Reinforcement Learning

Lecture 4
Model Free Prediction

Stergios Christodoulidis

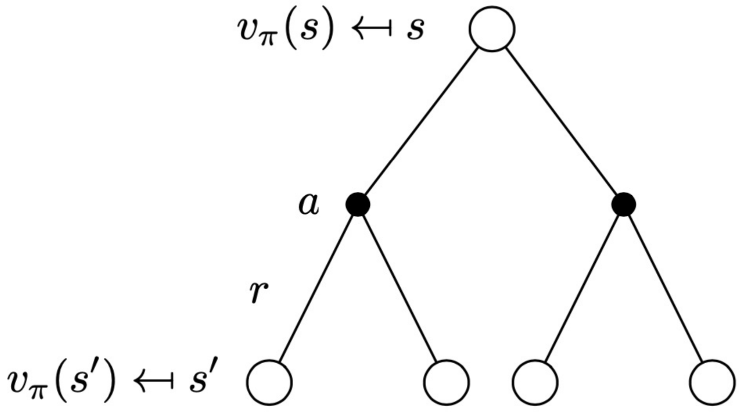
MICS Laboratory
CentraleSupélec
Université Paris-Saclay

https://stergioc.github.io/

CentraleSupélec

Last Lecture

Iterative Policy Evaluation



$$\mathrm{v}_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \mathrm{v}_k(s') \right)$$

[An Introduction to Reinforcement Learning, Sutton and Barto]

Iterative Policy Evaluation

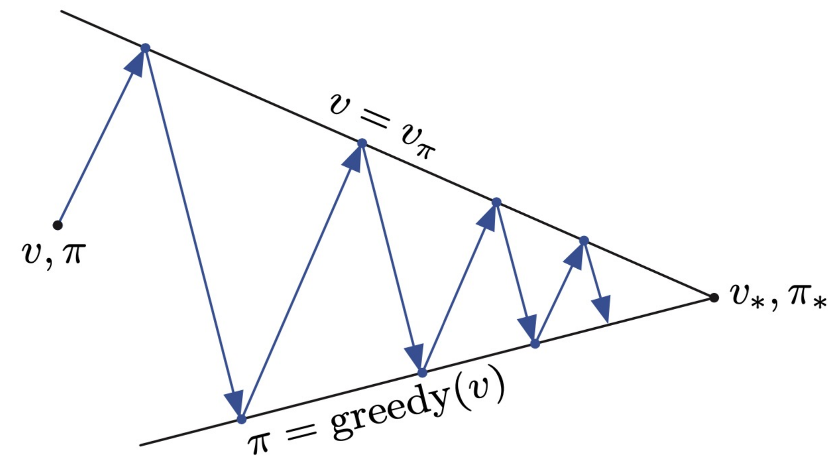
Iterative Policy Evaluation, for estimating $V \approx v_\pi$

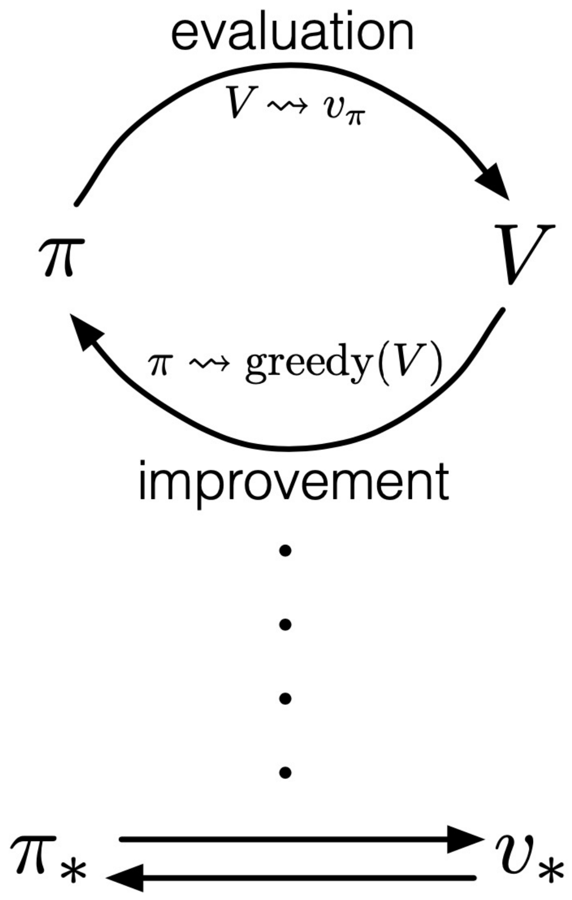
Input π, the policy to be evaluated
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$ arbitrarily, for $s \in \mathcal{S}$, and $V(\textit{terminal})$ to 0

