

在之前的视频中，我们谈到了使用generalized policy iteration来寻找最优策略。我们还谈到了使用TD来估计价值函数。如果我们用TD来做generalized policy iteration中的策略评估（policy evaluation）步骤，会是什么样子？

generalized policy iteration或GPI，结合了两个部分：策略评估（policy evaluation）和策略改进（policy improvement）。我们看到的第一个采用这种形式的算法是政策迭代（policy iteration）。政策迭代在满足政策之前运行政策评估到收敛。然后，我们看到了蒙特卡洛的GPI，它每集都进行一次政策评估和改进的循环。

为了更好地记住蒙特卡洛的GPI，想象一只老鼠在一个四态的走廊里，走廊的尽头是奶酪。老鼠一开始什么都不知道，遵循一个随机的政策。最终，老鼠会因为随机移动而绊倒在奶酪上。在这一点上，老鼠会更新它的行动值。然后，它通过对其行动值的贪婪定义来改进其政策。随着这个过程的重复，它最终会学到最优策略。请注意，带蒙特卡罗的GPI在改进前不执行完整的策略评估步骤。相反，它在每一集之后进行评估和改进。更进一步，我们可以在一个政策评估步骤之后改进政策。我们将用TD来做到这一点。为了在GPI中使用TD，我们需要学习一个行动价值函数。因此，我们需要看一下与你过去看到的TD略有不同的版本。与其看从状态到状态的转换并学习每个状态的值，不如看从状态动作对到状态动作对的转换并学习每个动作对的值。这种算法被称为Sarsa预测。让我们更详细地看一下它。sarsa的缩写描述了更新中使用的数据，状态、行动、奖励、下一个状态和下一个行动。Sarsa对状态行动对的值进行预测。代理人选择了一个行动，在初始状态下创建第一个状态行动对。接下来，它在当前状态下采取该行动，观察奖励 $R_{t+1}$ 和下一个状态 $S_{t+1}$ 。

$R_{t+1}$

$S_{t+1}$

# Recall Generalized Policy Iteration

Monte Carlo : 8th 100 episode 完成后 1's update :

Policy Evaluation



Policy Improvement



⋮



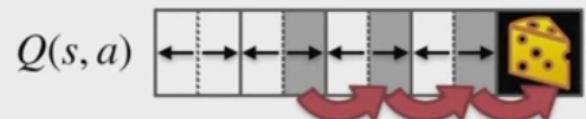
## From state-values to action-values



State to state



State-action to state-action  $\Rightarrow$  SARSA prediction.



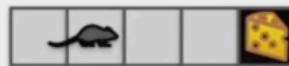
## The Sarsa algorithm

→ makes predictions about the values of the state-action pairs.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$$

Recall:  $V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$

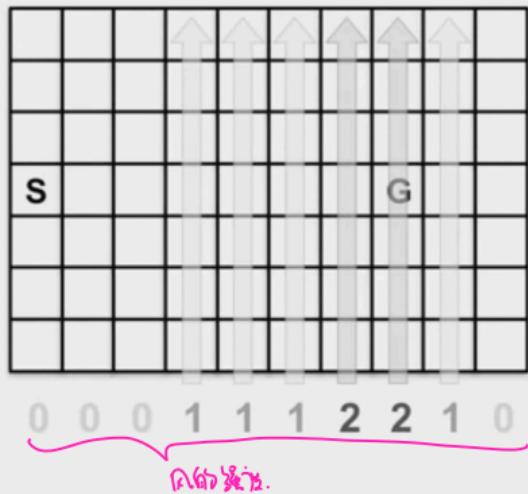
$S_t$        $A_t$        $R_{t+1}$        $S_{t+1}$        $A_{t+1} \sim \varepsilon\text{-greedy}$



