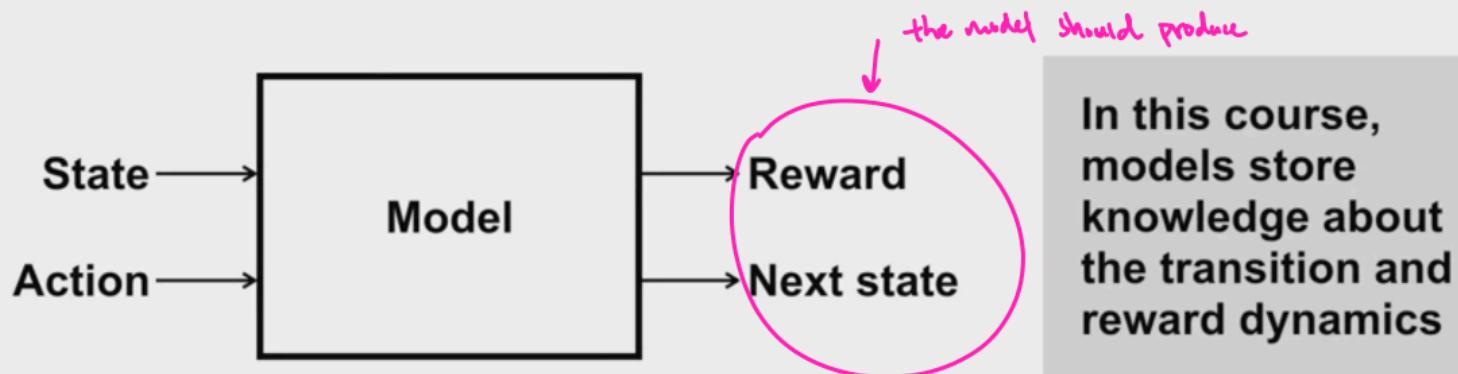


有时，我们在做决定时不会考虑太多，比如决定如何开车去上班。其他时候，我们坐下来，根据我们对世界的理解，想象许多可能的情况。我们可以利用这些想象的结果来决定做什么或不做什么。例如，如果我们用一只手拿着一个脆弱的物体，我们可以想象它掉下来并摔碎的情景，我们可以做出相应的选择。我们已经看到了基于样本的方法，如TD，它只从采样的经验中学习。我们也见过像动态编程或DP这样的方法，它通过使用关于事物如何运作的完整信息来进行规划，而不需要做出决定。如果我们能够得到一种中间方法，能够利用两个极端的优点，那就更好了。本周，我们将做到这一点，并将这些策略与Dyner架构统一起来。

模型是用来存储关于动态的知识(store knowledge about the dynamics)。当对一个决策想象出不同的情景时，这根植于我们对世界如何运作的知识。从行动中的一个特定状态，模型应该产生一个可能的下一个状态和奖励。这使我们能够看到一个行动的结果，而不必实际采取该行动。

一个模型可以进行规划(planning)。规划是指使用模型来改进政策的过程。用模型进行规划的一种方法是使用模拟的经验，并执行价值函数更新，就像这些经验发生了一样。通过改进价值估计，我们可以做出更明智的决定。模拟经验提高了样本的效率。模拟经验的增加意味着代理人需要更少的与世界的互动来得出相同的政策。

Models store knowledge



现在，让我们来谈谈可能对我们有用的不同类型的模型。一个是样本模型 (sample model)，它产生一个从一些基本概率中提取的实际结果。例如，抛掷硬币的样本模型可以产生一个头和尾的随机序列。另一种类型的模型是分布模型，它完全规定了每个结果的可能性或概率。在扔硬币的例子中，它可以说正面有50%的时间出现，反面有50%的时间出现。它还可以利用这些信息产生任何正面和反面的序列的概率。样本模型的计算成本很低，因为随机结果可以根据一组规则来产生。例如，要翻转五个硬币，一个样本模型可以独立地随机挑选五次零或一。它只需要为每次翻转产生一个结果。作为样本模型在实践中的一个例子，2017年，CloudFlare公司使用从熔岩灯墙中采样的随机模式来加密10%的全球互联网流量。这些样本是熔岩灯墙在特定时间点的随机配置，生成这些样本的规则程序由熔岩灯的动态定义。

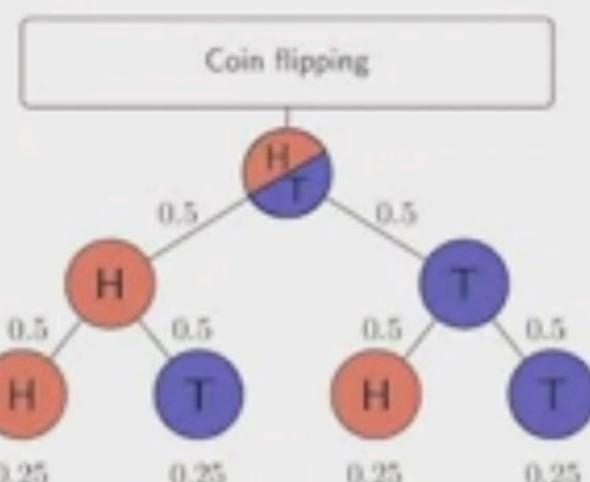
分布模型(Distribution models)包含更多的信息，但可能很难指定，而且会变得非常大。让我们回到翻转五个硬币的例子。分布模型会列举出你在五个硬币上可能得到的所有正面和反面的序列，并为每个序列分配一个概率。对于这个简单的问题，这将包括完全描述32种可能的结果。分布模型可以作为样本模型使用，根据每个结果的明确概率来~~预测~~结果。但它们包含更多的信息，可能严格来说只是为了获得样本。在这段视频中，我们谈到了模型如何通过一个叫做规划的过程来改善政策。我们讨论了两种不同类型的模型，样本模型和分布模型。下一次，我们将更深入地讨论这些模型之间的差异。

