**基于时间差分的强化学习方法：SARSA和Q\_Learning**

通过马尔科夫决策过程MDP将序贯决策问题进行形式化之后，根据状态转移矩阵是否已知，可以将解决问题的方法分为基于模型的强化学习和无模型的强化学习两大类。对于模型已知（即已知）的，可采用动态规划DP方法，用自举算法和贝尔曼方程进行策略评估和策略改善；对于模型未知的，需要用采样方法来计算状态值函数。

蒙特卡罗MC是基于值函数的强化学习方法之一，它利用“经验平均”计算状态值函数，具体有第一次访问蒙特卡罗方法（Formula（1））和每次访问蒙特卡罗方法（Formula（2））。但需要每次试验eposide结束后，才能计算出所有状态的值函数。在某些情况下（比如，试验数据需要大量，或是连续性任务无法获取试验数据），蒙特卡罗的学习速度慢，即收敛过程长。

 Formula（1）

 Formula（2）

时间差分（Temporal-Difference Learning, TD）综合利用了蒙特卡罗MC的采样方法和动态规划DP的自举算法，利用后继状态的值函数乘以折扣因子，再加上回报，得出当前状态的值函数（Formula（3））。

 Formula（3）

时间差分TD是n-step方法的特例，只使用one-step之后的状态的值函数，因此，时间差分TD也记作。

**1、状态值函数的更新过程**

以**增量式更新**方式计算状态值函数，如Figure 1所示。在一次eposide中，假设取该过程中的一个状态，其值函数为可被称为**估计值**。通过与Environment交互，得到回报和后继状态的估计值，对于当前状态而言，这两个值在局部（短期）内是真实的，因而称为**真实值**。称真实值与估计值的差为**TD偏差**。在偏差和当前状态估计值的基础上，更新当前状态值函数。注：在一次eposide没结束之前，得到状态的更新值仍旧为估计值。同理，当eposide执行到状态时，利用上述过程，不断向前更新状态值函数。

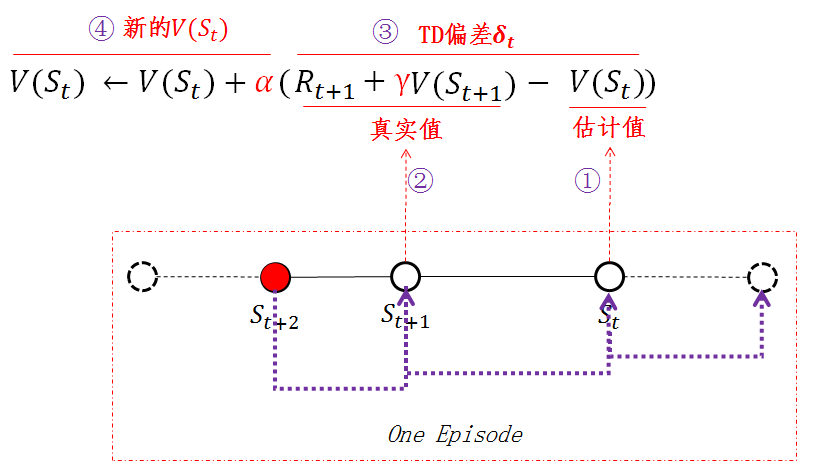


Figure 1 的增量式更新过程

**2、Example：Driving Home**

本文以Driving Home为示例，以为的特例，分别用时间差分TD和蒙特卡罗MC方法来介绍的后向视角和前向视角。

1. **问题背景**

假设每天下班回家时，A都要估计所需时间，它会受到下班时间点（是否是高峰期）、天气、路况等因素的影响。如Table 1所示，A准时18:00下班（Leaving Office），凭借以往的经验估计到家估计30分钟。18:05，A到达停车场（坐上车），并发现正在下雨（Rearch Car, Raining），由于其影响估计35分钟后能到家。十五分钟后，A驾离主干路（Exit Highway），此时估计15分钟后到家。在次干路上行驶10分钟后，发现前方有一辆行驶缓慢的卡车（2ndary Road, Truck），由于道路狭窄无法超车，A只能跟着开车后面行驶，并估计10分钟后到家。10分钟后，A终于驶向小区的街道（Enter Home Street），不再受卡车阻碍的影响，估计3分钟后到家。的确，三分钟后，A顺利到家，实际用时43分钟。

