1 Reinforcement Learning

1.1 Foundation

增强学习（五）—— 时间差分学习(Q learning, Sarsa learning)

⋆ 动态规划算法的特性：

- 需要环境模型，即状态转移概率P_{sa}.
- 状态值函数的估计是自举的(bootstrapping)，即当前状态值函数的更新依赖于已知的其他状态值函数.

⋆ 蒙特卡罗方法的特点：

- 可以从经验中学习不需要环境模型.
- 状态值函数的估计是相互独立的.
- 只能用于episode tasks.

⋆ Monte Carlo

Monte Carlo的状态值函数更新公式如下：

$$V(s_t) \leftarrow V(s_t) + \alpha[R_t - V(s_t)] \tag{1}$$

其中R_t是每个episode结束后获得的实际累积回报，α 是学习率，这个式子的直观的理解就是用实际累积回报R_t 作为状态值函数$V(s_t)$ 的估计值。具体做法是对每个episode，考察实验中s_t的实际累积回报R_t 和当前估计$V(s_t)$的偏差值，并用该偏差值乘以学习率来更新得到$V(S_t)$的新估值。

⋆ TD(0)

把等式1中R_t换成$r_{t+1} + \gamma V(s_{t+1})$，就得到了TD(0)的状态值函数更新公式：

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \tag{2}$$

为什么修改成这种形式呢，回忆一下状态值函数的定义：

$$V^\pi(s) = E_\pi[r(s^{'}|s,a) + \gamma V^\pi(s^{'})] \tag{3}$$

容易发现这其实是根据等式3的形式，利用真实的立即回报r_{t+1}和下个状态的值函数$V(s_{t+1})$来更新$V(s_t)$，这种方式就称为时间差分(temporal difference)。由于没有状态转移概率，所以要利用多次实验来得到期望状态值函数估值。类似MC方法，在足够多的实验后，状态值函数的估计是能够收敛于真实值的。

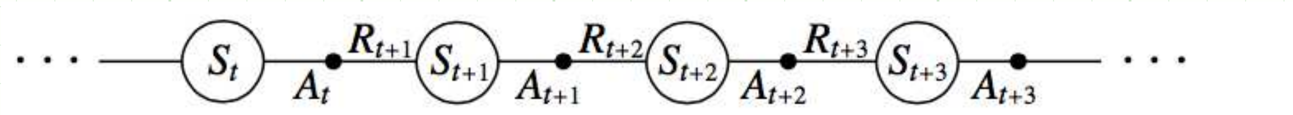
⋆ 策略会计(TD prediction)

输入：待估计的策略π
任意初始化所有$V(s)$，(e.g., $V(s) = 0, \forall s \in s^+$)
Repeat(对所有episode):
    初始化状态s
    Repeat(对每步状态转移):
        $a \leftarrow$策略π下状态s采取的动作
        采取动作a，观察回报r，和下一个状态$s^{'}$
        $V(s) \leftarrow V(s) + \alpha[r + \lambda V(s^{'}) - V(s)]$
        $s \leftarrow s^{'}$
    Until s_t is terminal
Until 所有$V(s)$收敛
输出$V^{\pi}(s)$

⋆ Sarsa算法

强化学习算法可以分为在策略(on-policy)和离策略(off-policy)两类，sarsa算法属于on-policy算法。

Sarsa算法估计的是动作值函数(Q函数)而非状态值函数，也就是说，估计的是策略π下，任意状态s上所有可执行的动作a的动作值函数$Q^{\pi}(s, a)$，Q函数同样可以利用TD Prediction算法估计。如下就是一个状态-动作对序列的片段及相应的回报值。



给出Sarsa的动作值函数更新公式如下：

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \lambda Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{4}$$

可见公式4与公式2的形式基本一致。需要注意的是，对于每个非终止的状态s_t，在到达下个状态s_{t+1}后，都可以利用上述公式更新$Q(s_t, A_t)$，而如果s_t是终止状态，则要令$Q(s_{t+1} = 0, a_{t+1})$。由于动作值函数的每次更新都与$(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$相关，因此算法被命名为sarsa算法。sarsa算法的完整流程图如下：