Types of Models

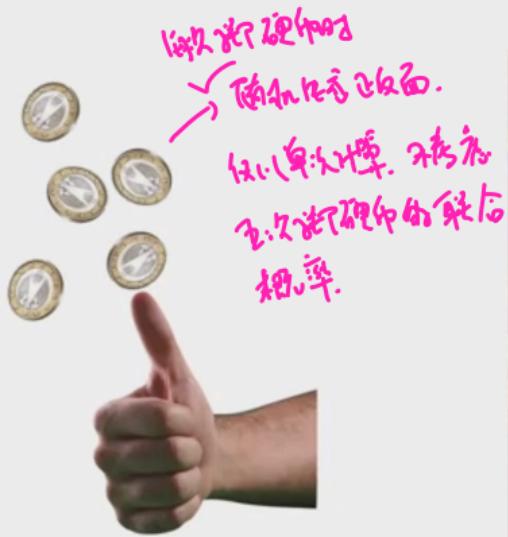
Sample



Distribution



Sample models



Distribution models



First flip



Second flip



Third flip



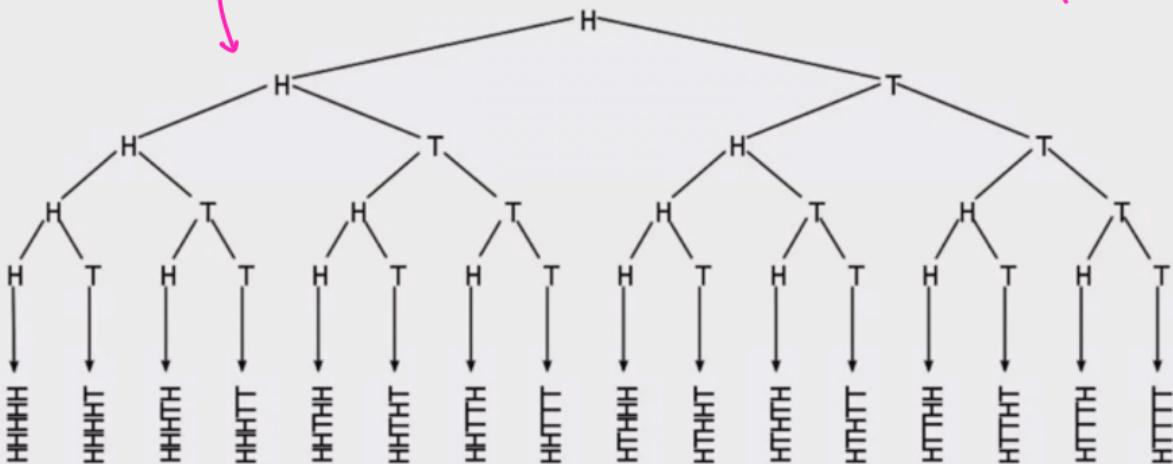
Fourth flip



Fifth flip

Outcome

树状图展示了五次掷硬币的所有可能结果，并为每次掷硬币分配概率。



$$p = \frac{1}{32}$$

模型代表了关于世界如何运作的知识。我们已经谈到了两种类型的模型；样本模型和分布模型。在这段视频中，我们将专注于一个具体的例子来说明样本模型和分布模型之间的关键区别。在本视频结束时，你将能够描述样本模型和分布模型的优点和缺点，你也将能够解释为什么样本模型可以比分布模型更紧凑地表示。

初心

让我们来考虑掷12个骰子的问题。这可以用一个骰子和足够多的样本轻松完成。掷一个骰子12次是一个样本模型的例子。在程序上，人们会从1-6中生成一个随机数，12次。这非常紧凑，没有考虑联合概率。对掷12次骰子的完整联合分布进行建模要花很多功夫。我们必须考虑12个骰子的每个可能的结果，并为每个结果分配一个概率。用一个骰子，我们有六个结果，用两个骰子，我们有36个结果，三个有216个，四个有1296个，当我们到了12个骰子时，我们有超过20亿个可能的结果需要考虑。但是，一个分布模型产生一个结果的确切概率可能是有用的。例如，我们可以直接计算出预期结果，或者量化结果的变异性。

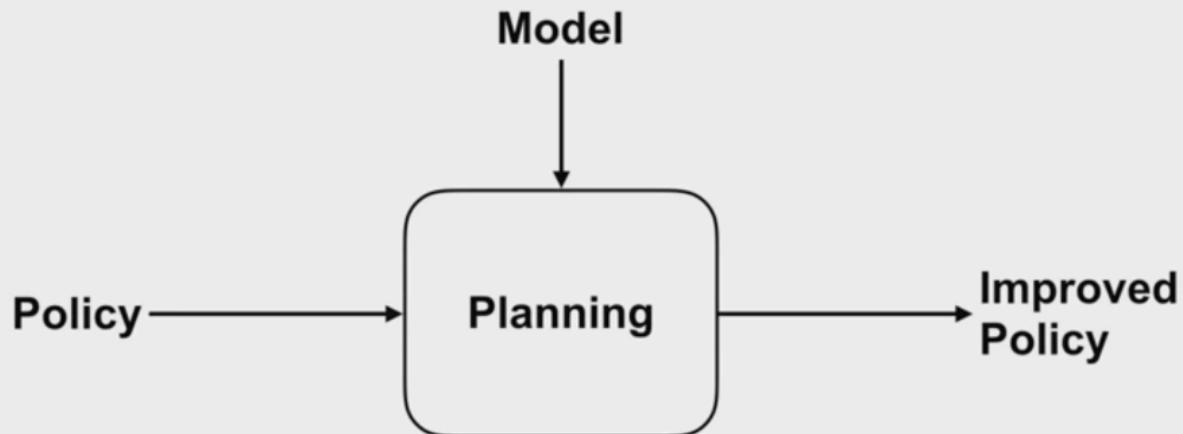
让我们总结一下每种类型的好处。样本模型需要较少的内存。另一方面，分布模型可以用来计算准确的预期结果，通过对所有结果加权的概率进行求和。请注意，样本模型只能通过将许多样本平均在一起来近似计算这个预期结果，知道确切的概率也有评估风险的灵活性。例如，当医生开药时，他们会考虑许多可能的副作用以及它们发生的可能性。在这段视频中，我们描述了样本模型如何比分布模型更紧凑，以及明确的概率和分布模型在某些情况下如何有用，如计算预期。

现在我们知道了什么是模型，我们可以讨论如何在强化学习中使用它们。也就是说，人们如何利用一个模型来更好地通知决策，而不需要与世界互动，我们把这个过程称为用模型经验进行规划。在这段视频中，我们将谈论一种特殊的规划观点。在本视频结束时，你将能够解释规划是如何被用来改进政策的，并描述一个步骤的表列规划。

规划是一个将模型作为输入并产生改进的策略的过程。规划的一个可能的方法是首先从模型中抽取经验。这就像根据你对世界运作方式的理解，想象世界上可能出现的情景。然后，这种生成的经验可以被用来执行价值函数的更新，就像这些互动在世界中实际发生一样。对这些改进的价值采取贪婪的行为会导致政策的改进。

回顾一下，Q-learning使用来自环境的经验，他们执行更新以改善政策。在Q-planning中，我们使用来自模型的经验，并执行类似的更新来改善政策。随机抽样的单步表格Q-planning (Random-sample one-step tabular Q-planning) 说明了这个想法。这种方法假设我们有一个过渡动态的样本模型。它还假设我们有一个对相关状态行动对进行抽样 (sampling relevant state action pairs) 的策略。一个可能的选择是对状态和行动进行统一采样。这种算法首先从所有的状态和动作集合中随机选择一个状态动作对。然后，它用这个状态动作对查询样本模型，产生下一个状态和奖励的样本。然后，它对这个模型转换进行Q-Learning更新。最后，它通过对更新的行动值进行美化来改进策略。一个关键点是，这种规划方法只使用想象的或模拟的经验。所有这些更新都可以在不在世界中行为的情况下进行，或者与交互循环平行进行。

Planning improves policies



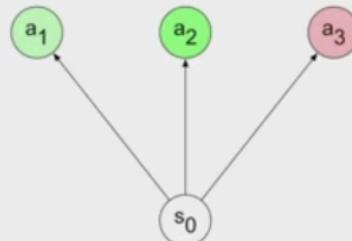
take model as an input and produce improved policy