Table 1：Driving Home各状态及用时（分钟）

|  |  |  |
| --- | --- | --- |
| **用时（分）** | **状 态** | **估计用时（分）** |
| 0 | Leaving Office | 30 |
| 5 | Rearch Car, Raining | 35 |
| 15 | Exit Highway | 15 |
| 10 | 2ndary Road, Truck | 10 |
| 10 | Enter Home Street | 10 |
| 3 | Arrive Home | 0 |

Table 1第二列共有6个状态，其中Leaving Office为起始状态，Arrive Home为终止状态。第一列可作为从当前状态转移到后继状态的回报，第三列可作为相应状态的值，该值为估计值。本示例中。

1. **后向视角：时间差分TD方法**

如Figure 2所示，从状态“Exit Highway”转移到状态“2ndary Road, Truck”时，得到，根据Formula（3），依次向前分别更新状态“Exit Highway”、“Rearch Car, Raining”、“Leaving Office”的值函数，如下：







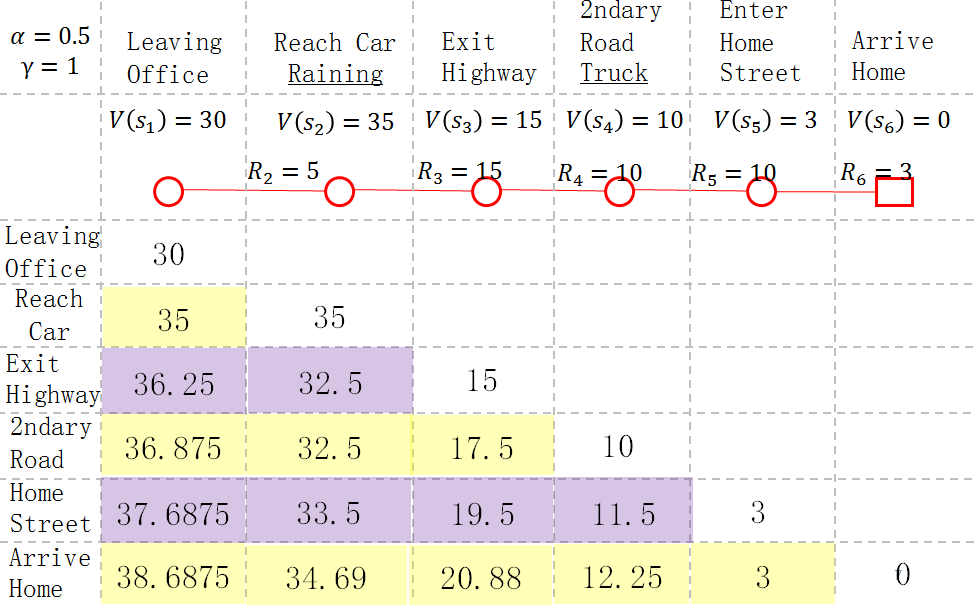


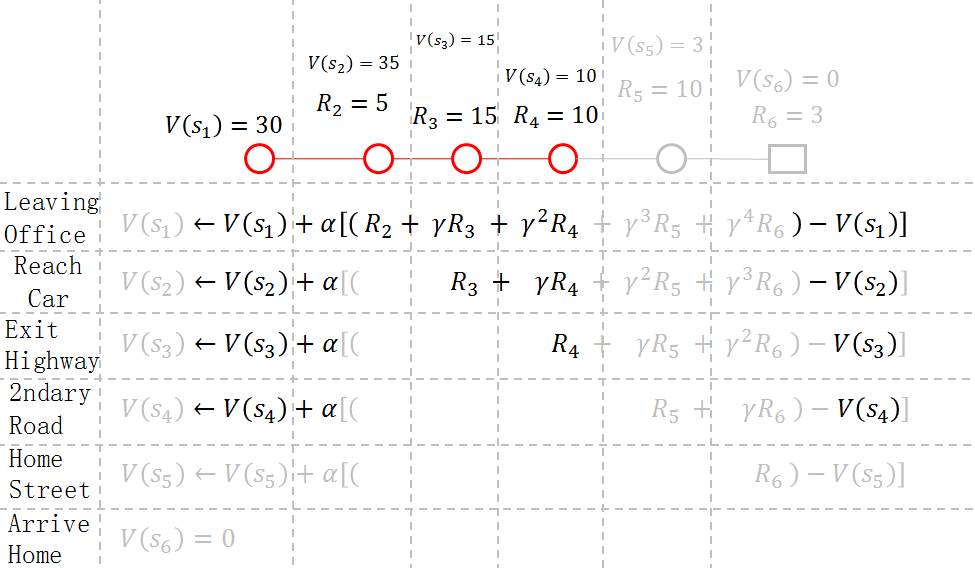
Figure 2 基于TD方法的各状态间转移以及状态值函数的更新过程

1. **前向视角：蒙特卡罗MC方法**

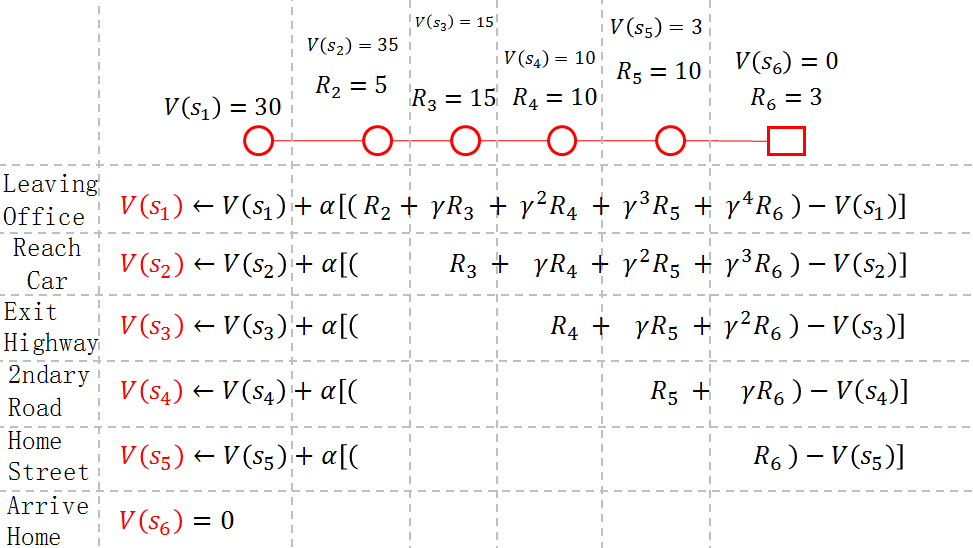
首先给出蒙特卡罗MC方法的值函数更新公式：

 Formula（4）

