强化学习 Reinforcement learning

概念

机器处于环境E中， 状态空间为X，其中每个状态$x \in X$ 是机器感知到的环境的描述，**例如： 种瓜任务中对西瓜长势的描述。** 机器能采取的动作构成了动作空间 A 。**例如：施肥，浇水，使用农药等。** 若某个动作$a \in A$作用在当前状态x上，则潜在的转移函数P 将使得环境从当前状态按某种概率转移到另一个状态。**例如：瓜苗状态为缺水时候，执行浇水的操作，有一定概率会回复健康也可能不回复。** 在转移到另一个状态的同时，环境会根据潜在的“奖赏”函数R反馈给机器一个奖赏。**例如：保持健康奖赏+1，凋零则-10，奖赏可以简单地理解为一个抽象的指标。**

综合来看，强化学习对应了四元组 $E =< X, A, P, R >$，其中$P : X \times A \times X \mapsto \mathbb{R}$ 指定了状态转移概率，$R : X \times A \times X \mapsto \mathbb{R}$ 指定了奖赏。在某些特殊情况，奖赏函数可能纸盒状态转移相关 $R : X \times X \mapsto \mathbb{R}$

目的： 机器要做的是在这种环境下学习出一种策略π，根据这个策略，在状态x下就能得知要执行的动作$a = \pi(x)$，**例如：看到瓜苗状态为缺水，就能得知动作“浇水”。**

K-摇臂赌博机

探索和利用

K-摇臂赌博机是一种模型，有K个摇臂，赌徒再投入一个硬币后，可以选择一个摇臂，每个摇臂以一定概率吐出硬币，但这个概率赌徒并不知道，赌徒的目标是通过一定的策略最大化奖赏。

探索：估计摇臂优劣 利用：选择当前最优摇臂

仅探索法：将所有的尝试机会**平均**分给每个摇臂，最后以每个摇臂各自的平均吐币概率作为其奖赏期望的近似估计。**目的是为了获取每个要比的期望奖赏**

仅利用法： 按下目前最优的(即到目前为止**平均奖赏最大**的)摇臂，若有多个则随机选取其中一个。 **目的是为了执行奖赏最大的动作**

区别:仅探索法能很好估计每个摇臂的奖赏，但是会失去很多选择最优机会的摇臂，仅利用法刚好相反，他很难估计每个摇臂的期望奖赏。

为了能够比较好的对探索和利用进行折中，有两种算法优化，ϵ-贪心和softmax

ϵ-贪心

ϵ-贪心法基于一个概率来对探索和利用进行这种：每次尝试时，以ϵ的概率进行探索，即以均匀概率随机选取一个摇臂；以$1-\epsilon$的概率进行利用，即选择当前平均奖赏最高的摇臂。

令$Q(k)$记录摇臂k的平均奖赏。若摇臂k被尝试了n次，得到的奖赏为$v_1, v_2, \ldots, v_n$，则平均奖赏为：

$$Q (k) = \frac { 1 } { n } \sum_ { i = 1 } ^ { n } v _ { i }$$

然而上式需要记录多个奖赏值， 不够高效，更方便的方式是没尝试一次就更新，用下表来表示尝试的测试，则在经过n此尝试后，平均奖赏应该更新为

$$Q _ { n } (k) = \frac { 1 } { n } \left((n - 1) \times Q _ { n - 1 } (k) + v _ { n } \right)$$

$$= Q _ { n - 1 } (k) + \frac { 1 } { n } \left(v _ { n } - Q _ { n - 1 } (k) \right)$$

这样只需要记录最近平均奖赏和已经尝试的次数n-1。当摇臂奖赏的不确定性比较大时，需要更多探索，则需要较大的ϵ。 当不确定较小时，则刚好相反。然而，若尝试次数非常大，则当一段时间后，奖赏可以比较好的近似出来就不需要探索，这个时候可以让ϵ随着次数增加而逐渐减小。

Softmax

Softmax算法是基于当前已知的摇臂的平均奖赏来对探索和利用进行这种。若当前各摇臂的平均奖赏相当，则选取各摇臂的概率也相当，若某些摇臂的平均奖赏明显高于其他摇臂，则选取概率也更高

Softmax算法中概率分配是基于BOltzmann分布

$$P (k) = \frac { e ^ { \frac { Q (k) } { \tau } } } { \sum _ { i = 1 } ^ { K } e ^ { \frac { Q (i) } { \tau } } }$$

其中，$Q(i)$记录当前摇臂的平均奖赏；$\tau > 0$称为"温度"， τ越小则平均奖赏高的摇臂被选取的概率越高。τ趋于0时，Softmax将趋于"仅利用"，τ趋于无穷大时softmax则将趋于"仅探索"。

有模型学习

在已知模型的环境中，即马尔科夫决策过程中四元组$E = \langle X, A, P, R \rangle$均为已知，学习的过程称为"有模型学习"。为方便讨论，先假设状态空间X和动作空间A均为有限。

