Model Free Prediction

岩延 yany@ucas.ac.cn

Model Free Reinforcement Learning

Introduction

- **Last Chapter**
  - Planning by dynamic programming
  - Solve a **known** MDP
- **Model Free prediction**
  - Estimate the value function of an **unknown** MDP
- **Model Free control**
  - Optimize the value function of an **unknown** MDP

Monte Carlo Methods

Introduction

- **MC methods learn directly from episodes of experience**
- **MC is model-free: no knowledge of MDP transitions / rewards**
- **MC learns from complete episodes: no bootstrapping**
- **MC uses the simplest possible idea: value = mean return**
- **Can only apply MC to episodic MDPs**
  - All episodes must terminate

Monte Carlo Methods

Monte Carlo Prediction

- 目标
  - 使用MC方法从经验中学习value function v_π
- **Return is the total discounted reward:**
  - $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-1} R_T$
- **Value function is the expected return:**
  - $v_\pi(s) \doteq \mathrm{E}_\pi[G_t | S_t = s]$
- **MC**方法实质上就是利用回报的**Empirical Return**去代替**Expected Return**

Monte Carlo Methods

First-Visit Monte-Carlo

- **To evaluate $\boldsymbol{v_\pi(s)}$**
  - The **first** time-step t that state s is visited in an episode,
  - Increment counter $N(s) \leftarrow N(s) + 1$
  - Increment total return $S \leftarrow S + G_t$
  - Value is estimated by mean return $V(s) = S/N(s)$
- 由大数定律可得，当$N(s) \rightarrow \infty$
  - $V(s) \rightarrow v_\pi(s)$

Monte Carlo Methods

Every-Visit Monte-Carlo

- **To evaluate $\boldsymbol{v_\pi(s)}$**
  - **Every** time-step t that state s is visited in an episode,
  - Increment counter $N(s) \leftarrow N(s) + 1$
  - Increment total return $S \leftarrow S + G_t$
  - Value is estimated by mean return $V(s) = S/N(s)$
- 由大数定律可得，当$N(s) \rightarrow \infty$
  - $V(s) \rightarrow v_\pi(s)$

Monte Carlo Methods

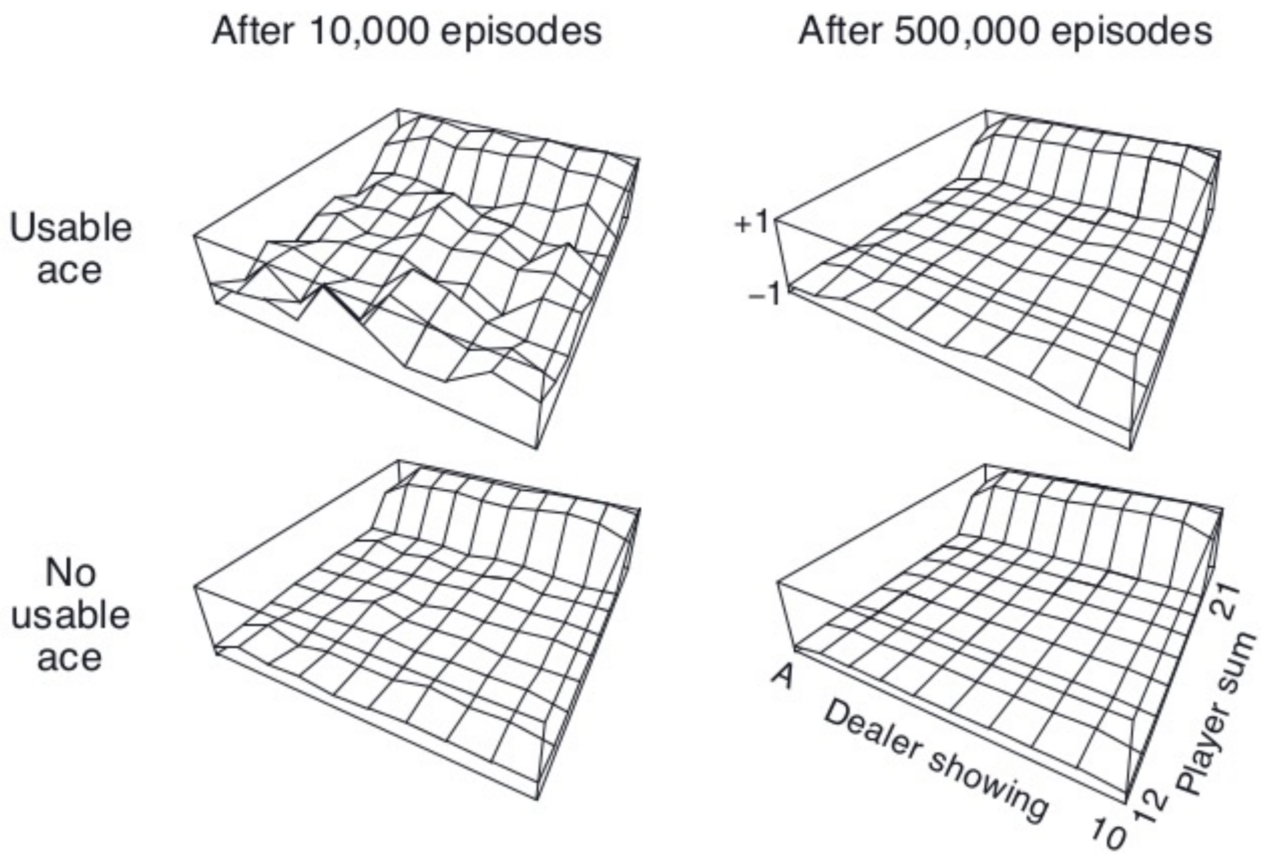
Example

- **二十一点Black Jack**
  - 21点可以看成一个回合式的有限马尔科夫过程（episodic finite MDP）
  - 每次游戏都是一个回合（episode）
  - 赢、输、draw的奖励分别为1、-1、0
  - 游戏过程中的任意动作奖励（reward）都为0
  - No discount （$\gamma = 1$）
  - 玩家的动作（action）只有要牌（hits）或者停止要牌（sticks）两种
  - 如果玩家将A当成11点来算的话（不能爆掉），我们称它*usable*
  - 总共有200个不同的状态

Monte Carlo Methods

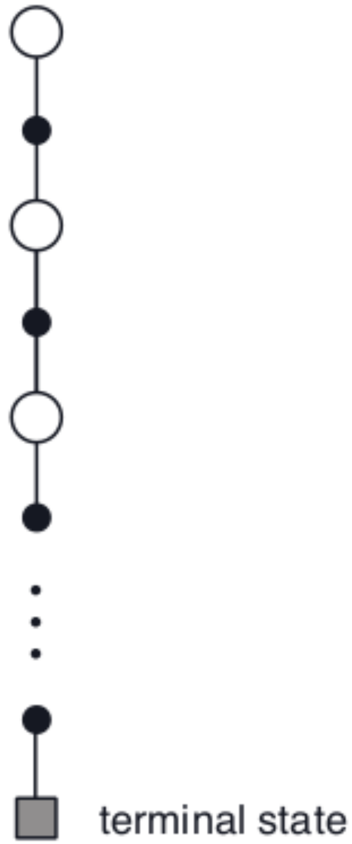
Example

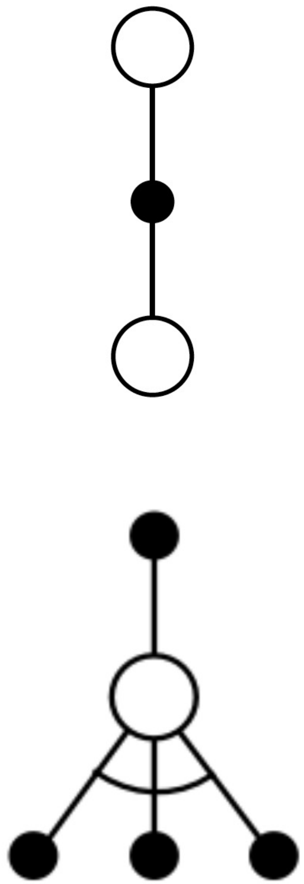
- 二十一点Black Jack
  - 策略：一直要牌，直到点数和等于20或21时停止



Monte Carlo Methods

Backup Diagram of MC





Monte Carlo Methods

Incremental Implementation

- **Incremental Mean**
  - 可以通过增量方式计算平均值
    - $Q_n = \frac{R_1 + R_2 + \cdots + R_{n-1}}{n-1}$
    - $Q_{n+1} = \frac{1}{n} \sum_{i=1}^{n} R_i$
    - $\quad = \frac{1}{n} \left(R_n + \sum_{i=1}^{n-1} R_i \right)$
    - $\quad = \frac{1}{n} \left(R_n + (n-1) \frac{1}{n-1} \sum_{i=1}^{n-1} R_i \right)$
    - $\quad = \frac{1}{n} \left(R_n + (n-1) Q_n \right)$
    - $\quad = \frac{1}{n} \left(R_n + n Q_n - Q_n \right)$
    - $\quad = Q_n + \frac{1}{n} \left(R_n - Q_n \right)$
  - 通用形式为
    - NewEstimate ← OldEstimate + StepSize[Target - OldEstimate]