Planning improves policies

Planning

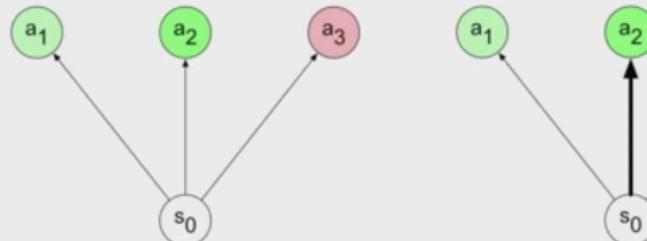


$$Q_{\pi}(s, a)$$



$$\pi(s_0) = a_2$$

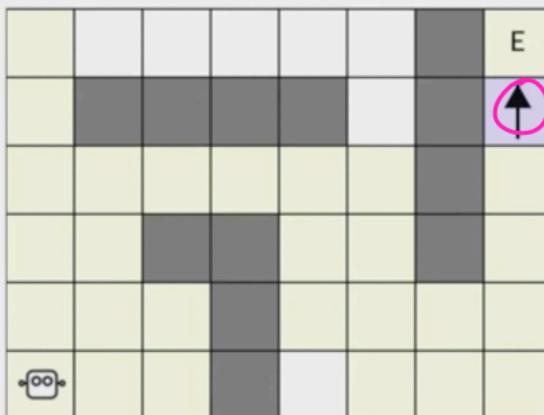
根据此动作值
选择最贪心
的政策
评价



想象一下，行动只能作为特定的时间点，但学习更新可以相对快速地执行。这就导致了从学习更新之后到采取下一个行动的时候会有一些等待时间。我们可以用规划更新来填补这个等待时间。想象一下，我们有一个站在悬崖边的机器人。假设它的模型知道跳下悬崖的结果，但在它的价值函数或策略中并没有准确体现。它可以生成踏出悬崖的模拟经验，并在这些过渡中执行许多计划步骤。它的价值函数现在能更好地反映出走下悬崖是不好的，它的政策也会有其远离悬崖的步骤。总结一下，我们讨论了规划方法如何使用模拟经验来改进政策，随机抽样一旦我表Q-规划如何使用样本模型随机产生经验，以及价值函数如何使用Q-学习更新来改进。

One direct RL update, using Q-learning:

$$Q(s, a) \leftarrow Q(s, a) + \alpha (r + \gamma \cdot \max_{a'} Q(s', a') - Q(s, a))$$



only updates here.

After 2 trips reward 250.

Dyna performs planning on every time step. However, planning has no impact on the policy during the first episode even though the model became more accurate on each time step. After the first episode completes, planning can really shine. In our cartoon, the thought bubble represents planning happening in the robot's mind. The grid world in the bubble represents the robots model of the world. The yellow states represent where the model knows what will happen next. The model knows about all the states the robot visited during the first episode. Let's look what happens if we run the planning loop at the beginning of the second episode. Dyna can simulate transitions from any of the state action pairs the agent visited during the first episode. In this case, there are a lot of them. Planning replays the simulated transitions as if they happened in the real world. Each step of planning simply applies the Q-learning update to a simulated transition and updates the value function. With enough planning steps, the agent can approve the policy in all the states visited during the first episode. Here, planning produce a pretty good policy. The policy is not perfect. Can you see where take suboptimal actions? The policy is still random in states that were not visited before. We represent these dates for the policy was not updated with white squares.

X

Dyna在每个时间步骤上都进行规划。然而，在第一集期间，尽管模型在每个时间步骤上都变得更准确，但规划对政策没有影响。在第一集完成后，规划可以真正发挥作用。在我们的动画片中，思维泡泡代表了机器人头脑中发生的规划。气泡中的网格世界代表机器人的世界模型。黄色的状态代表模型知道接下来会发生什么。该模型知道机器人在第一集里访问的所有状态。让我们看看如果我们在第二集开始时运行规划循环会发生什么。Dyna可以模拟代理人在第一集期间访问的任何一个状态行动对的过渡。在这种情况下，有很多这样的情况。规划会复制模拟的过渡，就像它们发生在现实世界中一样。规划的每一步只是将Q-learning更新应用于模拟的过渡，并更新价值函数。有了足够的规划步骤，代理人可以在第一集期间访问的所有状态中批准政策。在这里，规划产生了一个相当好的政策。该政策并不完美。你能看到哪里采取了次优的行动吗？该政策在以前没有访问过的状态中仍然是随机的。我们用白色方块表示这些政策没有更新的日期。

让我们详细看看Dyna-Q的算法。首先，我们有一个通常的代理环境互动循环。在当前状态下，代理人根据其epsilon贪婪策略选择一个行动。然后，它观察下一个状态下的奖励结果。它通过这个过渡执行Q-learning更新，我们称之为^{direct} Q-RL。如果我们在那里停止，我们将得到完全的Q-learning算法。这就是事情开始与无模型方法不同的地方。Dyna-Q将采用这个过渡并对其进行模型学习步骤。为了做到这一点，该算法记住了下一个状态和给定的状态动作对的奖励。这样做是因为我们假设环境是确定性的。然后，Dyna-Q执行了几个规划步骤。每个规划步骤由三个步骤组成：搜索控制 (search control)、模型查询(model query) 和 价值更新(value update)。在这个算法中，搜索控制随机选择一个以前访问过的状态行动对。它必须是代理人以前见过的状态动作对。否则，模型将不知道接下来会发生什么。给出状态动作对，我们为下一个状态和奖励创建模型。现在我们已经生成了一个模型转换。最后，Dyna-Q用模拟的过渡执行Q-learning更新。这个计划步骤重复了很多次。最重要的是要记住，Dyna-Q为每个环境转换执行许多规划更新。这就是tabular Dyna-Q，Dyna架构的一个简单实例。

让我们重新审视我们的机器人朋友，以更好地理解Dyna-Q算法。记住，这只是一个建立直觉的卡通。像以前一样，它一开始对环境一无所知。机器人在与环境互动的同时逐渐建立起一个模型。第一集需要184步。在这段时间里，代理只改变了目标旁边的状态动作对的值。如果没有计划，接下来的几集可能也会一样长。让我们看看下一集，研究在每个时间步骤上做100步计划的影响。上一次我们谈到这个机器人时，它在每一步上做的计划要多得多。所以这一次，Dyna-Q需要更长的时间来找到一个好的策略。仅仅过了一步，规划又更新了两个状态的值。现在，代理知道了在最后的走廊上的正确行动。让我们再运行一下Dyna-Q。规划的影响是相当巨大的。Dyna-Q将奖励信息传播到整个状态空间。最终，代理人知道了从大多数状态导航到目标的有效策略。

More planning → faster learning

→ Dyna Q is far more simple efficient.



