

Bootstrapping

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

$$= \underbrace{R_{t+1} + \gamma G_{t+1}}_{\Rightarrow \text{ writing the return function recursively}}$$

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s]$$

$$= \mathbb{E}_\pi[R_{t+1} + \gamma \boxed{G_{t+1}} | S_t = s]$$

$$= \underbrace{R_{t+1} + \gamma v_\pi(S_{t+1})}_{\Rightarrow \text{ writing the value function recursively as well}}$$

Temporal Difference

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

$$V(S_t) \leftarrow V(S_t) + \alpha [G_t - V(S_t)]$$

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

the TD error, always represented by δ_t

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

The TD target: (S_t, R_{t+1}) state & value (return).

事实上，我们以前在动态编程中也做过类似的事情。在DP中，我们朝着所有可能的下一个状态的价值更新它。主要区别在于，在DP中，我们使用所有可能的下一个状态的期望值。我们需要一个环境模型来计算这个期望值，在TD中我们只需要下一个状态。我们可以直接从环境中获得，而不需要一个模型。让我们来谈谈我们如何直接从环境中获得下一个状态。把时间t加1看作是当前的时间步骤，把时间t看作是上一个时间步骤。因此，我们简单地存储前一个时间步骤的状态，以便进行我们的TD更新。我们看到一个经验流：状态、行动、奖励、下一个状态等等。从时间t的状态，我们可以采取一个行动，并观察时间t加1的下一个状态。只有这样，我们才能更新之前状态的值。现在我们可以全面描述表格中的TD零点算法。TD把要评估的政策作为输入，它还需要一个步长参数和一个价值函数的初始估计。每个情节都从某个初始状态S开始，从那里开始，代理人根据它的政策采取行动，直到它到达终端状态。在情节的每一步，我们用TD学习规则更新数值。我们只需要跟踪以前的状态来进行更新。这就是TD零点。许多算法和强化学习都是基于TD的。如果你不明白这个算法到底是怎么运作的，不要担心。在接下来的几期节目中，我们将通过几个例子来说明TD的作用。今天就讲到这里。在这段视频中，我们介绍了时差学习，它使用的是回报率的自举估计。我们了解了TD误差，并讨论了TD零算法。

How we can get the next state directly from the environment?

1-Step TD

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

the previous timestep

the current timestep

$S_t \leftarrow S_{t+1}$ \Rightarrow ; we simply store the state from the previous timestep in order to make our TD updates.

update

$S, A, R, S, A, R, S, A, R, S, A, R, \dots$

From the state of time t we can take an action, and observe the next state at time $t+1$.

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated

Algorithm parameter: step size $\alpha \in (0, 1]$

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

Loop for each episode:

Initialize $S \Rightarrow$ *beginning episode begins in some initial state S*

Loop for each step of episode:

$A \leftarrow$ action given by π for S

Take action A , observe $R, S' \Rightarrow$ *the agent takes action according to its policy until it reaches the terminal state*

$V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$

2) update the values with the TD learning rule on each step of the episode

$S \leftarrow S'$

until S is terminal

预测学习 (Prediction learning) 意味着你已经试图预测接下来会发生什么，你做了一个预测，你等待，你看会发生什么，然后当你发现发生了什么，你就学习。正因为如此，因为你只需要等待，你不需要训练集。你不需要人们做大量的工作和准备东西。你甚至不需要一个目标。

预测学习是唯一最可扩展的一种学习方式。而 时空差异学习 (temporal difference learning) 是一种专门用于学习预测的学习。那么，这可能意味着什么？学习预测有什么特别之处？嗯，一个是时间过去了，你做了一个预测。实际上，你每时每刻都在做一系列的预测。你做了一个又一个预测，然后，你实际上发现了所有这些预测的正确答案是什么。当你发现实际发生了什么。因此，为了处理这种时间性，这种时间性结构的后果，你需要有专门的学习规则，这就是为什么我们有时间性差异学习。

现在，我相信你也已经了解或听说过监督学习 (supervised learning) 的方法。监督下的学习意味着有某种监督者告诉你答案是什么。也许这发生在学校，但在正常生活中，我们没有监督者或导师告诉我们该怎么做。但在预测学习中，我们确实在这样做。在预测学习中，你做了一个预测，然后你只是等待，你会发现会发生什么。这是很特别的。因此，你可以说预测学习是有监督的，因为你等待，你发现发生了什么，然后这就指导你。你也可以认为它是无监督的，因为你不必有一个监督者，你只是等待，你发现到事件的自然过程，你应该说什么。因此，我有时喜欢认为预测学习是无监督的监督学习 (unsupervised supervised learning)。

时差学习 (Temporal difference learning) , 记住是一种专门用于预测学习的方法, 用于多步骤预测学习。现在让我说说关于监督学习的另一件事。你可能经常听到有人把监督学习描述为预测学习, 从某种意义上说, 它确实是。但最重要的是, 它不是。它是当你做一个预测, 然后告诉你答案是什么。它不是你真正需要等待并看看会发生什么。监督学习通常是训练集, 或者, 你可以在特殊情况下使用预测学习, 即你只试图预测一步, 然后你可以做一个预测, 等待一步, 看到结果, 你可以把它当作一个传统的监督学习问题。但我们大多数的预测都不是一步到位的预测。我们只是看发生了什么, 比如说在一天的晚些时候, 或者一秒钟之后, 一个步骤。什么是一步? 一步可能被认为是十分之一秒, 并保证我们在十分之一秒进行预测, 但在我们的生活中更重要, 我们的长期预测。

