强化学习模型基础

马尔科夫决策过程 MDPs

价值函数：



=

=

动态规划DP

马尔科夫决策过程MDPs中状态价值函数的Bellman equation：







* 策略评估求解预测问题-求解给定策略的状态价值函数

基本思路是从任意一个状态价值函数开始，依据给定的策略更新迭代下得到最终的状态价值函数。



* 策略迭代求解控制问题-动作策略

一种方法是基于任意一个给定策略评估得到的状态价值函数来及时调整动作策略，这种方法叫做 Policy Iteration，最简单的方法是贪婪法。

例子：在计算出最终的状态价值后，在相应的状态选择相应的动作。个体在某个状态下选择的行为是其能够达到后续所有可能的状态中状态价值最大的那个状态。在计算出最终的状态价值之后，来调整动作。

* 异步动态规划算法

每次迭代并不对所有的状态进行更新，而是依据一定的原则有选择的更新部分状态的价值，

In-place dynamic programming: 即时计算即时更新。

Prioritised strategy: 对每一个状态进行优先级分级，优先级越高的状态其状态价值优先得到更新。通常使用bellman误差来评估状态的优先级。

Real-time dynamic programming: 直接使用agent与环境的交互产生的实际经历来更新状态价值。其中个体经常访问过的状态得到较高频次的价值更新，较少访问到的状态及其价值较少更新。

* 总结

动态规划算法使用 full-width的回溯机制来进行状态价更新，也就是说每一次的回溯更新某一状态的价值时，都要回溯到该状态的所有可能的后续状态，并使用bellman方程更新该状态的价值。

这种价值更新方式适用于状态较少的强化学习，当问题规模很大的情况下，还需要找到其他面向复杂问题的求解方法。

蒙特卡洛（MC）

蒙特卡洛法通过采样若干经历完整的状态序列（episode）来估计状态的实际价值。在计算某一个状态的状态价值时候，MC只需要求出所有完整序列中该状态出现时候的收获，再取平均值可近似求解。优化考虑的点：

* 同样一个状态可能在一个完整的状态序列当中重复出现，有两种解决方法：（1）仅把转台序列当中第一次出现该状态时候的收获纳入到收获平均值的计算当中。（2）每一个状态序列当中出现该状态时候，都计算对应的收获并纳入收获平均值的计算当中。
* 增量更新平均值。计算均值需要保存所有的计算项，比较好的方法是迭代计算收获均值，即保存上一次的收获均值与次数，计算当前的收获均值和次数



该计算公式可以表示为：





在处理海量数据分布式迭代当中，当前的次数可能不方便记录，可以使用一个系数，即为：

* 蒙特卡洛求解控制问题

MC求解控制问题的思路和动态规划价值迭代的思路类似，但是不同点主要体现在：（1）策略评估方法不同，（2）一般是优化最优动作价值函数，而不是状态价值函数，（3）动态规划一般基于贪婪法更新策略，MC一般采用贪婪法更新，通过设置一个较小的值，采用选择目前认为最大行为价值的行为，而用的概率随机从m个可选行为中进行选择。



同时在实际求解控制问题的过程当中，一般会随着算法的迭代过程逐渐减小，并趋于0

时序差分（TD）

采用为TD目标值（bootstrapping 方法），为TD误差

* TD的预测问题求解







* TD与MC的差别

1. TD在知道结果之前就可以学习
2. TD在更新状态价值时使用的TD目标值，是对当前状态价值的有偏估计
3. TD的方差要低，更加高效

* TD(n)

由于TD为前一步收获来近似收获，那么前两部的情况下，则有：



向前n步的情况下：



对于n步时序差分来说，和普通时序差分的区别在于收获的计算方式的差异。那么n到底为多少步好呢

* TD(λ)

定义λ-收获是所有步的收获乘以权重的和，每一步的权重为，这样λ-的收获为：



则TD(λ)的价值函数的迭代公式为：





从前向看TD(λ)，一个状态的价值由得到，而间接由后续状态的状态价值计算得到，因此可以认为更新一个状态的价值需要知道所有的后续状态的价值，必须要经历完整的状态序列包括终止状态的每一个状态的即时奖励才能更新当前状态的价值。这和MC的要求相似，因此TD(λ)和MC存在同样的问题。

从反向看TD(λ)，它可以分析状态对后续状态的影响。

* 给每一个状态引入一个数值，E (eligibility)来表示状态对后续状态的影响，所有状态的效用值，总称为ES (eligibility traces)，定义为：



 s is visited once at time k

于是TD(λ)的价值函数更新式子可以表示为：





* TD控制问题求解

对于TD也可以采用-贪婪法来价值迭代，和MC在线控制的区别主要在于收获的计算方式不同。TD on-policy算法最常见的是SARSA算法。对于off-policy一般有两个策略，一个策略（最常见的是-贪婪法）用于选择新的动作，另一个策略（最常用的贪婪法）用于更新价值函数，比较常见的是Q-learning算法