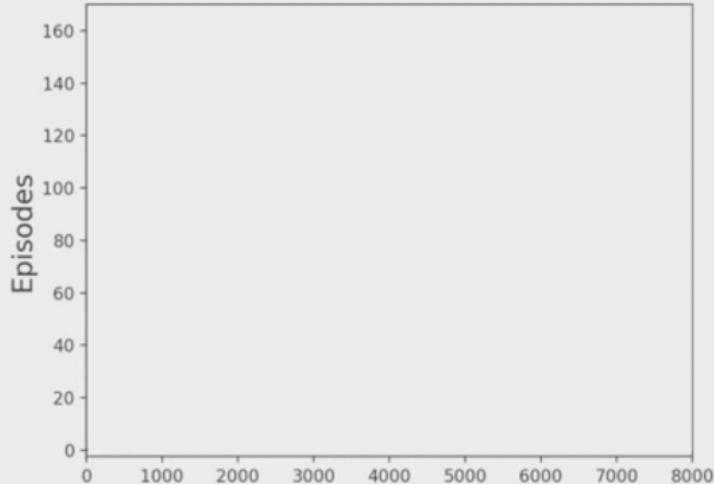
+0



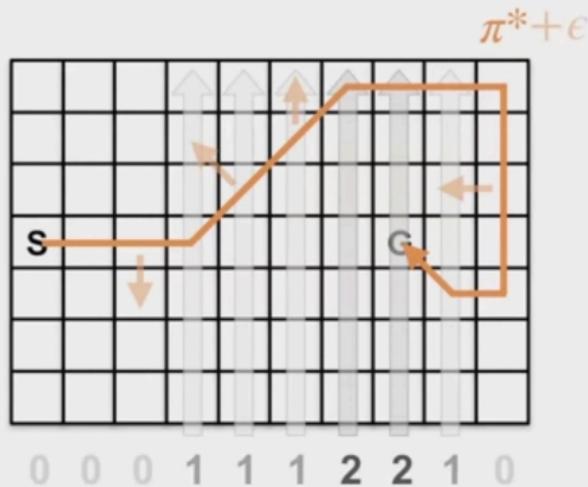
# Sarsa on the Windy Gridworld



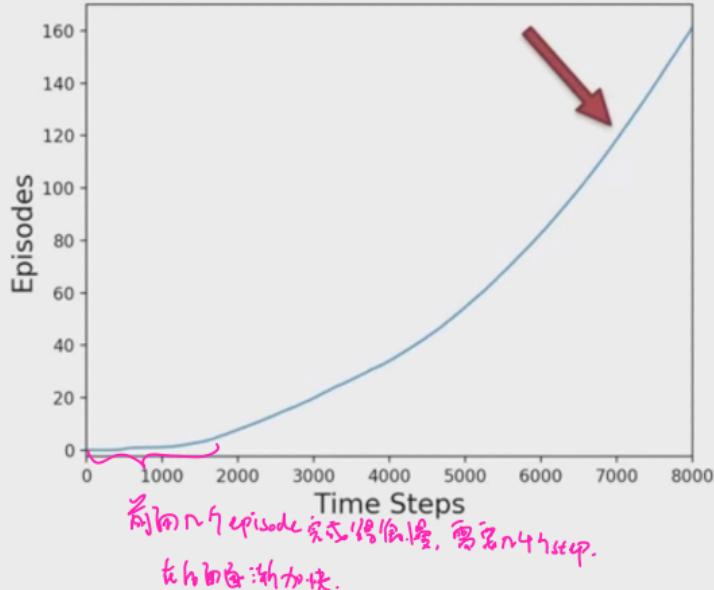
**Sarsa:**  $\epsilon = 0.1$  *⇒ we take a random action 1 in every 10 steps*  
 $\alpha = 0.5$



# Sarsa on the Windy Gridworld



**Sarsa:**  $\epsilon = 0.1$   
 $\alpha = 0.5$



强化学习的几个理由应用，学习玩雅达利游戏，控制交通信号，甚至自动配置网络系统，都是建立在一个单一的算法上。这种算法被称为Q-learning。

Q-learning开发于1989年，是第一批主要的在线强化学习算法之一。让我们来看看这个伪代码。我们以前见过这种格式。代理人在一个状态下选择一个行动，采取该行动，并观察下一个状态和奖励。然后代理人做一个更新，循环往复。有了这么多熟悉的元素，Q-learning有什么新的内容呢？