Loop:
    $\Delta \leftarrow 0$
    Loop for each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r \,|\, s, a)\big[r + \gamma V(s')\big]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

[An Introduction to Reinforcement Learning, Sutton and Barto]

Policy Iteration





- Policy evaluation Estimate v_π
  - Iterative policy evaluation
- Policy improvement Generate $\pi' \geq \pi$
  - Greedy policy improvement

[An Introduction to Reinforcement Learning, Sutton and Barto]

Policy Iteration

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

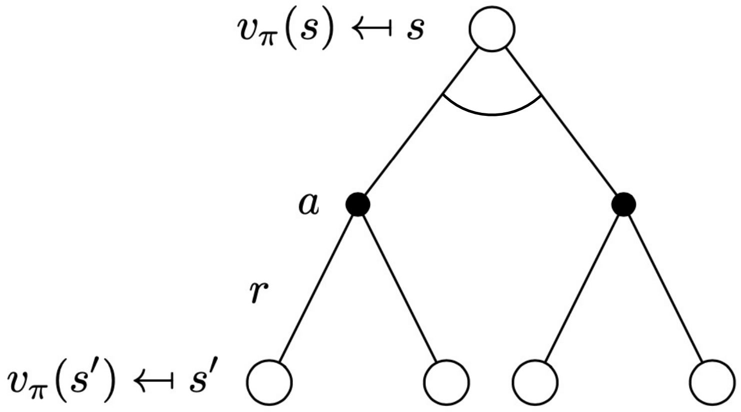
1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$; $V(\mathit{terminal}) \doteq 0$

2. Policy Evaluation
   Loop:
       $\Delta \leftarrow 0$
       Loop for each $s \in \mathcal{S}$:
           $v \leftarrow V(s)$
           $V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))\big[r + \gamma V(s')\big]$
           $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   $\mathit{policy\text{-}stable} \leftarrow \mathit{true}$
   For each $s \in \mathcal{S}$:
       $\mathit{old\text{-}action} \leftarrow \pi(s)$
       $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
       If $\mathit{old\text{-}action} \neq \pi(s)$, then $\mathit{policy\text{-}stable} \leftarrow \mathit{false}$
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

[An Introduction to Reinforcement Learning, Sutton and Barto]

