Model-Free Prediction (1)

Junni Zou

Institute of Media, Information and Network
Dept. of Computer Science and Engineering
Shanghai Jiao Tong University
http://min.sjtu.edu.cn

Spring, 2021

Institute of Media, Information, and Network

Outline

Table of Contents

Model-Free Reinforcement Learning

- Last lecture
  - Planning by dynamic programming
  - Solve a *known* MDP
- This lecture
  - Model-free prediction
  - Estimate the value function of an *unkown* MDP
- Next lecture
  - Model-free control
  - Optimize the value function of an *unkown* MDP

Table of Contents

Monte-Carlo Reinforcement Learning

- MC methods learn directly from episodes of experience
- MC is *model-free*: no knowledge of MDP transitions / rewards
- MC learns from simplest possible idea: value = average sample returns
- Caveat: can only apply MC to *episodic* MDPs
  - All episodes must terminate

Monte-Carlo Policy Evaluation

- Goal: learn v_π from episodes of experience under policy π

$$S_1, A_1, R_2, \cdots, S_k \sim \pi$$

- Recall that the *return* is the total discounted reward:

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T$$

- Recall that the value function is the expected return:

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

- Monte-Carlo policy evaluation uses *empirical* average sample returns, instead of *expected* return

First-Visit Monte-Carlo Policy Evaluation

- To evaluate state s
- The first time-step t that state s is visited in an episode,
- Increment counter $N(s) \leftarrow N(s) + 1$
- Increment total return $S(s) \leftarrow S(s) + G_t$
- Value is estimated by average return $V(s) = S(s)/N(S)$
- By law of large numbers, $V(s) \rightarrow v_\pi(s)$ as $N(s) \rightarrow \infty$

First-Visit Monte-Carlo Policy Evaluation (2)

First-visit MC prediction, for estimating $V \approx v_\pi$

Initialize:

$\pi \leftarrow$ policy to be evaluated

$V \leftarrow$ an arbitrary state-value function

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Repeat forever:

Generate an episode using π

For each state s appearing in the episode:

$G \leftarrow$ return following the first occurrence of s

Append G to $Returns(s)$

$V(s) \leftarrow \text{average}(Returns(s))$

Every-Visit Monte-Carlo Policy Evaluation

- To evaluate state s
- Every time-step t that state s is visited in an episode,
- Increment counter $N(s) \leftarrow N(s) + 1$
- Increment total return $S(s) \leftarrow S(s) + G_t$
- Value is estimated by mean return $V(s) = S(s)/N(s)$
- Again, $V(s) \rightarrow v_\pi(s)$ as $N(s) \rightarrow \infty$

Blackjack Example

- State (200 of them)
  - Current sum (12-21)
  - Dealer's showing card (ace-10)
  - Do I have a "useable" ace? (yes-no)
- Action stick: Stop receiving cards (and terminate)
- Action twist: Take another card (no replacement)
- Reward for stick:
  - $+1$ if sum of cards $>$ sum of dealer cards
  - 0 if sum of cards $=$ sum of dealer cards
  - -1 if sum of cards $<$ sum of dealer cards
- Reward for twist:
  - -1 if sum of cards > 21 (and terminate)
  - 0 otherwise
- Transitions: automatically twist if sum of cards < 12

Blackjack Example (2)

状态空间：（多达200种，根据对状态的定义可以有不同的状态空间，这里采用的定义是牌的分数，不包括牌型）

1. 当前牌的分数（12 - 21），低于12时，你可以安全的再叫牌，所以没意义。
2. 庄家出示的牌（A - 10），庄家会显示一张牌面给玩家
3. 我有“useable” ace吗？（是或否）A既可以当1点也可以当11点。

行为空间：

1. 停止要牌 stick
2. 继续要牌 twist

奖励（停止要牌）：

1. +1：如果你的牌分数大于庄家分数
2. 0：如果两者分数相同
3. -1：如果你的牌分数小于庄家分数

奖励（继续要牌）：

1. -1：如果牌的分数>21，并且进入终止状态
2. 0：其它情况

状态转换（Transitions）：如果牌分小于12时，自动要牌

当前策略：牌分只要小于20就继续要牌。

求解问题：评估该策略的好坏。

Blackjack Example (3)

求解过程：使用庄家显示的牌面值、玩家当前牌面总分值来确定一个二维状态空间，区分手中有无A分别处理。统计每一牌局下决定状态的庄家和玩家牌面的状态数据，同时计算其最终收获。通过模拟多次牌局，计算每一个状态下的平均值

我们仅研究初始状态下庄家一张明牌为4，玩家手中前两张牌和为15的情形，不考虑A牌。在给定策略下，玩家势必继续要牌，则可能会出现如下多种情形：

Blackjack Example (4)

庄家总序列	玩家最终序列	玩家获得奖励	当前估计的状态价值
4, 10, 3	Q, 5, 5	+1	+1
4, J, 7	9, 6, 8	-1	0
4, 10, 2, 8	7, 8, 9	+1	0.333
4, 4, 2, 7	J, 5, Q,	-1	0
4, 9	5, K, 4, 8	-1	-0.2
4, 3, K	8, 7, 5	+1	0
...	...	...	...