Q-learning的新元素是 行动值更新 (action value update)。在这里，目标是奖励 $R_{t+1}$ 加Gamma乘以下面状态下的最大行动值。这与Sarsa不同，Sarsa在其目标中使用下一状态的动作对的值。为什么Q-learning使用Emacs而不是下一个状态的动作对？这种联系可以追溯到动态编程的材料上。如果你看一下Sarsa的更新方程，它与动作值的贝尔曼方程可疑地相似。事实上，Sarsa是一种基于样本的算法，用于解决行动值的贝尔曼方程。Q-learning也使用环境中的样本来解决贝尔曼方程。但是，Q-learning没有使用标准的贝尔曼方程，而是使用了行动值的贝尔曼最优方程。优化方程使Q-learning能够直接学习Q-star，而不是在政策改进和政策评估步骤之间切换。

## The Q-learning algorithm

Q-learning (off-policy TD control) for estimating  $\pi \approx \pi_*$

Algorithm parameters: step size  $\alpha \in (0, 1]$ , small  $\varepsilon > 0$

Initialize  $Q(s, a)$ , for all  $s \in \mathcal{S}^+$ ,  $a \in \mathcal{A}(s)$ , arbitrarily except that  $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

Initialize  $S$

Loop for each step of episode:

Choose  $A$  from  $S$  using policy derived from  $Q$  (e.g.,  $\varepsilon$ -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

→ maximum action value in the following state

## Action Value update:

## Revisiting Bellman equations

**Sarsa:**  $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$

$$q_{\pi}(s, a) = \sum_{s', r} p(s', r | s, a) \left( r + \gamma \sum_{a'} \pi(a' | s') q_{\pi}(s', a') \right) \Rightarrow \text{Standard Bellman Equation}$$

**Q-learning:**  $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$

$$q_{*}(s, a) = \sum_{s', r} p(s', r | s, a) \left( r + \gamma \max_{a'} q_{\pi}(s', a') \right) \Rightarrow \text{Bellman's Optimality Equation}$$

$\therefore Q\text{-learning}\rightarrow \text{Learn } Q^*$

尽管Sarsa和Q-learning都是基于贝尔曼方程的，但它们基于非常不同的贝尔曼方程。Sarsa是基于样本的政策迭代版本，它使用行动值的贝尔曼方程，每个行动值都取决于一个固定的政策。Q-learning是基于样本的价值迭代版本，它迭代地应用贝尔曼最优方程。应用贝尔曼最优方程严格地改善了价值函数，除非它已经是最优的。因此，价值迭代不断提高价值函数估计值，最终收敛到最优解。出于同样的原因，只要老生继续探索并对状态行动空间的所有区域进行采样，Q-learning也会收敛到最优价值函数。让我们回顾一下今天所学的内容。在这个视频中，我们介绍了Q-learning算法，并描述了它与贝尔曼最优方程的联系。

我们刚刚向你介绍了Q-Learning。你可能对它为什么能很好地工作没有很直观的认识。在这个视频中，我们将演示如何在Windy Gridworld上使用Q-Learning，并讨论其结果。

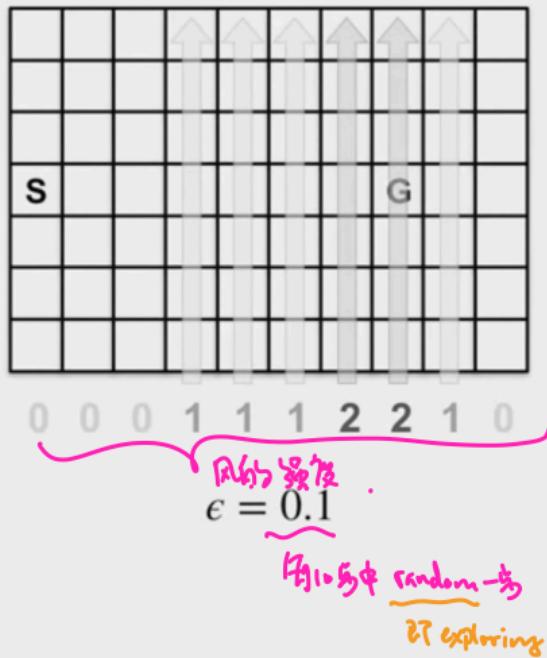
让我们回到他们的Windy Gridworld。这是一个未贴现的网格世界，当代理人多风的状态下移动时，风会将其吹向上方。风的强度显示在网格的每一列下面。代理人可以向四个方向移动，每一步都会得到减一的奖励。作为比较，这里是SARSA在Epsilon为0.1和Alpha为0.5时的表现。像以前一样，这张图在Y轴上显示了完成的剧集数量，在X轴上显示了时间步数。记得SARSA在最初的几集里学习得很慢。然后，它在达到最高点之前呈指数级增长。