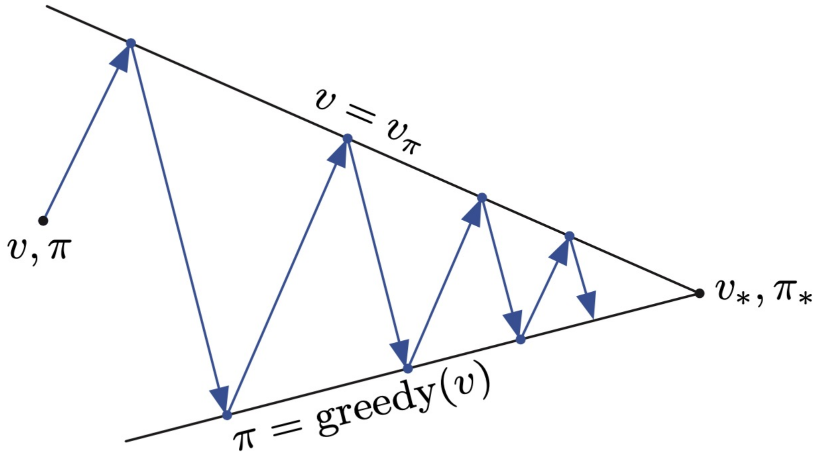
Value Iteration

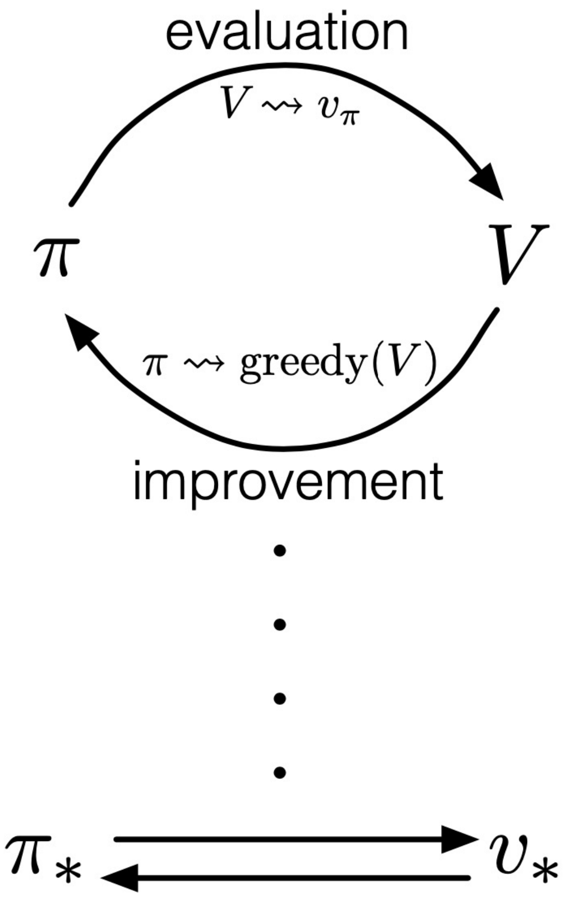


$$\mathrm{v}_{k+1}(s) = \max_{a\in\mathcal{A}}\left(\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a \mathrm{v}_k(s')\right)$$

[An Introduction to Reinforcement Learning, Sutton and Barto]

Generalized Policy Iteration





- Policy evaluation Estimate v_π
  - Any policy evaluation algorithm
- Policy improvement Generate $\pi' \geq \pi$
  - Any policy improvement algorithm

[An Introduction to Reinforcement Learning, Sutton and Barto]

Synchronous Dynamic Programming Algorithms

Problem	Bellman Equation	Algorithm
Prediction	Bellman Expectation Equation	Iterative Policy Evalution
Control	Bellman Expectation Equation + Greedy Policy Improvement	Policy Iteration
Control	Bellman Optimality Equation	Value Iteration

- Algorithms are based on state-value function $\mathrm{v}_\pi(s)$ or $\mathrm{v}_*(s)$
- Complexity $O(mn^2)$ per iteration, for m actions and n states

Today’s Lecture

Today’s Lecture

- Monte-Carlo Learning
- Temporal-Difference Learning
- n-step TD

Model-Free RL

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

Monte-Carlo Learning

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 complete episodes: no bootstrapping
- MC uses the simplest possible idea: value = mean return
- 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

$$\mathrm{S}_1, A_1, R_1, \ldots, S_k \sim \pi$$

- Recall that the return is the total discounted reward

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

- Recall that the value function is the expected return:

$$\mathrm{v}_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$$

- Monte-Carlo policy evaluation uses empirical mean return 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 mean return, $V(s) = S(s)/N(s)$
- By law of large numbers, $V(s) = v_\pi$ as $N(s) \rightarrow \infty$