好吧, 如果预测学习和时间差异学习是一种重要的学习, 你可能期望它可能出现在动物和人的自然系统中。这对我们在人工智能领域的发展几乎是偶然的, 但确实, 时差学习在动物如何学习的理论中发挥了重要作用, 特别是在奖励系统中。时差模型现在是大脑中奖励系统的标准模型。我们可以看到大脑中与TD错误相对应的信号。而这些似乎驱动着学习, 正如时差学习模型所预测的那样。因此, 你可能已经听说过多巴胺, 在动物和哺乳动物中的立场。携带时差误差的东西是多巴胺神经递质。在其他动物中, 它是一种不同的递质, 如在蜜蜂中, 携带时差区的信号被称为八胺 (octopamine) 。但是, 即使在这些非常不同的动物系统中, 昆虫、哺乳动物和灵长类动物, 甚至在海蛞蝓中, 它都是真的一样。

在之前的视频中，我们介绍了时差学习。TD优雅地结合了动态编程和蒙特卡洛方法的关键思想。像动态编程一样，TD方法是自举（bootstrap）的。像蒙特卡洛方法一样，TD可以直接从经验中学习。这种组合比其各部分的总和更强大。

让我们考虑一下下班开车回家的例子。每天，你预测回家需要多长时间。你的预测基于时间、星期几、天气和其他因素。想象一下，你以前曾多次开车回家。从路上可能遇到的各种情况来看，你对回家需要多长时间有一个很好的估计。让我们看看这些对一次回家旅程的估计。一天晚上，当你离开办公室时，你预测需要30分钟才能到家。大约5分钟后，你走出停车场，发现正在下雨。雨中的交通比较缓慢。所以你估计需要35分钟才能到家。15分钟后，你比预计的时间提前离开高速公路，估计现在需要15分钟才能到家。在旅途中的30分钟，你被堵在一辆缓慢的卡车后面，预测还需要10分钟才能到家。10分钟后，你进入家的街道，从那里到家通常需要3分钟。虽然在三分钟后，你就到家了。因此，圆圈中的数字代表了我们对剩余驾驶时间的预测，我们怎样才能改进这些预测？为了了解如何更新我们的预测，我们需要指定奖励。

