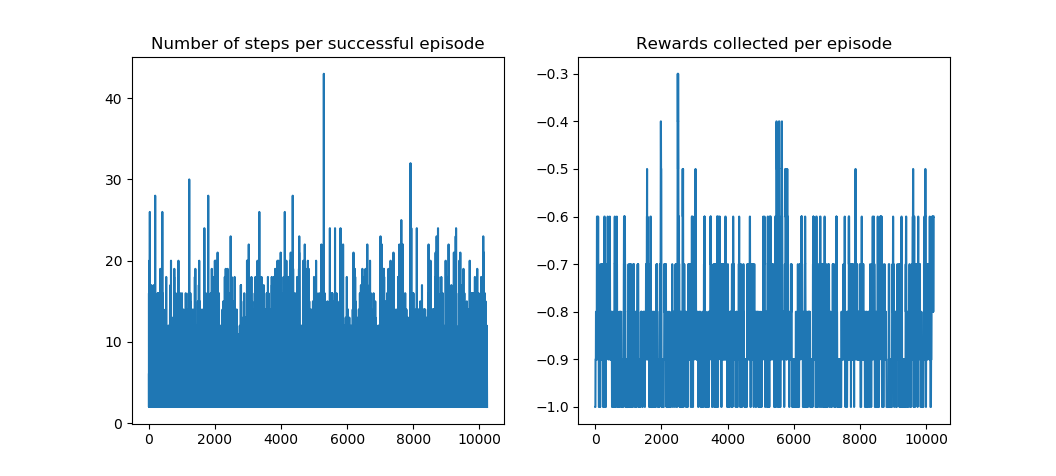
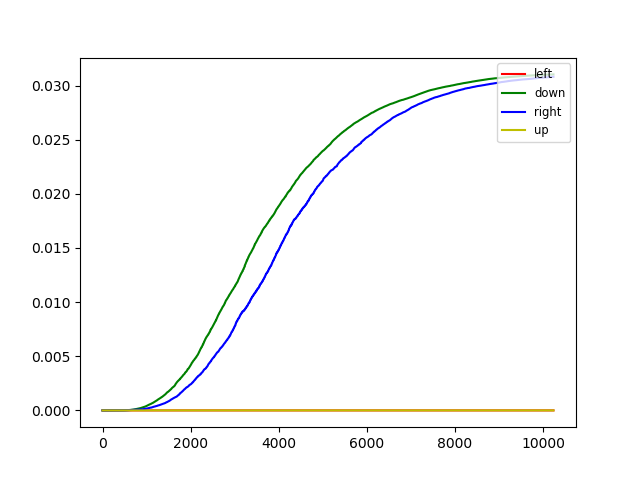
## 1，On-vs Off-policy

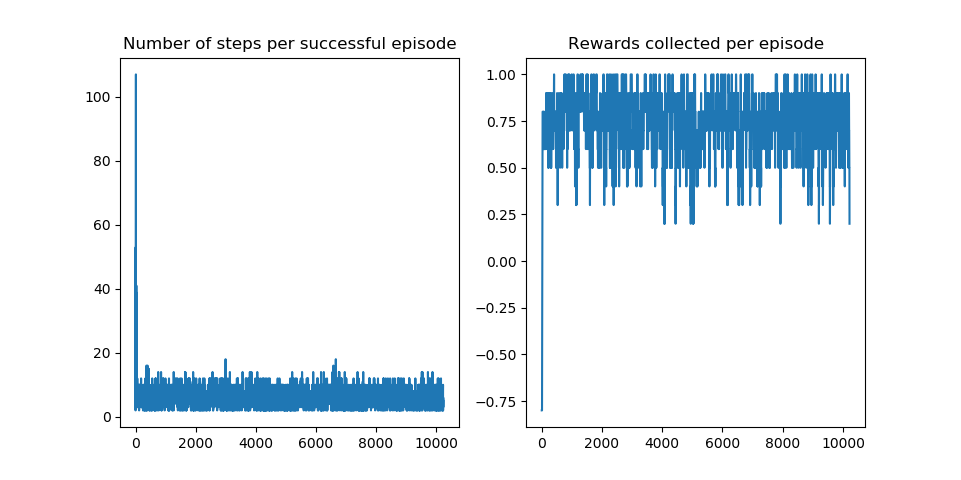
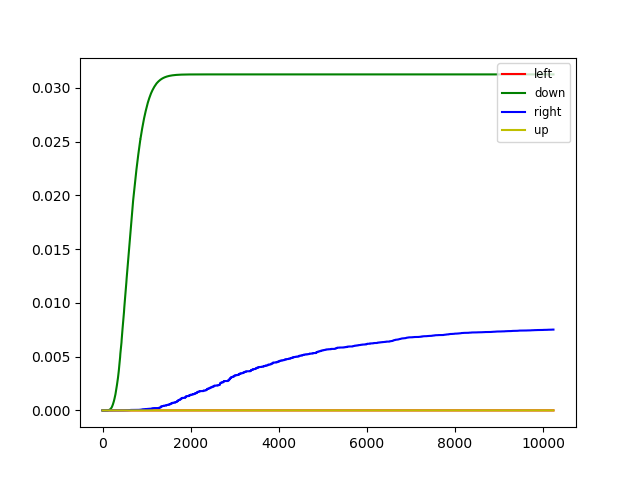
* **RL Algorithm: Q-learning; ε = 0.8.**