Figure 3描述了各状态值函数的具体计算公式（部分根据Formula（1）或Formula（2）），灰色字体表示该变量值未知（因暂未转移到相应状态）。（a）从当前状态“Exit Highway”转移到状态“2ndary Road, Truck”时，由于未执行到状态“Enter Home Street”或“Arrive Home”，回报或未知，因而无法估计出当前状态以及其之前状态的值函数。只有当终止状态“Arrive Home”时，才能确定状态“Leaving Office”、“Rearch Car, Raining”、“Exit Highway”、“2ndary Road, Truck”、“Enter Home Street”的值函数。



（a）从状态“Exit Highway”到 “2ndary Road, Truck”时，各变量已知情况



（b）到达终止状态“Arrive Home”时，各变量均可知

Figure 3基于MC方法的各状态间转移以及状态值函数的更新过程

**3、SARSA和Q\_Learning**

SARSA和Q\_Learning算法的原理与相似，都是以增量式方式更新，不同的是，它们用后继<状态，动作>值函数来更新当前<状态，动作>值函数，如Figure 4所示。

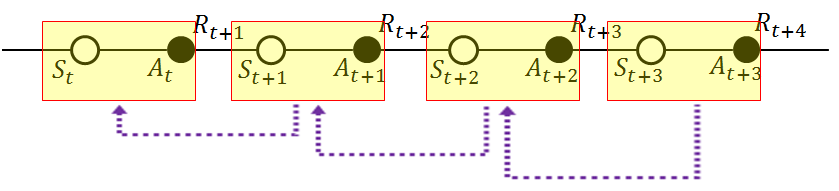


Figure 4 SARSA和Q\_Learning算法的更新方式

SARSA和Q\_Learning的更新公式分别为Formula（5）和Formula（6）。

 Formula（5）

 Formula（6）

SARSA和Q\_Learning算法选取的策略应能够满足“利用和探索”要求，常用的策略有（Formula（7））、贪婪策略（Formula（8））等。平衡了利用和探索，选取动作值函数最大的部分为利用，其他非最优动作仍有概率作为探索部分。