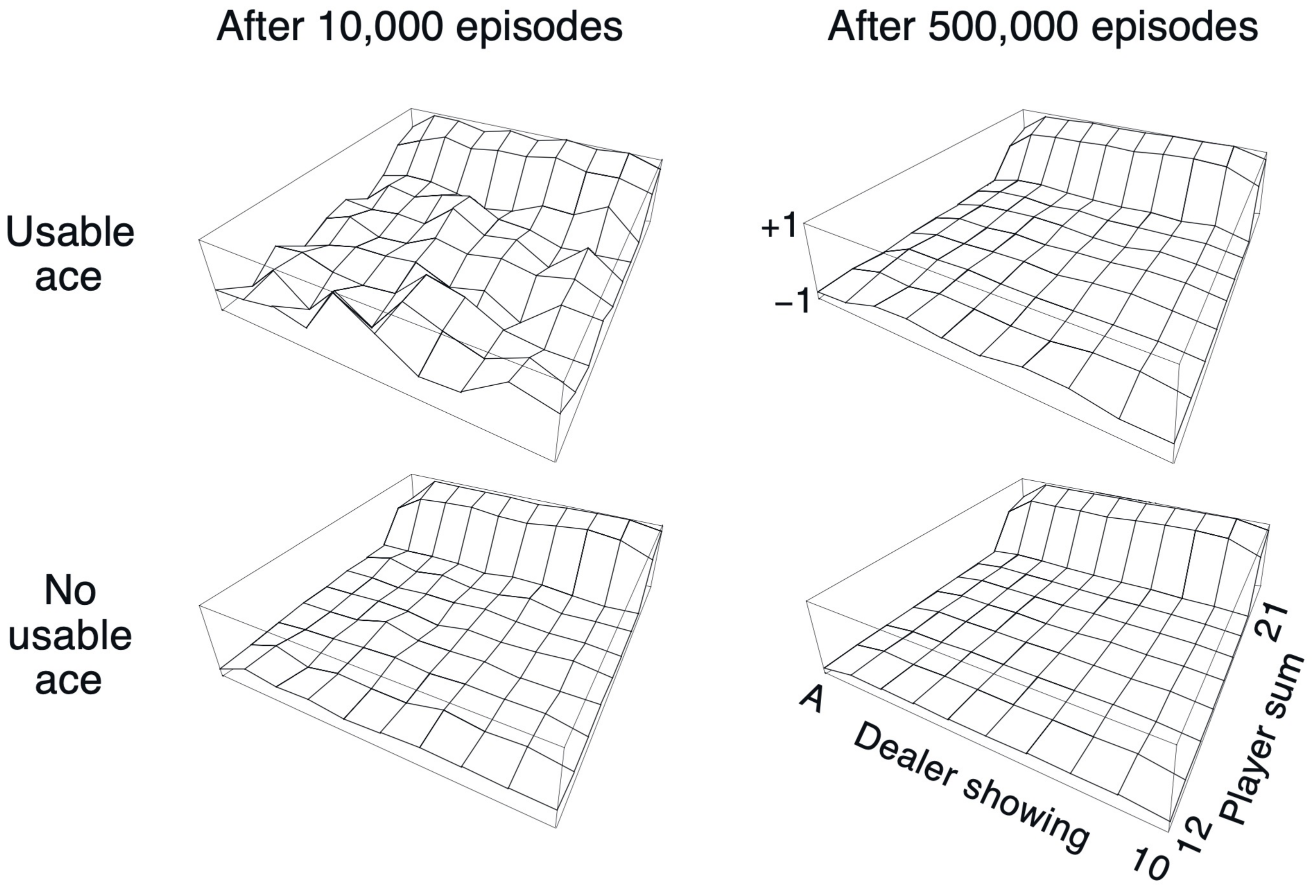
Every-Visit Monte-Carlo Policy Evaluation

- 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, $V(s) = v_\pi$ as $N(s) \rightarrow \infty$

Example: Blackjack

- States (200 in total):
  - Current sum (12-21)
  - Dealer’s showing card (ace or 2-10)
  - Do I have a “useable” ace? (yes-no)
- Actions
  - *hit*: Take another card (no replacement)
  - *stick*: Stop receiving cards (and terminate)
- Rewards
  - 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
  - for hit:
    - -1 if sum of cards > 21 (and terminate)
    - 0 otherwise
- Policy: stick if sum of cards $\geq$ 20, otherwise hit

Blackjack Value Function after MC Learning



[An Introduction to Reinforcement Learning, Sutton and Barto]

Incremental Mean

- The mean μ_k of a sequence $x_1, x_2, ..., x_k$ can be computed incrementally,

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

Incremental Monte-Carlo Updates

- Update $V(s)$ incrementally after episode $\mathrm{S}_1, A_1, R_1, \ldots, S_k$
- 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.