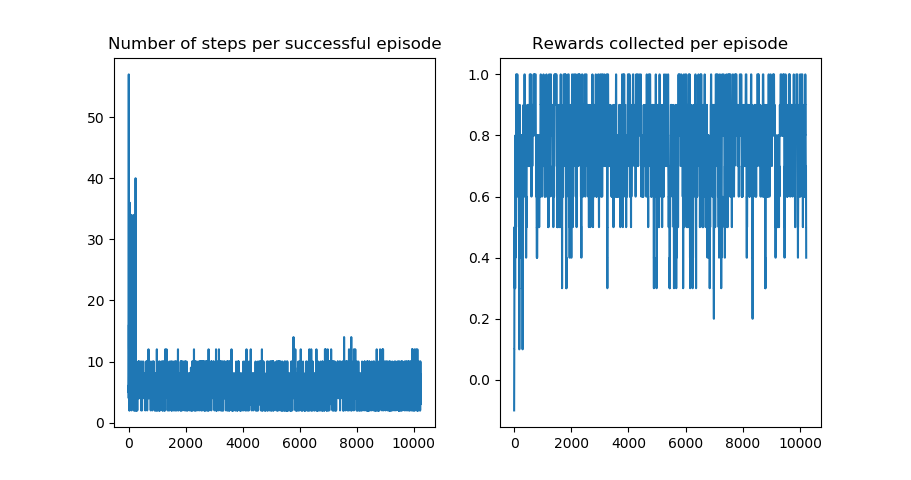
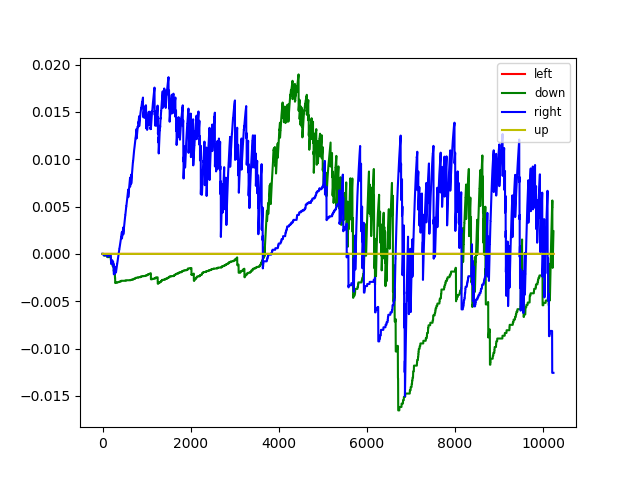
* **RL Algorithm: Sarsa; ε = 0.8.**