Monte Carlo Methods

Incremental Monte-Carlo Updates

- **To evaluate $v_\pi(s)$**
  - The first or every time-step t that state s is visited in an episode,
  - Increment counter $N(s) \leftarrow N(s) + 1$
  - Increment total return $S \leftarrow S + G_t$
  - Value is estimated by mean return $V(s) = S/N(s)$
- **增量更新$V(s)$**
  - For each state S_t with return G_t
    - $N(S_t) \leftarrow N(S_t)+1$
    - $V(S_t) \leftarrow V(S_t)+\frac{1}{N(S_t)}(G_t - V(S_t))$
  - 对于non-stationary问题，可以使用固定的步长
    - $V(S_t) \leftarrow V(S_t)+\alpha(G_t - V(S_t))$

Temporal-Difference Learning

Introduction of Temporal-Difference Learning

- **TD直接从episodes的经验中学习**
- **TD不需要完整的episodes**
  - bootstrapping
- **TD是model-free的**
  - 不需要了解MDP transitions / rewards
- **TD updates a guess towards a guess**

Temporal-Difference Learning

TD Prediction

- **目标**
  - 从经验中在线学习策略π的v_π
- **MC方法**
  - 使用实际的G_t更新$V(S_t)$
    - $V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$
- **TD方法**
  - 使用推测的return $R_{t+1} + \gamma V(S_{t+1})$更新$V(S_t)$
    - $V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$
    - $R_{t+1} + \gamma V(S_{t+1})$为TD target
    - 称为TD(0)或one-step TD
- **MC更新的Target是G_t，而TD更新的Target是$R_{t+1} + \gamma V(S_{t+1})$**

Temporal-Difference Learning

TD Prediction

- **与DP类似，TD(0)是bootstrapping方法**
  - $v_\pi(s) \doteq \mathrm{E}_\pi[G_t | S_t = s]$
  - $\quad = \mathrm{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s]$
  - $\quad = \mathrm{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$
  - $\quad = \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r | s, a)[r + \gamma \mathrm{E}_\pi[G_{t+1} | S_{t+1} = s']]$
  - $\quad = \sum_a \pi(a|s) \sum_{s', r} p(s', r | s, a)[r + \gamma v_\pi(s')]$

- $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$**为TD Error**
  - 存在多种形式
  - MC error可以写成TD error之和
  - $G_t - V(S_t) = R_{t+1} + \gamma G_{t+1} - V(S_t) + \gamma V(S_{t+1}) - \gamma V(S_{t+1})$
  - $\quad = \delta_t + \gamma (G_{t+1} - V(S_{t+1}))$
  - $\quad = \delta_t + \gamma \delta_{t+1} + \gamma^2 (G_{t+2} - V(S_{t+2}))$
  - $\quad = \delta_t + \gamma \delta_{t+1} + \gamma^2 \delta_{t+2} + \cdots + \gamma^{T-t}(G_T - V(S_T))$
  - $\quad = \delta_t + \gamma \delta_{t+1} + \gamma^2 \delta_{t+2} + \cdots + \gamma^{T-t}(0)$
  - $\quad = \sum_{k=t}^{T-1} \gamma^{k-t} \delta_k$

TD(0)

Temporal-Difference Learning

Driving Home Example

状态	过去的时间	预测的时间	预测的总时间
周五六点，离开办公室	0	30	30
到车上，下雨	5	35	40
出主路	20	15	35
辅路，卡车后	30	10	40
进入居住街道	40	3	43
到家	43	0	43

Temporal-Difference Learning

Driving Home Example

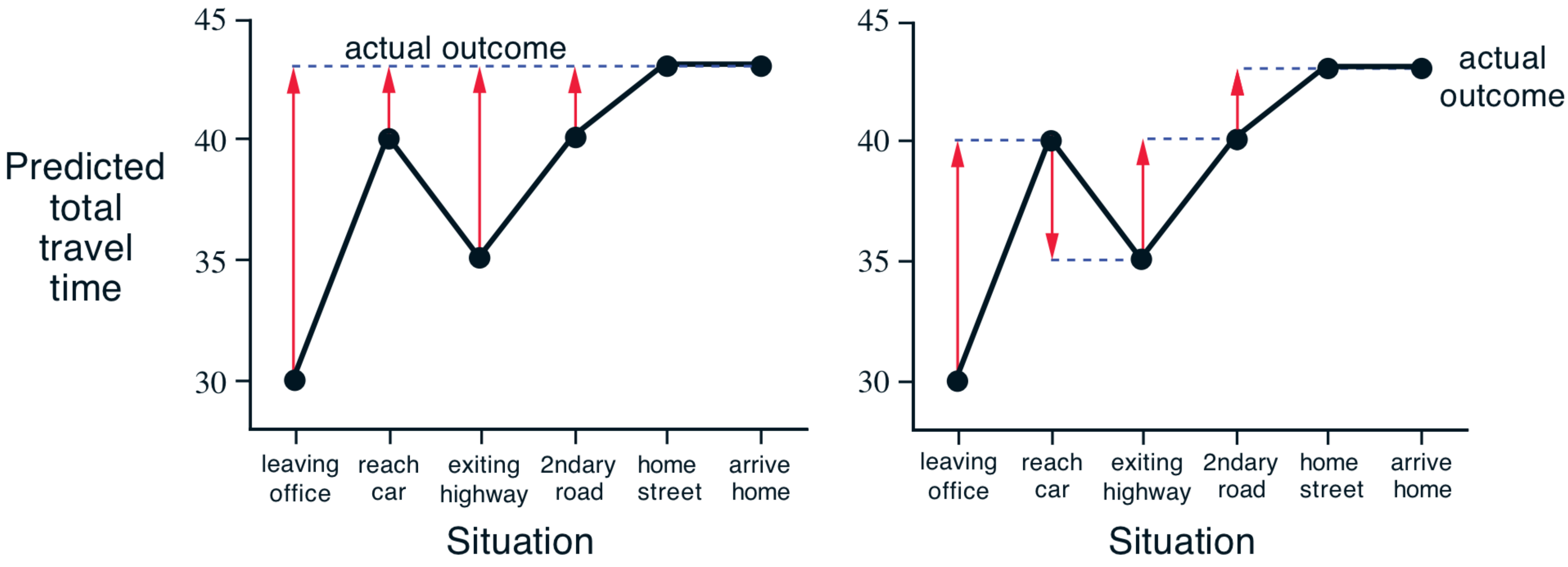


Figure 6.1: Changes recommended in the driving home example by Monte Carlo methods (left) and TD methods (right).

Temporal-Difference Learning

TD Prediction

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 $V(\mathit{terminal}) = 0$

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

Temporal-Difference Learning

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

Temporal-Difference Learning

MC vs TD

- $v_\pi(S_t)$**的Unbiased推测**
  - $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-1} R_T$
  - $R_{t+1} + \gamma v_\pi(S_{t+1})$
- $v_\pi(S_t)$**的biased推测**
  - TD target $R_{t+1} + \gamma V(S_{t+1})$
- **TD target的variance比return低**
  - Return depends on many random actions, transitions, rewards
  - TD target depends on one random action, transition, reward

Temporal-Difference Learning

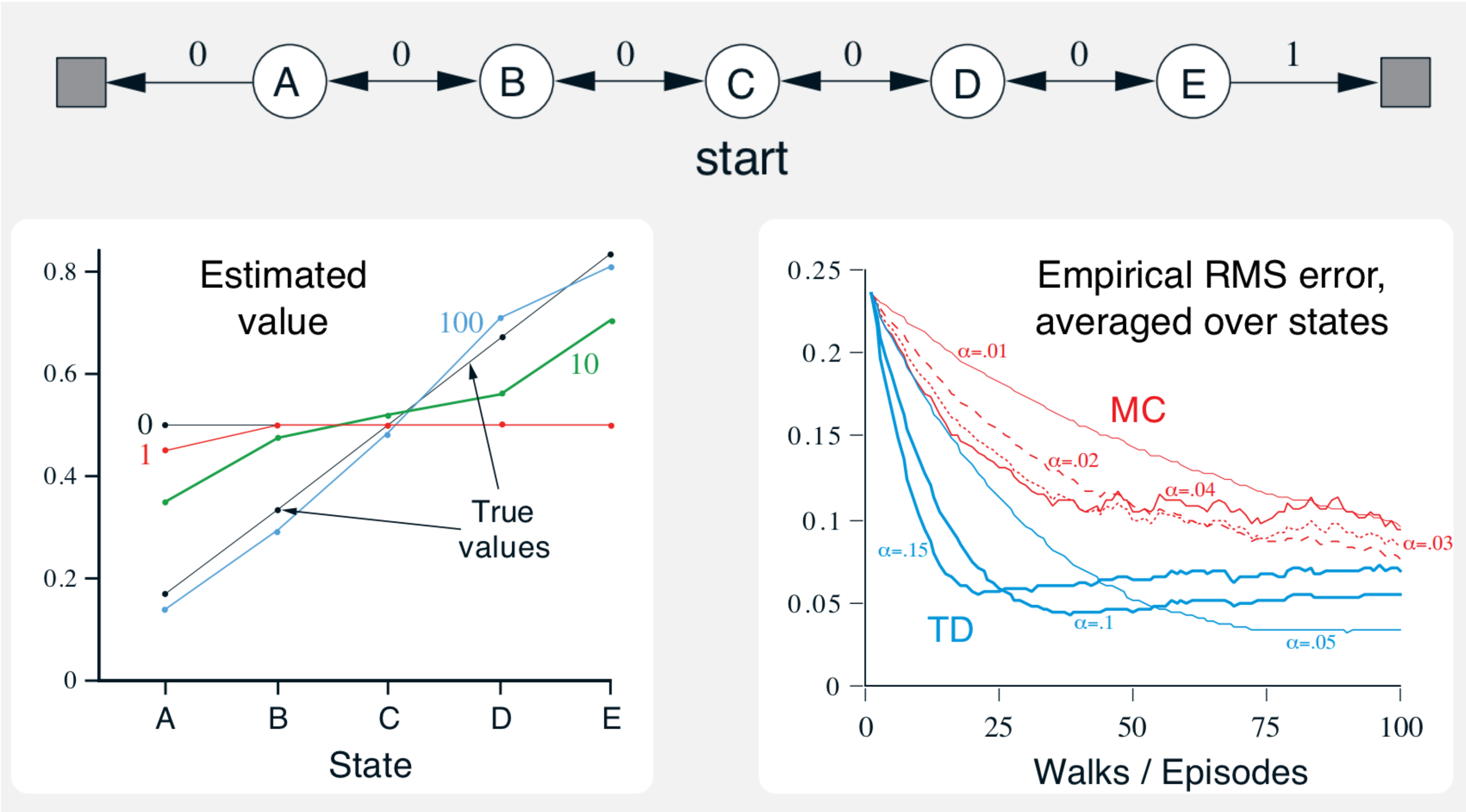
MC vs TD

- **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 $v(s)$ (but not always with function approximation)
  - More sensitive to initial value