leave

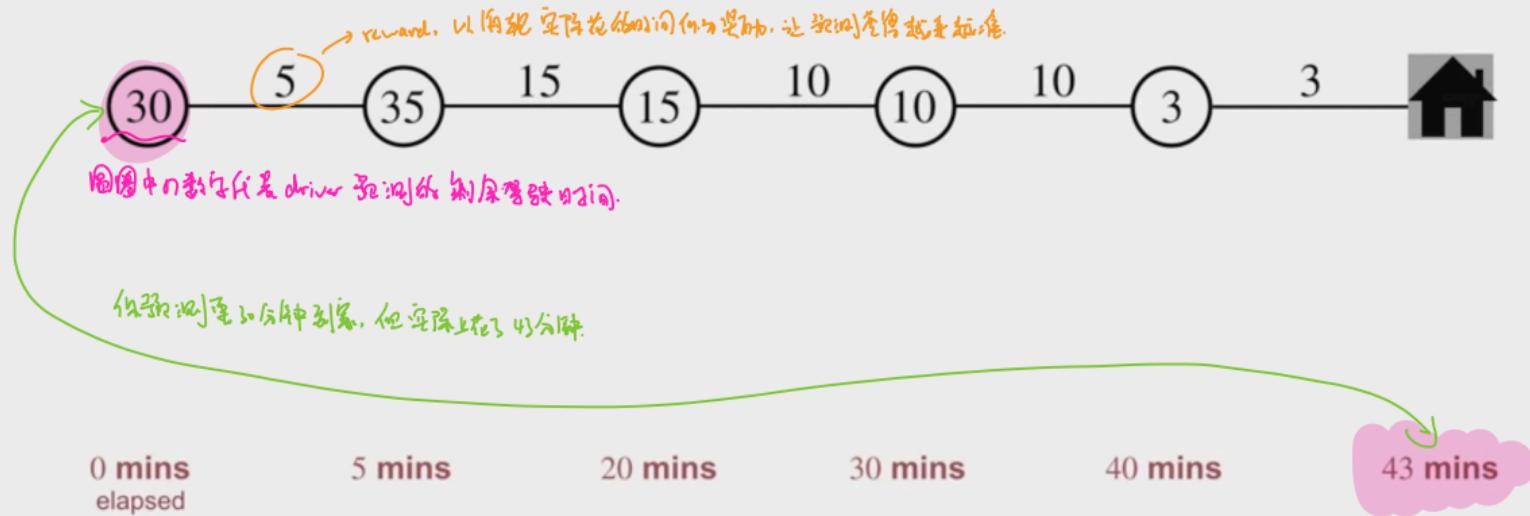
exit

exit highway

secondary road

enter home
street

arrive home



让我们使用所花的时间量作为奖励。你花了5分钟离开停车场。我们可以给这个过渡贴上一个5的奖励。你又花了15分钟离开高速公路，以此类推。你预测要花30分钟才能到家，但实际上你花了43分钟。也许你可以从这个经验中学习。让我们先来看看蒙特卡洛方法。特别是，我们将使用 α 等于1的恒定 α 蒙特卡洛方法。在蒙特卡洛方法中，你向回报率更新你的估计值，而回报率只有在剧情结束时才能得到。我们只能在我们到达家后更新每个状态的估计值。我们可以从离开办公室时开始。从办公室出发，你总共花了43分钟才到家。因此，零点时的回报是43。由于 α 是1，我们把我们的估计完全移向实际返回。从你离开停车场开始，你花了38分钟才到家，我们相应地更新我们的估计。我们以同样的方式更新其余州的估计。但是，真的有必要等到最后的结果出来后再开始学习吗？让我们把时间倒回去，看看我们会如何用TD进行更新。从办公室出发，你花了5分钟离开停车场，从那里你做出了35分钟的预测。从出口状态，你可以更新对办公室的估计。你目前从办公室的估计是30分钟。你得到了5的奖励。下一个状态的值是35，所以我们更新我们的估计值为40分钟。我们来考虑下一个状态。你目前的估计是35分钟。从停车场到出高速路，你花了15分钟。从你离开高速路到回家的估计时间也是15分钟。所以我们把从出停车场时的估计时间减少到30分钟。这是有道理的，因为我们比预期更早地离开了高速公路。我们类似地更新了对其余各州的估计。在我们回家的路上，我们可以在线学习，而不像蒙特卡洛那样等待知道最后的结果。让我们总结一下使用TD的优势。与动态编程不同，TD方法不需要一个环境模型。他们可以直接从经验中学习。与蒙特卡洛不同，TD可以在每一步更新数值。Bootstrapping允许我们在其他估计的基础上更新估计。TD渐进地收敛到正确的预测值。此外，TD方法通常比蒙特卡洛方法收敛得更快。

② Monte Carlo method update:

leave

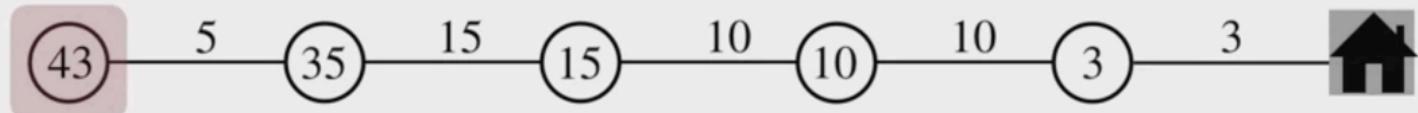
exit

exit highway

secondary road

enter home street

arrive home



$$G_0 = 5 + 15 + 10 + 10 + 3 = 43$$

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$

$$\alpha = 1 \quad \gamma = 1$$

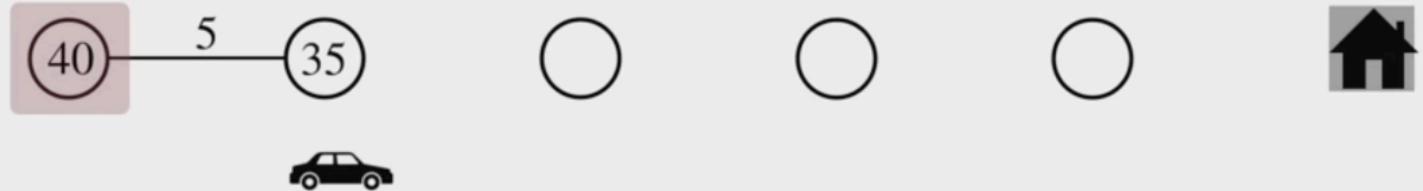
$$V(\text{leave}) \leftarrow V(\text{leave}) + \alpha[G_0 - V(\text{leave})]$$

30 43 30

QTD method update:

leave

exit



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

$$\alpha = 1 \quad \gamma = 1$$

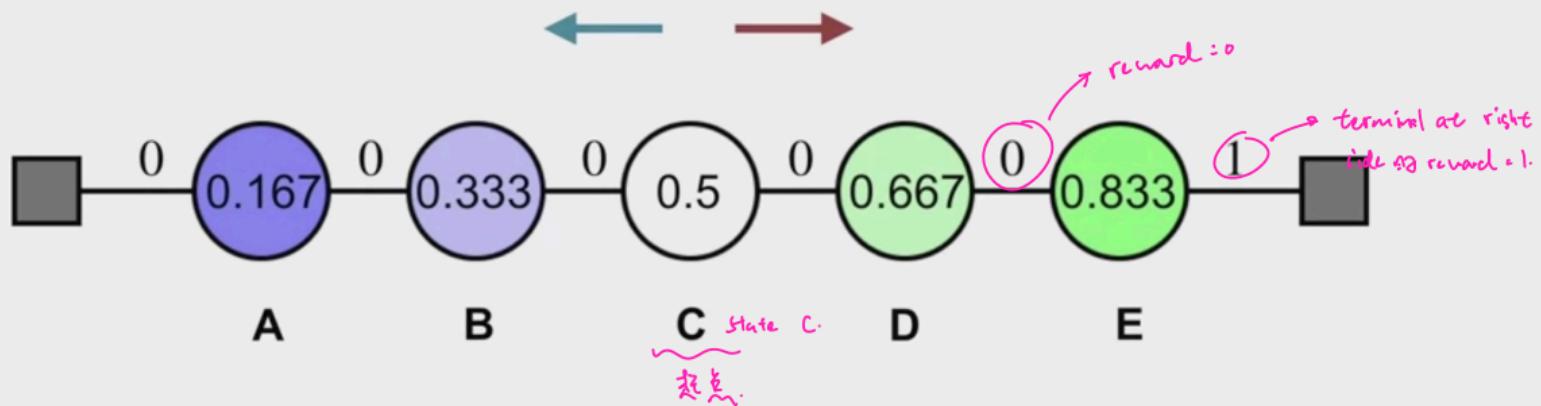
$$V(\text{leave}) \leftarrow V(\text{leave}) + \alpha [R_l + \gamma V(\text{exit}) - V(\text{leave})]$$