Institute of Media, Information, and Network

Blackjack Example (5)

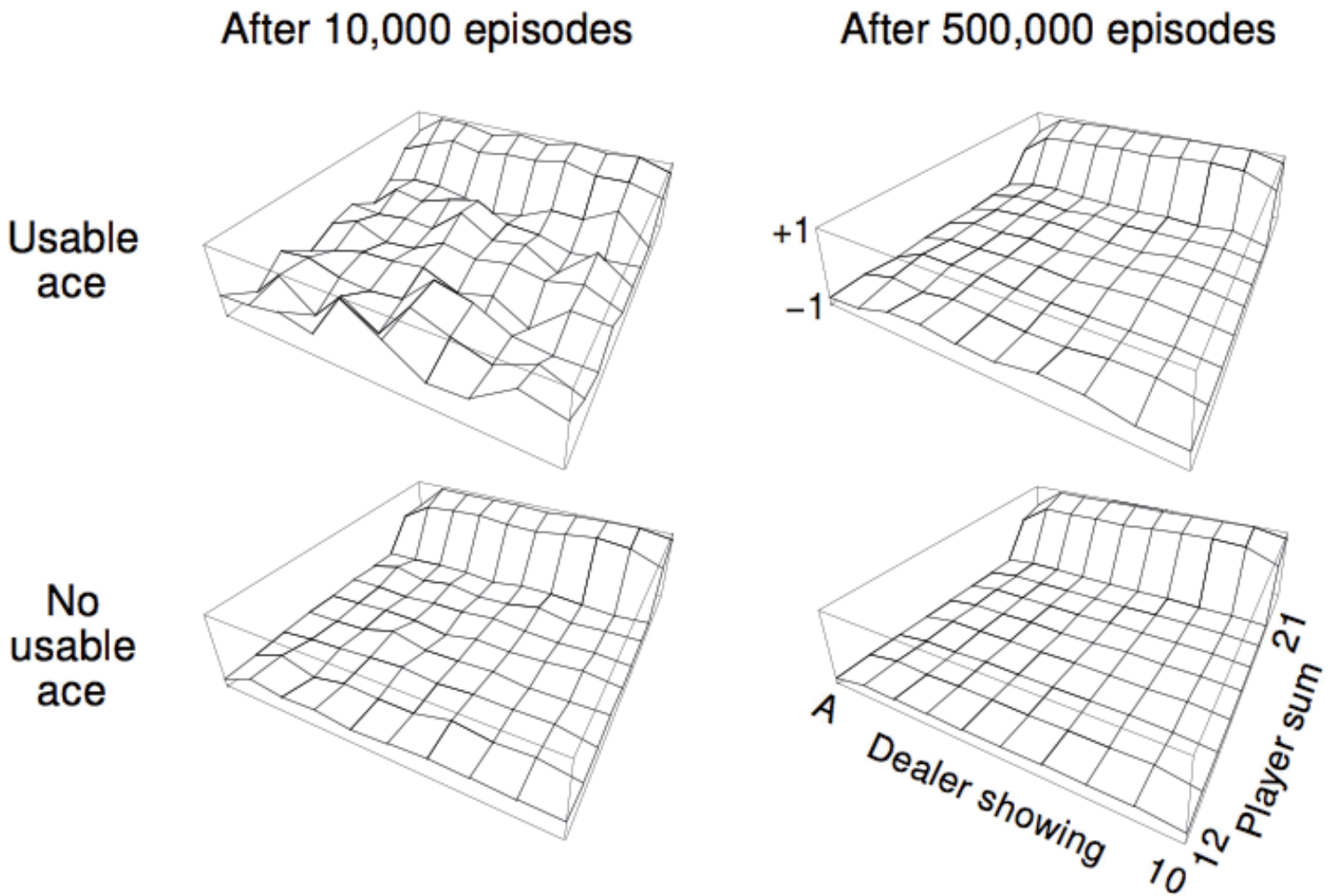
可以看到，使用只有当牌不小于20的时候才停止叫牌这个策略，前6次平均价值为0，如果玩的牌局足够多，按照这样的方法可以针对每一个状态（庄家第一张明牌，玩家手中前两张牌分值合计）都可以制作这样一张表，进而计算玩家奖励的平均值。通过结果，可以发现这个策略并不能带来很高的玩家奖励。

这里给出表中第一个对局对应的信息序列（Episode）：

S_0 <4,15>, A_0 <要牌>, R_1 <0>, S_1 <4,20>, A_1 <停止要牌>, R_2 <+1>.

可以看出，这个完整的Episode中包含两个状态，其中第一个状态的即时奖励为0，后一个状态是终止状态，根据规则，玩家赢得对局，获得终止状态的即时奖励+1。

Blackjack Value Function after Monte-Carlo Learning



Policy: stick if sum of cards $>$ 20, otherwise twist

Institute of Media, Information, and Network

Incremental Mean

The mean $\mu_1, \mu_2, \cdots$ of a sequence $x_1, x_2, \cdots$ can be computed incrementally,

$$\begin{aligned}
\mu_k &= \frac{1}{k}\sum_{j=1}^{k} x_j \\
&= \frac{1}{k}\left(x_k + \sum_{j=1}^{k-1} x_j\right) \\
&= \frac{1}{k}(x_k + (k-1)\mu_{k-1}) \\
&= \mu_{k-1} + \frac{1}{k}(x_k - \mu_{k-1})
\end{aligned}$$

Institute of Media, Information, and Network

Increment Monte-Carlo Updates

- Update $V(s)$ incrementally after episode $S_1, A_1, R_2, \cdots, S_T$
- For each state S_t with return G_t

$$N(S_t) \leftarrow N(S_t) + 1$$
$$V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)}(G_t - V(S_t))$$

- In non-stationary problems, it can be useful to track a running mean, i.e. forget old episodes.