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
    Initialize S
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Repeat (for each step of episode):
        Take action A, observe R, S'
        Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)
        $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma Q(S', A') - Q(S, A)]$
        $S \leftarrow S'$; $A \leftarrow A'$;
    until S is terminal

算法最终得到所有状态-动作对的Q函数，并根据Q函数输出最优策略π。

⋆ Q-learning

在sarsa算法中，选择动作时遵循策略和更新动作值函数时遵循的策略是相同的，即ϵ-greedy的策略，而在Q-learning中，动作值函数更新则不同于选取动作时遵循的策略，这种方式称为离策略(off-policy)。Q-learning的动作值函数更新公式如下：

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \lambda \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \tag{5}$$

可以看到，Q-learning与sarsa算法最大的不同在于更新Q值的时候，直接使用了最大的$Q(s_{t+1,a})$值——相当于采用了$Q(s_{t+1}, a)$值最大的动作，并且与当前执行的策略，即选取动作a_t时采用的策略无关。Off-policy方式简化了证明算法分析和收敛性证明的难度，使得它的收敛性很早就得到了证明。Q-learning的完整流程图如下：

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode): Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
    $S \leftarrow S'$
  until S is terminal

[xxxx]http://www.cnblogs.com/jinxulin/p/5116332.html
Baidu
$\max_{a<x<b} \{f(x)\}$

XXX

Figure 1: XXX

adfasd
adfasd

- aaa.
- bbb.

1. AAA.
2. BBB.
   - b1.
   - b2.

   BBB.
3. CCC.

Round 1: Participant U_i performs the following steps:

1) Choose $x_i \in Z_q^*$ and compute $X_i = g^{x_i}$.

2) Broadcast message (U_i, X_i).

Round 2: Upon receiving messages (U_{i-1}, X_{i-1}) and (U_{i+1}, X_{i+1}), each U_i does as following:

1) Compute $Y_i^L = X_{i-1}^{x_i}$, $Y_i^R = X_{i+1}^{x_i}$, $Y_i = Y_i^R / Y_i^L$.

2) Broadcast message (U_i, Y_i).

Session Key Generation: Upon receiving all messages $(U_j, Y_j)_{j \in \{1,\cdots,n\}, j \neq i}$, each U_i carries out the following steps:

1) Compute orderly $\hat{Y}_{i+1}^R = Y_{i+1} \cdot Y_i^R$, $\hat{Y}_{i+2}^R = Y_{i+2} \cdot \hat{Y}_{i+1}^R$, $\cdots$, $\hat{Y}_{i+(n-1)}^R = Y_{i+(n-1)} \cdot \hat{Y}_{i+(n-2)}^R$.

2) Check $Y_i^L \stackrel{?}{=} \hat{Y}_{i+(n-1)}^R$. If it is true, continue; Otherwise, abort.

3) Generate the session key $sk = \hat{Y}_1^R \cdot \hat{Y}_2^R \cdot \cdots \cdot \hat{Y}_n^R = g^{x_1 x_2 + x_2 x_3 + \cdots + x_n x_1}$.

Figure 2: One unauthenticated GKA Protocol

section name goes here

⋆ term definition

DEF: **term - and it's definition**

⋆ an example

EX: **example heading**

⋆ a system of equations

$$\begin{cases} 2x + 4y = 2 \\ 2x + 6y = 3 \end{cases}$$

⋆ working a multistep problem

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \xrightarrow{R_1+R_2} \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \Rightarrow \begin{cases} x = 1 \\ y = 2 \\ z = 3 \end{cases}$$

⋆ a vector in R^3

$$v = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

⋆ a multi-step process

$A \xrightarrow{\text{do stuff}} B \xrightarrow{\text{more stuff}} C$

⋆ an enumerated list

1. this is the first item in an enumerated list
2. this is the second item in an enumerated list

⋆ manually broken lines

the first line
the second line
the third line

⋆ some math

$$\int_a^b f(x)\,dx \int f(x)\,dx \; \frac{\pi}{2} \; \sqrt{\theta} \; n = 1, 2, 3 \ldots 4$$