30 5 35 30

在过去的几个视频中，我们介绍了TD，并讨论了它比动态编程和蒙特卡洛的优势。在这段视频中，我们将深入研究，在一个精心构建的科学实验中比较TD和蒙特卡洛。

考虑一下这个简单的NDP。我们有五个非终结状态(non-terminal states)。在每个状态下，我们有两个决定性的行动(deterministic actions)，左边和右边。让我们评估一下统一的随机政策。我们想估计它的价值函数。所有的剧情都从状态C开始，剧情要么在左边要么在右边终止。除了在右边终止的奖励是加一之外，在所有的转换中奖励都是零。让我们把折扣系数设为1。在这个问题中，值有一个直观的含义。每个状态的值是指从每个状态开始时在右边终止的概率。开始状态的值是0.5，这意味着从中心终止的概率是一半。从中心随机游走，有50-50的概率在任何一边终止。这很有意义。让我们给其余的状态贴上它们的数值。我们现在就来做个实验吧。我们在幻灯片的顶部给NDP贴上了真实值的标签。代理人，我们可爱的小机器人，用TD估计价值函数。在底部，我们有一个蒙特卡洛代理。让我们把两个代理的近似值函数初始化为0.5。让我们看一下两个代理在第一集的表现。

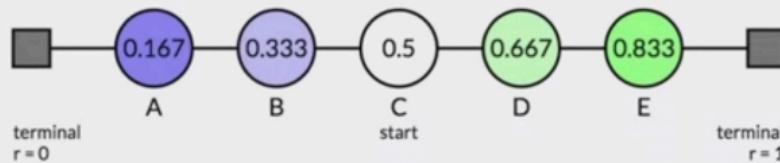
Random Walk



$$\pi(\cdot | s) = 1/2 \quad \forall s \in \mathcal{S} \quad \gamma = 1$$

unif. random policy, 50% left, 50% right.
均匀随机策略，50% 左，50% 右。

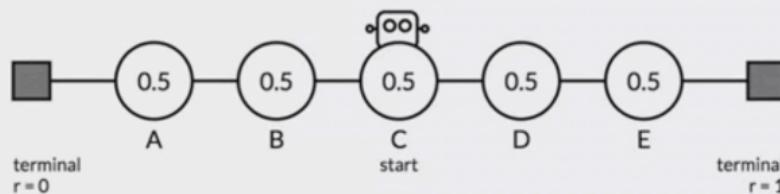
Target / Exact Values



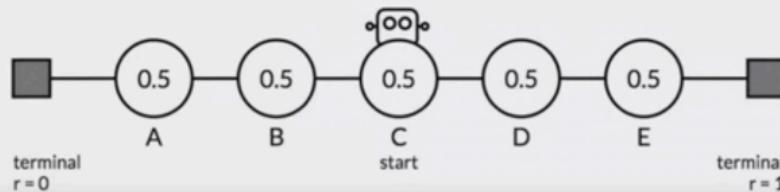
Updates using TD Learning

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

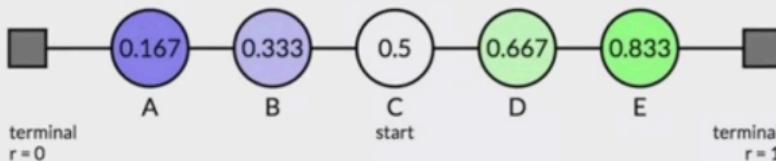
estimate the value functions using TD.



Updates using Monte Carlo



Target / Exact Values after An Episode:



Updates using TD Learning

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

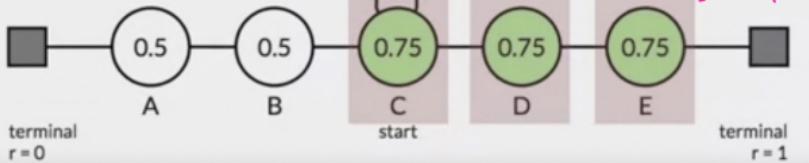
1x 0.5
0.5
因为
The TD Agent only update the value in State E.