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

Table of Contents

Temporal-Difference Learning

- TD methods learn directly from episodes of experience
- TD is *model-free*: no knowledge of MDP transitions / rewards
- TD learns from *incomplete* episodes, by *bootstrapping*
- TD updates a guess towards a guess

MC and TD

- Goal: learn v_π online from experience under policy π
- Incremental every-visit Monte-Carlo
  - Update value $V(S_t)$ toward actual return G_t

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

- Simplest temporal-difference learning algorithm: TD(0)
  - Update value $V(S_t)$ toward *estimated* return $R_{t+1} + \gamma V(S_{t+1})$

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

  - $R_{t+1} + \gamma V(S_{t+1})$ is called the TD *target*
  - $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is called the *TD error*

TD(0) Policy Evaluation

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 $V(terminal) = 0$

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'
    $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
    $S \leftarrow S'$
  until S is terminal

Institute of Media, Information, and Network

Driving Home Example

想象一下你下班后开车回家，需要预估整个行程花费的时间。假如一个人在驾车回家的路上突然碰到险情：对面迎来一辆车感觉要和你相撞，严重的话他可能面临死亡威胁，但是最后双方都采取了措施没有实际发生碰撞。如果使用蒙特卡洛学习，路上发生的这一险情可能引发的负向奖励不会被考虑进去，不会影响总的预测耗时；但是在TD学习时，碰到这样的险情，这个人会立即更新这个状态的价值，随后会发现这比之前的状态要糟糕，会立即考虑决策降低速度赢得时间，也就是说你不必像蒙特卡洛学习那样直到他死亡后才更新状态价值，那种情况下也无法更新状态价值。

TD算法相当于在整个返家的过程中（一个Episode），根据已经消耗的时间和预期还需要的时间来不断更新最终回家需要消耗的时间。

Driving Home Example (2)

状态	已消耗时间（分）	预计仍需耗时（分）	预测总耗时（分）
离开办公室	0	30	30
取车，发现下雨	5	35	40
离开高速公路	20	15	35
被迫跟在卡车后面	30	10	40
到达家所在街区	40	3	43
进入家门	43	0	43

这里使用的是**从某个状态预估的到家还需耗时**来**间接**反映某状态的价值

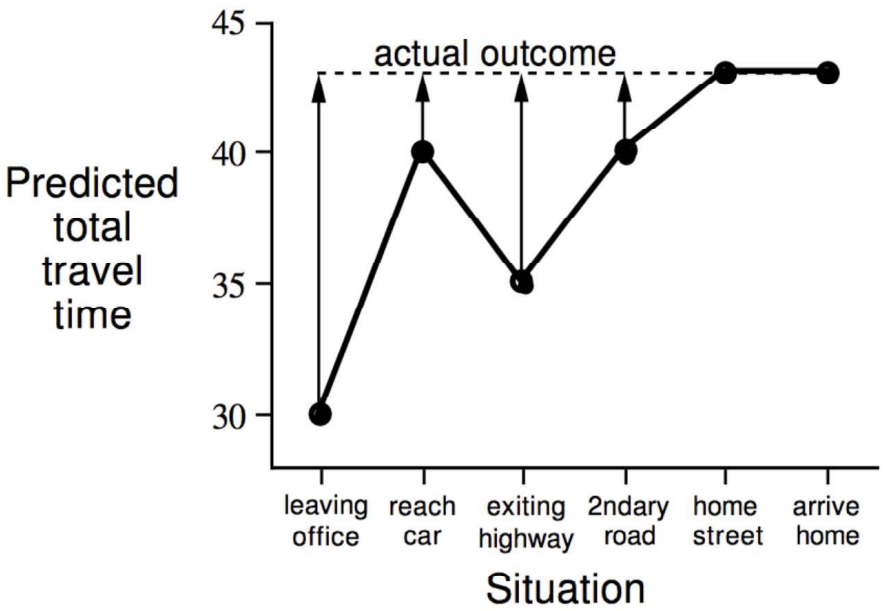
Institute of Media, Information, and Network

Driving Home Example (3)

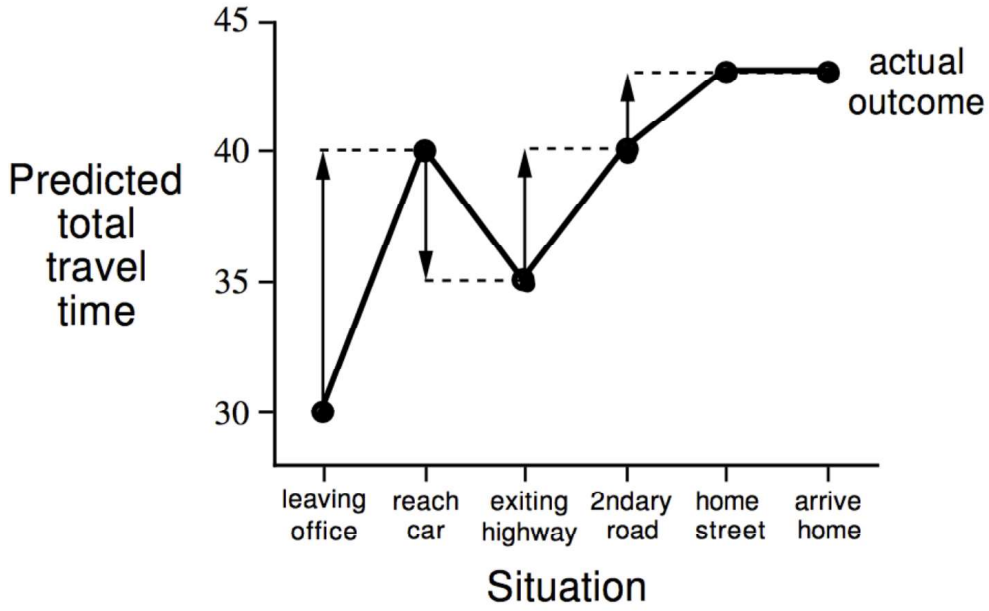
位置预估的到家时间越长，该位置价值越低，在优化决策时需要避免进入该状态。对于蒙特卡洛学习过程，驾驶员在路面上碰到各种情况时，他不会更新对于回家的预估时间，等他回到家得到了真实回家耗时后，他会重新估计在返家的路上着每一个主要节点状态到家的时间，在下一次返家的时候用新估计的时间来帮助决策；而对于TD学习，在一开始离开办公室的时候你可能会预估总耗时30分钟，但是当你取到车发现下雨的时候，你会立刻想到原来的预计过于乐观，因为既往的经验告诉你下雨会延长你的返家总时间，此时你会更新目前的状态价值估计，从原来的30分钟提高到40分钟。同样当你驾车离开高速公路时，会一路根据当前的状态（位置、路况等）对应的预估返家剩余时间，直到返回家门得到实际的返家总耗时。这一过程中，你会根据状态的变化实时更新该状态的价值。