让我们使用Q-Learning，参数设置相同。在展示结果之前，让我们思考一下会发生什么。也许SARSA的策略会表现得更好，因为其价值函数考虑到了它的Epsilon-greedy行为。也许Q-Learning会学得更快，因为它直接学习了最优策略的价值函数。你认为会发生什么？下面是结果：

在开始时，两种算法的学习速度相似。到了最后，Q-Learning似乎学到了一个更好的最终政策。

为什么Q-Learning在这里做得这么好？如果不进行更有针对性的实验，我们无法确定。也许Q-Learning的更新目标更稳定。Q-Learning取的是下一个行动值的最大值。因此，它只在代理人了解到一个行动比另一个行动更好时才会改变。相反，SARSA在其目标中使用下一个行动值的估计值。这在代理人每次采取探索性行动时都会改变。也许，你可以设计一个实验来弄清这个问题的真相。

# The Windy Gridworld



现在，让我们继续前进。我们怎样才能使SARSA表现得更好？对于这个实验来说，步长的参数值为0.5可能有点高了。当代理人采取探索性行动时，这些大的更新可能会给SARSA带来麻烦。SARSA的最终策略肯定不是最优的。让我们再运行一下这个实验，这次Alpha等于~~0.01~~<sup>0.1</sup>。我们期望SARSA在较小的 $\alpha$ 值下学习得更慢，但找到一个更好的政策。让我们看看会发生什么。预测得到证实。SARSA学习的最终策略与Q-Learning相同，但更慢。我们知道两个算法都收敛到了相同的政策，因为线条的斜率是相等的。斜率相等意味着两个代理都以相同的速度完成剧情。这个实验还强调了强化学习中参数选择的影响。Alpha、Epsilon、初始值和实验的长度都会影响最终的结果。在这个视频中，我们评估了Windy Gridworld中的Q-Learning，并深入了解了Q-Learning和SARSA在一个简单的MDP上的差异。

Q-Learning是一种非政策性 (off-policy) 的算法。但到目前为止，我们只见过使用重要采样的非政策性算法。在不使用重要抽样的情况下，Q-learning如何能够脱离政策？让我们深入了解一下吧。

回顾一下，一个代理人根据他们的目标政策 (target policy) 下的预期收益来估计其价值函数。他们实际上是根据他们的行为政策 (behavior policy) 来行事的。当目标政策和行为政策相同时，代理人是在学习~~on-policy~~<sup>off-policy</sup>政策，否则，代理人是在学习~~非~~<sup>on-policy</sup>政策。Sarsa是一种政策上的算法。在Sarsa中，代理从它接下来要采取的行动的值中引导，这是从它的行为策略中采样的。而Q-learning则从其下一个状态的最大行动值中提取。这就像在最优策略的估计下，而不是在行为策略的估计下，对行动进行采样。由于Q-learning学习的是它可能采取的最佳行动，而不是它实际采取的行动，所以它是在政策外学习。

每当你看到一个强化学习算法，一个自然的问题是，“行为政策中的目标是什么？”Q-learning的目标政策对于它的当前值来说总是greedy的。然而，行为策略可以是任何在学习过程中继续访问所有状态行动对的东西。一个可能的策略是 $\epsilon$ 贪婪。目标政策和行为政策之间的差异证实了Q-learning是非政策性的。

## Comparison with Sarsa

**Sarsa:**  $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, \tilde{\pi}(A_{t+1})) - Q(S_t, A_t))$

↳ i. learning on-policy

**Q-learning:**  $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, \tilde{\pi}_*(a')) - Q(S_t, A_t))$

↳ learning off-policy

# Behavior and target policies



Target policy:  $\pi^* = \operatorname{argmax}_a Q(s, a)$

→ target policy is

greedy (it's policy  
(all optimal))