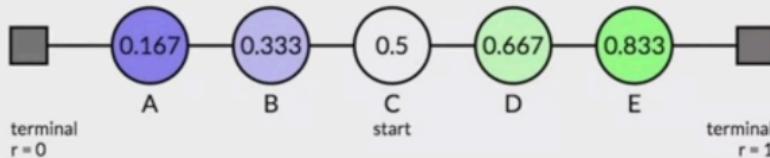
Updates using Monte Carlo

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$

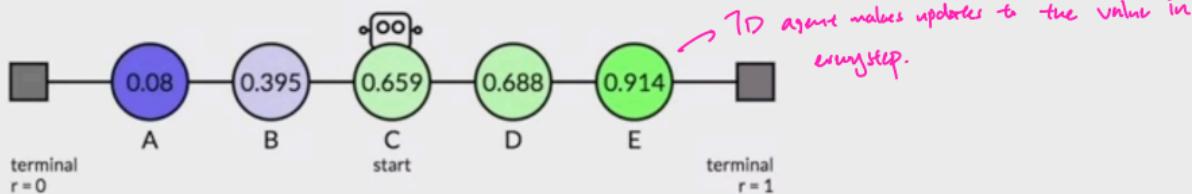
→ update 3 to 8-9 episode of robot 進到終點 state.



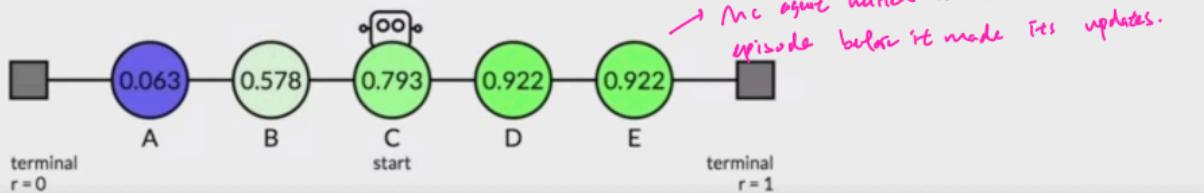
Target / Exact Values *After A few more episodes:*



Updates using TD Learning *→ seem to converge to the true value.*

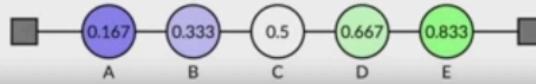
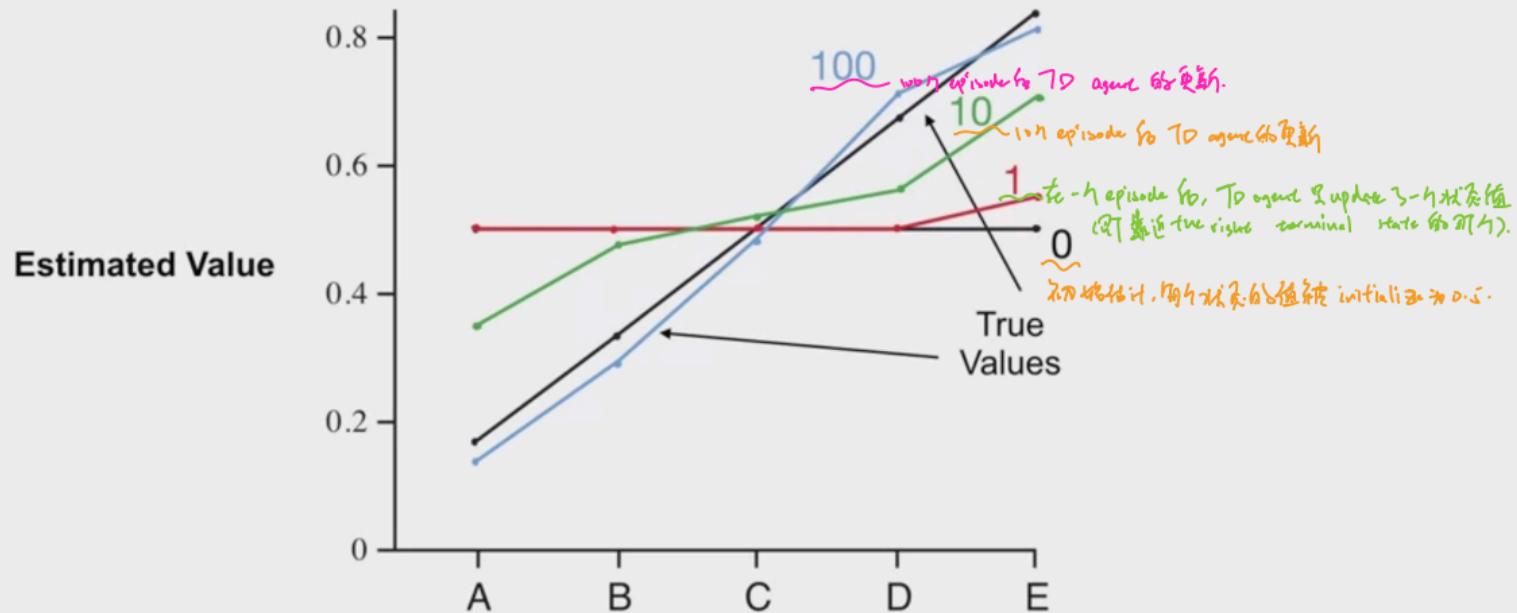