Driving Home Example: MC vs. TD

Changes recommended by Monte Carlo methods (α=1)



Changes recommended by TD methods (α=1)



Advantages and Disadvantages of MC vs. TD

- TD can learn *before* knowing the final outcome
  - TD can learn online after every step
  - MC must wait until end of episode before return is known
- TD can learn *without* the final outcome
  - TD can learn from incomplete sequences
  - MC can only learn from complete sequences
  - TD works in continuing (non-terminating) environments
  - MC only works for episodic (terminating) environments

Bias/Variance Trade-Off

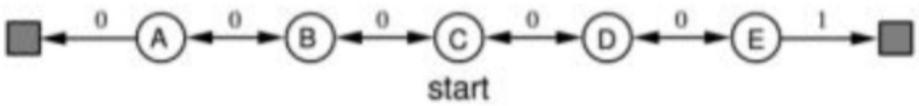
- Return $G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T$ is *unbiased* estimate of $v_\pi(S_t)$
- True TD target $R_{t+1} + \gamma v_\pi(S_{t+1})$ is *unbiased* estimated of $v_\pi(S_t)$
- TD target $R_{t+1} + \gamma V(S_{t+1})$ is *biased* estimated of $v_\pi(S_t)$
- TD target is much lower variance than the return:
  - Return depends on many random actions, transitions, rewards
  - TD target depends on one random action, transition, reward

Advantages and Disadvantages of MC vs. TD (2)

- MC has high variance, zero bias
  - Good convergence properties
  - (even with function approximation)
  - Not very sensitive to initial value
  - Very simple to understand and use
- TD has low variance, some bias
  - Usually more efficient than MC
  - TD(0) converges to $v_\pi(s)$
  - (but not always with function approximation)
  - More sensitive to initial value