Temporal-Difference Learning

Example

- **Random Walk**



Temporal-Difference Learning

Batch MC and TD

- **随着经验趋于无穷MC和TD可保证收敛**
  - $V(s) \to v_\pi(s)$
- **如果经验有限？**
  - K个episode
  - $s_1^1, a_1^1, r_1^1, \dots, s_{T_1}^1$
  - $s_1^k, a_1^k, r_1^k, \dots, s_{T_k}^k$
- **Batch Update**
  - 重复从这K个episode中进行采样
  - 对于某一次采样得到的episode $k \in [1, K]$应用MC或者TD(0)方法

Temporal-Difference Learning

Certainty-equivalence estimate

- **Batch MC和TD的收敛性**
  - MC收敛于最小均方误差(minimum mean-squared error)解
    - $\sum_{k=1}^{K}\sum_{t=1}^{T_k}(G_t^k - V(s_t^k))^2$
  - TD收敛于极大似然Markov模型对应的解
    - MDP $\langle \boldsymbol{S}, \boldsymbol{A}, \boldsymbol{P}, \boldsymbol{R}, \gamma \rangle$为产生这些数据概率最大的模型
    - $p_{ss'}^{a} = \frac{1}{N(s,a)}\sum_{k=1}^{K}\sum_{t=1}^{T_k}\mathbf{1}(s_t^k, a_t^k, s_{t+1}^k = s, a, s')$
    - $r_s^a = \frac{1}{N(s,a)}\sum_{k=1}^{K}\sum_{t=1}^{T_k}\mathbf{1}(s_t^k, a_t^k = s, a) r_t^k$