Behavior policy:  $\pi$



但是，如果Q-learning学习的是非政策性的，为什么我们没有看到任何 important sampling ratios呢？这是因为代理人正在用未知的政策来估计行动值。它不需要important sampling ratios来纠正行动选择上的差异。行动值函数表示在给定状态下每个行动之后的回报。代理人的目标政策代表了在给定状态下采取每个行动的概率。把这两个元素放在一起，代理人可以计算出其目标政策下的预期收益，从任何给定的状态，特别是下一个状态， $S_{t+1}$ 。Q-learning正是使用这种技术来学习非政策。由于代理人的目标政策是贪婪的，就其行动值而言，所有非最大行动的概率为0，因此，该状态的预期回报等于该状态的最大行动值。

早些时候，我们谈到了Q-learning不在政策评估和政策改进之间迭代，而是直接学习最优价值函数。直接学习最优价值函数和策略听起来很好，但是有一些微妙的地方使得它在特定情况下不太理想。让我们来看看这个悬崖行走的例子，以了解其中的一些微妙之处。悬崖上行走的环境是一个未贴现的偶发网格世界，底部边缘有一个悬崖。在大多数步骤中，代理人得到的奖励是负1。然而，掉下悬崖会把代理人送回起点，并得到减去100的奖励。我们在这个环境中比较了Q-learning和Sarsa， $\epsilon$ 为0.1。下面是Q-learning的平均表现。由于Q-learning学习的是最优的价值函数，它很快就学会了一个最优的政策在悬崖边上行驶。然而，由于他的行动或 $\epsilon$ 贪婪，在悬崖边行驶偶尔会导致掉下悬崖。Sarsa考虑到 $\epsilon$ 贪婪行动选择的影响，了解了他当前的政策。考虑到偶尔的探索性行动，它学会了走更长但更可靠的路径。它们通常会避免随机掉入悬崖。因为它的路径更安全，Sarsa能够更可靠地到达目标。在这段视频中，我们展示了Q-learning在不使用重要抽样的情况下是非政策性的，在政策性或非政策性的学习中，根据任务的不同，在控制方面的表现可能不同。

我们刚刚讨论了两种TD控制方法，Sarsa和Q-learning。在这一课中，我们将讨论另一种TD控制方法，称为预期的Sarsa (expected Sarsa)。

回顾一下行动值的贝尔曼方程。在这里你可以看到对可能的下一状态行动对的期望值。把这个期望值拆开，我们可以看到对可能的下一个状态以及可能的下一个行动的总和。Sarsa通过从环境中抽取下一个 ~~state~~ <sup>state</sup> 和从其政策中抽取下一个行动来估计这个期望值。但是，代理人已经知道了这个政策，那么它为什么要对其下一个行动进行采样呢？相反，它应该直接计算出期望值。在这种情况下，我们可以对所有可能的下一步行动的值进行加权求和。权重是代理人政策下采取每个行动的概率。明确计算下一步行动的期望值是预期萨尔萨算法的主要思想。现在让我们来看看预期Sarsa的学习更新。由于expected Sarsa仍然是基于行动值的贝尔曼方程，它使用了熟悉的学习更新形式。该算法几乎与Sarsa相同，只是T误差使用了下一个行动值的预期估计，而不是下一个行动值的样本。这意味着在每个时间步骤上，代理人必须根据政策下的可能性大小来平均下一个状态的行动值。例如，在以下数值和政策下，expected Sarsa会使用1.4的数值来估计预期的下一个行动值。

然而，明确地计算期望值有一个巨大的好处。expected Sarsa比Sarsa有一个更稳定的更新目标。让我们看一个例子来更清楚地说明这一点。在这个例子中，媒体奖励是确定的1。Sarsa和预期的Sarsa，都是以下一个状态的真实际行动值启动的。即使在这种理想化的情况下，Sarsa所做的下一个动作采样也会导致它在错误的方向上更新其数值。它所依赖的事实是，在跨越多次更新的预期中，方向是正确的。相比之下，expected Sarsa更新目标是完全正确的，并不会使其估计值偏离真实值。一般来说，expected Sarsa更新目标的方差比Sarsa低得多。不过，较低的方差也有一个缺点。随着行动数量的增加，计算下一个行动的平均值变得更加昂贵。当有许多行动时，计算平均数可能需要很长的时间，特别是由于平均数必须在每个时间步长进行计算。

