强化学习基础篇（八）动态规划扩展

1、异步动态规划算法（Asynchronous Dynamic Programming）

同步动态规划（Synchronous Dynamic Programming）是在每次迭代都会同时保存所有状态的值函数。他的确定是对马尔可夫决策过程中的所有状态集S进行扫描（遍历），每一次迭代都会完全更新所有的状态值，该方法称为同步备份（Synchronous Backup）。如果环境中的状态集非常庞大（即状态空间过大），那么即使是单次遍历状态空间，其时间代价也会非常高。例如，西洋双陆棋的状态超过10^{20}种，即使按照每秒执行1000万个状态的值更新，使用普通电脑也需要100年的时间才能完成一次遍历。

异步动态规划（Asynchronous Dynamic Programming，ADP），又称为异步备份（Asynchronous Backup）能够更高效地完成强化学习任务。其思想是通过某种方式，使得每一次扫描（遍历）不需要更新所有的状态值，可以以任意顺序更新状态值，甚至某些状态值可能会在其他状态值更新一次之已经更新过多次。事实上，经过实践和理论证明，很多状态值是不需要被更新的。但如果想要算法正确收敛，那么异步动态规划法必须持续地更新完所有的状态值。不同的是，在选择更新状态时，异步动态规划具有很大的灵活性。

异步动态规划可以非常明显得减少计算量，其收敛必须满足：在任一时刻，任何状态都有可能被选中。

下面将分别介绍异步动态规划中常见的3种方法:

- In-Place动态规划（In-place dynamic programming））
- 加权扫描动态规划（Prioritized sweeping）
- 实时动态规划（Real-time dynamic programming）

1.1、In-Place动态规划（In-place dynamic programming）

在基于同步动态规划的值迭代算法中，存储了两个值函数的备份，分别是$v_{new}(s)$和$v_{old}(s)$。

$$v_{new}(s) = \max_a(r + \gamma \sum_{s' \in S} p(s'|s,a)v_{old}(s'))$$

即在计算过程中，通过赋值的方式使旧的状态值作为下一次计算新的状态值。
而In-place动态规划（In-Place Dynamic Programming，IPDP）则是去掉旧的状态值$v_{old}(s)$，只保留最新的状态值$v_{new}(s)$，在更新的过程中可以减少存储空间的浪费。

$$v(s) = \max_a(r + \gamma \sum_{s' \in S} p(s'|s,a)v(s'))$$

直接原地更新下一个状态值$v(s)$，而不像同步迭代那样需要额外存储新的状态值$v_{new}(s)$。在这种情况下，按何种次序更新状态值有时候会更具有意义。

1.2、加权扫描动态规划（Prioritized sweeping）

加权扫描动态规划（Prioritized Sweeping Dynamic Programming，PSDP）基王贝尔曼误差的大小来指导动作的选择。具体而言，加权扫描的思想是根据贝尔曼误差确定每一个状态是否重要，对于重要的状态，进行更多的更新；对于不重要的状态，则减少其更新次数。

- 状态值的更新顺序:

  使用优先队列来确定状态值函数的更新次序。按照权重优先原则，每次对优先权最高的状态值进行更新。

- 权重设计规则：

  使用贝尔曼误差确定状态的优先权，如：

  $$|\max_{a\in A}(R_s^a+\gamma\sum_{s'\in S}P_{ss'}^a v_{s'}(s'))-v(s)|$$

  通过状态s'和状态的误差作为当前状态的评估标准。如果某种状态上一次的状态值与当前的状态值相差不大，则可以认为该状态趋于稳定，从而减小更新权重，否则放大更新权重。

1.3、实时动态规划（Real-time dynamic programming)

实时动态规划（Real-time dynamic programming，RTDP）的思想是只更新与智能体当前时间点相关的状态，暂时不更新与智能体当前时间点无关的状态。如此，算法便能根据智能体的经验来指导状态选择。

在每个时间步有S_t, A_t, R_{t+1}，进行如下更新

$$v(s)\leftarrow\max_{a\in A}(R_{S_t}^a+\gamma\sum_{s'\in S}P_{S_ts'}v(s'))$$

例如，如果智能体在状态执行动作a，得到环境反馈r，那么此时要做是仅更新当前时间步的状态值函数，而不需要更新全部的状态值函数。

2、全宽备份(Full-width backup)

这种全宽备份(Full-width backup)方法中，对于每一次备份来说，所有的后继状态和动作都要被考虑在内，并且需要已知MDP的转移矩阵 P 与奖励函数 R，因此动态规划将面临维数灾难问题。所以我们就有了样本备份（Sample Backup），这种方法主要思想是利用样本进行backup，优点有：

- 用于无模型方法（Model-free），此时不需要知道MDP的MDP的转移矩阵 P 与奖励函数 R。
- 避免了维数灾难的问题
- 备份（backup)的时间复杂度是固定的，与状态的数量无关。

3、近似动态规划（Approximate Dynamic Programming)

使用其他技术手段（例如神经网络）建立一个参数较少，消耗计算资源较少、同时虽然不完全精确但却够用的近似价值函数：

$$\tilde{v}_k(s)=\max_{a\in A}(R_s^a+\gamma\sum_{s'\in S}P_{ss'}^a\hat{v}(s',w_k))$$

4、压缩映射定理（Contraction Mapping)

压缩映射定理（Contraction Mapping），是度量空间理论的一个重要工具。它保证了度量空间的一定自映射的不动点的存在性和唯一性，并提供了求出这些不动点的构造性方法。

压缩映射定理（Contraction Mapping）在这里应用可以证明如下几个问题：

- 值迭代收敛是否到 v_*（How do we know that value iteration converges to v_* ?)
- 迭代策略评价是否收敛到v_*(iterative policy evaluation converges to v_* ?)
- 策略迭代是否收敛到v_*(policy iteration converges to v_* ?)
- 解是否唯一(Is the solution unique?)
- 算法收敛速度如何(How fast do these algorithms converge?)

证明过程太难，从略。