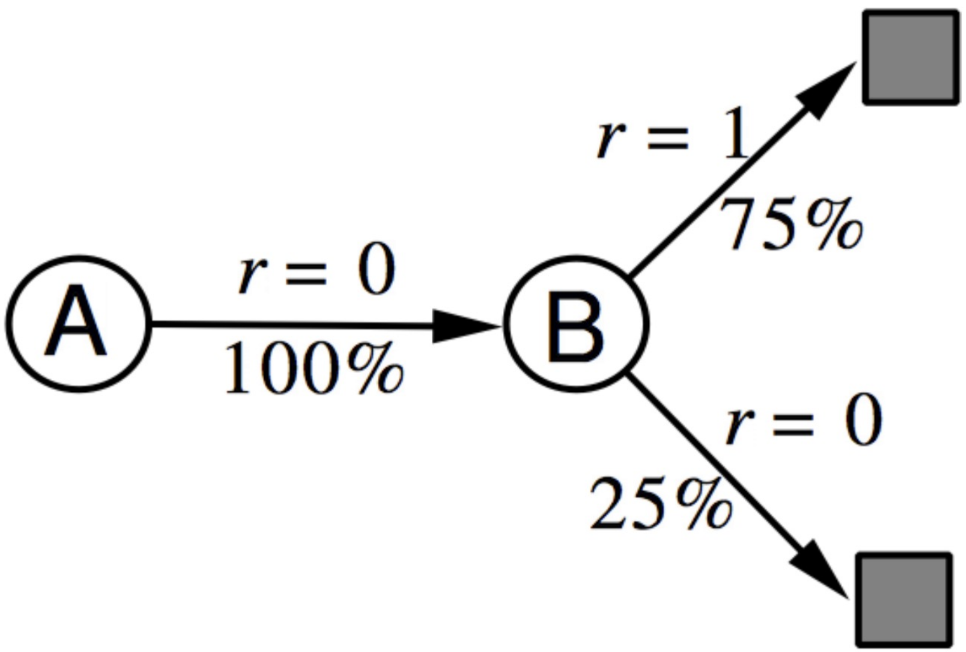
Temporal-Difference Learning

Example

- **MC $V(A)$=0**
- **TD $V(A)$=0.75**

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

A, 0, B, 0
B, 1
B, 1
B, 1
B, 1
B, 1
B, 1
B, 0



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

Temporal-Difference Learning

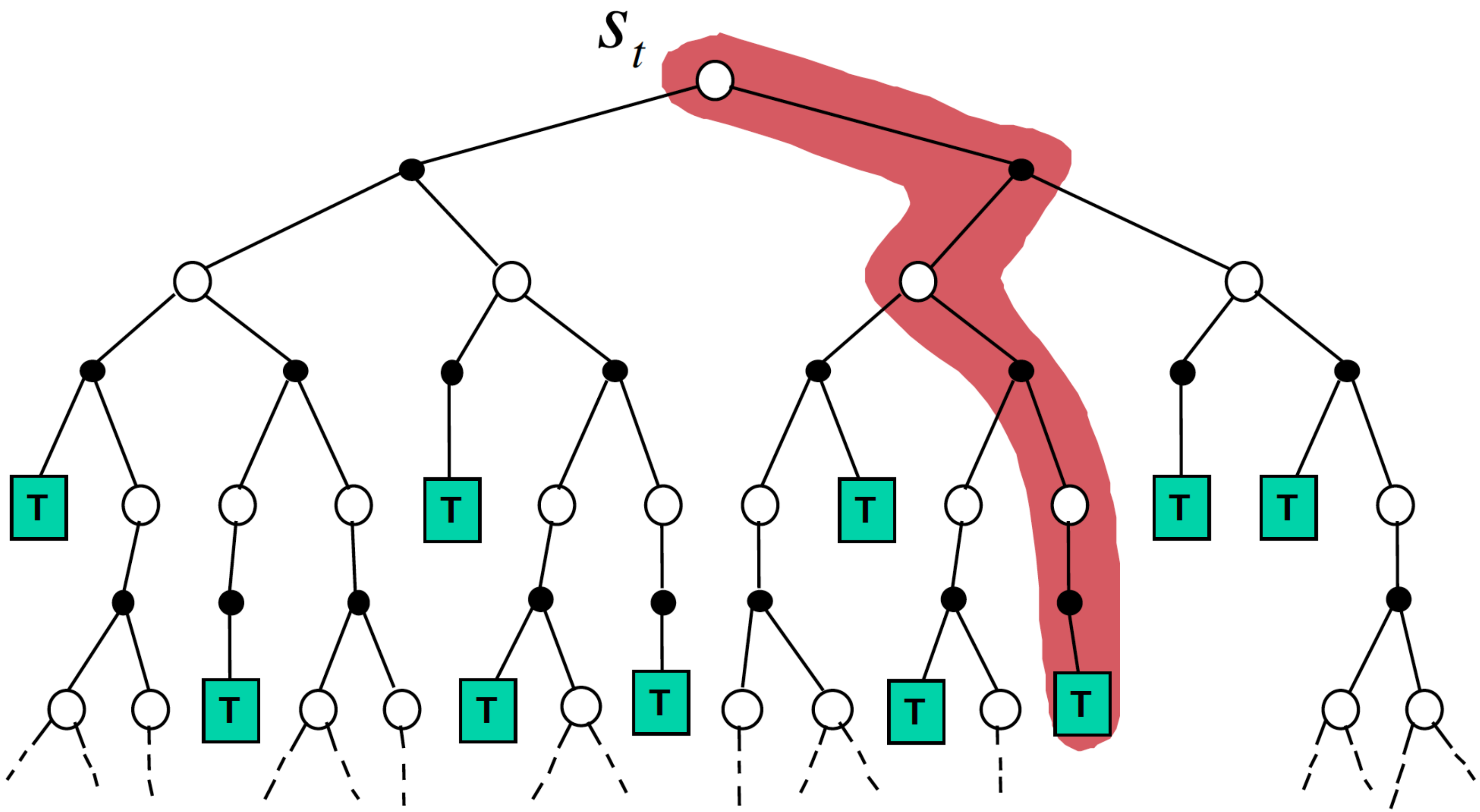
Batch MC and TD

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

Temporal-Difference Learning

Unified View

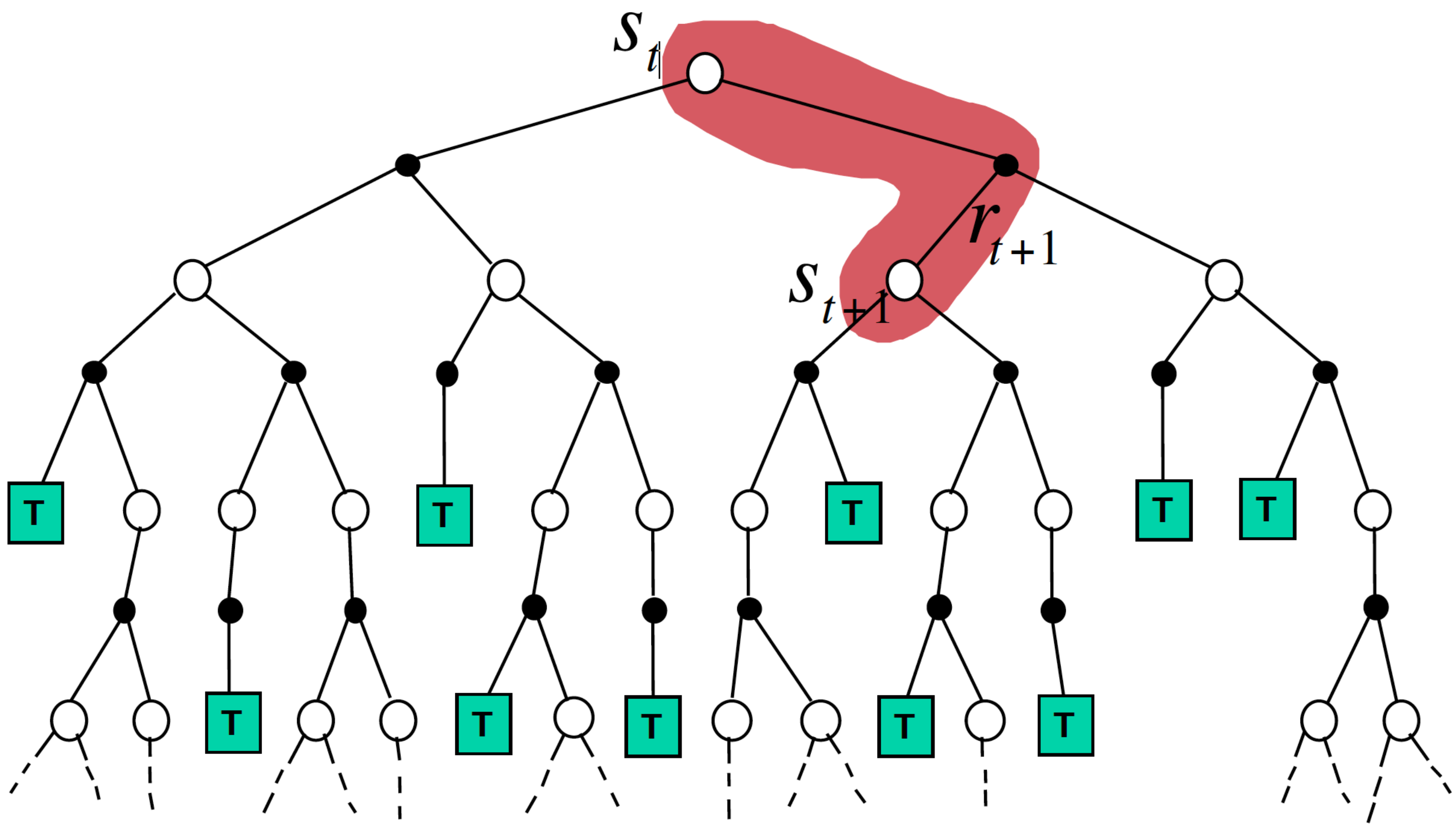
- **MC backup**
  - $V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$



Temporal-Difference Learning

Unified View

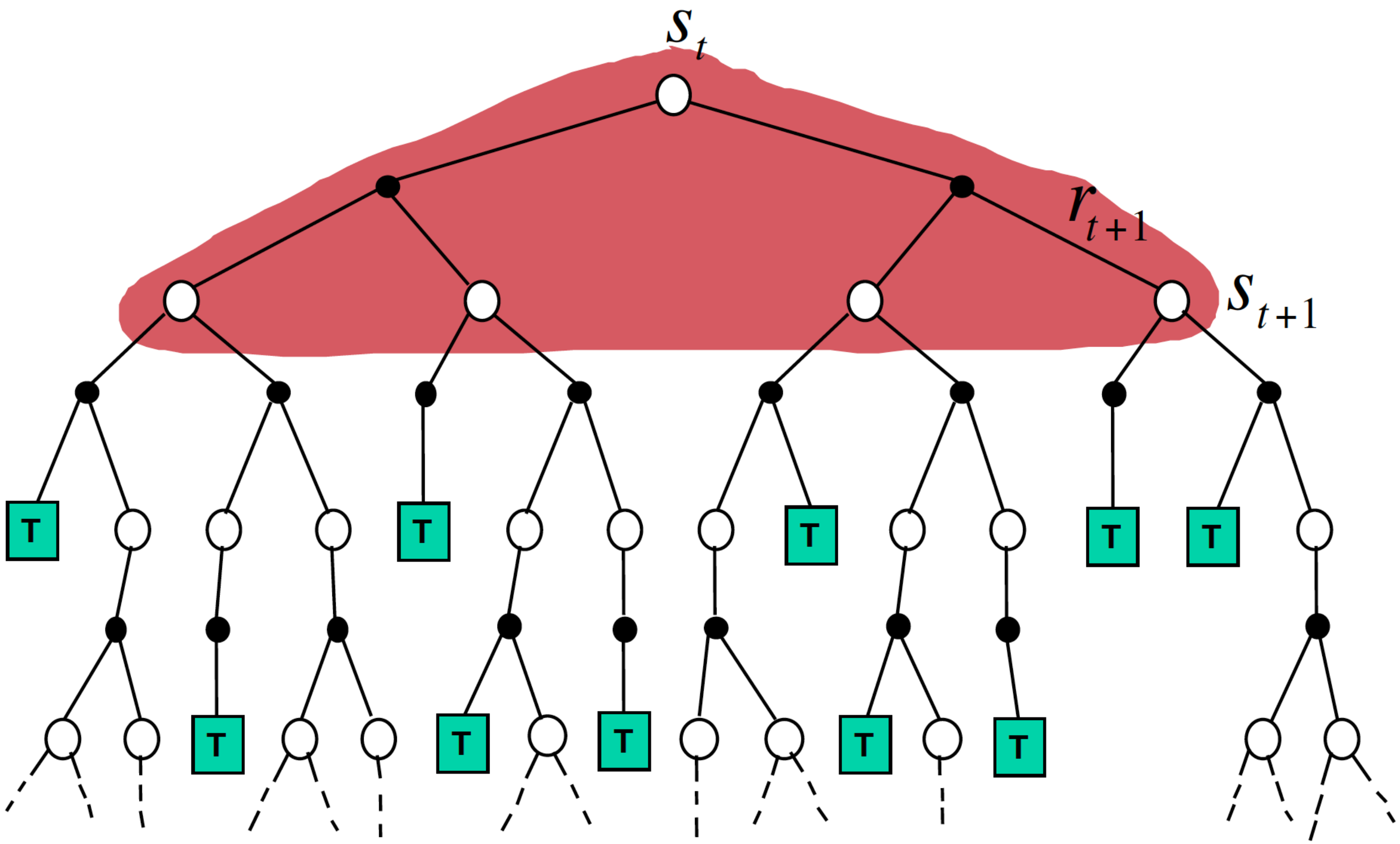
- **TD backup**
  - $V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$



Temporal-Difference Learning

Unified View

- **DP backup**
  - $V(s) = \mathrm{E}_{\pi}[R_{t+1} + \gamma V\ (S_{t+1})]$



Temporal-Difference Learning

Unified View

- **Bootstrapping and Sampling**
  - Bootstrapping：update involves an estimate
    - MC does not bootstrap
    - TD bootstraps
    - DP bootstraps
  - Sampling：update samples an expectation
    - MC samples
    - TD samples
    - DP does not sample（it's full-width）