 Formula（7）

 Formula（8）

1. **SARSA是一种on-policy算法**

|  |
| --- |
| **Algorithm 1**：SARSA (on-policy TD Control) for estimating |
| Parameters: step size , small  Initialize , for all , , arbitrarily except that  Loop for each episode:  Initialize  Choose  from  using policy derived from  (e.g., )  Loop for each step of episode:  Take action , observe ,  Choose  from  using policy derived from  (e.g., )      Until  is Terminal |

SARSA的策略评估和策略改善阶段使用相同的策略（如，），其算法（Algorithm 1）执行过程如Figure 5所示。在一次episode中，SARSA初始化状态，并利用策略选择动作。在每一步中，SARSA执行动作，并产生回报和后继状态，利用策略选择动作后，根据Formula（5）用更新，同时将后继状态更新为当前状态，后继动作更新为当前动作。重复执行上述过程，直到最终状态（Terminal）为止。

从Figure 5可以看出，**SARSA在每一步更新值的同时已经选择好下一步要执行的动作，意味着基本上决定了未来的状态**。

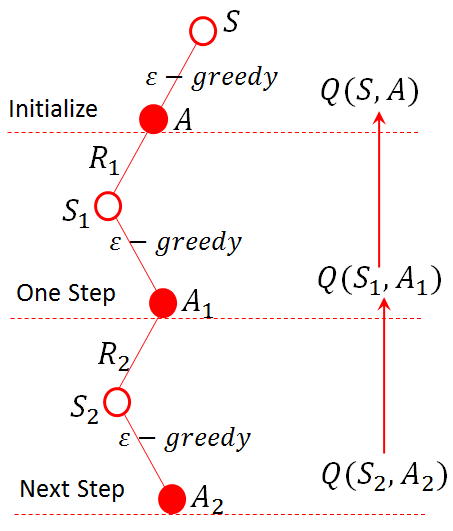


Figure 5 SARSA算法的执行过程

1. **Q\_Learning是一种off-policy算法**

|  |
| --- |
| **Algorithm 2**：Q\_Learning (off-policy TD Control) for estimating |
| Parameters: step size , small  Initialize , for all , , arbitrarily except that  Loop for each episode:  Initialize  Loop for each step of episode:  Choose  from  using policy derived from  (e.g., )  Take action , observe ,      Until  is Terminal |

Q\_Learning的行动策略（如）和目标策略（如贪婪策略）使用不同的策略，其算法（Algorithm 2）执行过程如Figure 6所示。在一次episode中，Q\_Learning初始化状态。在每一步中，Q\_Learning利用策略选择动作并执行，产生回报和后继状态，此时为策略评估阶段。在策略改善阶段，Q\_Learning在状态下利用贪婪策略选择使得值最大的动作，根据Formula（6），用更新，同时将后继状态更新为当前状态。重复执行上述过程，直到最终状态（Terminal）为止。

从Figure 6可以看出，**Q\_Learning在每一步更新值时，仅仅选取的是使得值最大化的方向，并不执行动作。当要确定未来的状态时，再根据等策略重新选择动作**。

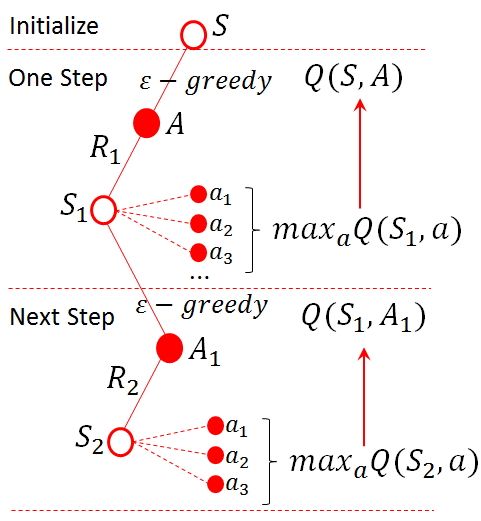


Figure 6 Q\_Learning算法的执行过程

1. **SARSA和Q\_Learning对比**

Figure 7描述了SARSA和Q\_Learning算法的异同。初始时处于状态，根据策略选择了动作（往右走），自然而然地转移到状态，SARSA和Q\_Learning在这一过程是相同的（红线所示）。接下来，Q\_Learning算法首先计算在状态下最大的值，但在真正执行动作时会“三思而后行”，不一定会执行那个使得值最大的动作，仅仅是估计了未来的动作值（蓝线所示）。SARSA却相反，它是“言必行，行必果”，评估的动作也是它未来要执行的动作（紫线所示）。

