\pi(s^{'})]
\end{aligned}
$$

如果大于v_π，那么新策略总体更好。因此需要选取最好的动作来更新策略。那么新的贪心策略$\pi^{'}$有：

$$
\begin{aligned}
\pi^{'}(s) &= \arg\max_a q_\pi(s,a) \\
&= \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s, A_t = a] \\
&= \arg\max_a \sum_{s^{'},r} p(s^{'}, r|s,a)[r + \gamma v_\pi(s^{'})]
\end{aligned}
$$

策略迭代

若策略π被v_π提升为更好的策略$\pi^{'}$，就可以继续计算$v_{\pi^{'}}$，再提升得到$\pi^{''}$。我们可以得到单调提升的函数：

$$\pi_0 \to^E v_{\pi_0} \to^I \pi_1 \to^E v_{\pi_1} \to^I \cdots \to^I \pi_* \to^E v_{\pi_*}$$

E为评估，I为提升。每一次都能保证严格递增，而有限MDP只有有限个策略，由单调有界定理可知一定能收敛到最优策略与最有价值函数。

MC

蒙特卡罗方法不基于我们对环境的了解，而是一种真正意义上的学习方法。它是基于对样本回报求平均的方法解决问题。为了得到良定义的回报，蒙特卡罗方法这里适用于回合制任务。对于每个回合，无论什么动作都会结束，且结束后价值估计与策略才能改变，这样蒙特卡罗方法就可以写作逐个回合增量的形式。

对于模型不可用的情况而言，估计动作价值会比估计状态价值更为有用，因此蒙特卡罗方法更注重q的估计，即估计$q_\pi(s,a)$，从状态s开始做出动作a，并依据策略π得到的期望回报。但是有可能有许多(s,a)不被访问到，而若π是一个确定的策略，那么每个状态只能观察到一个动作的回报，因此需要估计每个状态可能的动作，而非当前应该选择的动作，这里需要用epsilon-greedy策略选择动作并执行。

接着需要做的是一个策略迭代与策略提升，从一个随机策略π_0开始，最优策略与最优动作-价值函数结束：

$$\pi_0 \to^E q_{\pi_0} \to^I \pi_1 \to^E q_{\pi_1} \to^I \cdots \to^I \pi_* \to^E q_{\pi_*}$$

策略评估与DP类似，而策略提升的方法是让策略贪心。因为我们有动作-价值函数，因此不需要建模来构建贪心策略。对于动作-价值函数，应该

$$\forall s \in \mathcal{S}, \pi(s) = \arg\max_a q(s,a)$$

之后构建π_{k+1}作为q_{π_k}的贪婪策略。

TD

SARSA

TD是MC与DP思想的融合，它吸取了MC的经验，可以从原始经验学习而无需对环境建模；同时又与DP一样，TD基于其他学习估计更新估计，而不需要等待最终的结果。

我们先看TD与蒙特卡罗相近的部分——利用经验来解决预测问题。对于基于策略π的经验，两种方法都更新了其对经验中发生的非终止状态S_t的v_π的估计V。但对于MC来说，它需要等到访问后的回报已知，然后使用这个回报作为$V(S_t)$的目标。即：

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t] - V(S_t)$$

但是，对于TD而言，我们可以在下一个时间$t+1$，立即形成目标并使用观察到的奖励R_{t+1}来进行更新。最简单的TD即：

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t))]$$

过渡到S_{t+1}并接收R_{t+1}。即TD的更新目标为$R_{t+1}+\gamma V(S_{t+1})$。这即是TD(0)方法，是一种自举的方法。

SARSA则是基于TD(0)，力求完成一个控制问题。和MC一样，TD也有在策略和离策略之分，而SARSA是在策略的TD控制方法。

首先，SARSA需要学习动作价值函数，而非状态价值函数。回顾一下，一个回合是由一系列的状态-动作对组成：

$$S_t \rightarrow^{A_t} \rightarrow^{R_{t+1}} S_{t+1} \rightarrow^{A_{t+1}} \rightarrow^{R_{t+2}} S_{t+2} \rightarrow \ldots$$

在TD(0)中，我们考虑状态间的转变，学习了状态的价值，但在SARSA中，需要学习的是状态-动作对的价值：

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

这个学习公式中，需要五元组$(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$，这也是SARSA名称的由来。

这个算法的收敛性取决于对Q的依赖性，可利用ϵ-弹性策略，只要所有状态-动作对被无限次访问，并且策略收敛于贪婪策略的限制（$\epsilon \rightarrow 0$），SARSA就以概率1收敛到最优策略和动作价值函数。