n-step Bootstrapping

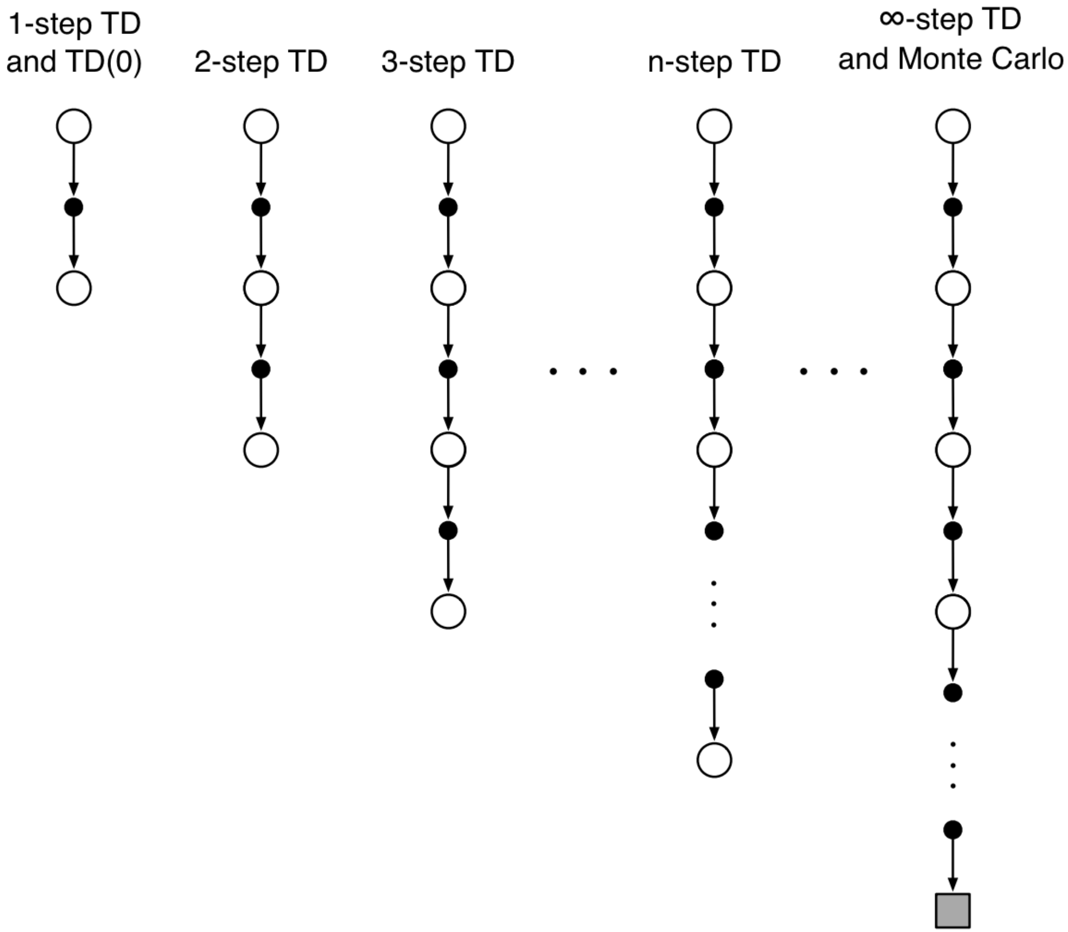
n-step Bootstrapping

- **MC**
  - wait until end of episode
    - $V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$
    - MC error $G_t - V(S_t)$
- **TD(0)**
  - Wait until next time step
    - $V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$
    - TD error $R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$
    - TD target $R_{t+1} + \gamma V(S_{t+1})$
- **n-step Bootstrapping**
  - 介于两者之间
  - 称为n-step TD
  - 利用eligibility traces

n-step Bootstrapping

n-step Prediction

- Let TD target look n steps into the future



n-step Bootstrapping

n-step Prediction

- **考虑S_t的State-value估计值**
  - 基于序列值$S_t, R_{t+1}, S_{t+1}, R_{t+2}, \dots, R_T, S_T$
  - MC: complete return
    - $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-1} R_T$
  - TD: one-step return
    - $G_{t:t+1} = R_{t+1} + \gamma V_t(S_{t+1})$
  - two-step return
    - $G_{t:t+2} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V_{t+1}(S_{t+2})$
- **n-step return**
  - Truncated
  - Use n rewards and bootstrap
    - $G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n})$
- **n-step TD learning**
    - $V_{t+n}(S_t) \leftarrow V_{t+n-1}(S_t) + \alpha(G_{t:t+n} - V_{t+n-1}(S_t))$

n-step Bootstrapping

n-step Prediction

n-step TD for estimating $V \approx v_\pi$

Input: a policy π
Algorithm parameters: step size $\alpha \in (0, 1]$, a positive integer n
Initialize $V(s)$ arbitrarily, for all $s \in \mathcal{S}$
All store and access operations (for S_t and R_t) can take their index mod $n+1$

Loop for each episode:
    Initialize and store $S_0 \neq$ terminal
    $T \leftarrow \infty$
    Loop for $t = 0, 1, 2, \ldots$:
    | If $t < T$, then:
    | Take an action according to $\pi(\cdot|S_t)$
    | Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
    | If S_{t+1} is terminal, then $T \leftarrow t+1$
    | $\tau \leftarrow t - n + 1$ (τ is the time whose state's estimate is being updated)
    | If $\tau \geq 0$:
    | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$
    | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
    | $V(S_\tau) \leftarrow V(S_\tau) + \alpha [G - V(S_\tau)]$
    Until $\tau = T - 1$

n-step Bootstrapping

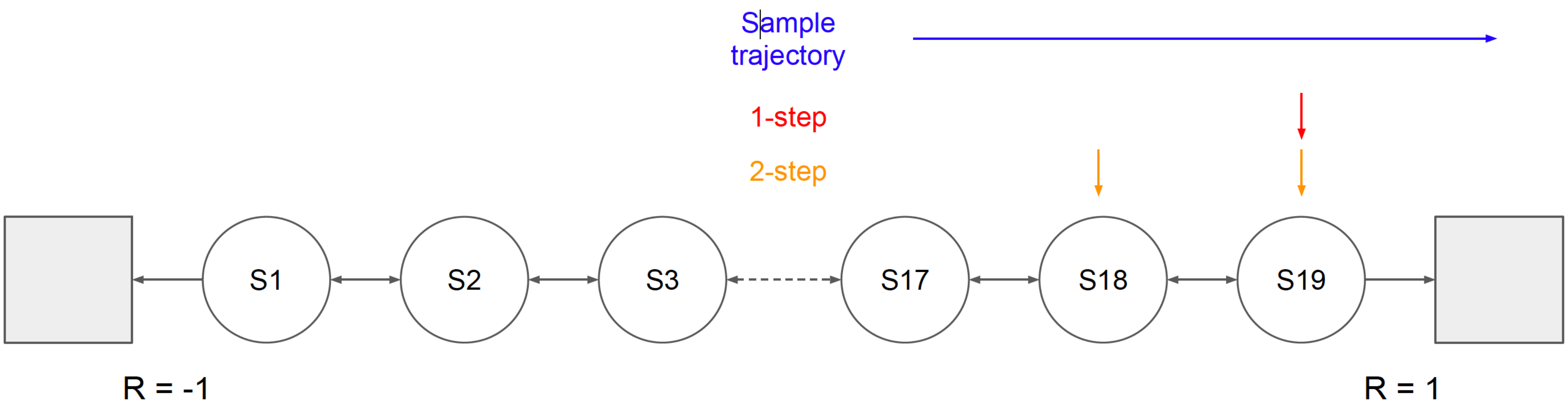
n-step Prediction

- **Convergence**
  - The n-step return has **Error reduction property**
  - 在最差情况下，n-step return相比$V_{t+n-1}(S_{t+n})$是$v_\pi(s)$的更好估计值
    - $\max_s |\mathrm{E}_\pi[G_{t:t+n}|S_t = s] - v_\pi(s)| \leq \gamma^n \max_s |V_{t+n-1}(S_{t+n}) - v_\pi(s)|$
  - 在合适的条件下可收敛至正确的value

n-step Bootstrapping

n-step Prediction

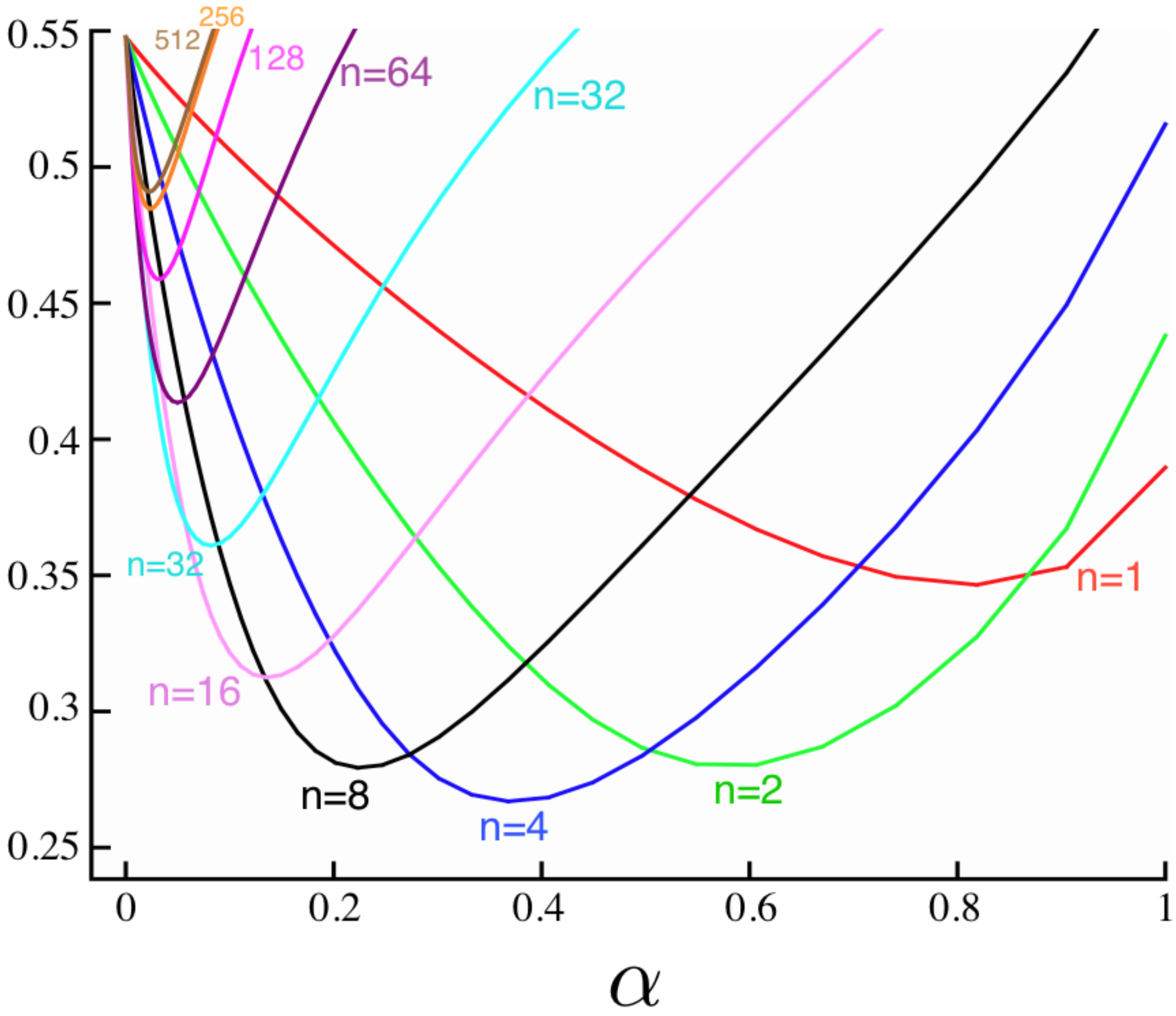
- **Random Walk Example**
  - Rewards only on exit (-1 on left exit, 1 on right exit)
  - n-step return: propagate reward up to n latest states