让我们在一个稍有不同的迷宫中直观地看到正在发生的事情。我们用一个箭头来表示根据每个状态下的估计值而采取的贪婪行动。没有箭头的状态意味着每个行动在政策下都是同样可能的。在一个 ~~小~~ ^{episode} 之后, Q-learning 只更新了一个行动值, 即与目标旁边的状态中的向上行动相对应的那个。这是唯一一个经历过奖励为非零的过渡的状态。要想把这个状态的值引导到其他附近的状态, 还需要几个情节。让我们更仔细地看一下搜索控制是如何影响规划的。我们的机器人将以与 Dyna-Q 略有不同的方式做事。但这只是为了更清楚地强调我们的观点。我们将从 10 个规划步骤开始, 连续调用 10 次规划循环, 所以总共有 100 个规划步骤。正如你所看到的, 许多规划更新未能改变价值函数。事实上, 代理在 100 步规划后只更新了 2 个状态行动对的值。

让我们继续下去。这次我们将尝试每次调用 100 个规划步骤, 让代理运行的时间更长一些。即使有 100 个规划步骤, 前几个调用也只更新了少数几个动作值。为什么代理需要这么多规划步骤来学习价值函数和合理的政策?

代理人需要许多规划步骤, 因为搜索控制随机地对状态行动对进行采样。如果抽样的状态动作对产生了 $O(T)$ 的错误, 规划更新就不会有任何影响。这种情况在这个环境中经常发生, 因为所有的奖励都是 0, 初始值也是 0。如果我们以一种更有效的方式对状态行动对进行排序, 会发生什么? 事实证明, 代理人只能用六分之一的规划更新来学习一个好的政策。在更大的环境中, 随机搜索控制变得更有问题。如果你想了解更多关于这个主题的信息, 请查看教科书的第 8.4 节。

在这个视频中, 我们在一个简单的网格世界中测试了表格 Dyna-Q 算法。我们看到了规划是如何极大地加快学习速度的。在我们的实验中, 我们做的规划越多, 代理人的表现就越好。

到目前为止，我们在课程中谈到了代理人如何通过用模型产生的经验进行规划来改进他们的政策或价值函数。但是，如果代理人用一个不准确的模型的经验进行规划，会发生什么？在本视频中，你将学习如何识别模型不准确的方式，解释用不准确的模型进行规划的影响，并描述Dyna如何用不完整的模型成功植入。

首先，让我们谈一谈模型不准确的含义。**当模型存储的过渡与环境中发生的过渡不同时，模型是不准确的。**在学习的开始阶段，代理还没有在几乎所有状态下尝试过大部分的行动。与在这些状态下尝试这些行动相关的过渡在模型中是缺失的。我们把缺失过渡的模型称为**不完整模型**(incomplete models)。如果环境发生变化，该模型也可能是一个准确的模型。在一个状态下采取的行动可能会导致与代理人在变化前观察到的不同的下一个状态和奖励。**我们说模型是不准确的，因为实际发生的情况与模型所说的不同。**那么，当我们用不准确的模型进行规划时会发生什么？用不准确的模型进行规划的效果取决于模型是如何不准确的。在开始时，模型是不完整的。由于模型不能产生下一步或奖励，它不能用于规划。然而，**随着代理人与环境的互动，模型储存了越来越多的转换。**然后，代理**可以通过模拟它以前看到的过渡来执行更新。**这意味着，只要代理看到了一些过渡，它就可以用模型进行规划。现在让我们思考一下环境的变化如何影响规划。想象一下，**代理已经访问了每一个状态行动对，并将其经验储存在模型中。**然后，环境发生了变化。**模型中的一个过渡不再反映环境中的过渡。**

当代理人试图使用其不准确的模型进行规划时，会发生什么？如果代理人试图用模型中的一个不正确的过渡来执行规划更新，代理人更新的价值函数或策略可能会向错误的方向变化。请记住，规划是针对模型认为会发生的事情而改进政策的，而不是环境中真正会发生的事情。当环境发生变化时，该模型就会变得过时。用该模型进行规划可能会使代理人的政策在环境方面变得更糟。现在我们已经讨论了模型可能不准确的两种主要方式，让我们思考一下代理人如何在不完整的模型下成功地进行计划。实际上，我们已经讨论了一个这样的算法，即Dyna Q。让我们看看Dyna Q如何在不完整的模型下进行规划。在计划步骤中，我们必须确定哪些状态下的行动对可以查询模型。模型只知道下一个状态和它已经访问过的状态动作对的奖励。因此，Dyna Q只能从以前访问过的状态动作对中进行规划更新。Dyna Q只计划它已经看到的过渡。因此，在学习的最初几个时间步骤中，Dyna Q可能会对相同的过渡做相当多的规划更新。然而，随着Dyna Q访问环境中更多的状态动作对，它的规划更新在整个状态动作空间中的分布变得更加均匀。

在这段视频中，我们谈到了如果模型不完整或者环境发生变化，模型就会不准确。用一个不准确的模型进行规划，可以改善模型而不是环境的策略或价值函数。Dyna Q可以通过只对以前访问过的状态行动对进行抽样，从而用不完整的模型进行规划。

Dyna with an incomplete model

Tabular Dyna-Q

Initialize $Q(s, a)$ and $Model(s, a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$

Loop forever:

- (a) $S \leftarrow$ current (nonterminal) state
- (b) $A \leftarrow \varepsilon\text{-greedy}(S, Q)$
- (c) Take action A ; observe resultant reward, R , and state, S'
- (d) $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$
- (e) $Model(S, A) \leftarrow R, S'$ (assuming deterministic environment)
- (f) Loop repeat n times: \Rightarrow planning step.

$S \leftarrow$ random previously observed state

$A \leftarrow$ random action previously taken in S

$R, S' \leftarrow Model(S, A)$

$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$

The model only knows the next state and reward from state action pairs it has already visited.
i. Dyna Q can only do planning updates from previously visited state action pairs.

我们之前谈到了代理如何用不完整的模型进行计划。现在，让我们来讨论代理如何利用因环境变化而不准确的模型进行规划。

当代理人的模型不准确时，计划可能会使政策或价值函数相对于环境变得更糟。这意味着代理人在确保它的模型保持准确方面有既得利益。在这个例子中，健壮的模型 (the robust model) 最初说，如果兔子选择向右移动，它将直接去找胡萝卜。然而，在向右移动后，兔子发现自己在中间的方格。在经历了这个转变之后，兔子可以更新它的世界模型。这个更新反映出，在最左边的方格中，选择向右移动的动作，将导致移动到中间的方格。我们看到，兔子在经历了环境中的一个过渡后修正了它的模型。

一般来说，一个代理可能想反复检查它的所有模型的转换是否正确。然而，用低价值的行动重复检查过渡，往往会导致低回报。在不断变化的环境中，代理人的模型可能在任何时候都变得不准确。因此，代理人必须做出选择；探索以确保它的模型是准确的，或者利用模型来计算最佳策略，假设模型是正确的。

