5 Model-Free Control

上个章节，我们讨论了如何仅在环境交互的情况也就是不清楚环境是如何运作的条件下去估计给定策略的价值。我们学习到了Model-free policy evaluation方法:Monte Carlo policy evaluation和TD Learning。在本讲座中，我们将讨论model-free control，在相同的约束条件下学习良好的策略(只有交互，不知道奖励的结构或转移概率)。该框架在两种类型的领域中很重要:

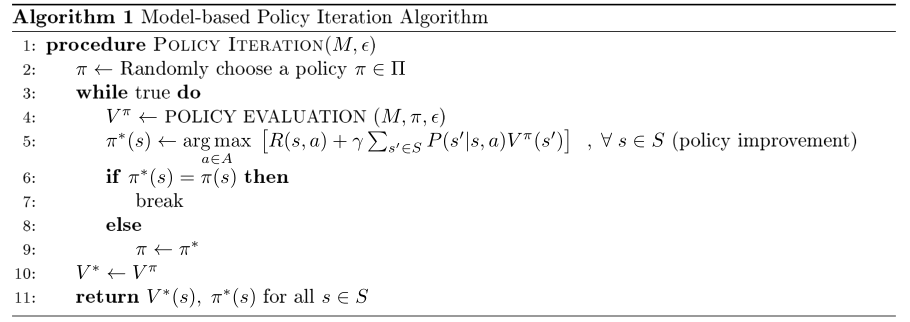
1.当MDP模型未知时，我们通过MDP采集大量样本(sample trajectories)或者

2.当MDP模型已知时,由于域的大小我们通过基于模型的控制方法计算值函数是不可行的。

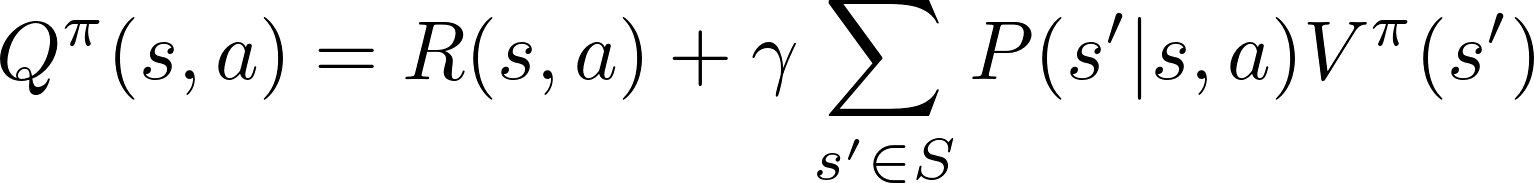
在本章节中，我们仍将局限于表格法，我们可以将每个state或state-action的值表示为查找表的一个元素。在下一章节中，我们将基于值函数近似的条件下重新研究今天和上一章节的算法，其中涉及尝试将一个函数转换为state或state-action值函数。

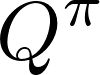
5.1 Generalized Policy Iteration

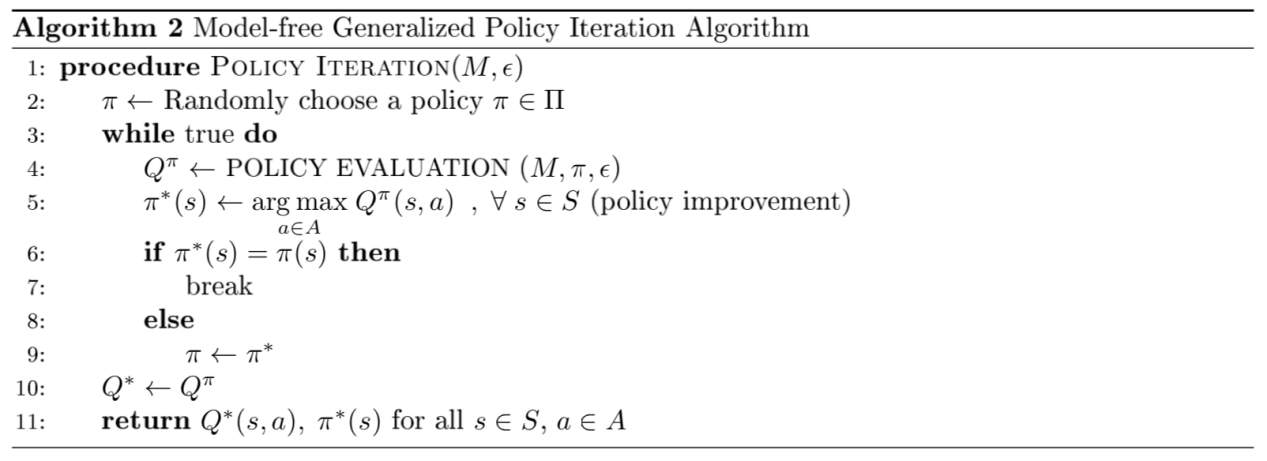
首先回顾我们基于模型的策略迭代(model-based policy iteration)算法，为了方便起见，我们在算法1中重写了该算法:



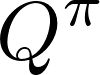
上个章节，我们介绍了无模型策略评估(model-free policy evaluation)方法，因此我们可以用model-free的方式完成第4行。然而，为了使整个算法是model-free的，我们还必须找到一种方法，以model-free的方式执行第5行，即策略改进步骤。根据定义我们有:



因此,我们能够通过代入的值到model-based policy iteration算法得到一个model-free policy iteration算法(Algorithm 2)，并且始终使用state-action，该算法如下:



由于我们在第5行进行了替换，因此对该算法有一些警告：

1. 如果policy π是已知的(确定的)或者每次行动a都没有使用正数的概率采取行动，那么实际上我们是无法计算第五行的argmax的。
2. Policy evalutaion algorithm给我们一个估计值，所以不清楚第五行代码是否会像model-based的情况那样一直改善我们的策略(可能有时会变得糟糕?)。

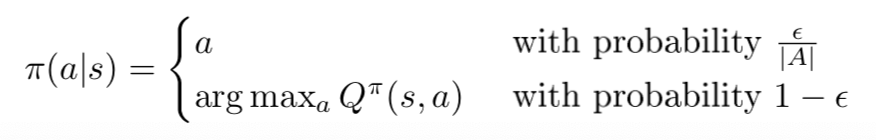
5.2 Importance of Exploration

### 5.2.1 Exploration

在上一个小节中，我们看到一个关于model-free policy iteration算法的警告，是说策略π的每一个行动的概率都要求不小于0，所以每个state-action pair的值都是可以确定的。换句话说，策略π需要对动作进行探索，即使这个探索的结果对我们当前Q值的估计可能不是最优的。

### 5.2.2 ε-greedy Policies

为了探索相对于我们当前的Q值估计而言的次优行为,我们需要一个系统的方法来平衡次优行为的探索和最优或贪婪行为的选择。一个很自然的策略就是采取概率很小的随机行动，而其余时间选择选择贪心算法。这种探索策略被称为ε-greedy policy。数学上，ε-greedy policy是关于state-action值wpsoffice的函数，它有如下形式:

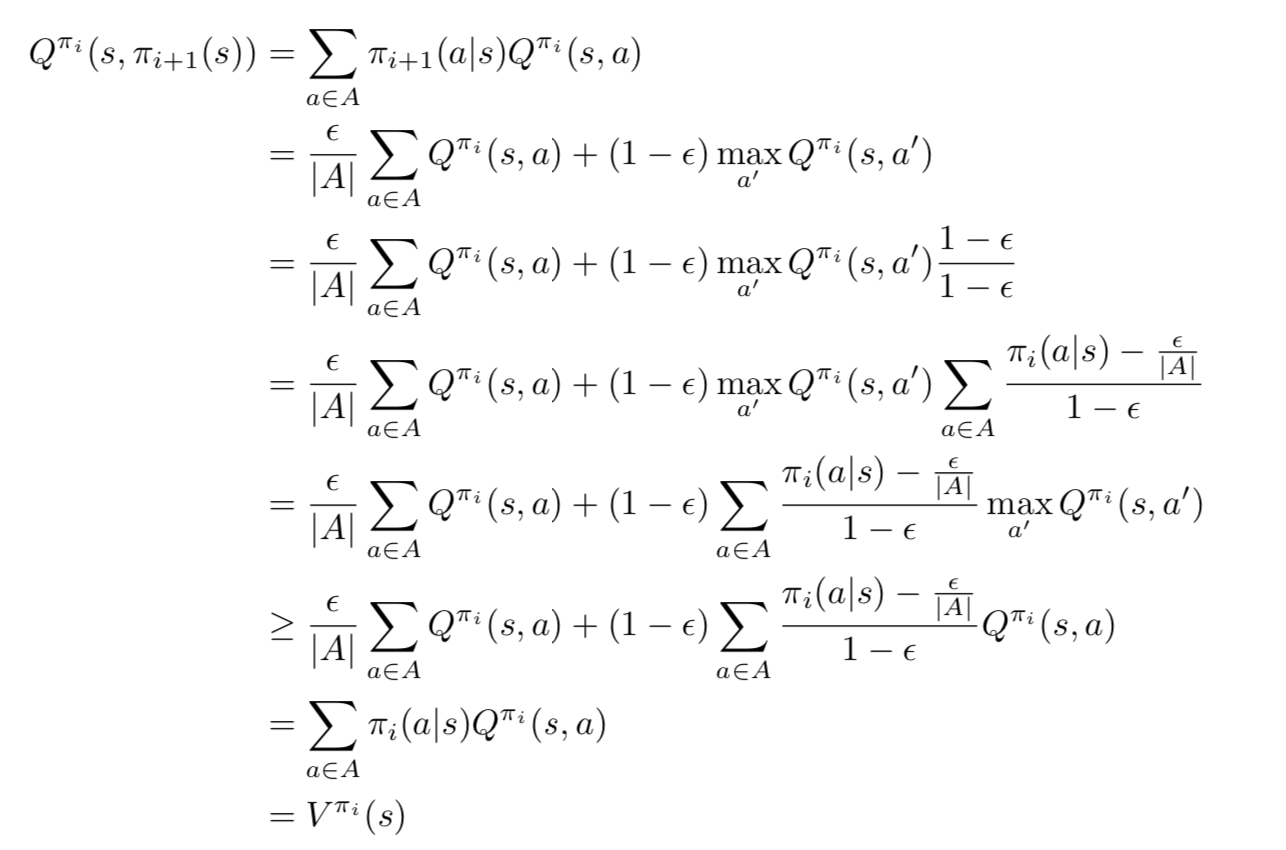


### 5.2.3 Monotonic ε-greedy Policy Improvement

通过策略改进定理，在第二课中我们看到，如果对当前值采取贪婪的行为，然后遵循策略π，则此策略是对策略π的改进。那么一个很自然的问题就是，关于ε-greedy policy π的ε-greedy policy是否是对π真正的进行了改善。这帮助我们定位了对于generalized policy iteration算法的第二个警告。幸运的是，有一个关于ε-greedy policy的策略改进定理的类似定理，我们在下面陈述和推导。

定理5.1(Monotonic ε-greedy Policy Improvement).令wpsoffice为ε-greedy policy.则关于wpsoffice的ε-greedy policy,定义/var/folders/4_/wsyh4np55ms9lgn8jhhpylgr0000gp/T/com.kingsoft.wpsoffice.mac/wpsoffice.tzc655wpsoffice是基于wpsoffice的Monotonic improvement。换句话说:wpsoffice。

证明:我们首先证明对于所有的状态s,都能得到:wpsoffice:



第一个等式源于这样一个事实，即我们采取的第一个动作是关于策略，然后我们遵循策略。接下来是第四个等式成立是因为:



根据策略改进定理，我们可以得出:意味着 for all state s。

因此，monotonic ɛ-greedy policy improvement向我们表明，如果我们对当前ɛ-greedy policy采取ɛ-greedy行动，我们的策略实际上确实是会改进。

### 5.2.4 Greedy in the limit of exploration

目前我们介绍了ɛ-greedy，它是作为一种朴素的方式来对新状态的探索(或者说新的行动)和对现有知识利用之间的平衡；然而，我们可以通过引入一种新的探索策略来进一步改善这种平衡，这种策略允许我们对算法做出收敛保证。这类策略被称为Greedy in the Limit of Infinite Exploration(GLIE)。