在这个视频中，我们展示了expected Sarsa明确地计算其策略下的期望值，这比抽样更昂贵，但方差更低。

Up until now, we've seen three TD algorithms for control, Sarsa, Q-learning, and Expected Sarsa. Two of those algorithms, Sarsa and Expected Sarsa both approximate the same Bellman equation. Today, let's look at how Expected Sarsa is related to Q-learning.

Let's start with the on-policy case, where the behavior policy and the target policy are equal. Consider the Expected Sarsa update. The next action is sampled from  $\pi_i$  in this case. However, notice that the expectation over actions is computed independently of the action actually selected in the next state. In fact,  $\pi_i$  need not be equal to the behavior policy. This means that Expected Sarsa, like Q-learning, can be used to learn off-policy without importance sampling. Now, what happens if the target policy is greedy with respect to its action value estimates? We can see that only the highest value action is considered in the expectation. This is equivalent to computing the maximum over actions in the next state just like in Q-learning. In other words, Q-Learning is a special case of Expected Sarsa. In this video, we showed that expected Sarsa and Q-Learning both use the expectation over their target policies in their update targets. This allows them to learn off-policy without importance sampling. Expected Sarsa with the target policy that's greedy with respect to its action values, is exactly Q-learning.

×

到目前为止，我们已经看到了三种用于控制的TD算法，Sarsa、Q-learning和Expected Sarsa。其中两个算法，Sarsa和Expected Sarsa都近似于同一个贝尔曼方程。今天，我们来看看Expected Sarsa与Q-learning有什么关系。

让我们从on-policy的情况开始，即行为策略(behavior policy)和目标策略(target policy)是相等的。考虑一下Expected Sarsa的更新。在这种情况下，下一个行动是从 $\pi_i$ 中取样的。然而，注意到行动的期望值是独立于在下一个状态中实际选择的行动来计算的。事实上， $\pi_i$ 不需要等于行为策略。这意味着，像Q-learning一样，Expected Sarsa可以用来学习非政策，而不需要importance sampling。现在，如果目标政策在它的行动值估计方面是贪婪的，会发生什么？我们可以看到，在期望值中只考虑最高价值的行动。这就相当于在下一个状态中计算行动的最大值，就像在Q-learning中一样。换句话说，Q-learning是Expected Sarsa的一个特例。在这段视频中，我们展示了Expected Sarsa和Q-Learning都在其更新目标中使用其目标政策的期望值。这使得它们可以在没有重要性采样的情况下学习非政策。具有目标政策(target policy)的Expected Sarsa在其行动值方面是贪婪的，这正是Q-learning。

We learned about three of them. Sarsa uses a sample based version of the Bellman equation. It learns Q-pi. Q-learning uses the Bellman optimality equation. It learns Q-star. Expected sarsa uses the same Bellman equation as Sarsa, but samples it differently. It takes an expectation over the next action values.

What's the story with on-policy and off-policy learning? Sarsa is an on-policy algorithm that learns the action values for the policy it's currently following. Q-learning is an off-policy algorithm that learns the optimal action values. And Expected Sarsa is both an on-policy and an off-policy algorithm that can learn the action values for any policy.

Sarsa can do better than Q-learning when performance is measured online. This is because on-policy control methods account for their own exploration. In the cliff world we saw that q-learning frequently fell off the cliff because of its exploratory actions. Sarsa learned the longer but safer path that rarely fell off the cliff, this resulted in higher reward. We then studied an improvement over Sarsa called Expected Sarsa. In the cliff world Expected Sarsa outperformed Sarsa for all the step size parameter values we tested. This is because Expected Sarsa mitigates the variance due to its own policy. Expected Sarsa, like the name suggests, takes the expectation over the next action.