运行不出结果。

* **RL Algorithm: Q-learning; ε = 0.1.**



* **RL Algorithm: Sarsa; ε = 0.1**



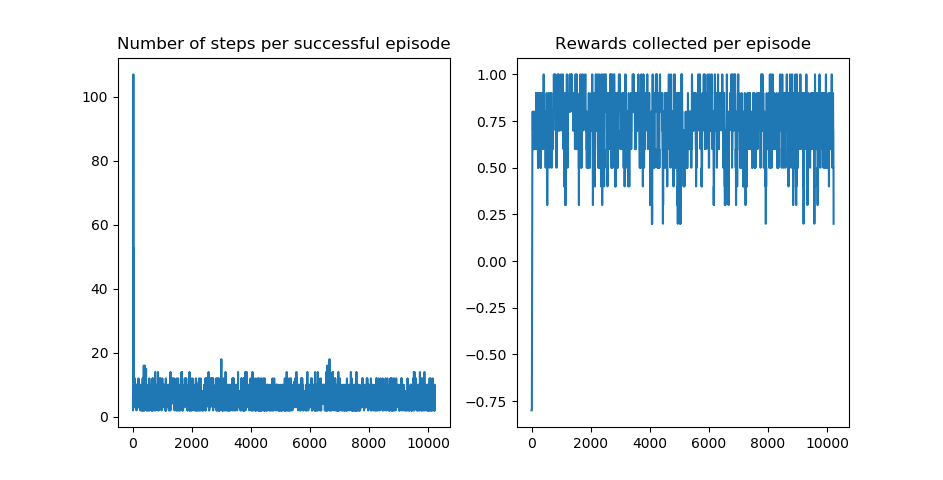
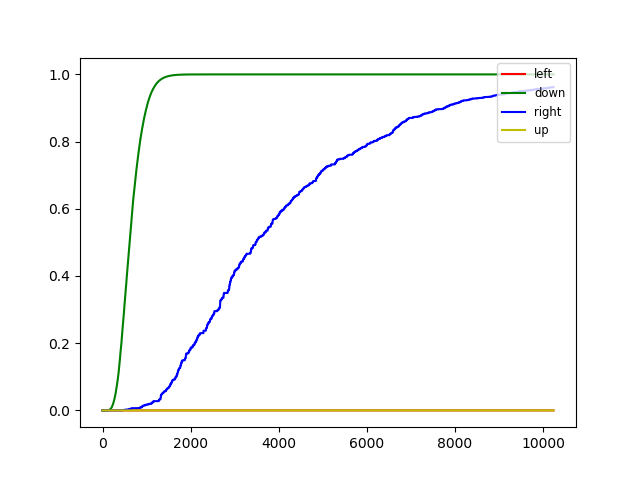
Q-learning在每一步TD中贪心的获取下一步最优的状态动作值函数。而Sarsa则是e-greedy的选取TD中的下一个状态动作值函数。在这种情况下，Q-learning更倾向于找到一条最优policy，而Sarsa则会找到一条次优的policy。这是由于Sarsa在TD误差中随机的选取下一个状态动作值函数，这样可能会使整体的状态值函数降低。

**on-policy**：生成样本的policy（value function）跟网络更新参数时使用的policy（value function）相同。典型为**SARAS**算法，基于当前的policy直接执行一次动作选择，然后用这个样本更新当前的policy，因此生成样本的policy和学习时的policy相同，算法为on-policy算法。该方法会遭遇探索-利用的矛盾，只利用目前已知的最优选择，可能学不到最优解，收敛到局部最优，而加入探索又降低了学习效率。epsilon-greedy 算法是这种矛盾下的折衷。优点是直接了当，速度快，劣势是不一定找到最优策略。