n-step Bootstrapping

n-step Prediction

Average RMS error over 19 states and first 10 episodes



Eligibility traces

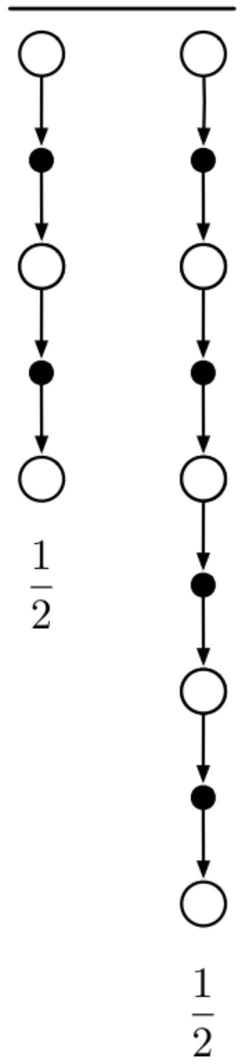
Eligibility Traces

- **Eligibility traces are one of the basic mechanisms of RL**
- **Eligibility traces unify and generalize TD and MC methods**
- **Advantage of eligibility traces over n-step methods**
  - Learning occurs continually and uniformly in time rather than being delayed and then catching up at the end of the episode.
  - Learning can occur and affect behavior immediately after a state is encountered rather than being delayed n steps.

Eligibility traces

Averaging n-Step Returns

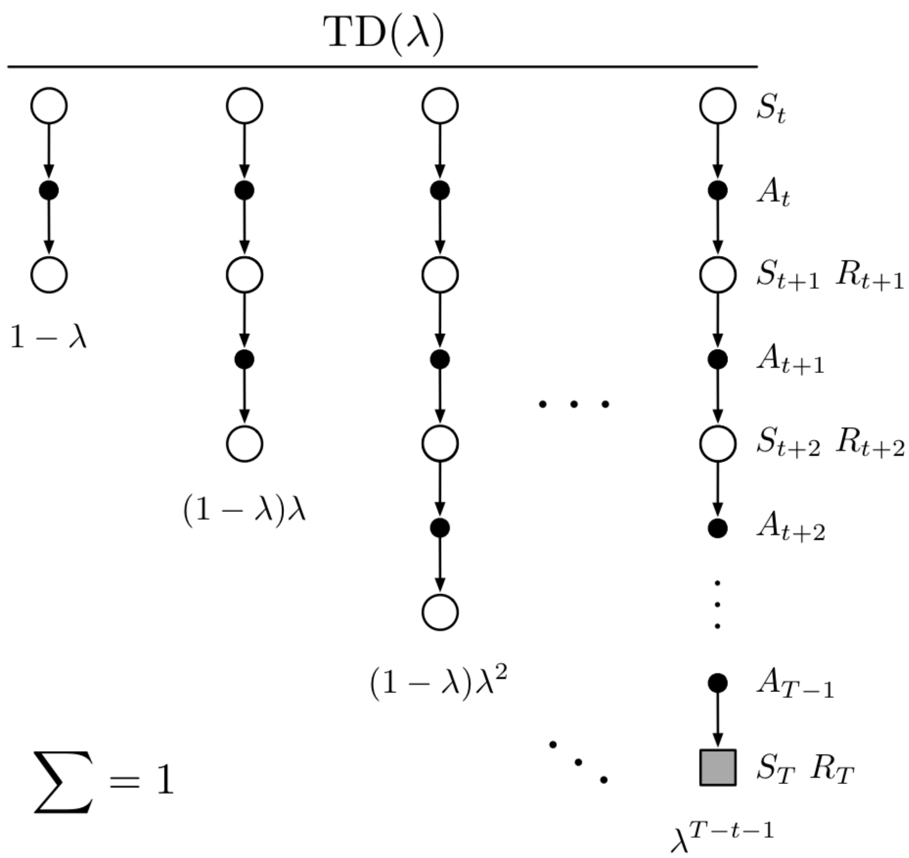
- **n-step return**
  - $G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n})$
- **We can average n-step returns over different n**
  - e.g. average the 2-step and 4-step returns
    - $\frac{1}{2} G_{t:t+2} + \frac{1}{2} G_{t:t+4}$
  - Combines information from two different time-steps
  - Can we efficiently combine information from all time-steps?



Eligibility traces

λ-return

- **The λ-return G_t^λ combines all n-step returns $G_{t:t+n}$**
  - Each has weight λ^{n-1}
  - Normalized by $1-\lambda$
  - $G_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_{t:t+n}$



  - $(1-\lambda)+(1-\lambda)\lambda+(1-\lambda)\lambda^2+\cdots+(1-\lambda)\lambda^n = 1+\lambda^n \approx 1$

Eligibility traces

λ-return

- **TD(λ) Weighting Function**

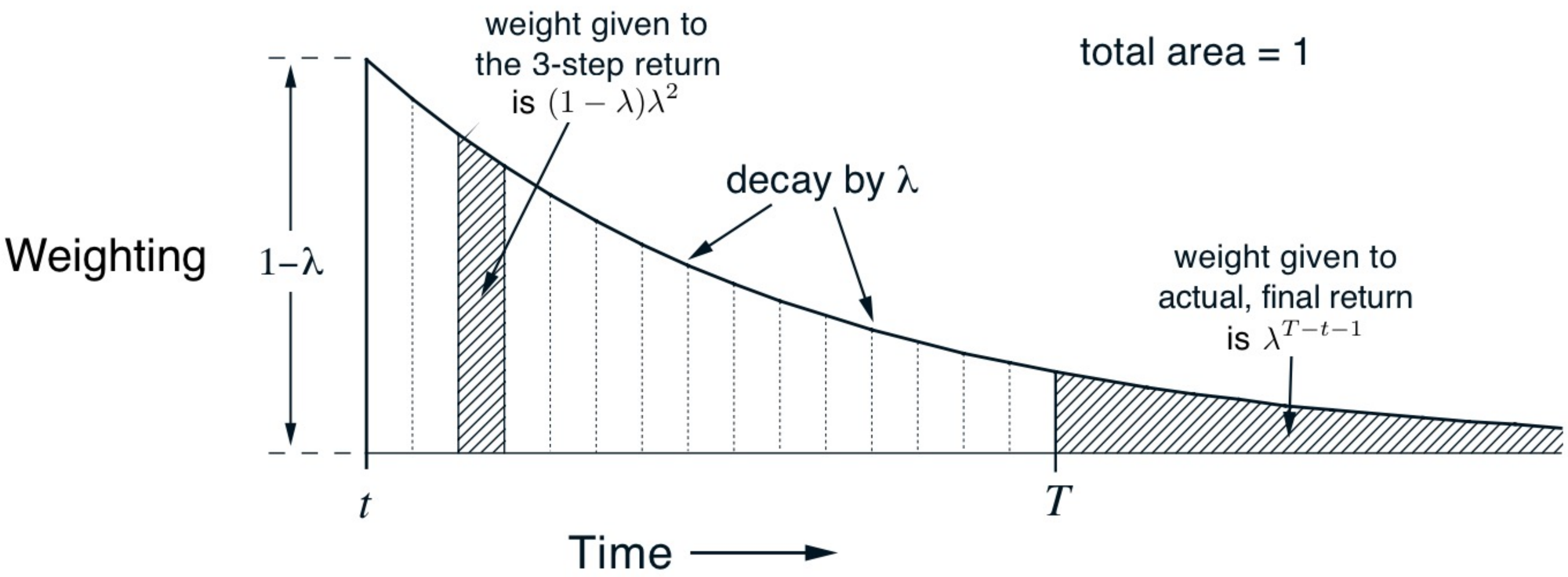


Figure 12.2: Weighting given in the λ-return to each of the n-step returns.

