马尔可夫决策过程

http://www.dosrc.com/

1 马尔可夫决策过程模型

80至90年代期间，人们逐渐的认识到把不完全感知的马尔可夫决策过程作为强化学习问题的模型是合适的。目前在人工智能中，马儿可夫决策过程也已被作为新的，特别适合的一种规划问题加以广泛的研究。

强化学习通常用马尔可夫决策过程来对问题进行建模，马尔可夫决策过程用一个五元组表示:

$$MDP\left(S,A,P_{sa},\gamma,R\right)$$

其中S表示状态集，A表示动作集，P_{sa}表示状态转换概率分布（$\sum_{s'}P\left(s'\right)=1$,$s'$为$s$状态下执行$a$动作转移到的可能状态），$\gamma$表示折扣因子（$0\leq\gamma<1$），$R$是奖励函数，它与状态集对应

2 评价

那么怎么来评价策略的好坏呢？使用奖励函数显然不行，奖励函数是对一个状态（动作）的即时评价，我们需要一个能够考虑长远利益的东西。由此我们定义了值函数，值函数是从长远的角度来考虑一个状态（或状态-动作对）的好坏。值函数又称为评价函数。一种值函数的定义如下：

$$V^{\pi}\left(s\right)=E^{\pi}\left(r_t+\gamma r_{t+1}+\gamma^2 r_{t+2}+\ldots|s_0=s\right)=E^{\pi}\left(\sum_{i=0}^{\infty}\gamma^i r_{t+i}|s_0=s\right)$$

Q函数是另外一种评价函数。在默某些时候，记录状态-动作对的值比只记录状态的值更有用，Watkins把状态-动作对的值称为Q值。一种Q函数的定义如下：

$$Q^{\pi}\left(s,a\right)=E^{\pi}\left(r_t+\gamma r_{t+1}+\gamma^2 r_{t+2}+\ldots|s_0=s,a_0=a\right)=E^{\pi}\left(\sum_{i=0}^{\infty}\gamma^i r_{t+i}|s_0=s,a_0=a\right)$$

不同的策略下有不同的评价函数的值，我们的目的就是要最大化评价函数的值：

$$V^{*}\left(s\right)=\max_{\pi}\left(V^{\pi}\left(s\right)\right)$$

$$Q^{*}\left(s,a\right)=\max_{\pi}\left(Q^{\pi}\left(s,a\right)\right)$$

奖励值是即时的奖励，而状态值（Q值）是长远的，是Agent在其整个生命周期内通过一系列观察，不断估计得出的。事实上，绝大部分强化学习算法的研究就是针对如何有效快速的估计值函数。因此，值函数是强化学习方法的关键。

3 算法类型

当前，解决强化学习问题主要有两大类算法：一类是值函数估计法，这是目前强化学习领域研究最为广泛，发展最为迅速的方法；另一类是策略空间直接搜索法，如遗传算法、遗传程序设计、模拟退火以及其它一些进化方法。

4 值函数估计法

4.1 model-base VS model-free

如果知道系统的模型，即转移概率和奖励函数，那么这属于经典的规划问题，解此方程有各种各样的动态规划方法，算法更新所有状态，常见的算法有Policy Iteration，Value Iteration。但是实际系统中，很多时候我们不知道模型，这种情况下典型的算法有Q(0)、Sarsa(0)、Monta Carlo等。在无模型算法中，为了加快收敛的速度可以在边学习的同时学习系统的模型，利用学习的模型进行值函数的更新以加快收敛速度，这一算法实际上是把规划和强化学习结合起来，典型的算法有Dyna-Q

4.1.1 Policy Iteration(model-base)

Algorithm 1 Policy Iteration

Initialization

$V(s) \in R$ and $\pi(s) \in A(s)$ arbitrarily for all $s \in S$

Policy Evaluation

repeat
  $\Delta \leftarrow 0$
  for each $s \in S$ **do**
    $u \leftarrow V(s)$
    $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s)) [r + \gamma V(s')]$
    $\Delta \leftarrow \max(\Delta, |u - V(s)|)$
  end for
until $\Delta < \theta$ ($a\ small\ positive\ number$)

Pilicy Improvement

policy-stable $\leftarrow$ true
for each $s \in S$ **do**
  $a \leftarrow \pi(s)$
  $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a) [r + \gamma V(s')]$
  if $a \neq \pi(s)$ **then**
    policy-stable $\leftarrow$ false
  end if
end for
if policy-stable **then**
  return V and π
else
  got to Policy Evaluation
end if

4.1.2 Value Iteration(model-base)

Algorithm 2 Value Iteration

Initialize array V arbitrarily ($e.g., V(s) = 0\ for\ all\ s \in S^+$)
repeat
  $\Delta \leftarrow 0$
  for each $s \in S$ **do**
    $u \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a) [r + \gamma V(s')]$
    $\Delta \leftarrow \max(\Delta, |u - V(s)|)$
  end for
until $\Delta < \theta$ ($a\ small\ positive\ number$)
Output a deterministic policy,π,such that
$\pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a) [r + \gamma V(s')]$

4.1.3 Q Learning(model-free)

Algorithm 3 Q Learning

Initialize $Q(s,a), \forall s \in S, a \in A(s)$, arbitrarily, and$Q(terminal-state, \cdot) = 0$
repeat
  $\Delta \leftarrow 0$
  for each $s \in S$ **do**
    $u \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
    $\Delta \leftarrow \max(\Delta, |u - V(s)|)$
  end for
until $\Delta < \theta$ $(a\ small\ positive\ number)$
Output a deterministic policy,π,such that
$\pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$