Exploration and exploitation for model accuracy

Exploration



- Higher model accuracy

Exploitation



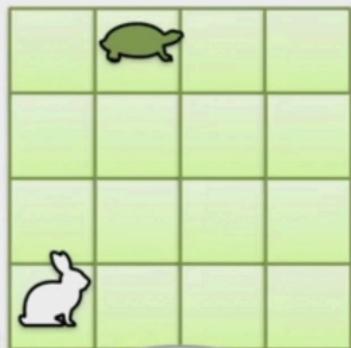
- Better policy with respect to model

当环境发生变化时，模型将是不正确的。它将保持不正确，直到代理人重新访问环境中发生变化的部分并更新模型。这表明，代理应该探索它有一段时间没有去过的地方。让我们想一想这可能是怎么回事。粗略地说，模型更有可能是错误的，而且是代理人很久没有去过的状态。例如，假设兔子知道乌龟在下面的单元格开始。由于乌龟移动缓慢，它在最初的几个时间步骤中不可能移动很远。然而，在很长一段时间后，乌龟可能会在一个与它开始时完全不同的单元中，使兔子的模型不正确。为了鼓励代理人定期重访其状态，我们可以在规划中使用的奖励上增加一个奖励。这个奖励是简单的Kappa，乘以Tau的平方根，其中r是模型的奖励，Tau是环境中最[最后一次访问状态行动对后的时间](#)。Tau不在规划循环中更新，那不是真正的访问。[Kappa是一个小常数，控制奖金对规划更新的影响](#)。如果Kappa为零，我们将完全忽略奖金。在规划更新中加入这种探索奖励，就形成了Dyna-Q+算法。

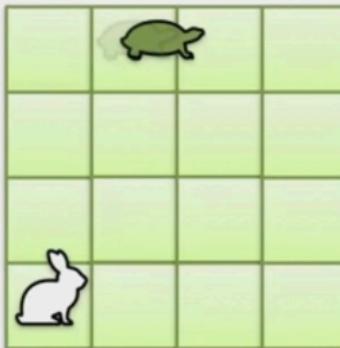
通过人为地增加规划中使用的奖励，我们增加了最近没有访问过的状态行动对的价值。这究竟是如何鼓励探索的呢？想象一下，一个很久没有被访问过的状态动作对。这意味着Tau会很大。随着Tau的增长，奖金变得越来越大。最终，由于奖金大，规划会改变政策，直接去S。当代理最终访问状态S时，它可能会看到一个大的奖励，或者它可能会感到失望。无论哪种情况，模型都会被更新以反映环境的动态变化。让我们看一下捷径迷宫，以了解Dyna-Q和Dyna-Q+之间的区别。

Model inaccuracy and time

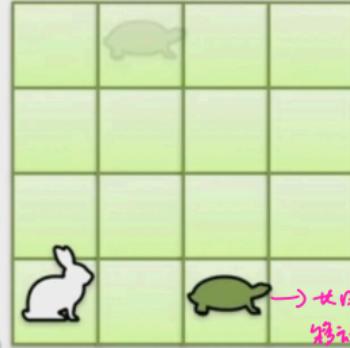
$t = 0$



$t = 2$



$t = 1000$



the rabbit's model.
totally correct.



model 基本正确.



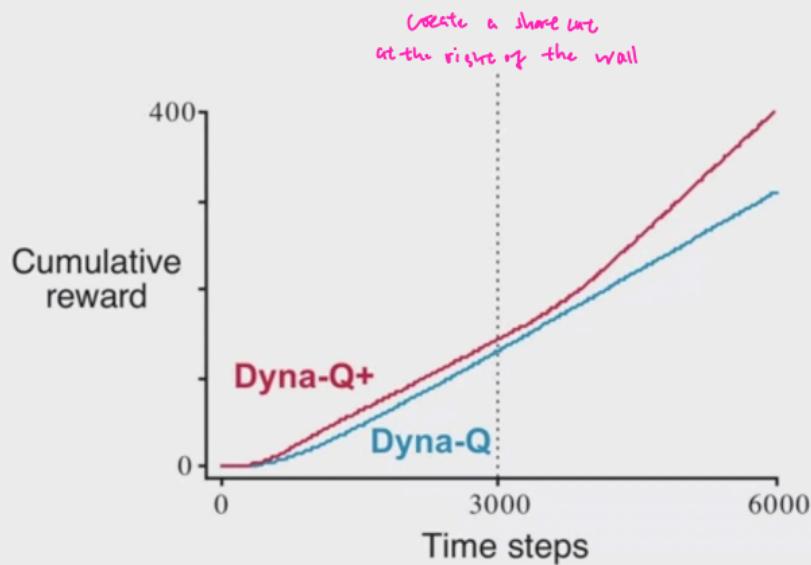
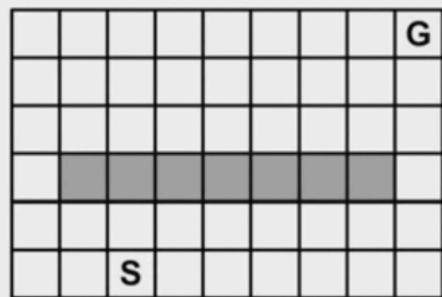
model 完全正确.

→ 太晚时间轴, 乌龟
移动到了兔子的位
置.

在这个例子中，目标是尽可能快地从起始状态到达目标。奖励在任何地方都是零，除了终端转换 (terminal transition)，它发出的奖励是+1。折扣系数小于1，因为我们想鼓励代理人快速到达目标。Dyna-Q和Dyna-Q+都使用 epsilon-greedy 行动选择。在实验的前半段，Dyna-Q和Dyna-Q+的表现非常相似。在这种情况下，Dyna-Q+增加的探索有助于更快地定义一个好的策略。这就是为什么Dyna-Q+的线高于Dyna-Q的线。现在，我们在墙的右侧创建一个捷径。让我们看看Dyna-Q和Dyna-Q+如何适应这一变化。他们能通过墙的右侧找到通往目标的更短路径吗？下面是结果，Dyna-Q+在环境变化后很快找到了捷径，另一方面，Dyna-Q在规定的时间内没有找到捷径。最终，Dyna-Q会通过使用epsilon-greedy重新探索整个状态动作空间来找到捷径。然而，这样做至少需要连续进行七次探索性行动。所以可能需要很长的时间。在这种环境下，Dyna-Q+的持续和系统探索是关键。

在这段视频中，我们谈到了模型的准确性如何产生一种新的探索-开发权衡的变体。代理人必须探索以确保其模型的准确性。我们还谈到了Dyna-Q+如何使用到期奖金来探索环境。

Dyna-Q vs Dyna-Q+ in a changing environment



你刚刚完成了模块4，在那里我们学习了计划(planning)、学习(learning) 和行动(acting)。让我们复习一下本模块中涉及的概念和算法。

在第一课中，我们介绍了 模型(models)，并谈到了 分布模型(Distribution models) 和 样本模型(sample models) 之间的区别。分布模型存储每个可能结果的概率。然而，这可能需要大量的内存。样本模型通常比分布模型要紧凑得多，因为它们不明确地存储每个结果的概率。

在第2课中，我们介绍了one-step Q-planning 计划。Q-planning使用的更新方式与Q-learning相同。然而，它使用的是由模型产生的经验，而不是真实的经验。

在第三课中，我们介绍了Dyna架构。Dyna在一个单一的代理中包含了规划和学习。我们看到，通过使用许多规划更新，Dyna Q-agent可以更快速地学习。模型并不总是准确的。