– $G_t^\lambda = (1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1}G_{t:t+n} + \lambda^{T-t-1}G_t$

Eligibility traces

Forward-view TD(λ)

- **Forward View TD(λ)**
  - $V(S_t) \leftarrow V(S_t) + \alpha(G_t^\lambda - V(S_t))$
  - Update value function towards the $\boldsymbol{\lambda}$-return
  - Forward-view looks into the future to compute G_t^λ
  - Like MC, can only be computed from complete episodes

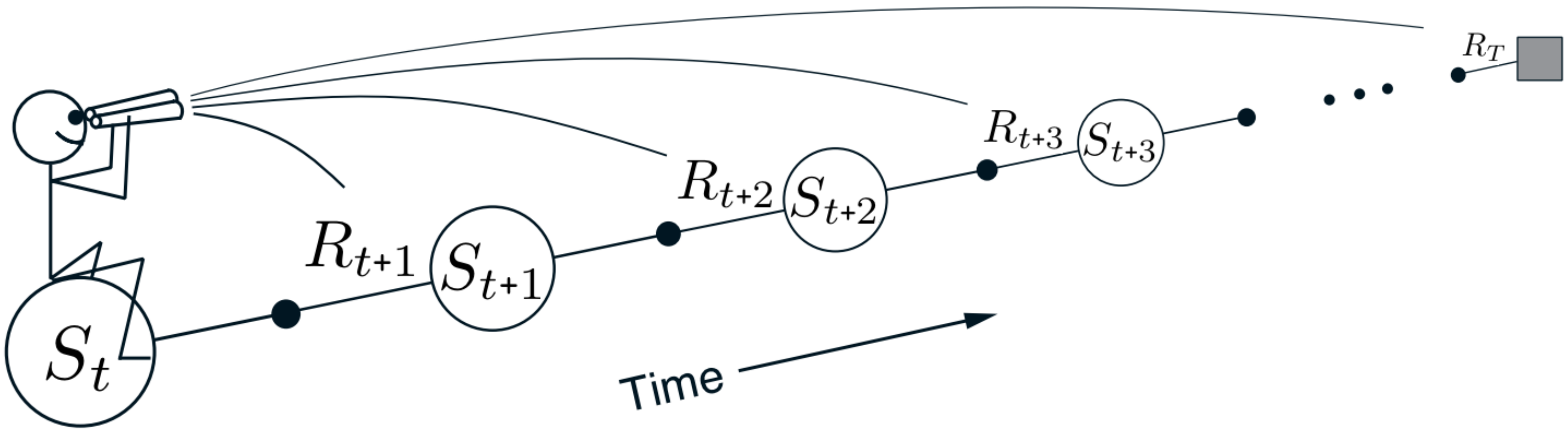
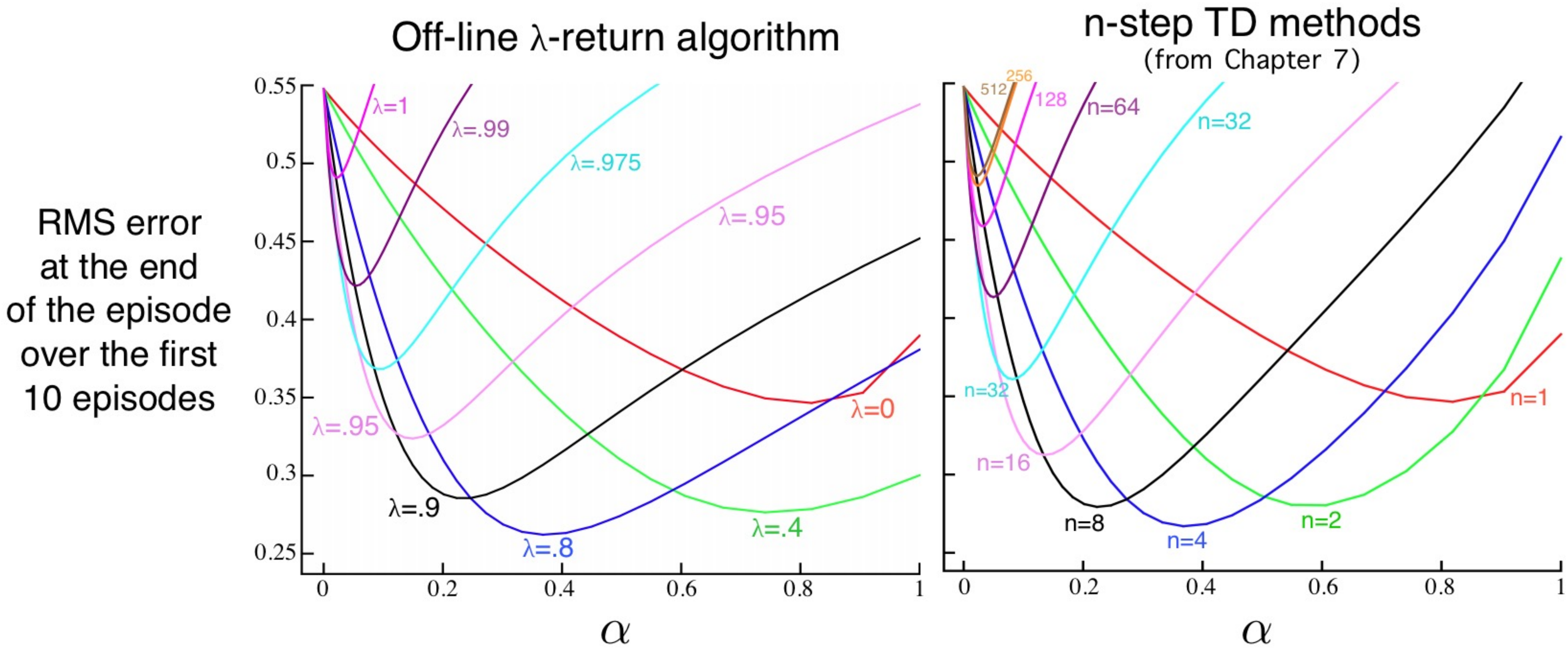


Figure 12.4: The forward view. We decide how to update each state by looking forward to future rewards and states.

Eligibility traces

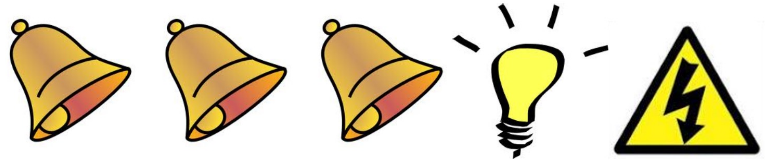
Forward-view TD(λ) on Large Random Walk

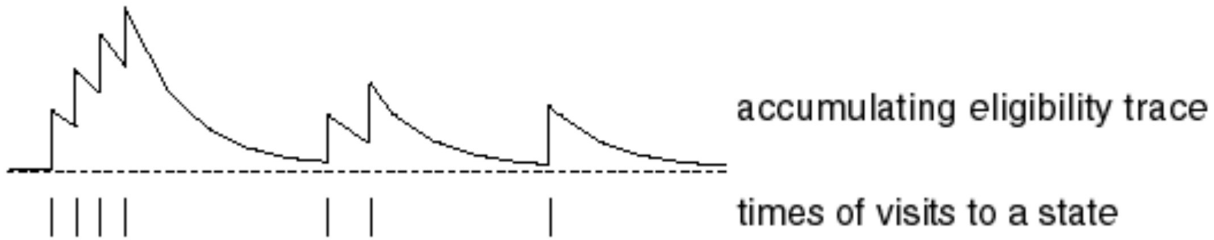


Eligibility traces

Backward-view TD(λ)

- **Backward view**
  - Forward view provides theory
  - Backward view provides mechanism
  - Update online, every step, from incomplete sequences
- **Eligibility Traces**
  - Credit assignment problem: did bell or light cause shock?
    - Frequency heuristic: assign credit to most frequent states
    - Recency heuristic: assign credit to most recent states
  - Eligibility traces combine both heuristics
    - $E_0(s) = 0$
    - $E_t(s) = \gamma\lambda E_{t-1}(s) + \mathbf{1}(S_t = s)$

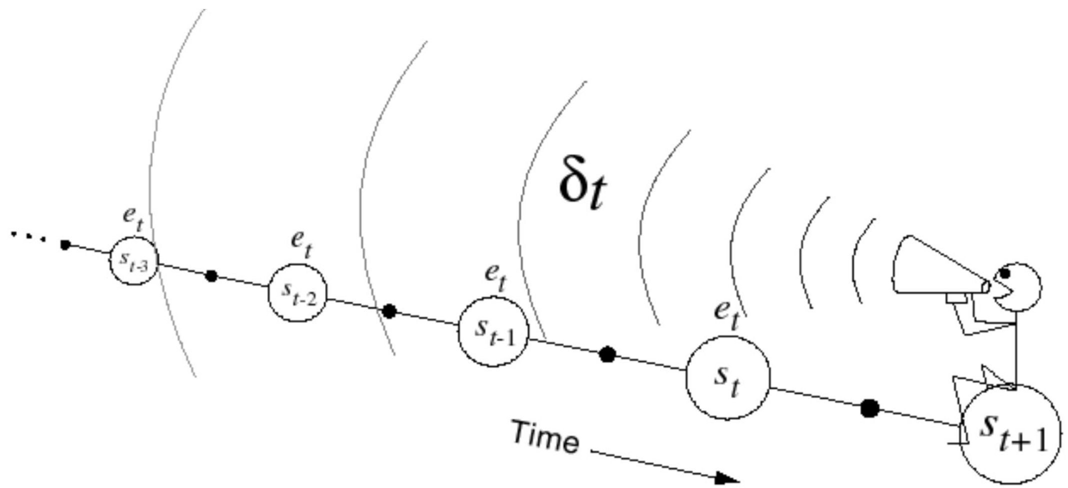




Eligibility traces

Backward-view TD(λ)