策略评估

在模型已知时，对于任意策略π能估计出该策略带来的期望累计奖赏。令函数$V^{\pi}(x)$表示从状态x出发，使用策略π所带来的累计奖赏。函数$Q^{\pi}(x, a)$表示从状态x出发，执行动作a以后在使用策略π带来的累计奖赏。$V(\cdot)$表示状态值函数。$Q(\cdot)$表示状态-动作值函数。分别表示指定"状态"上以及指定"状态-动作"上的累计奖赏。

根据T步累积奖赏和γ折扣累积奖赏的定义，有状态值函数：

$$\left\{ \begin{array} { l } { V _ { T } ^ { \pi } (x) = \mathbb { E } _ { \pi } \left[\frac { 1 } { T } \sum _ { t = 1 } ^ { T } r _ { t } | x _ { 0 } = x \right] } \\ { V _ { \gamma } ^ { \pi } (x) = \mathbb { E } _ { \pi } \left[\sum _ { t = 0 } ^ { + \infty } \gamma ^ { t } r _ { t + 1 } | x _ { 0 } = x \right] } \end{array} \right.$$

同理有状态-动作值函数：

$$\left\{ \begin{array} { l } { Q _ { T } ^ { \pi } (x , a) = \mathbb { E } _ { \pi } \left[\frac { 1 } { T } \sum _ { t = 1 } ^ { T } r _ { t } | x _ { 0 } = x , a _ { 0 } = a \right] } \\ { Q _ { \gamma } ^ { \pi } (x , a) = \mathbb { E } _ { \pi } \left[\sum _ { t = 0 } ^ { + \infty } \gamma ^ { t } r _ { t + 1 } | x _ { 0 } = x , a _ { 0 } = a \right] } \end{array} \right.$$

由于MDP具有马尔科夫性质，即系统下一时刻的状态仅由当前时刻的状态决定，不依赖于以往任何状态。

对于T步累积奖赏有

$$\left.\begin{aligned} V _ { T } ^ { \pi } (x) & = \mathbb { E } _ { \pi } \left[\frac { 1 } { T } \sum _ { t = 1 } ^ { T } r _ { t } | x _ { 0 } = x \right] \\ & = \mathbb { E } _ { \pi } \left[\frac { 1 } { T } r _ { 1 } + \frac { T - 1 } { T } \frac { 1 } { T - 1 } \sum _ { t = 2 } ^ { T } r _ { t } | x _ { 0 } = x \right] \\ & = \sum _ { a \in A } \pi (x , a) \sum _ { x ^ { \prime } \in X } P _ { x \rightarrow x ^ { \prime } } ^ { a } \left(\frac { 1 } { T } R _ { x \rightarrow x ^ { \prime } } ^ { a } + \frac { T - 1 } { T } \mathbb { E } _ { \pi } \left[\frac { 1 } { T - 1 } \sum _ { t = 1 } ^ { T - 1 } r _ { t } | x _ { 0 } = x ^ { \prime } \right] \right) \\ & = \sum _ { a \in A } \pi (x , a) \sum _ { x ^ { \prime } \in X } P _ { x \rightarrow x ^ { \prime } } ^ { a } \left(\frac { 1 } { T } R _ { x \rightarrow x ^ { \prime } } ^ { a } + \frac { T - 1 } { T } V _ { T - 1 } ^ { \pi } \left(x ^ { \prime } \right) \right) \end{aligned} \right.$$

类似的对于γ折扣累积奖赏有

$$V _ { \gamma } ^ { \pi } (x) = \sum _ { a \in A } \pi (x , a) \sum _ { x ^ { \prime } \in X } P _ { x \rightarrow x ^ { \prime } } ^ { a } \left(R _ { x \rightarrow x ^ { \prime } } ^ { a } + \gamma V _ { \gamma } ^ { \pi } \left(x ^ { \prime } \right) \right)$$

策略改进

对某个策略的累积奖赏评估后，发现并不是最优的，理想的策略应该是最大化累积奖赏。

$$\pi ^ { * } = \underset { \pi } { \arg \max } \sum _ { x \in X } V ^ { \pi } (x)$$

因此最优策略所对应的值函数V^*称为最优值函数，即

$$\forall x \in X : V ^ { * } (x) = V ^ { \pi ^ { * } } (x)$$

注意，当策略空间无约束时，上式才是最优策略对应的值函数。

由于最优值函数的累积奖赏值已达最大，因此对前面的累积奖赏函数改动一下，就可以得到最优值函数。

$$\left\{ \begin{array} { l } { V _ { T } ^ { * } (x) = \max _ { a \in A } \sum _ { x ^ { \prime } \in X } P _ { x \rightarrow x ^ { \prime } } ^ { a } \left(\frac { 1 } { T } R _ { x \rightarrow x ^ { \prime } } ^ { a } + \frac { T - 1 } { T } V _ { T - 1 } ^ { * } \left(x ^ { \prime } \right) \right) } \\ { V _ { \gamma } ^ { * } (x) = \max _ { a \in A } \sum _ { x ^ { \prime } \in X } P _ { x \rightarrow x ^ { \prime } } ^ { a } \left(R _ { x \rightarrow x ^ { \prime } } ^ { a } + \gamma V _ { \gamma } ^ { * } \left(x ^ { \prime } \right) \right) } \end{array} \right.$$