Random Walk Example

状态空间：如下图：A、B、C、D、E为中间状态，C同时作为起始状态。灰色方格表示终止状态；



行为空间：除终止状态外，任一状态可以选择向左、向右两个行为之一；

即时奖励：右侧的终止状态得到即时奖励为1，左侧终止状态得到的即时奖励为0，在其他状态间转化得到的即时奖励是0；

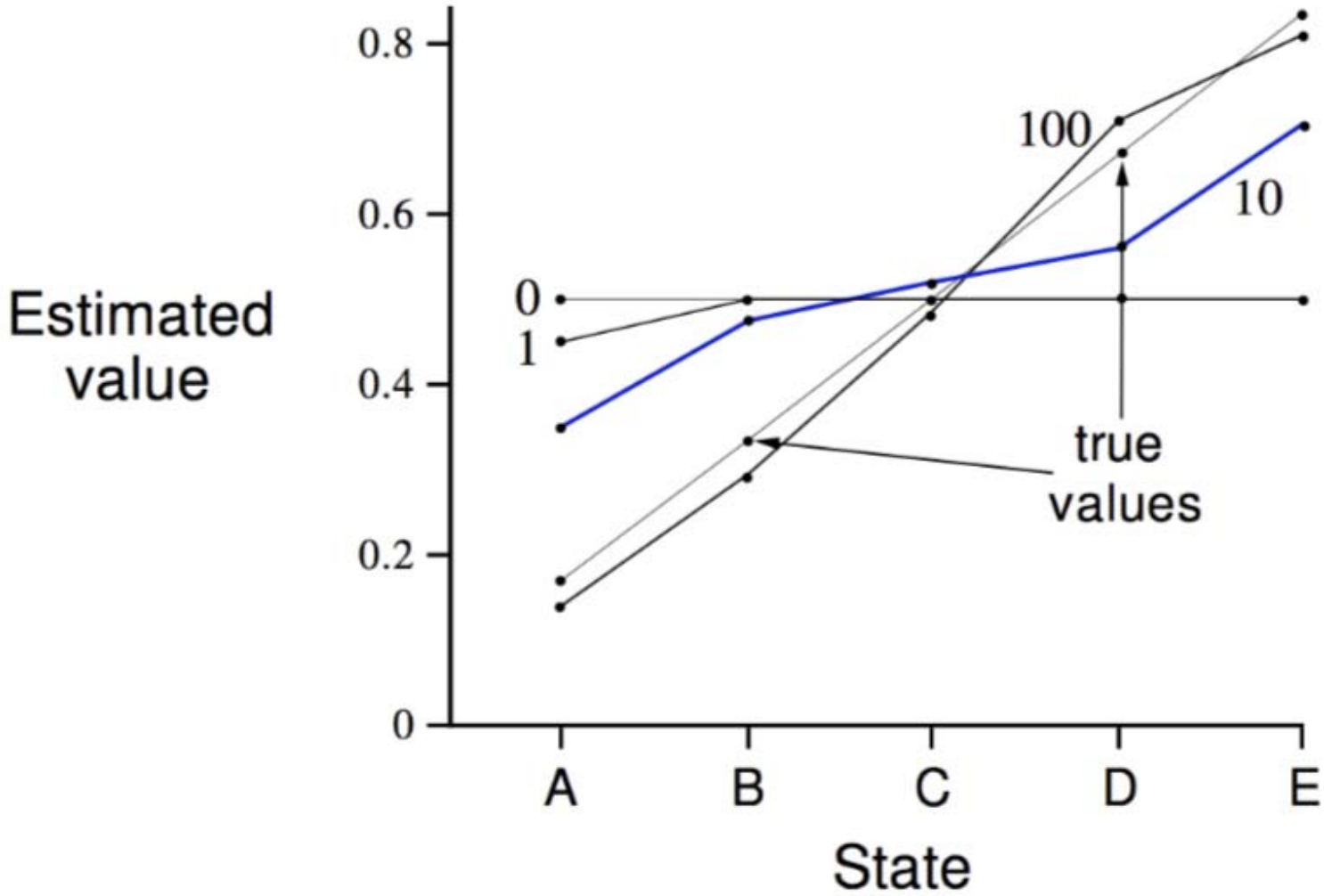
状态转移：100%按行为进行状态转移，进入终止状态即终止；

衰减系数：1；

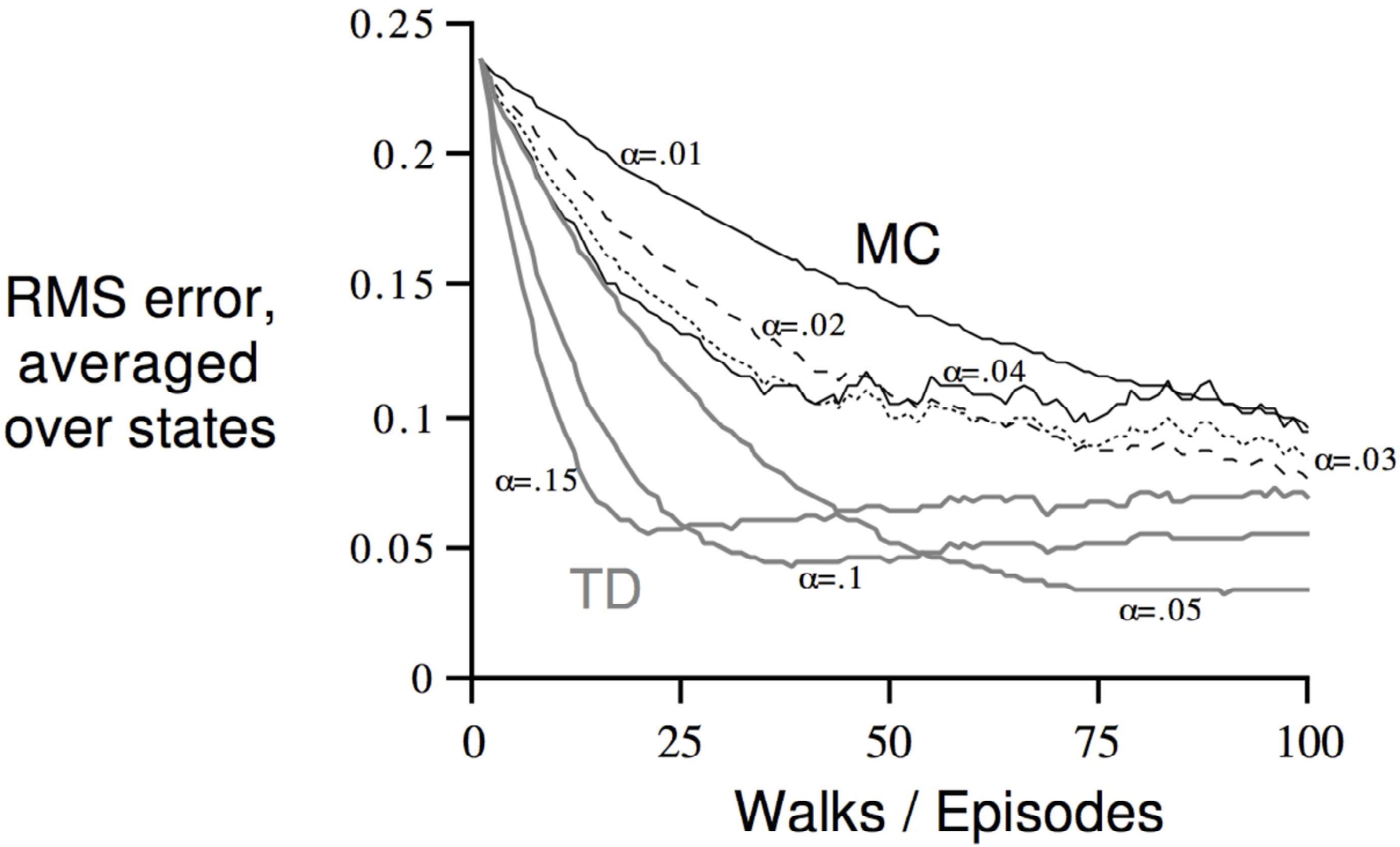
给定的策略：随机选择向左、向右两个行为。

Institute of Media, Information, and Network

Random Walk Example (2)



Random Walk: MC vs. TD



Institute of Media, Information, and Network

Batch MC and TD

- MC and TD converge: $V(s) \to v_\pi(s)$ as experience $\to \infty$
- But what about batch solution for finite experience?

$$s_1^1, a_1^1, r_2^1, \cdots, s_{T_1}^1$$
$$\vdots$$
$$s_1^K, a_1^K, r_2^K, \cdots, s_{T_K}^K$$

  - e.g. Repeatedly sample episode $k \in [1, K]$
  - Apply MC or TD(0) to episode k

Institute of Media, Information, and Network

AB Example (1)