定义5.1(Greedy in the Limit of Infinite Exploration(GLIE)).如果满足以下两个属性，策略π is greedy in the limit of infinite exploration :

1.所有的state-action pairs都被访问了无数次，例如,对于所有的:



其中,是在episode i状态s执行动作a的次数。

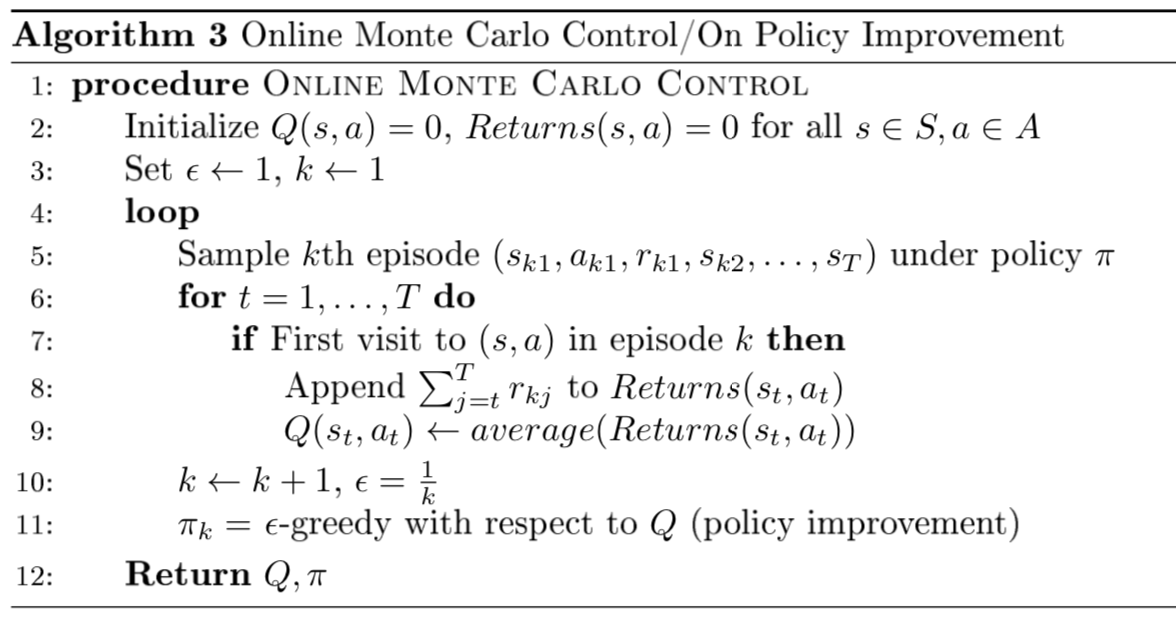
2.(The behavior policy converges to the policy that is greedy with respect to the learned Q-function)行为策略收敛于学习到的Q-function的贪婪策略。即:



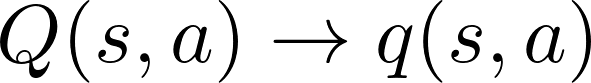
GLIE策略的一个例子是一个ɛ-greedy policy，其中ɛ会慢慢被衰减到零,,其中i是episode数,我们可以看到，由于 for all i，因此我们将在每个时间步以一定的正概率进行探索，从而满足第一个GLIE条件。因为等于，所以我们也知道该策略在极限内是greedy的，因此满足第二个GLIE条件。

5.3 Monte Carlo Control

现在，我们将把上面描述的探索策略与上节课的model-free policy evaluation算法结合起来，给出一些model-free control方法。我们讨论的第一个算法是Monte Carlo online controle算法。在算法3中，我们给出了first-visit online Monte Carlo Control算法。如果在算法3的第七行检测是否为第一次访问给删除，就变成了every-visit control。