SARSA

SARSA算法是一种使用时序差分求解强化学习控制问题的方法，和MC类似，都是价值迭代，即通过价值函数的更新，来更新当前的策略，产生新的状态和及时奖励，进而更新价值函数，直到价值函数和策略不再更新。SARSA属于on-policy，即一直使用一个策略来更新价值函数和选择新的动作。

SARSA算法首先基于-贪婪法在当前状态S选择一个动作A，系统进入新的状态，同时获得一个即时奖励R,在新的状态，基于-贪婪法在选择一个动作，只是用来更新价值函数，价值函数的更新公式为：



* SARSA(λ)

SARSA(λ)默认基于反向来进行价值函数迭代

TD off-policy 算法 Q-learning

对于Q-learning算法，基于状态S采用-贪婪法选择新的动作A，得到奖励R,并进入状态， Q-learning使用贪婪法选择，而不是-贪婪法，这是Q-learning与SARSA的本质区别。选择使最大的a作为来更新价值函数，其公式如下：



此时选择的动作只会参与价值函数的更新，不会执行。价值函数更新后，新的执行动作需要基于状态，用-贪婪法选择得到。相对于SARSA，价值函数更新使用的会作为下一阶段开始时候的执行动作。

* SARSA vs Q-learning

1. Q-learning 学习的是最优策略，SARSA在学习最优策略的同时还在做探索，所以SARSA需要超参数在迭代的过程中逐渐变小
2. Q-learning直接学习最优策略，但是最优策略会以来于训练中产生的一系列数据，所以受样本数据的影响较大，因此受到训练数据方差的影响很大。同样Q-learning的深度强化学习版Deep Q-learning存在同样的问题。
3. SARSA在学习的过程中鼓励探索，学习过程会比较平滑，不至于过于激进，导致出现像Q-learning可能遇到的一些特殊的最优“陷阱”，比如经典强化学习问题“Cliff walk”

在实际应用当中，对于模拟环境中训练强化学习模型，推荐使用Q-learning，如果在线生产环境中训练模型，推荐使用SARSA.

在大数据情况下，异常复杂的状态和可选动作，是Q-learning和SARSA要维护的Q表异常的大，在深度血虚兴起之后，基于深度学习的强化学习开始占主导地位，也是一种比较好的解决方案。

Deep Q-learning

* 价值函数的近似表示方法

由于状态集合规模较大，一个可行的建模方法是价值函数的近似表示，通过引入一个状态价值函数,这个函数由参数w描述，并接受状态作为输入，计算后得到状态s的价值，即期望：



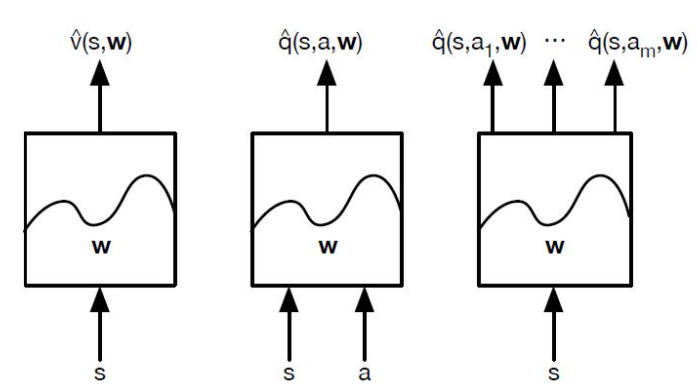
类似的，引入一个动作价值函数，这个函数由参数w描述，并接受状态s和动作a作为输入，计算后得到动作价值，即期望：



价值函数近似的方法很多，比如简单的线性表示法，用表示状态s的特征向量，则此时的状态价值函数可以近似表示为：



通常情况下，比较常见和引用比较广泛的表示方法是神经网络，可以使用DNN，CNN或者RNN。通常近似过程可以由以下三种情况：



对于状态价值函数，神经网络的输入是s的特征向量，输出是状态价值。对于动作价值函数，有两种方法，一种树输入状态s的特征向量和动作a，输出对应的动作价值，另外一种是只输入状态s的特征向量，动作集合有多少个动作就有多少个输出。这里隐含了动作是有限个的离散动作。

* Deep Q-learning算法思路

Q值的计算不是直接通过状态和动作来计算，而是通过Q网络进行，DQN主要使用的技巧是经验回放（experience replay），即每次和环境交互得到的奖励与状态更新情况都保存起来，用于后面目标Q值的更新。由于在Q-learning当中有一张Q表来保存所有的Q值的当前结果，但是DQN是没有的，在动作价值函数更新的时候，采用经验回放。

通过经验回放的到的目标Q值和通过Q网络计算的Q值存在误差，通过梯度的反向传播来更新神经网络的w，从而得到近似的Q值的计算方法。