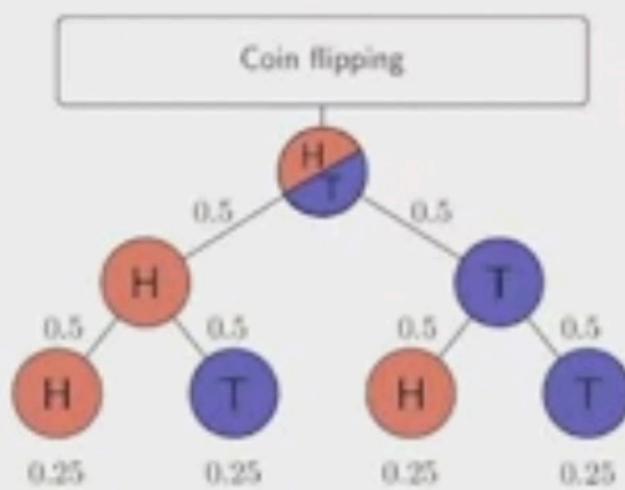
在第4课中，我们谈到了不准确的模型如何影响规划。我们指出，Dyna-Q 可以使用不完整的模型进行有效的规划，但如果模型是错误的，则不能。

最后，我们介绍了Dyna-Q+。该算法在其规划更新中使用奖励奖金来鼓励探索。通过探索，Dyna-Q+保持其模型的最新性和准确性，从而获得更好的性能。

Types of models

Distribution

存储向事件的概率.



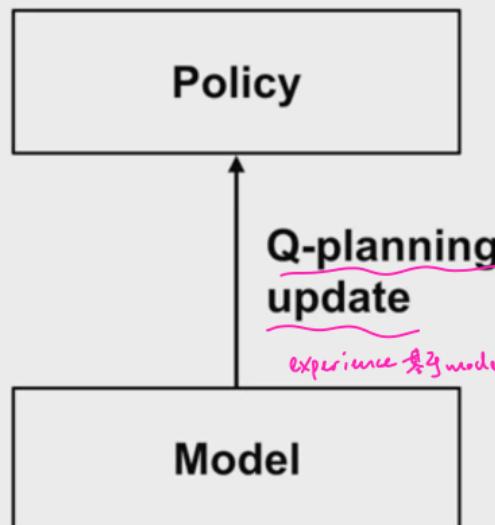
Sample

每次掷硬币的任一结果.

掷硬币同时出现正面/反面的随机概率.



Random-sample one-step tabular Q-planning



$$Q(S, A) \leftarrow Q(S, A) + \alpha \left[R + \gamma \max_a Q(S', a) - Q(S, A) \right]$$

Practice Assessment

最新提交作业的评分 100%

1. Which of the following are the most accurate characterizations of sample models and distribution models? (Select all that apply)

1/1分

- A sample model can be used to compute the probability of all possible trajectories in an episodic task based on the current state and action.
- A distribution model can be used as a sample model.

正确

Correct; a distribution model contains all the information about the transition dynamics of the system, which can be used to 'sample' new states and rewards given the current state and action – just like a sample model.

- A sample model can be used to obtain a possible next state and reward given the current state and action, whereas a distribution model can only be used to compute the probability of this next state and reward given the current state and action.
- Both sample models and distribution models can be used to obtain a possible next state and reward, given the current state and action.

正确

Correct; given any state and action, you can sample the next state and reward using a sample model or distribution model.

2. Which of the following statements are **TRUE** for Dyna architecture? (Select all that apply)

1/1分

- Simulated experience can be used to improve the model
- Simulated experience can be used to improve the value function and policy

✓ 正确

Correct; we do this in the planning step of the tabular Dyna-Q algorithm

- Real experience can be used to improve the model

✓ 正确

Correct; we do this in the model-learning step of the tabular Dyna-Q algorithm

- Real experience can be used to improve the value function and policy

✓ 正确

Correct; we do this in the direct-RL step of the tabular Dyna-Q algorithm

3. Mark all the statements that are TRUE for the tabular Dyna-Q algorithm. (Select all that apply)

1/1 分

- The algorithm **cannot** be extended to stochastic environments.
- For a given state-action pair, the model predicts the next state and reward

正确

Correct; this is because in the tabular Dyna-Q algorithm, the model stores the next state and action for every state-action pair that is encountered

- The environment is assumed to be deterministic.

正确

Correct; the algorithm assumes that the environment deterministically transitions to a single next state and reward for a given state-action pair. If the environment is stochastic, the update-model step in its current form would simply overwrite a state-action pair with a different next state and reward transition. So unless the update-model step is modified, we would be losing a lot of useful information. This may lead to a poor performance even though we are using a planning-based method.

- The memory requirements for the model in case of a deterministic environment are quadratic in the number of states

- Model-based methods often suffer more from bias than model-free methods, because of inaccuracies in the model.

正确

Correct; the performance of model-based methods depends heavily on the model.

- The amount of computation per interaction with the environment is larger in the Dyna-Q algorithm (with non-zero planning steps) as compared to the Q-learning algorithm.

正确

Correct; apart from the direct RL steps performed in the Q-learning algorithm, Dyna-Q performs additional steps of model-learning and planning.

- When compared with model-free methods, model-based methods are relatively more sample efficient. They can achieve a comparable performance with comparatively fewer environmental interactions.

正确

Correct; we have seen examples of this in the lectures and [Chapter 8](#) of Sutton and Barto's RL textbook



Model-based methods like Dyna typically require more memory than model-free methods like Q-learning.



正确

Correct; additional memory is required to store the model.

5. Which of the following is generally the most computationally expensive step of the Dyna-Q algorithm?
Assume $N > 1$ planning steps are being performed (e.g., $N=20$).

Tabular Dyna-Q

Initialize $Q(s, a)$ and $Model(s, a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
 Loop forever:

- $S \leftarrow$ current (nonterminal) state
- $A \leftarrow \varepsilon\text{-greedy}(S, Q)$
- Take action A ; observe resultant reward, R , and state, S'
- $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$
- $Model(S, A) \leftarrow R, S'$ (assuming deterministic environment)
- Loop repeat n times:
 - $S \leftarrow$ random previously observed state
 - $A \leftarrow$ random action previously taken in S
 - $R, S' \leftarrow Model(S, A)$
 - $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$

Model learning (step e)

Direct RL (step d)

Action selection (step b)

Planning (Indirect RL; step f)



正确

Correct; the planning step performs search control ($O(1)$ with an appropriate dictionary implementation), generates a simulated experience ($O(1)$), and updates the action-value function ($O(|A|)$). This is repeated N times, for overall $O(N*|A|)$ time complexity.

6. What are some possible reasons for a learned model to be inaccurate? (Select all that apply)

1/1分

- The agent's policy has changed significantly from the beginning of training.
- There is too much exploration (e.g., epsilon is epsilon-greedy exploration is set to a high value of 0.5)
- The environment has changed.

正确

Correct; if the environment has changed (e.g., a new wall has come up in the gridworld, changing the transition probabilities), then the learned model is no longer accurate

- The transition dynamics of the environment are stochastic, and only a few transitions have been experienced.