- **Backward-view TD(λ)**
  - Keep an eligibility trace for every state s
  - Update value $V(s)$ for every state s
  - In proportion to TD-error δ_t and eligibility trace $E_t(s)$
    - $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$
    - $V(s) \leftarrow V(s) + \alpha \delta_t E_t(s)$
  - Compare with TD(0)
    - $V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$



Eligibility traces

Forwards and Backwards TD(λ)

- **Consider an episode where s is visited once at time-step k**
- **TD(λ) eligibility trace discounts time since visit**
  - $E_t(s) = \gamma\lambda E_{t-1}(s) + \mathbf{1}(S_t = s) = \begin{cases} 0 & \text{if } t < k \\ (\gamma\lambda)^{t-k} & \text{if } t \geq k \end{cases}$
- **Backwards TD(λ) updates accumulate error online**
  - $\sum_{t=1}^{T} \alpha\delta_t E_t(s) = \alpha \sum_{t=k}^{T} (\gamma\lambda)^{t-k} \delta_t = \alpha(G_k^\lambda - V(S_k))$
- **By end of episode it accumulates total error for λ –return**
- **When $\lambda = 1$, sum of TD errors telescopes into MC error**
  - $G_t - V(S_t) = R_{t+1} + \gamma G_{t+1} - V(S_t) + \gamma V(S_{t+1}) - \gamma V(S_{t+1})$
  - $= \delta_t + \gamma(G_{t+1} - V(S_{t+1}))$
  - $= \delta_t + \gamma\delta_{t+1} + \gamma^2 (G_{t+2} - V(S_{t+2}))$
  - $= \delta_t + \gamma\delta_{t+1} + \gamma^2\delta_{t+2} + \cdots + \gamma^{T-t}(G_T - V(S_T))$
  - $= \delta_t + \gamma\delta_{t+1} + \gamma^2\delta_{t+2} + \cdots + \gamma^{T-t}(0)$
  - $= \sum_{k=t}^{T-1} \gamma^{k-t} \delta_k$

Eligibility traces

Offline Equivalence of Forward and Backward TD(λ)

- **Offline updates**
  - Updates are accumulated within episode
  - but applied in batch at the end of episode
  - $\sum_{t=1}^{T} \alpha \left(G_t^{\lambda} - V(S_t) \right) \mathbf{1}(S_t = s) = \sum_{t=1}^{T} \alpha \delta_t E_t(s)$
- **Online updates**
  - TD($\boldsymbol{\lambda}$) updates are applied online at each step within episode
  - Forward and backward-view TD($\boldsymbol{\lambda}$) are slightly different

Eligibility traces

TD(λ) and TD(0)

- **When $\boldsymbol{\lambda} = 0$, only current state is updated**
  - $E_t(s) = \mathbf{1}(S_t = s)$
  - $V(S) \leftarrow V(S) + \alpha\delta_t E_t(s)$
- **This is exactly equivalent to TD(0) update**
  - $V(S) \leftarrow V(S) + \alpha\delta_t = V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$

Eligibility traces

TD(1) and MC

- **TD(1) when $\lambda = 1$**
  - The credit given to earlier states falls only by γ per step
  - credit is deferred until end of episode
  - Achieve MC behavior
- **TD(1) can be applied to discounted continuing tasks**
- **TD(1) can be performed incrementally and online**
- **Over the course of an episode, total update for TD(1) is the same as total update for MC**

Eligibility traces

TD(1) and MC

- **Consider an episode where *s* is visited once at time-step *k***
- **TD(1) eligibility trace discounts time since visit**
  - $E_t(s) = \gamma E_{t-1}(s) + \mathbf{1}(S_t = s) = \begin{cases} 0 & \text{if } t < k \\ \gamma^{t-k} & \text{if } t \geq k \end{cases}$
- **TD(1) updates accumulate error online**
  - $\sum_{t=1}^{T} \alpha \delta_t E_t(s) = \alpha \sum_{t=k}^{T} \gamma^{t-k} \delta_t = \alpha (G_k - V(S_k))$
- **By end of episode it accumulates total error**
  - $\sum_{t=k}^{T} \gamma^{t-k} \delta_t$

Eligibility traces

TD(1) and MC

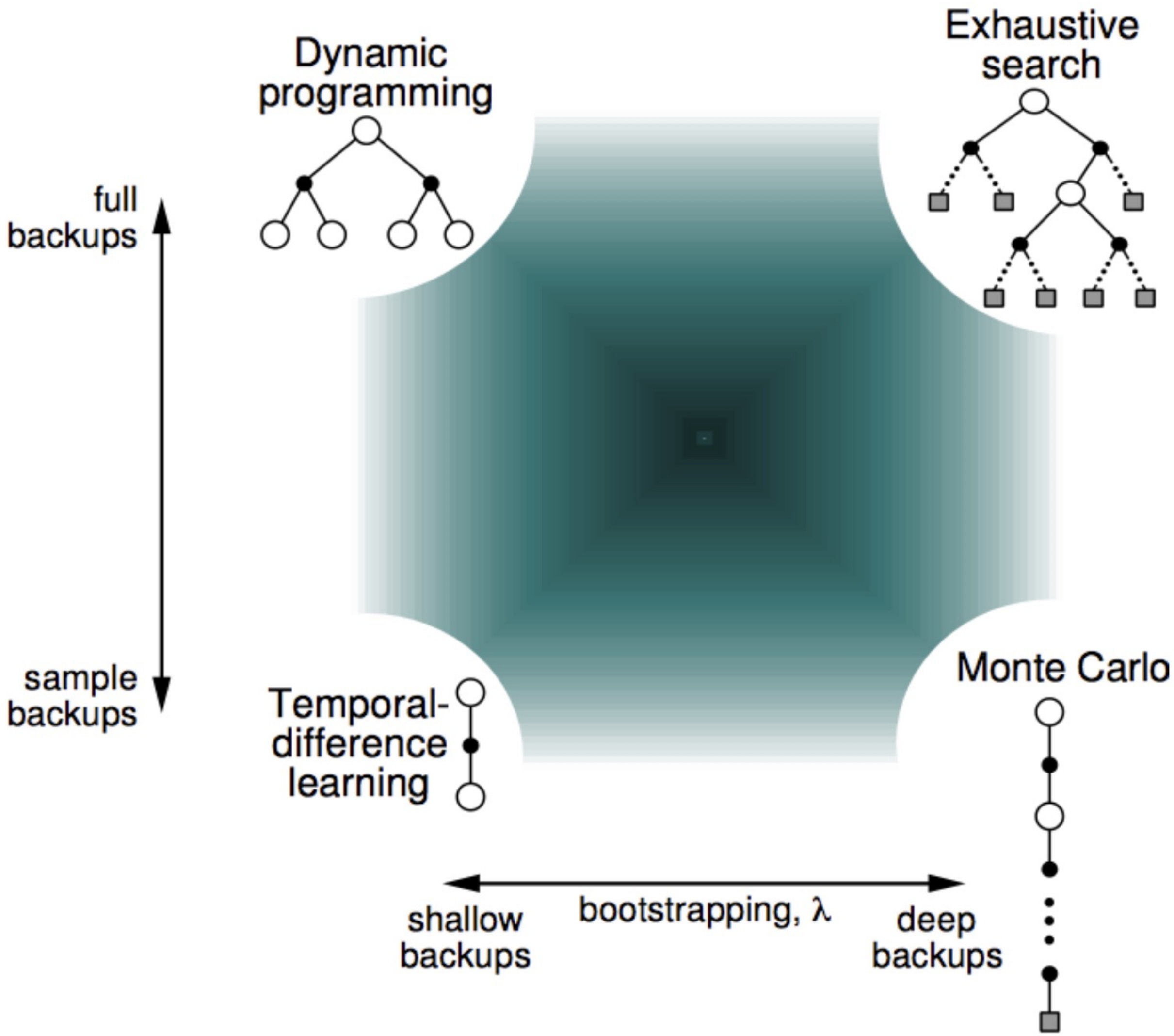
- **TD(1) is roughly equivalent to every-visit Monte-Carlo**
- **Error is accumulated online, step-by-step**
- **If value function is only updated offline at end of episode**
  - Then total update is exactly the same as MC

Eligibility traces

Forward and Backward TD(λ)

Offline updates	$\lambda = 0$	$\lambda \in (0, 1)$	$\lambda = 1$
Backward view Forward view	TD(0) $\parallel$ TD(0)	TD(λ) $\parallel$ Forward TD(λ)	TD(1) $\parallel$ MC
Online updates	$\lambda = 0$	$\lambda \in (0, 1)$	$\lambda = 1$
Backward view Forward view Exact Online	TD(0) $\parallel$ TD(0) $\parallel$ TD(0)	TD(λ) $\nparallel$ Forward TD(λ) $\parallel$ Exact Online TD(λ)	TD(1) $\nparallel$ MC $\parallel$ Exact Online TD(1)

Unfied View of Reinforcement Learning



Model Free Prediction

Suggested reading

- **Schultz, W., Dayan, P., and Montague, R., "A Neural Substrate of Prediction and Reward", 1999**
- **Redish, A. D., "Addiction as a Computational Process Gone Awry",2004**