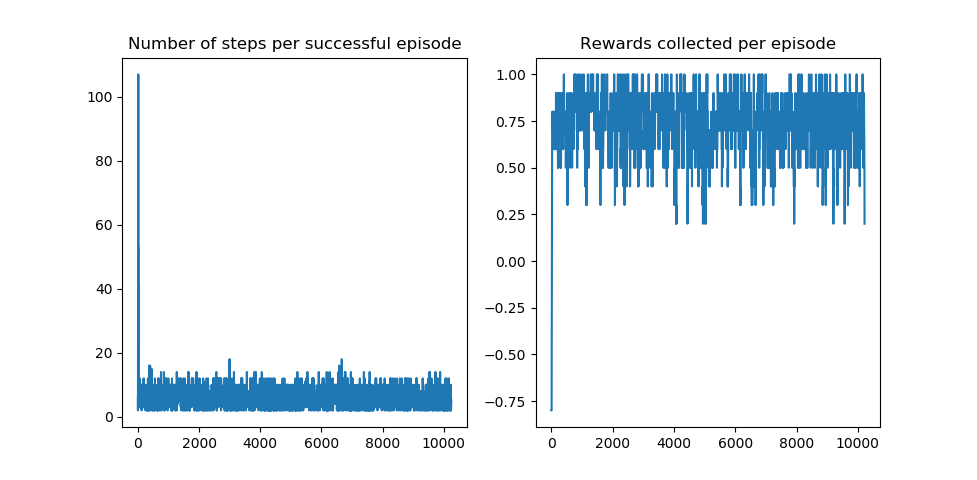
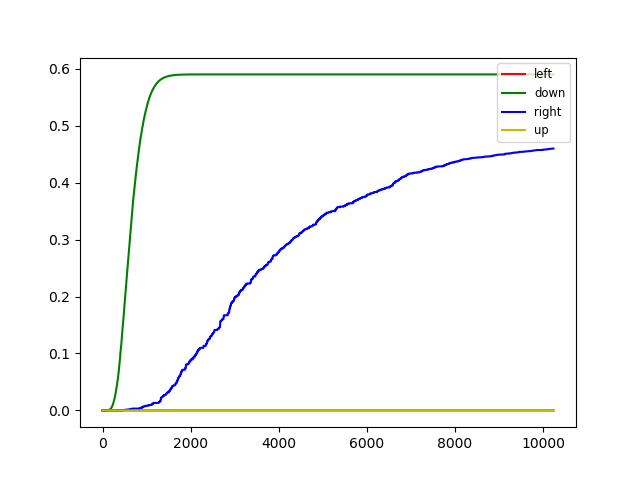
**off-policy**：生成样本的policy（value function）跟网络更新参数时使用的policy（value function）不同。典型为**Q-learning**算法，计算下一状态的预期收益时使用了max操作，直接选择最优动作，而当前policy并不一定能选择到最优动作，因此这里生成样本的policy和学习时的policy不同，为off-policy算法。先产生某概率分布下的大量行为数据（behavior policy），意在探索。从这些偏离（off）最优策略的数据中寻求target policy。当然这么做是需要满足数学条件的：假設π是目标策略, µ是行为策略，那么从µ学到π的条件是：π(a|s) > 0 必然有 µ(a|s) > 0成立。两种学习策略的关系是：on-policy是off-policy 的特殊情形，其target policy 和behavior policy是一个。劣势是曲折，收敛慢，但优势是更为强大和通用。其强大是因为它确保了数据全面性，所有行为都能覆盖。

## 2，折扣因素

* **γ = 1.**



* **γ = 0.9.**



* **γ = 0.**

运行不出结果。

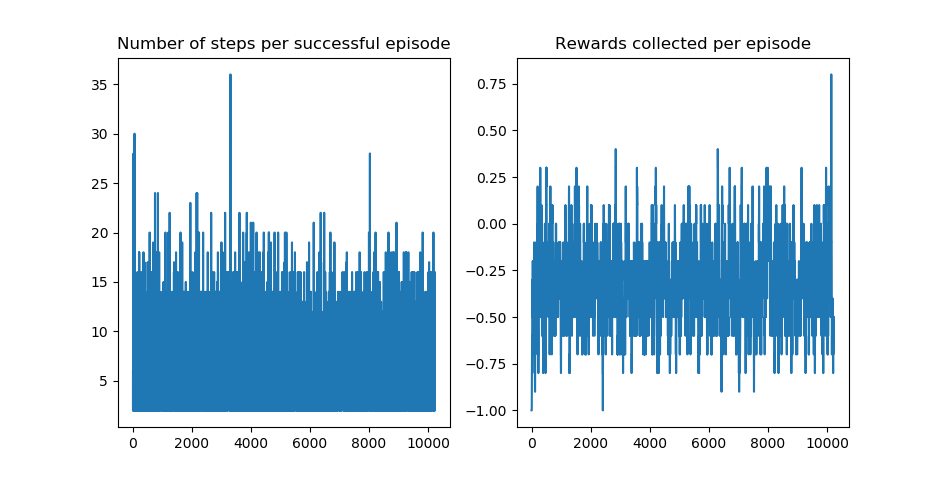
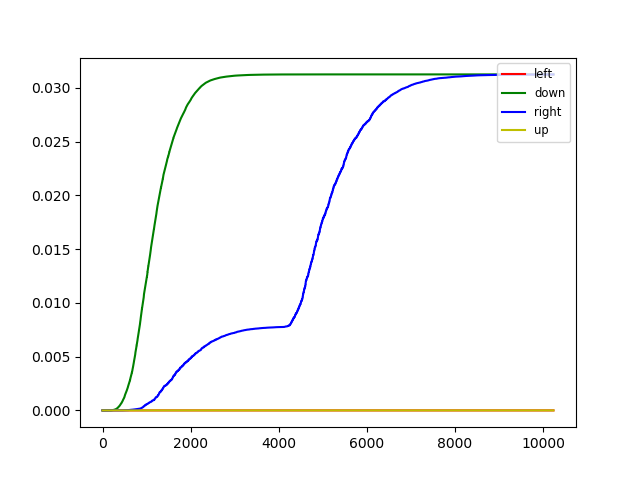
Q Learning算法利用下面公式进行off-policy学习，即用公式来表示Q-Learning中Q表的更新：