Updates using Monte Carlo

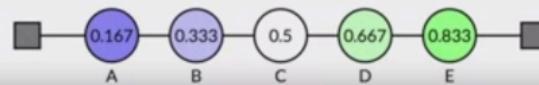
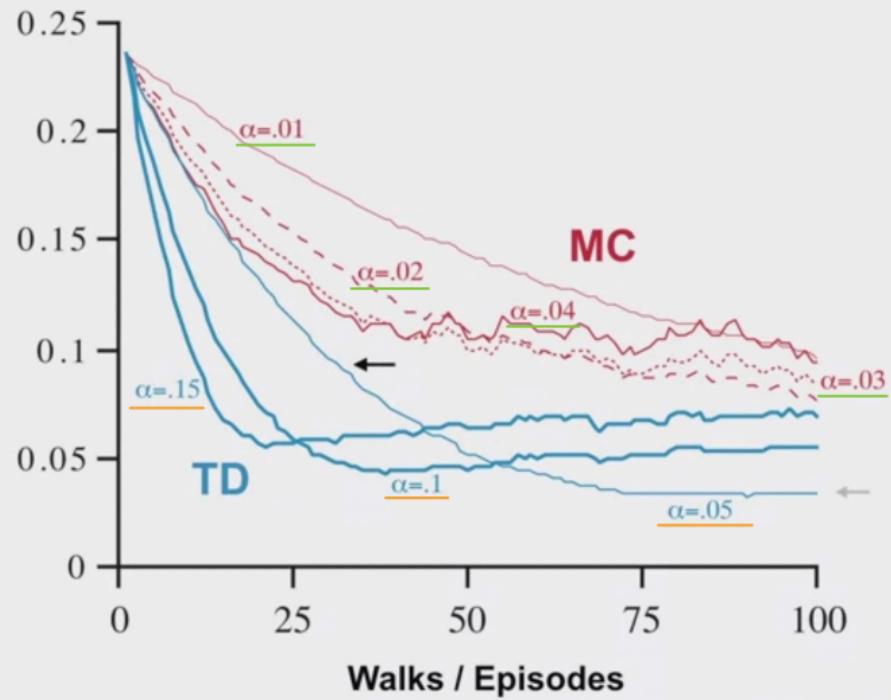


TD代理的价值估计似乎正在向真实价值移动。这花费的时间太长了。让我们跳到前面，评估渐进性能(asymptotic performance)。让我们在学习过程中绘制几个点的估计值。X轴代表NDP中的每个状态。在Y轴上，我们有估计值(estimated value)。这里是真实值。让我们也绘制一下初始值的估计。回顾一下，我们把它们初始化为0.5。红色曲线显示的是第一集之后学到的值。在第一集结束时，TD只更新了一个单一状态的值，正如我们讨论的那样。100集后的估计值与它们所能得到的一样好。记住，我们使用的是0.1的恒定步长。这意味着数值会随着最近几集的结果而波动。如果你使用一个更小的学习率，或者更好的是一个衰减的学习率(decaying learning rate)，我们可能会得到一个更好的估计。

下一个问题是，TD比Monte Carlo学习得更快吗？让我们在我们的例子问题上比较一下TD和Monte Carlo的性能。在这里，X轴代表事件的数量。y轴代表价值函数和学习估计之间的均方根误差(root mean squared error)。红色的学习曲线表示Monte Carlo在几个Alpha值下的表现。每条曲线是100次独立运行的平均数。例如，这一点代表了在学习率为0.01的情况下，50次运行后达到的平均误差。现在，让我们来看看TD的表现。我们看到，TD的表现一直比蒙特卡洛好。让我们再仔细看一下。注意，在学习率为0.15的情况下，误差降低得更快，但最终会导致更高的最终误差。在较小的学习率下，TD的学习速度更慢，但最终误差更低。总之，我们做了一个仔细的实验，比较了TD和蒙特卡洛，结果表明，在这个问题上，TD收敛得更快，最终误差更低。



RMS Error,
averaged over states



在以前的模块中，我们讨论了动态编程和蒙特卡洛。在这个模块中，我们在TD学习中吸取了两者的一些优点。我们首先借用了动态编程的重要概念- -自举(bootstrapping)。TD的更新是针对自举回报的。我们可以用奖励加上我们对下一个状态的估计值, 来代替时间步长后的回报 t 。我们说TD从一个猜测更新一个猜测。它把它的价值估计朝着下一个状态的价值引导。然后我们讨论了表格中的TD零点算法。这里的主要启示是，TD可以在每一步的情节中更新其价值估计。它不需要等待剧情完成。它只需要记住以前的状态。

TD与蒙特卡洛和动态编程相比如何？嗯，TD通常比蒙特卡洛收敛得更快，TD不需要像动态编程那样的模型，而且TD是在线(online)的，完全增量(fully incremental)的，不像蒙特卡洛或动态编程。我们学习了TD如何在收到新信息时更新其预测。我们通过预测下班回家需要多长时间来证明这一点。我们用TD在开车回家的每个阶段不断地完善我们的预测到达时间。最后，我们做了一个实验，比较TD和蒙特卡洛的随机行走。在每次转换中，TD都会更新其价值函数以反映最新的信息。相比之下，蒙特卡洛只在每集结束时更新其价值函数。从长远来看，TD比Monte Carlo学习得更快，并取得了更好的最终误差。