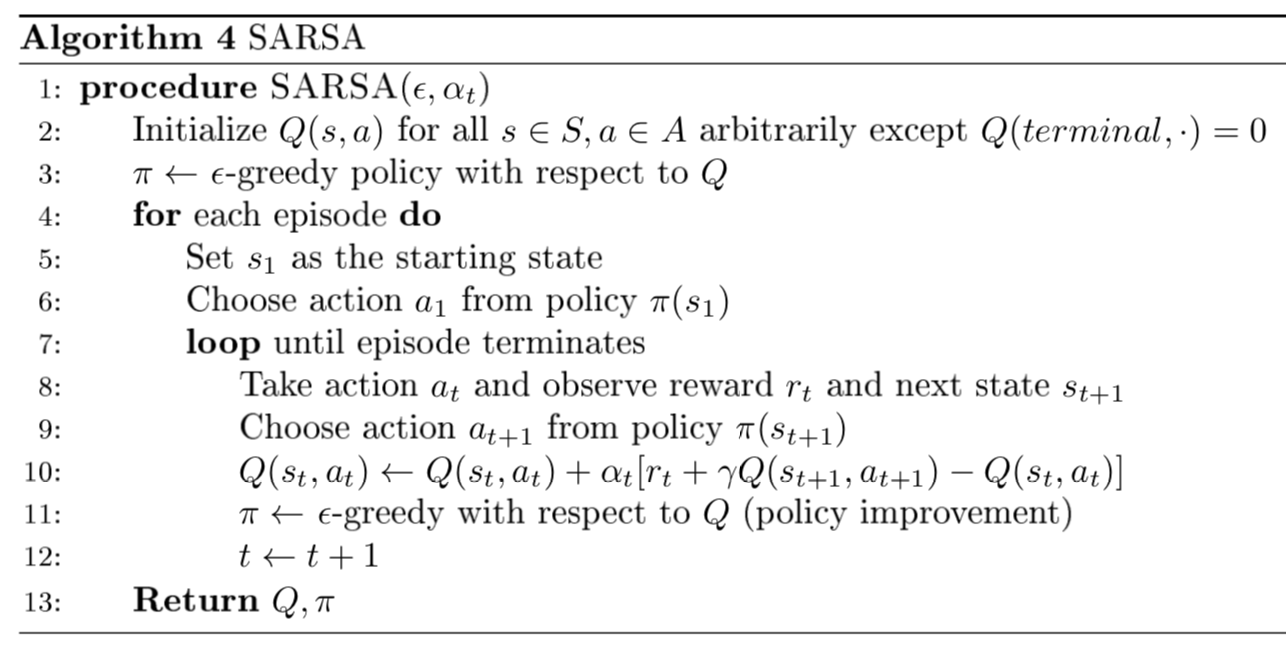
现在，如前所述，GLIE策略可以帮助我们实现无模型控制方法的收敛保证。特别是，我们有以下结果:

**定理5.2**.GLIE Monte Carlo control 收敛域optimal state-action value function。即:

换句话说，如果算法3中使用的ε-greedy policy是GLIE，那么从该算法导出的Q值将收敛到最优的Q函数。

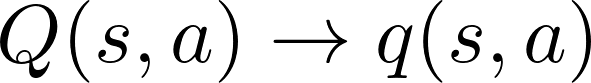
5.4 Temporal Difference Methods for Control

上一个章节，我们已经介绍了TD(0)作为求解model-free policy evaluation的一个方法。现在，我们可以在TD(0)的基础上，用我们的探索思想来获得TD-style的model-free control。有两种方法: on-policy和off policy。我们首先介绍基于on-policy的算法4，因为算法4使用(s,a,r,s’,a’)的值所以被称为SARSA。

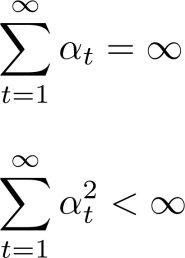


SARSA是一个on-policy方法，因为在方程式中更新所使用的操作a和a'都是来自即时更新时遵循的策略。

就像在Monte Carlo中一样，如果步长上新增一个额外的条件，我们可以得出SARSA的收敛，如下命题所示:

**定理5.3**.如果满足下列两个条件，对于finite-state和finite-action的马尔可夫决策过程,SARSA将会收敛到最优action-value，例如::

1. policy π遵循GLIE(The sequence of policies π from is GLIE)
2. The step-size wpsoffice 满足Robbins-Munro 序列:



练习5.1。 算法4第11行中的每次更新之后执行策略改进步骤有什么好处？ 减少执行政策改进步骤的频率好处是什么？

每次都执行改进的好处是每次都改进策略，更快收敛。

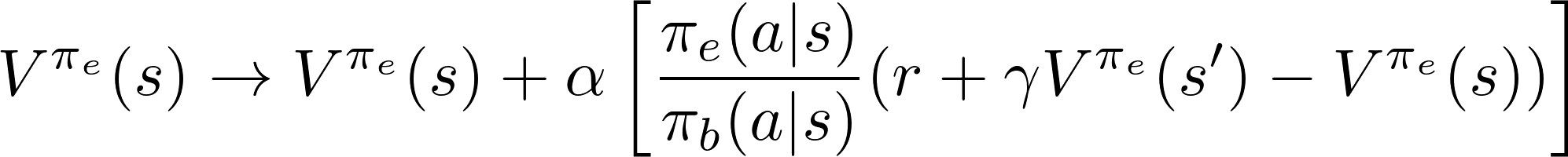
减少改进频率的好处是减少运行的次数，提高效率。

5.5 Importance Sampling for Off-Policy TD

在深入学习off-policy TD-style control之前，让我们先回退一步,看一种off-policy TD policy evaluation的方法。回顾一下TD更新的公式:



现在，让我们假设，就像在off-policy Monte Carlo案例中一样，我们有来自策略的数据，并且我们想要估计策略的值。然后，就像在Monte Carlo policy evaluation案例中一样，我们可以通过行为策略中看到样本的概率与通过重要性抽样评估的策略的比率来加权目标。然后，新的更新采用以下形式:

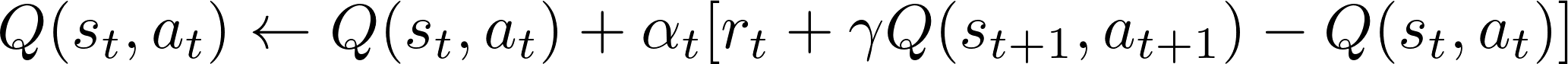


请注意，因为我们只使用一个轨迹样本，而不是像蒙特卡罗那样对整个轨迹进行采样，所以我们只从一个步骤中引入似然(likelihood)比。出于同样的原因，该方法的方差也明显低于蒙特卡罗方法。

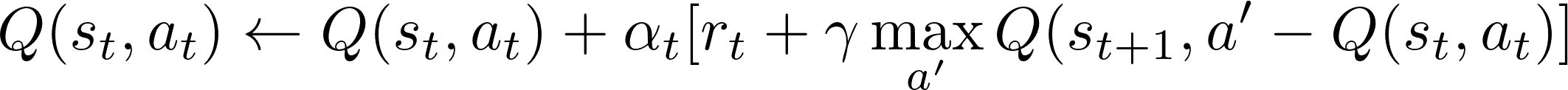
此外，不需要在每一步都相同，但是我们需要知道每一步的概率。就像在蒙特卡洛一样，我们需要这两个策略得到同样的支持。也就是说，如果,那么。

5.6 Q-Learning

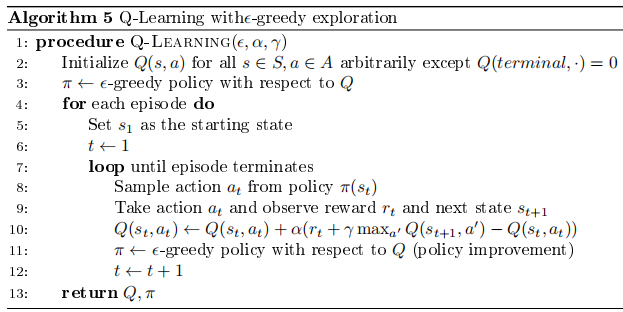
现在，我们回到为TD-style control寻找一个off-policy方法。在上面的公式中，我们再次利用了重要性抽样，但是在控制情况下，我们不需要依赖它。取而代之的是，我们可以保持state-action Q估计值，并引导未来最佳动作的值。我们的SARSA更新采取了以下形式:



但是我们可以基于下一个状态引导Q值，以获得以下更新:



这就产生了Q-Learning，这在算法5中有详细描述。现在我们对下一个状态的动作取最大值，这一行动不一定与我们从当前策略中得出的行动相同。因此，Q-Learning被认为是一种off-policy算法。



6 Maximization Bias

最后，我们将讨论一种被称为最大化偏差(**maximization bias**)的现象。我们将首先在一个小例子中考察最大化偏差。

6.1 Example: Coins

假设有两个相同(公平)的硬币，但是我们不知道它们是否是相同的(或者说个公平的)。如果硬币正面朝上，我们得到一美元，如果硬币反面朝上，我们损失一美元。我们想回答以下两个问题:

1.哪种硬币会给未来的翻转带来更多的钱(哪种硬币未来的期望收益更高)？

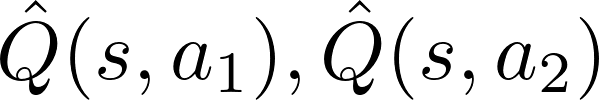
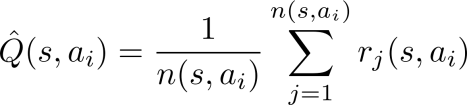
2.使用问题1中的硬币，我每次掷硬币能期望赢/输多少？

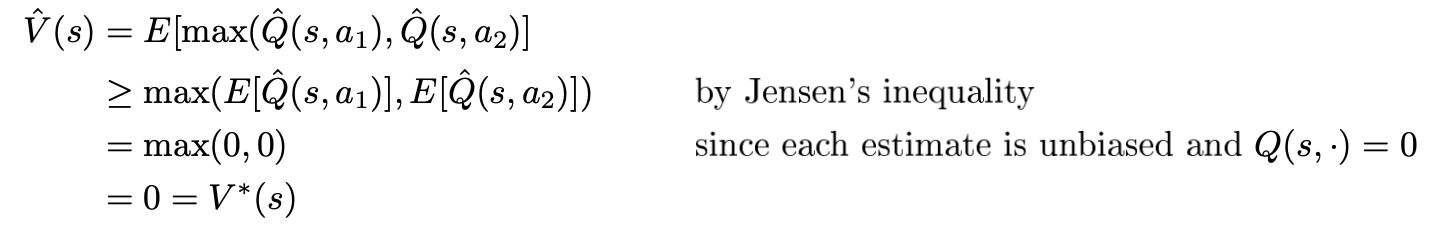
为了回答这个问题，我们每枚硬币投一次。然后我们选择能产生更多钱的硬币作为问题1的答案。不管硬币给了我们多少，我们都回答了问题2。例如，如果硬币1落在正面，硬币2落在反面，我们会用硬币1回答问题1也就是用一美元回答问题1。

让我们检查一下这个过程的可能结果。如果至少有一枚硬币是头像，那么我们对问题2的回答是一美元。如果两枚硬币都是尾部，那么我们的答案是负一美元。因此，我们对问题2的回答的期望值是(3/4)×1+(1/4)×-1 = 0.5。这使我们对投出更好硬币的期望值的估计高于投出该硬币的真实期望值。换句话说，我们会系统地认为硬币比它们实际上更好。

这个问题来自于这样一个事实，我们用我们的估计来选择更好的硬币并估计它的价值。我们可以通过分开这两个步骤来缓解这种情况。这样做的一个方法是改变程序如下:在选择了更好的硬币后，再次翻转更好的硬币，并使用这个值作为问题2的答案。这个答案的期望值现在是0，这与投硬币的真实期望值相同。

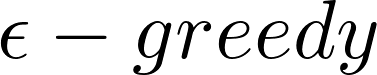
6.2 Maximization Bias in Reinforcement Learning