Q-Learning

Q学习是一种离策略TD控制算法，定义为：

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

这种情况下，学习的动作-价值函数Q直接近似于最佳动作价值函数，而与策略无关。

区别

模型、规划与学习

目前已经学习过的这几个算法，首先可以以是否基于模型进行分类，其中DP需要环境模型，而MC、TD则不依赖于环境模型。

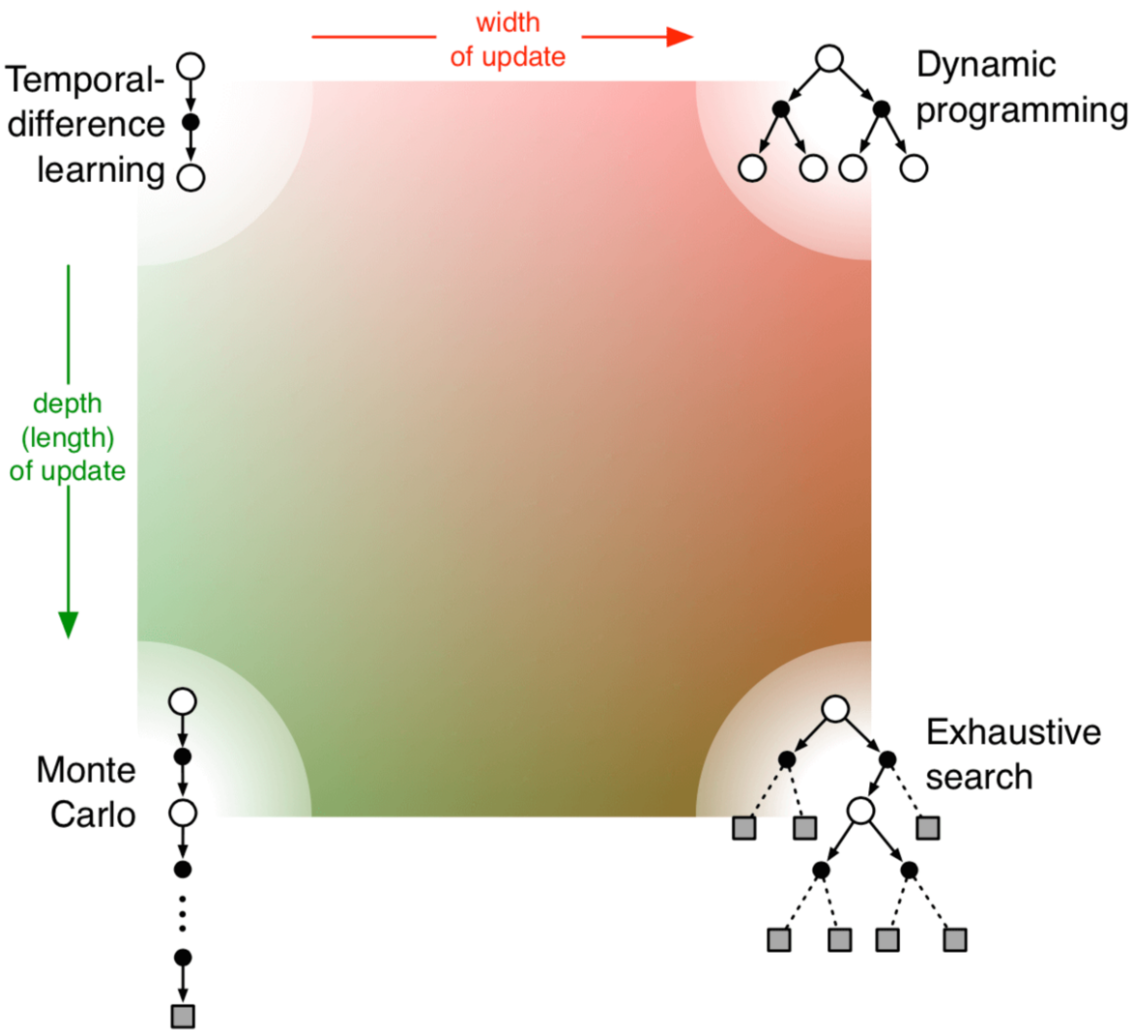
环境模型，可以让智能体来预测环境如何影响其行为。当给定状态与动作，环境模型就能对下一状态与下一奖励进行预测。模型可以用于模仿，当起始状态与动作给定后，模型发生转换，并产生整个回合。总言之，即：**模型用于模拟环境，最终产生经验。**

当模型一旦构建，如何产生最终的策略？规划则是最终产生策略的方法。规划可以认为是对状态空间进行搜索，产生最优策略。DP算法中，首先扫描空间，为每个状态生成可能的转换分布（trans_pro），再利用每个分部来计算备份值，并更新状态的估计值。

$$\text{模型} \longrightarrow \text{模拟经验} \rightarrow^{\text{备份}} \rightarrow \text{价值} \longrightarrow \text{策略}$$

可以看出，规划最优行为与我们所追求的学习最优行为之间关系密切。他们都是估计相同的价值函数。但是，一个基于模型，一个基于经验。可以说，规划与学习十分相似，只是一个来源于模型，一个来源于真实经验，来规划方法。

变化的维度



这张图表现了目前所涉及到的算法的变化的维度，而维度又与用于改进价值的更新类型有关。

横向，即width轴，表现了算法究竟是基于样本还是模型进行更新。可以看到，TD与MC都是基于样本或者实际经验，根本没有模型，但DP与穷举式搜索则需要样本的模型。

纵向，即depth轴，则体现了算法更新的深度（即自举程度）。左边一步TD更新到下面MC完全回报更新；右边DP到穷举式搜索。而其实在正方形的边和内部，也存在一些其他的方法。例如方格迷宫中也可以使用A^*算法，它是启发式算法，基于模型，但并不会进行穷举，所以在上图中应该在DP与穷举搜索之间。一些算法也混合了预期更新和样本更新，使得整个正方形内部有多种多样的算法。

在策略与离策略

在策略的情况中，智能体学习其当前遵循的策略的价值函数；离策略的情况下，它学习不同策略的策略价值函数，一般学习的都是当前智能体的最佳策略。由于探索的出现，策略生成的行为通常与当前认为的最佳行为不同。

回报

在不同的算法中，任务也不尽相同，MC中任务是回合的，而TD中是持续的。而回报是否是有折扣的也会产生不同

动作价值函数、状态价值函数

在不同的算法中，我们使用了不同的函数进行价值计算。DP、SARSA使用了状态估计价值，而MC、Q-Learning则选择了动作价值函数。这是我们由不同的规划（学习）方法决定的。

共同点

1. 都尝试进行估计价值函数实现价值评估。
2. 都通过沿着产生的状态轨迹备份值进行。
3. 都保持近似价值函数与近似策略，并不断尝试利用彼此更新。即遵循广义策略迭代的一般策略。