Bootstrapping

$$G_t = R_{t+1} + \gamma G_{t+1}$$

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

it bootstraps its value estimates towards the value of the next state

⇒ TD updates towards a bootstrap return

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated

Algorithm parameter: step size $\alpha \in (0, 1]$

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

Loop for each episode:

 Initialize S

 Loop for each step of episode:

$A \leftarrow$ action given by π for S

 Take action A , observe R, S'

$V(S) \leftarrow V(S) + \alpha [R + \gamma V(S') - V(S)]$ ←

$S \leftarrow S'$

 until S is terminal

To can update its value estimates on each step of the episode.

It doesn't have to wait for the episode to complete!!

It just has to remember the previous state!!

Advantages of TD



Can converge faster than Monte Carlo methods



Does not require a model of the environment *⇒ DP 需要.*



Online and incremental

TD Methods Practice Quiz

最新提交作业的评分 100%

1. TD(0) is a solution method for:

1 / 1 分

Control

Prediction

正确

Correct! TD(0) is used to estimate the value function for a given policy. In other words, it is a solution method for the prediction problem.

2. Which of the following methods use bootstrapping? (Select all that apply)

1 / 1 分

Dynamic Programming

正确

Correct! DP algorithms are obtained by turning Bellman equations into update rules for improving approximations of the desired value functions. These methods update estimates of the values of states based on estimates of the values of successor states. That is, they update estimates on the basis of other estimates.

Monte Carlo

TD(0)

正确

Correct! Temporal Difference methods update “a guess from a guess”. They estimate the value of the current state using the immediate reward and the estimate of the value in the next state. They bootstrap-off their own estimates.

3. Which of the following is the correct characterization of Dynamic Programming (DP) and Temporal Difference (TD) methods?

- Both TD methods and DP methods require a model: the dynamics function p .
- Neither TD methods nor DP methods require a model: the dynamics function p .
- TD methods require a model, the dynamics function p , but Monte-Carlo methods do not.
- DP methods require a model, the dynamics function p , but TD methods do not.



Correct! Dynamic Programming methods solve Bellman equations using a model. TD methods use sample updates from the environment, and do not need to explicitly have the dynamics function p .

4. Which of the following correctly pairs a prediction algorithm and an update?

- Monte Carlo: $V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$



Correct! Monte-Carlo methods update value estimates toward empirically observed returns.

- TD(0): $V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$

- TD(0): $V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$



Correct! TD(0) updates value estimates toward the TD(0)-target of the sum of the observed reward and discounted next state value.

- Monte Carlo: $V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$

5. Which of the following well-describe Temporal Difference (TD) and Monte-Carlo (MC) methods?

1/1分

- TD methods are used in *continuing* tasks.

TD :
 continuing *episodic*
 MC : only episodic

正确

Correct! The returns in continuing tasks are sums of rewards infinitely into the future. But, TD does not have to wait to get samples of these returns. The targets can be obtained immediately, using bootstrapping.

- MC methods are used in *continuing* tasks.

- TD methods are used in *episodic* tasks.

正确

Correct! TD updates on every step, using bootstrapped targets. This means it can be used in continuing and episodic tasks.

- MC methods are used in *episodic* tasks.

正确

Correct! Monte Carlo methods are used in episodic tasks. MC methods use observed returns as targets, obtained by waiting until the end of the episode.

6. In an episodic setting, we might have different updates depending on whether the next state is terminal or non-terminal. Which of the following TD error calculations are correct?

S_{t+1} is non-terminal: $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$

$\therefore V(S_{t+1}) \neq 0$. \therefore 這個狀態的 $V(S_{t+1})$ 是不為零的

正確

Correct! Review the "What is Temporal Difference (TD) learning?" video and in particular the TD(0) algorithm presented therein. The TD target for non-terminal states is indeed the reward plus the discounted value of the next state.

S_{t+1} is non-terminal: $\delta_t = R_{t+1} - V(S_t)$

S_{t+1} is terminal: $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ with $V(S_{t+1}) = 0$

$\therefore V(S_{t+1}) = 0$. $\therefore \gamma V(S_{t+1}) = 0$. \therefore 也可以寫成 $\delta_t = R_{t+1} - V(S_t)$

正確

Correct! Review the "What is Temporal Difference (TD) learning?" video and in particular the TD(0) algorithm presented therein. By using $V(\text{terminal}) = 0$, we can use the usual TD error calculation for non-terminal states.

S_{t+1} is terminal: $\delta_t = R_{t+1} - V(S_t)$

正確

Correct! Review the "What is Temporal Difference (TD) learning?" video and in particular the TD(0) algorithm presented therein. Note in particular that $V(s)$ can be initialized arbitrarily but terminal states should have value 0. Or, in other words, $V(\text{terminal}) = 0$. Thus, we can use that $\gamma V(S_{t+1}) = 0$ and have the TD target for terminal states as simply the reward.