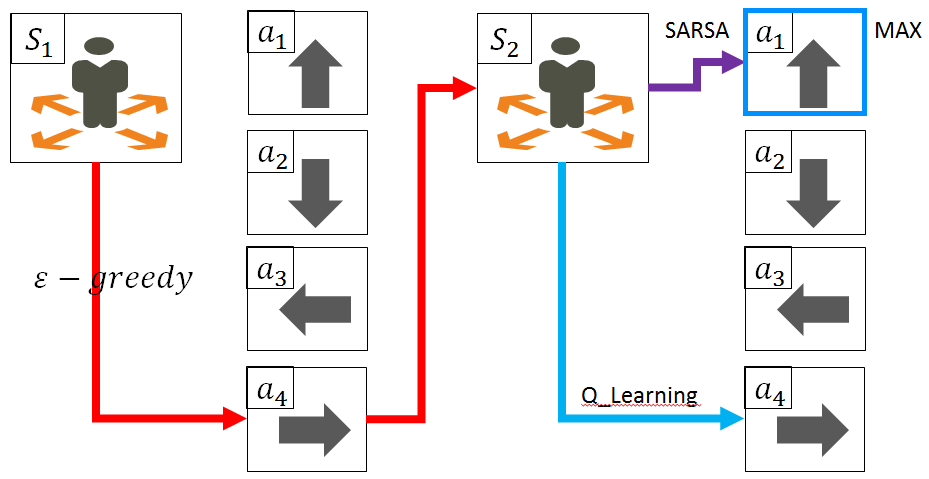


Figure 7 SARSA和Q\_Learning算法对比示例

**4、Example：Cliff Walking**

Cliff Walking描述的是一个在悬崖边走路的场景（Figure 8），目的是从入口以最少的步数走到出口。离悬崖边越远的路径就越安全，蓝色线是一条安全的路径（Safer path），红色线则是一条危险但最优的路径（Optimal path）。每个小格表示一个状态，动作有上下左右，白色方格的回报均为-1；一旦落入灰色区域，回报为-100，并重新开始。

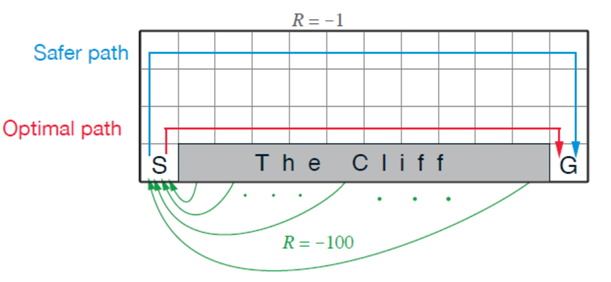
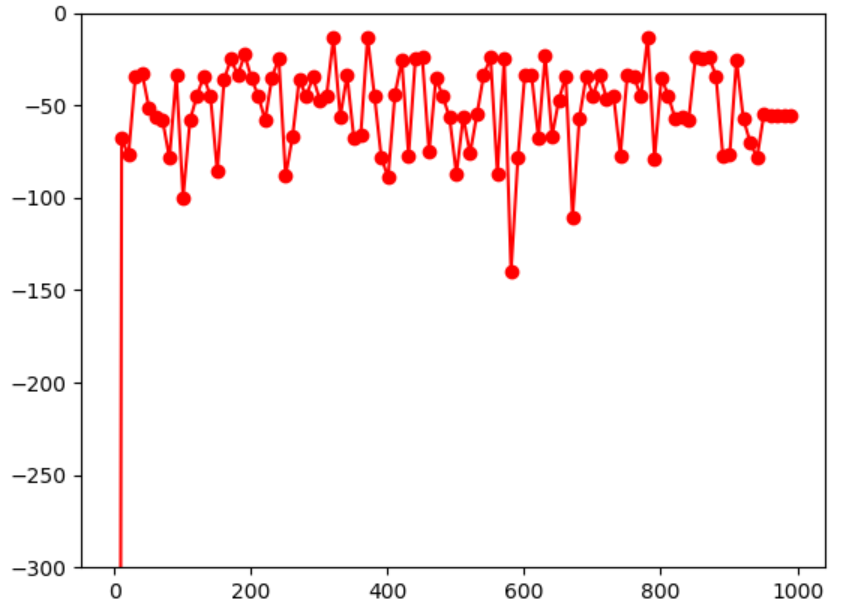
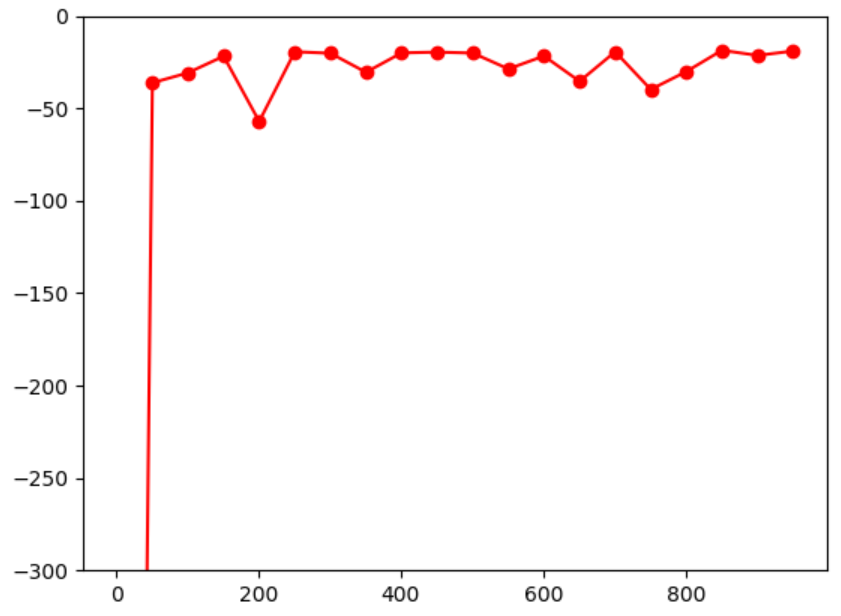
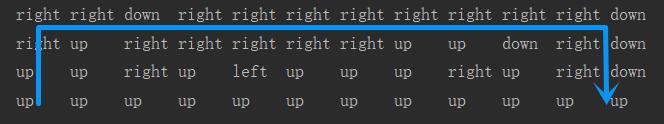