Two states A, B; no discounting; 8 episodes of experience

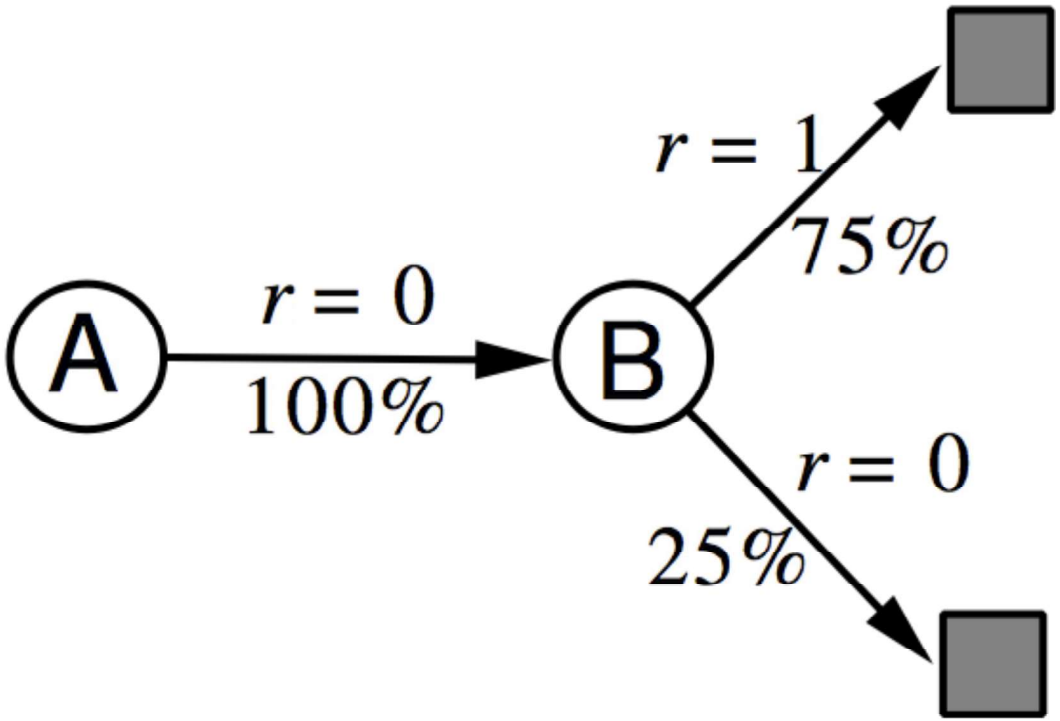
$$\begin{aligned}
&\mathrm{A}, 0, \mathrm{B}, 0 \\
&\mathrm{B}, 1 \\
&\mathrm{B}, 1 \\
&\mathrm{B}, 1 \\
&\mathrm{B}, 1 \\
&\mathrm{B}, 1 \\
&\mathrm{B}, 1 \\
&\mathrm{B}, 0
\end{aligned}$$

What is $V(A), V(B)$?

AB Example (2)

Two states A, B; no discounting; 8 episodes of experience

$\mathrm{A}, 0, \mathrm{B}, 0$
$\mathrm{B}, 1$
$\mathrm{B}, 1$
$\mathrm{B}, 1$
$\mathrm{B}, 1$
$\mathrm{B}, 1$
$\mathrm{B}, 1$
$\mathrm{B}, 0$



What is $V(A), V(B)$?

r

Certainty Equivalence

- MC converges to solution with minimum mean-squared error
  - Best fit to the observed returns

$$\sum_{k=1}^{K}\sum_{t=1}^{T_k}(G_t^k - V(s_t^k))^2$$

  - In the AB example, $V(A) = 0$
- TD(0) converges to solution of max likelihood Markov model
  - Solution to the MDP $\langle \mathcal{S}, \mathcal{A}, \hat{\mathcal{P}}, \hat{\mathcal{R}}, \gamma \rangle$ that best fits the data

$$\hat{\mathcal{P}}_{s,s'}^a = \frac{1}{N(s,a)}\sum_{k=1}^{K}\sum_{t=1}^{T_k}\mathbf{1}(s_t^k, a_t^k, s_{t+1}^k = s, a, s')$$

$$\hat{\mathcal{R}}_s^a = \frac{1}{N(s,a)}\sum_{k=1}^{K}\sum_{t=1}^{T_k}\mathbf{1}(s_t^k, a_t^k = s, a)r_t^k$$