7. Suppose we have current estimates for the value of two states: $V(A) = 1.0$, $V(B) = 1.0$ in an episodic setting. We observe the following trajectory: $A, 0, B, 1, B, 0, T$, where T is a terminal state. Apply $TD(0)$ with step-size, $\alpha = 1$, and discount factor, $\gamma = 0.5$. What are the value estimates for state A and state B at the end of the episode? Provide your answers to 1 decimal place in the following format, replacing $V(A)$ and $V(B)$ with your answers: $(V(A), V(B))$

(0.5,0)



Correct! The steps to the answer are presented below:

After observing $A, 0, B$: *→ this is the first step of the episode & reward. State 0 is the start.*

$$V(A) \leftarrow V(A) + \alpha \cdot [R + \gamma V(B) - V(A)]$$

Simplifying, $V(A) \leftarrow 1.0 + 1 \cdot [0 + 0.5 \cdot 1.0 - 1.0]$.

So, $V(A) \leftarrow 1 + [-0.5]$. Thus, $V(A) \leftarrow 0.5$.

$V(B)$ remains the same.

Therefore, after this transition, $V(A) = 0.5$, $V(B) = 1$.

After observing $B, 1, B$: *→ in this step, the episode & reward = 1.*

$$V(B) \leftarrow V(B) + \alpha \cdot [R + \gamma V(B) - V(B)]$$

Simplifying, $V(B) \leftarrow 1 + 1 \cdot [1 + 0.5 \cdot 1 - 1]$.

So, $V(B) \leftarrow 1 + [0.5]$. Thus, $V(B) = 1.5$.

$V(A)$ remains the same.

Therefore, after this transition: $V(A) = 0.5$, $V(B) = 1.5$.

After observing $B, 0, T$:

$$V(B) \leftarrow V(B) + \alpha \cdot [R + \gamma \tilde{V}(T) - V(B)]$$

$\tilde{V}(T)$: T is terminal, $\therefore V(T) = 0$.

Simplifying, $V(B) \leftarrow 1.5 + 1 \cdot [0 + 0.5 \cdot 0 - 1.5]$ and

$V(B) \leftarrow 1.5 + [-1.5]$. Thus, $V(B) = 0$.

$V(A)$ remains the same.

Therefore, after this transition: $V(A) = 0.5, V(B) = 0$.

Thus the answer is $(0.5, 0.0)$.

8. Which of the following pairs is the correct characterization of the targets used in TD(0) and Monte Carlo?

1/1分

TD(0): High Variance Target, Monte Carlo: High Variance Target

TD(0): High Variance Target, Monte Carlo: Low Variance Target

TD(0): Low Variance Target, Monte Carlo: High Variance Target

TD(0): Low Variance Target, Monte Carlo: Low Variance Target



正确

Correct! MC targets generally have higher variance while TD(0) targets usually have lower variance.

9. Suppose you observe the following episodes of the form (State, Reward, ...) from a Markov Decision Process with states A and B:

Episodes
A, 0, B, 0 \Rightarrow Starts in state A, transitioned to B with $r=0$, and then terminates from B with $r=0$.
B, 1
B, 1
B, 1
B, 0
B, 0
B, 1
B, 0

What would batch Monte Carlo methods give for the estimates $V(A)$ and $V(B)$? What would batch TD(0) give for the estimates $V(A)$ and $V(B)$? Use a discount factor, γ , of 1.

For Batch MC: compute the average returns observed from each state. For Batch TD: You can start with state B. What is its expected return? Then figure out $V(A)$ using the temporal difference equation:
 $V(S_t) = E[R_{t+1} + \gamma V(S_{t+1})]$.

Provide your answers to 1 decimal place in the following format:
 $(V^{\text{batch-MC}}(A), V^{\text{batch-MC}}(B), V^{\text{batch-TD}}(A), V^{\text{batch-TD}}(B))$ and replace

- $V^{\text{batch-MC}}(A) = \frac{0}{4} = 0$
- $V^{\text{batch-MC}}(B) = \frac{(1+1+1+1+0+0+0+0)}{8} = 0.5$
- $V^{\text{batch-TD}}(A)$, and $V(A) = E[R_{t+1} + \gamma V(S_{t+1})] = 0 + 1 \times 0.5 = 0.5$
- $V^{\text{batch-TD}}(B)$ $G_{t+8} = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots + \gamma^7 R_{t+8}$
 $= 1 + 1 \times 1 + 1 \times 1 + 1 \times 1 + 1 \times 1 = 4$
 $\frac{4}{8} = 0.5$ (?)

(0,0.5,0.5,0.5)



Correct! See Section 6.3 and Example 6.4 of the textbook for more details.

10. True or False: "Both TD(0) and Monte-Carlo (MC) methods converge to the true value function asymptotically, given that the environment is Markovian."

1/1分

True

False

 正确

Correct! See Section 6.2, "Advantages of TD Prediction methods" in the book.

11. Which of the following pairs is the correct characterization of the TD(0) and Monte-Carlo (MC) methods?

1/1分

Both TD(0) and MC are offline methods.

Both TD(0) and MC are online methods.

TD(0) is an online method while MC is an offline method.

MC is an online method while TD(0) is an offline method.

 正确

Correct! A primary advantage of TD(0) is that it can learn during an episode: it can learn online. However, Monte-Carlo methods need to wait for the episode to end. They cannot update on each step, and so are not online.