换言之就是

$$V ^ { * } (x) = \max _ { a \in A } Q ^ { \pi ^ { * } } (x , a)$$

带入状态动作值函数可得到

$$\left\{ \begin{array} { l } { Q _ { T } ^ { * } (x , a) = \sum _ { x ^ { \prime } \in X } P _ { x \rightarrow x ^ { \prime } } ^ { a } \left(\frac { 1 } { T } R _ { x \rightarrow x ^ { \prime } } ^ { a } + \frac { T - 1 } { T } \max _ { a ^ { \prime } \in A } Q _ { T - 1 } ^ { * } \left(x ^ { \prime } , a ^ { \prime } \right) \right) } \\ { Q _ { \gamma } ^ { * } (x , a) = \sum _ { x ^ { \prime } \in X } P _ { x \rightarrow x ^ { \prime } } ^ { a } \left(R _ { x \rightarrow x ^ { \prime } } ^ { a } + \gamma \max _ { a ^ { \prime } \in A } Q _ { \gamma } ^ { * } \left(x ^ { \prime } , a ^ { \prime } \right) \right) } \end{array} \right.$$

上述等式称为最优Bellman等式，其唯一解是最优值函数。其解释了非最优策略的改进方式：将策略选择的动作改变为当前最优的动作。

$$\left.\begin{aligned} V ^ { \pi } (x) & \leqslant Q ^ { \pi } \left(x , \pi ^ { \prime } (x) \right) \\ & = \sum _ { x ^ { \prime } \in X } P _ { x \rightarrow x ^ { \prime } } ^ { \pi ^ { \prime } (x) } \left(R _ { x \rightarrow x ^ { \prime } } ^ { \pi ^ { \prime } (x) } + \gamma V ^ { \pi } \left(x ^ { \prime } \right) \right) \\ & \leqslant \sum _ { x ^ { \prime } \in X } P _ { x \rightarrow x ^ { \prime } } ^ { \pi ^ { \prime } (x) } \left(R _ { x \rightarrow x ^ { \prime } } ^ { \pi ^ { \prime } (x) } + \gamma Q ^ { \pi } \left(x ^ { \prime } , \pi ^ { \prime } \left(x ^ { \prime } \right) \right) \right) \\ & = \dots \\&= V ^ { \pi ^ { \prime } } (x) \end{aligned} \right.$$

这个公式我其实没推出来过。。。 因此改进后的策略为

$$\pi ^ { \prime } (x) = \underset { a \in A } { \arg \max } Q ^ { \pi } (x , a)$$

策略迭代与值迭代

知道如何评估一个策略和改进一个策略后，将两者结合起来就可以求解得到最优解的方法：从一个初始策略出发，先进性策略评估，然后改进策略，评估改进的，再进一步改进一次类推，这称为”策略迭代“

但是策略迭代算法每次改进之后都需要对策略评估，因此比较耗时，因此可以把**策略改进视作值函数的改进**

$$\left\{ \begin{array} { l } { V _ { T } (x) = \max _ { a \in A } \sum _ { x ^ { \prime } \in X } P _ { x \rightarrow x ^ { \prime } } ^ { a } \left(\frac { 1 } { T } R _ { x \rightarrow x ^ { \prime } } ^ { a } + \frac { T - 1 } { T } V _ { T - 1 } \left(x ^ { \prime } \right) \right) } \\ { V _ { \gamma } (x) = \max _ { a \in A } \sum _ { x ^ { \prime } \in X } P _ { x \rightarrow x ^ { \prime } } ^ { a } \left(R _ { x \rightarrow x ^ { \prime } } ^ { a } + \gamma V _ { \gamma } \left(x ^ { \prime } \right) \right) } \end{array} \right.$$

强化学习任务可以归结基于动态规划的寻有问题，但是和监督学习不同在于，他并未涉及到泛化能力，只是为每一个状态寻找到最好的动作。

免模型学习

与前面提到的有模型学习相反，不提前知道四元组的前提下，不依赖环境建模就是”免模型学习“

蒙特卡罗强化学习

从起始状态出发，使用某种策略进行采样，执行该策略T步并获得轨迹

$$< x _ { 0 } , a _ { 0 } , r _ { 1 } , x _ { 1 } , a _ { 1 } , r _ { 2 } , \ldots , x _ { T - 1 } , a _ { T - 1 } , r _ { T } , x _ { T } >$$

然后更新Q，然后根据Q再生成新的策略，再利用新策略更新Q，如此迭代下去。

根据生成策略和原始策略是否一致分为同策略和异策略两种学习方法：

同策略：生成策略与更新策略都为原始策略π对应的ϵ-贪心策略 异策略：生成策略是原始策略的π对应的贪心策略，更新策略是原始策略

时序差分和模仿学习待更新