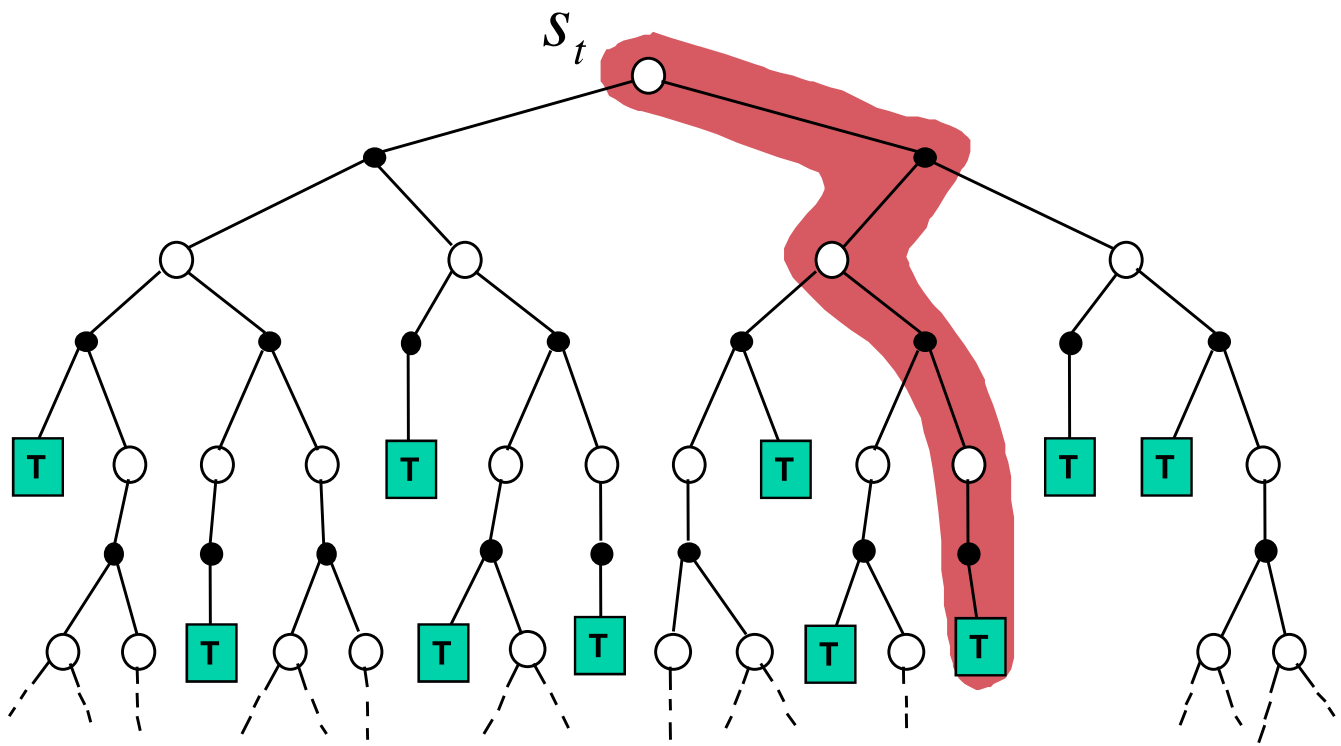
  - In the AB example, $V(A) = 0.75$

Advantages and Disadvantages of MC vs. TD (3)

- TD exploits Markov property
  - Usually more efficient in Markov environments
- MC does not exploit Markov property
  - Usually more effective in non-Markov environments

Monte-Carlo Backup

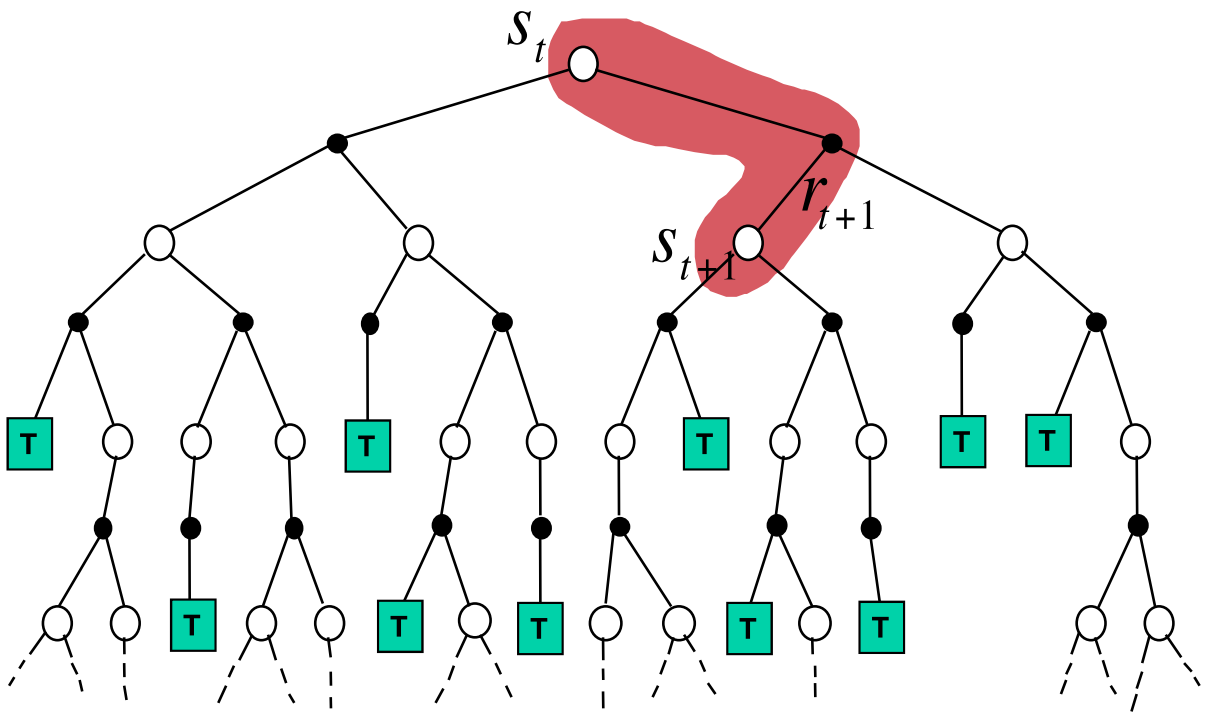
$$V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$$



Institute of Media, Information, and Network

Temporal-Difference Backup

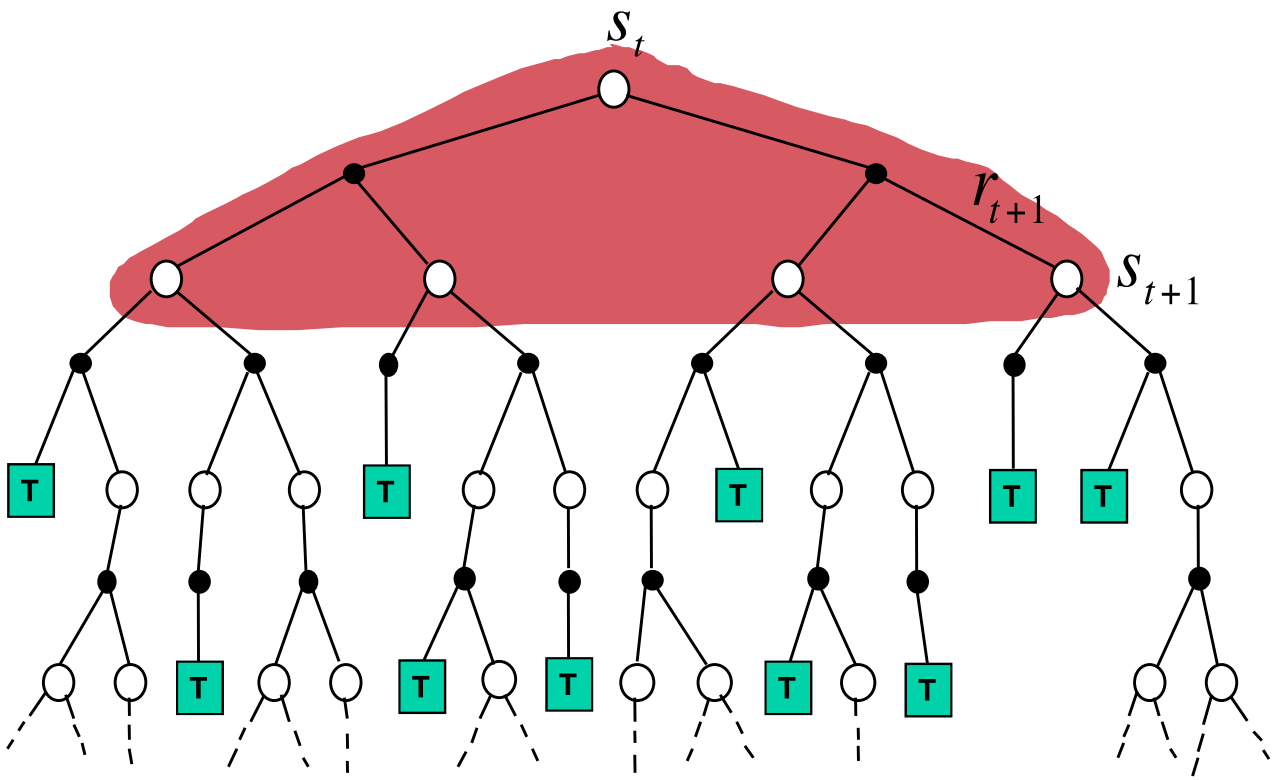
$$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$