正确

Correct; if there are stochastic transitions from certain states and actions, you might require many samples to form reliable estimates in the model. For a stochastic environment, we can keep counts of the number of times each next state and reward is experienced from each state-action pair. We can use this to estimate probabilities of next states and rewards, from a given state and action.

7. In search control, which of the following methods is likely to make a Dyna agent perform better in problems with a large number of states (like [the rod maneuvering problem](#) in Chapter 8 of the textbook)? Recall that search control is the process that selects the starting states and actions in planning. Also recall the navigation example in the video lectures in which a large number of wasteful updates were being made because of the basic search control procedure in the Dyna-Q algorithm. (Select the best option)

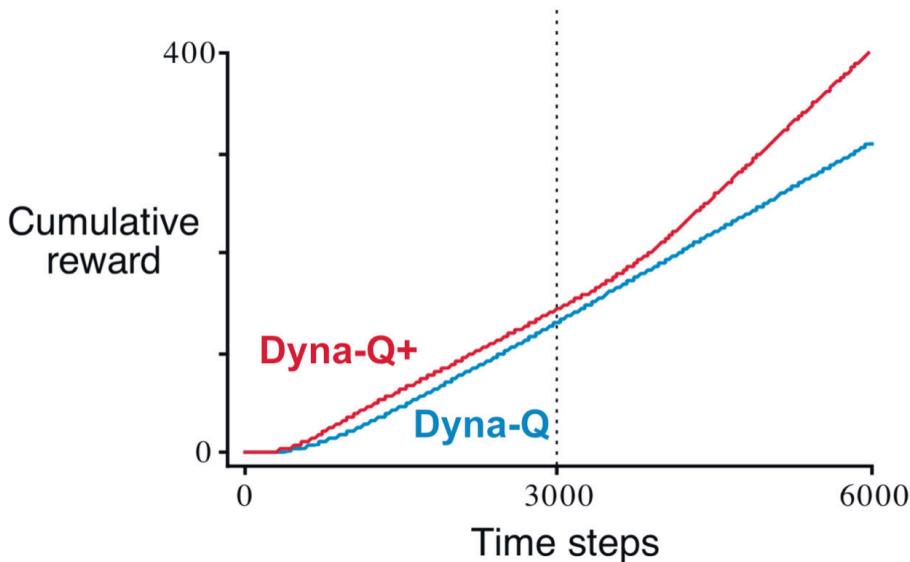
- Select state-action pairs uniformly at random from all previously experienced pairs.
- Start backwards from state-action pairs that have had a non-zero update (e.g., from the state right beside a goal state). This avoids the otherwise wasteful computations from state-action pairs which have had no updates.
- Start with state-action pairs enumerated in a fixed order (e.g., in a gridworld, states top-left to bottom-right, actions up, down, left, right)
- All of these are equally good/bad.



正确

Correct; such a heuristic allows us to focus the updates on station-action pairs which are expected to have non-zero updates. This speeds up the search for the optimal solution, and is the intuition behind backward focusing and prioritized sweeping (check out [Section 8.4](#) of Sutton and Barto's RL textbook).

8. In the lectures, we saw how the Dyna-Q+ agent found the newly-opened shortcut in the shortcut maze, whereas the Dyna-Q agent didn't. Which of the following implications drawn from the figure are TRUE? (Select all that apply)



- The Dyna-Q+ agent performs better than the Dyna-Q agent even in the first half of the experiment because of the increased exploration.



Correct; the increased exploration due to the reward bonus helps the agent discover the path to the goal relatively faster.

The Dyna-Q agent can never discover shortcuts (i.e., when the environment changes to become better than it was before).

The difference between Dyna-Q+ and Dyna-Q narrowed slightly over the first part of the experiment. This is because the Dyna-Q+ agent keeps exploring even when the environment isn't changing.

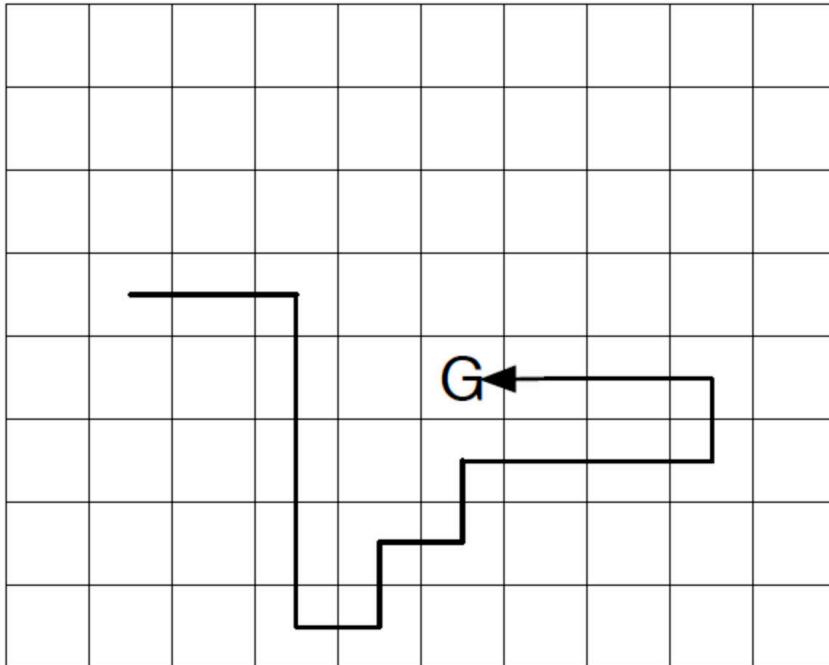
 正确

Correct; such exploration can lead to a slightly suboptimal behaviour even if the optimal policy has been learned for a stationary environment.

None of the above are true.

9. Consider the gridworld depicted in the diagram below. There are four actions corresponding to up, down, right, and left movements. Marked is the path taken by an agent in a single episode, ending at a location of high reward, marked by the G. In this example the values were all zero at the start of the episode, and all rewards were zero during the episode except for a positive reward at G.

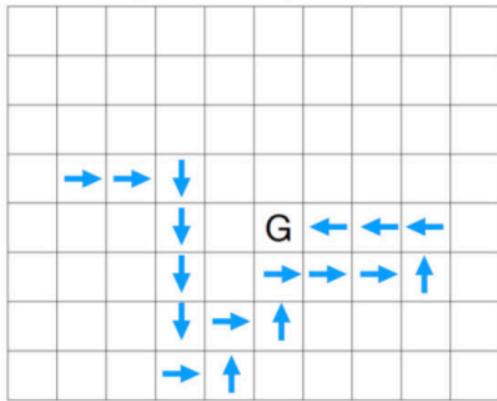
Path taken



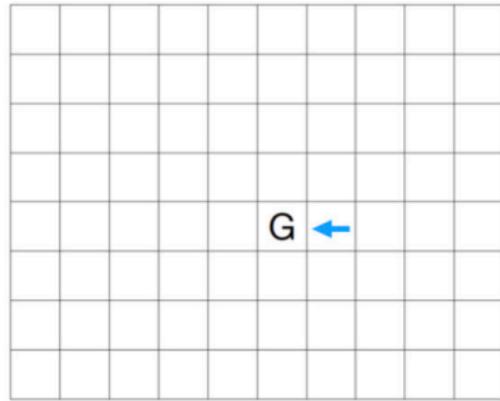
Now which of the following figures best depicts the action values that would've increased by the end of the episode using **one-step Sarsa** and **500-step-planning Dyna-Q**? (Select the best option)