我们了解了其中的三个: Sarsa使用一个基于样本的贝尔曼方程版本。它学习的是Q-pi。Q-learning使用贝尔曼最优化方程。它学习的是Q-star。Expected sarsa使用与Sarsa相同的贝尔曼方程,但以不同的方式取样。它采取的是对下一个行动值的期望值。

政策内(on-policy) 和 政策外学习(off-policy)是怎么回事? Sarsa是一个on-policy的算法, 它学习它当前遵循的政策的行动值。Q-learning是一种非政策性算法, 它学习最佳行动值。而预期的Sarsa既是一个on-policy算法, 也是一个off-policy算法, 可以学习任何政策的行动值。

当性能被在线测量时, Sarsa可以比Q-learning做得更好。这是因为政策上的控制方法说明了他们自己的探索。在悬崖世界里, 我们看到Q-learning由于其探索性的行动而经常跌落悬崖。Sarsa学习了更长但更安全的路径, 很少掉下悬崖, 这导致了更高的奖励。然后, 我们研究了对Sarsa的改进, 称为Expected Sarsa。在悬崖世界中, 在我们测试的所有步长参数值中, Expected Sarsa的表现都优于Sarsa。这是因为Expected Sarsa减轻了因其自身政策而产生的变异。正如其名称所暗示的, Expected Sarsa采取的是对下一步行动的预期。

## TD control and Bellman equations

$$q_{\pi}(s, a) = \sum_{s', r} p(s', r | s, a) \left( r + \gamma \sum_{a'} \pi(a' | s') q_{\pi}(s', a') \right)$$



**Sarsa**  $\Rightarrow$  learn  $Q_{\pi}$

$$R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$$



**On-policy**

$$q_{*}(s, a) = \sum_{s', r} p(s', r | s, a) \left( r + \gamma \max_{a'} q_{*}(s', a') \right)$$



**Expected Sarsa**

$$R_{t+1} + \gamma \sum_{a'} \pi(a' | S_{t+1}) Q(S_{t+1}, a')$$



**Off-policy**  
and

*On-policy*



**Q-learning**  $\Rightarrow$  learn  $Q_{*}$

$$R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a')$$



**Off-policy**

# Practice Quiz

## 最新提交作业的评分 100%

1. What is the target policy in Q-learning?

1/1分

- $\epsilon$ -greedy with respect to the current action-value estimates
- Greedy with respect to the current action-value estimates

正确

Correct! Q-learning's target policy is greedy with respect to the current action-value estimates.

2. Which Bellman equation is the basis for the Q-learning update?

1/1分

- Bellman equation for state values
- Bellman equation for action values
- Bellman optimality equation for state values
- Bellman optimality equation for action values

正确

Correct! The Q-learning update is based on the Bellman optimality equation for action values.

3. Which Bellman equation is the basis for the Sarsa update?

1/1分

- Bellman equation for state values
- Bellman equation for action values
- Bellman optimality equation for state values
- Bellman optimality equation for action values

正确

Correct! The Sarsa update is based on the Bellman equation for action values.

4. Which Bellman equation is the basis for the Expected Sarsa update?

1/1分

- Bellman equation for state values
- Bellman equation for action values
- Bellman optimality equation for state values
- Bellman optimality equation for action values

正确

Correct! The Expected Sarsa update is based on the Bellman equation for action values.

5. Which algorithm's update requires more computation per step?

1/1分

Expected Sarsa *沒有差值,但計算量費*

Sarsa

 正確

Correct! Expected Sarsa computes the expectation over next actions.

6. Which algorithm has a **higher variance target**?

1/1分

Expected Sarsa

Sarsa

 正確

Correct! We saw that Sarsa was more sensitive to the choice of step-size because its target has higher variance.

7. Q-learning does not learn about the outcomes of exploratory actions.

1/1分

True

False

正确



Correct! The update in Q-learning only learns about the greedy action. As demonstrated in Cliff World, it ignores the outcomes of exploratory actions.

8. Sarsa, Q-learning, and Expected Sarsa have similar targets on a transition to a terminal state.

1/1分

True

False

正确

Correct! The target in this case only depends on the reward.

9. Sarsa needs to wait until the end of an episode before performing its update.

1/1分

這是 Monte Carlo!!

True

False

正确

Correct! Unlike Monte Carlo methods, Sarsa performs its updates at every time-step using the reward and the next action-value estimate.