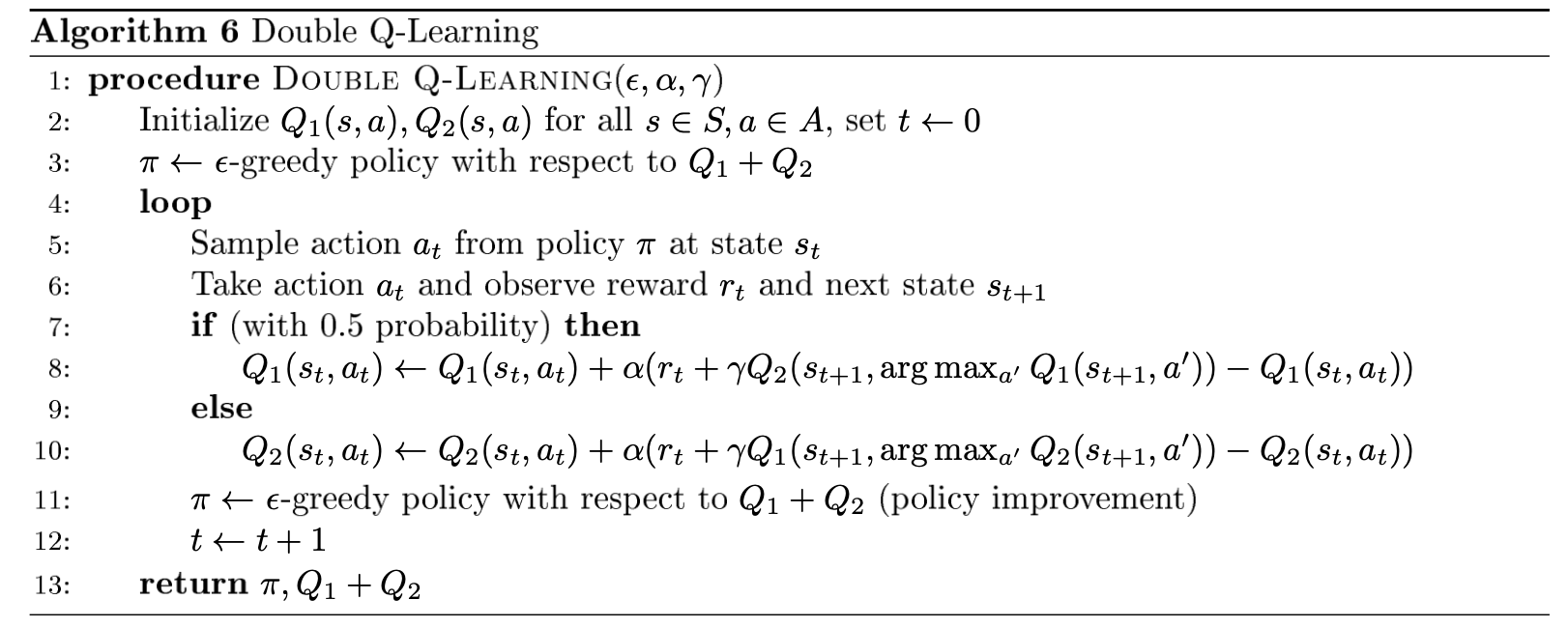
现在让我们将上文中提出的一个状态两个动作给出正式的定义。假设我们处于状态s，有两个动作和，它们的平均奖励为0。然后，我们得到state以及state-action的真实价值为零。也就是说，qt_temp。假设我们已经对每次的行动进行采样，并获得了奖励，这样我们就有了对状态行动值的一些估计。进一步假设这些样本是无偏的，因为它们是使用蒙特卡罗方法生成的。换句话说，，其中qt_temp是state-action二元组的样本数。令qt_temp，也就是对我们的Q值估计采取贪婪的政策。那么，我们有了



因此，我们的状态值估计值至少与状态s的真实值一样大，所以在有限样本存在的情况下，我们系统地高估了状态的值。

6.3 Double Q-Learning

正如我们在上个章节所看到的，当我们有有限多个样本时，状态值依然会遭受maximization bias。正如我们在硬币示例中讨论的，通过解耦最大值和估计最大值可以消除这种偏差。在Q-Learning中，我们能够保持两个独立的无偏估计,我们可以使用一个其中一个估计去选择最大值，另一个去估计最大值。这就产生了double Q-Learning，该算法如下所示。当我们提到关于的策略，意味着在状态s时的最优动作等同于:。



Double Q-Learning可以比正常Q-Learning学习更快地消除次优动作，从而显着加快训练速度。