Q(St,At)←Q(St,At)+α[Rt+1+γmax Q(St+1,a)−Q(St,At)]

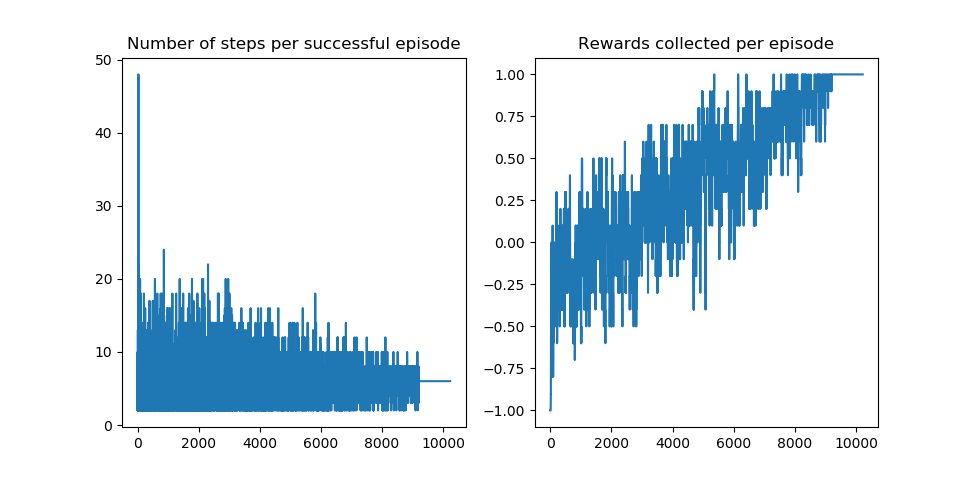
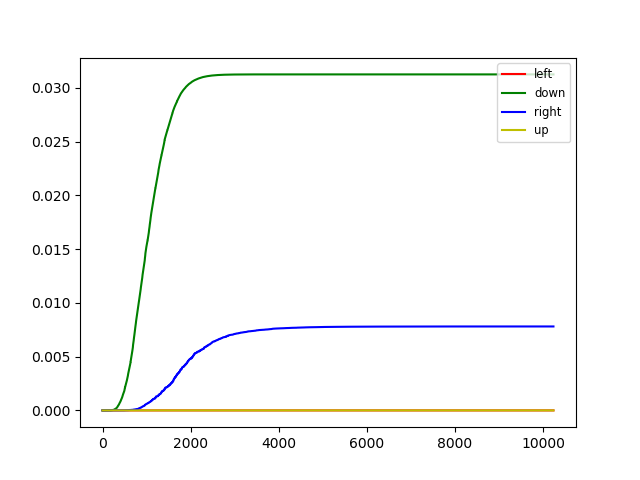
**γ为折扣因子**，0<= γ<1，γ=0表示立即回报，γ趋于1表示将来回报，γ决定时间的远近对回报的影响程度，表示牺牲当前收益，换取长远收益的程度。将累计回报作为评价策略优劣的评估函数。当前的回报值以及以前的回报值都可以得到，但是后续状态的回报很难得到，因此累计回报就难以计算。而Q-learning用Q函数来代替累计回报作为评估函数，正好解决这个问题。

## 3，环境随机性

* **Deterministic environment.---确定性环境。**



* **Stochastic environment.---随机环境。**



在我们探索环境（environment）之前，Q-table 会给出相同的任意的设定值（大多数情况下是 0）。随着对环境的持续探索，这个 Q-table 会通过迭代地使用 Bellman 方程（动态规划方程）更新 Q(s,a) 来给出越来越好的近似。

我们指定一个探索速率「epsilon」，一开始将它设定为 0.5。这个就是我们将随机采用的步长。在一开始，这个速率应该处于最大值，因为我们不知道 Q-table 中任何的值。这意味着，我们需要通过随机选择动作进行大量的探索。生成一个随机数。如果这个数大于 epsilon，那么我们将会进行「利用」（这意味着我们在每一步利用已经知道的信息选择动作）。否则，我们将继续进行探索。在刚开始训练 Q 函数时，我们必须有一个大的 epsilon。随着智能体对估算出的 Q 值更有把握，我们将逐渐减小 epsilon。