Institute of Media, Information, and Network

Dynamic Programming Backup

$$V(S_t) \leftarrow \mathbb{E}_\pi[R_{t+1} + \gamma V(S_{t+1})]$$

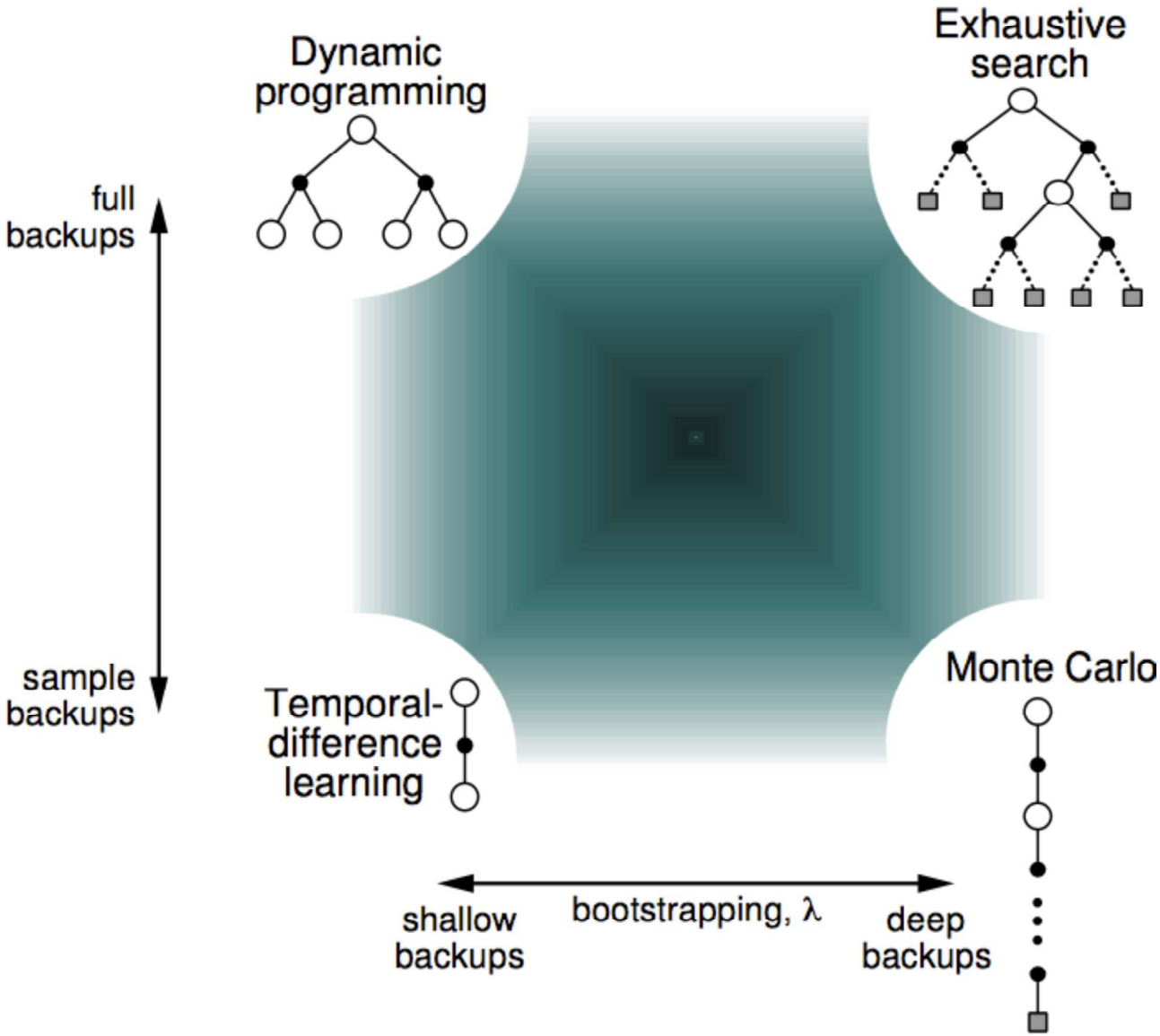


Institute of Media, Information, and Network

Bootstrapping and Sampling

- Bootstrapping: update involves an estimate
  - MC does not bootstrap
  - DP bootstraps
  - TD bootstraps
- Sampling: update samples an expectation
  - MC samples
  - DP does not sample
  - TD samples

Unified View of Reinforcement Learning



ıstitute of Media,
ıformation, and Network