Figure 8 Cliff Walking

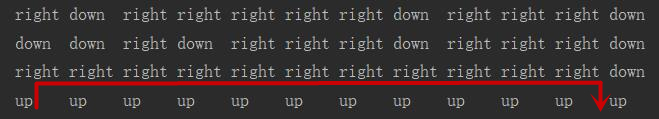
SARSA算法和Q\_Learning算法均迭代10次，每次使用1000个episode，对10次迭代的每个episode的reward取均值，并按照每隔10个episode采样值分别绘制其reward图（分别如Figure 9 (a)和(b)所示）。SARSA算法和Q\_Learning算法收敛后的Policy分别如Figure 9 (c)和(d)所示。



1. (b)



(c)



(d)

Figure 9 SARSA算法(a)和Q\_Learning算法(b)的reward图

SARSA算法(c)和Q\_Learning算法(d)收敛后的Policy

Q\_Learning可以收敛到Optimal path而SARSA是Safe path，是因为Q\_Learning的目标策略是贪婪策略，保证了Agent在值进入收敛后不会也不可能再记录可能掉入悬崖的状态动作的值，也可以说是Off-Policy类的控制方法并不会受到贪婪策略无探索性的影响，所以才能够产生Optimal Policy。

而SARSA算法使用的策略来更新值，即Agent总是有一定的概率选择非最优的动作，其中掉入悬崖的动作也始终有一定的概率被选中，并在函数更新时被记录下来，所以整个Grid的函数变成了越靠近悬崖，值越小的分布，最终导致了SARSA选择的是Safe path。

Q\_Learning的目标策略是基于值的贪婪策略，在学习探索过程中决定Agent的行动策略和SARSA同是策略。进入收敛后，由于Q\_Learning的Agent每次选择的都是Optimal path且又因为动作策略具有一定的探索性，所以会有一定的概率选择掉入悬崖的动作，虽然这些动作产生的值并不会被更新记录。SARSA的Agent进入收敛后，基本上选择的是Safe pass，掉入悬崖的概率比Q\_Learning要小很多。所以，SARSA算法平均每个episode获得的Reward值通常要比Q\_Learning更接近实际的步数乘上-1，也就是说SARSA的在线学习的效果比Q\_Learning更好。