○

Action values increased
by one-step Sarsa

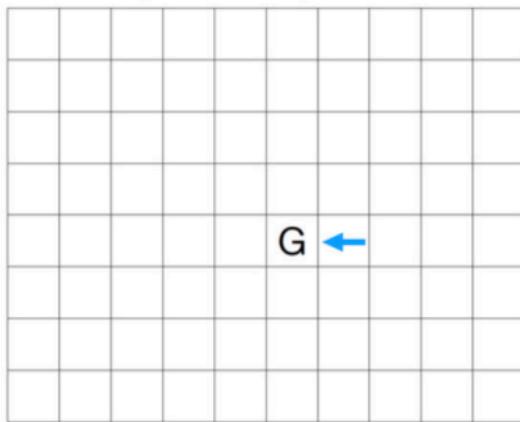


Action values increased
by Dyna-Q (500 planning steps)

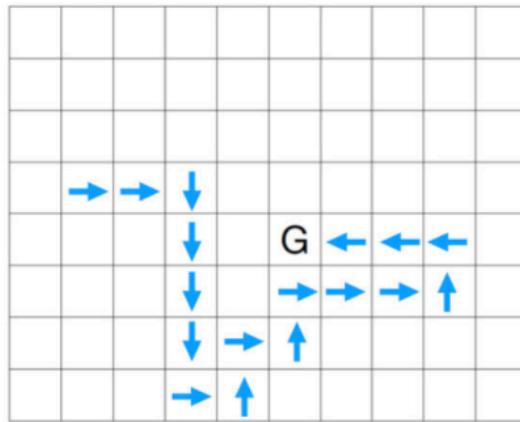




Action values increased
by one-step Sarsa

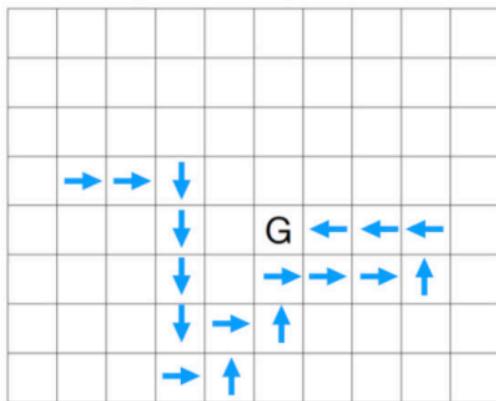


Action values increased
by Dyna-Q (500 planning steps)

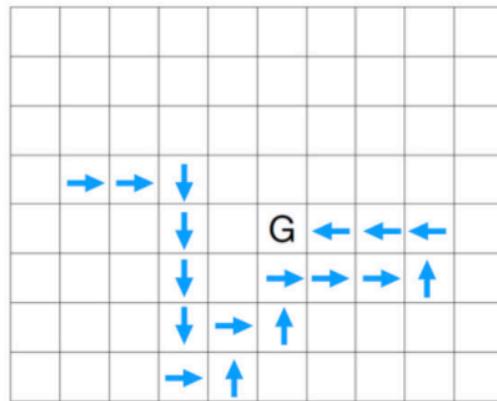


○

Action values increased
by one-step Sarsa

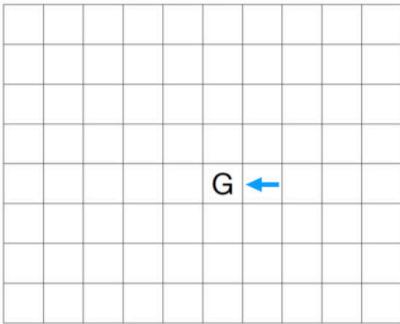


Action values increased
by Dyna-Q (500 planning steps)

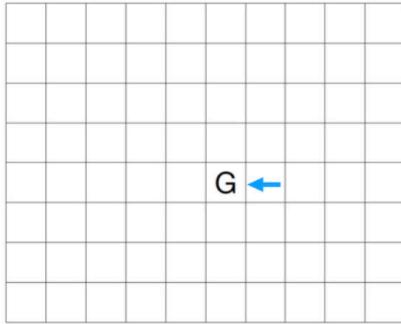




Action values increased
by one-step Sarsa



Action values increased
by Dyna-Q (500 planning steps)



正确

Correct; one-step Sarsa would make a single non-zero update for the state-action pair leading to the goal state, but 500 planning steps would lead to more non-zero steps along this trajectory.

10. Which of the following are planning methods? (Select all that apply)

1 / 1 分

↳ use experience from model.

Value Iteration

正确

Correct; Value Iteration is a Dynamic Programming method that uses a model to improve the policy.

Q-learning

Expected Sarsa

Dyna-Q

正确

Correct; Dyna-Q combines model-free Q-learning with planning. It uses both the experience from the environment as well as simulated experiment from the model in order to make updates to improve the policy.

Sample-based learning methods

Monte Carlo

$$R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$$

TD Learning

$$R_{t+1} + \gamma \underline{V(S_{t+1})}$$



From state-values to action values

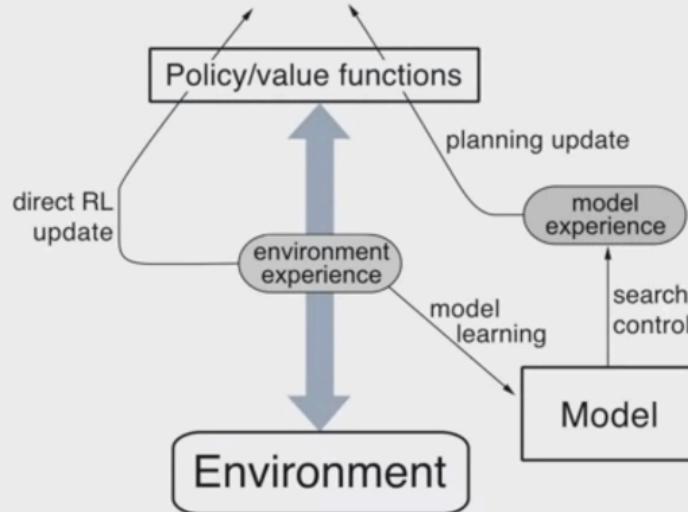
$$V(S_t) \leftarrow V(S_t) + \alpha [\hat{G}_t - V(S_t)] \quad \longrightarrow \quad Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha [\hat{G}_t - Q(S_t, A_t)]$$

Sarsa: $\hat{G}_t = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$ *on policy*

Q-learning: $\hat{G}_t = R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a')$ *off policy* (special case of Expected Sarsa)

Expected Sarsa: $\hat{G}_t = R_{t+1} + \gamma \sum_{a'} \pi(a' | S_{t+1}) Q(S_{t+1}, a')$ *on or off policy*.

Planning with a model



Dyna-Q
Q-learning + Q-planning

Dyna-Q+
Reward bonus in
planning updates