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

Temporal-Difference Learning

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 from episodes of experience under policy π
- Incremental 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 $R_{t+1} + \gamma V(S_{t+1})$

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

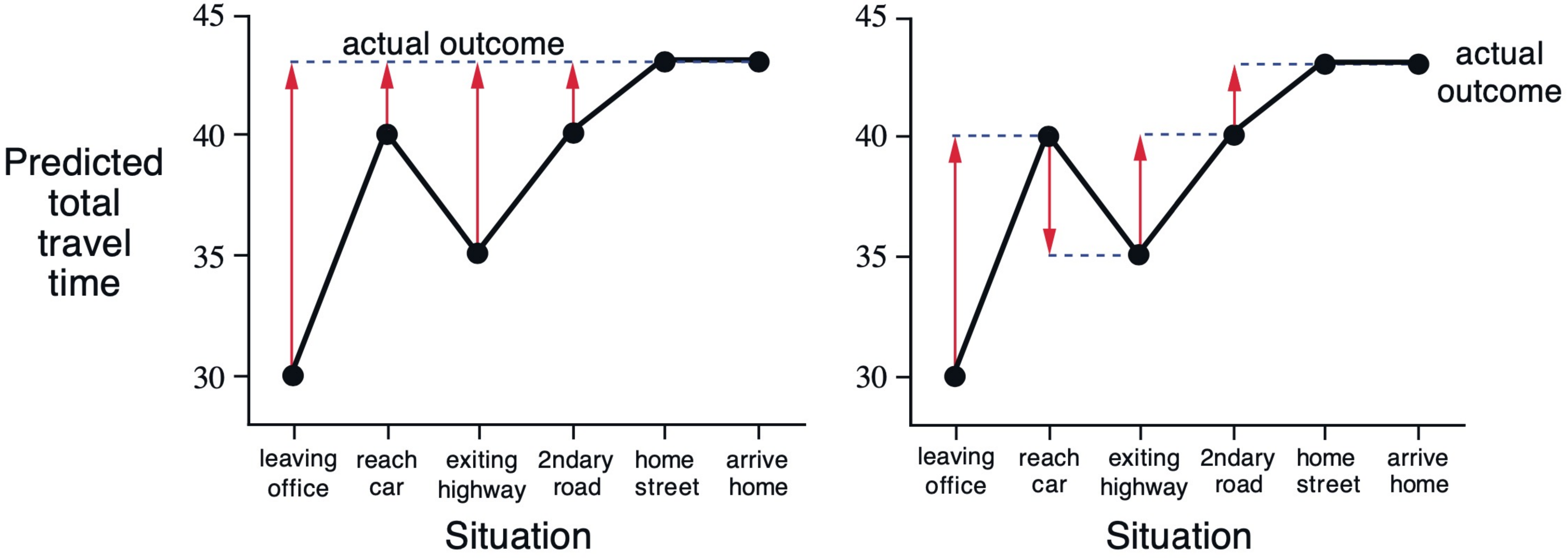
  - $R_{t+1} + \gamma V(S_{t+1})$ is called TD target
  - $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is called the TD error

Example: Driving Home

State	*Elapsed Time (minutes)*	*Predicted Time to Go*	*Predicted Total Time*
leaving office, friday at 6	0	30	30
reach car, raining	5	35	40
exiting highway	20	15	35
2ndary road, behind truck	30	10	40
entering home street	40	3	43
arrive home	43	0	43

[An Introduction to Reinforcement Learning, Sutton and Barto]

Example: Driving Home – MC vs TD



[An Introduction to Reinforcement Learning, Sutton and Barto]

Advantages and Disadvantages of MC vs TD (1/3)

- TD can learn before knowing the outcome
  - TD can learn online after every step
  - MC must wait until end of episode before return is known
- TD can learn without the 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

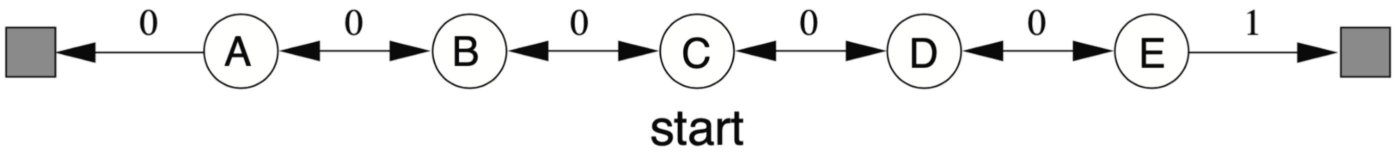
Bias/Variance Trade-Off

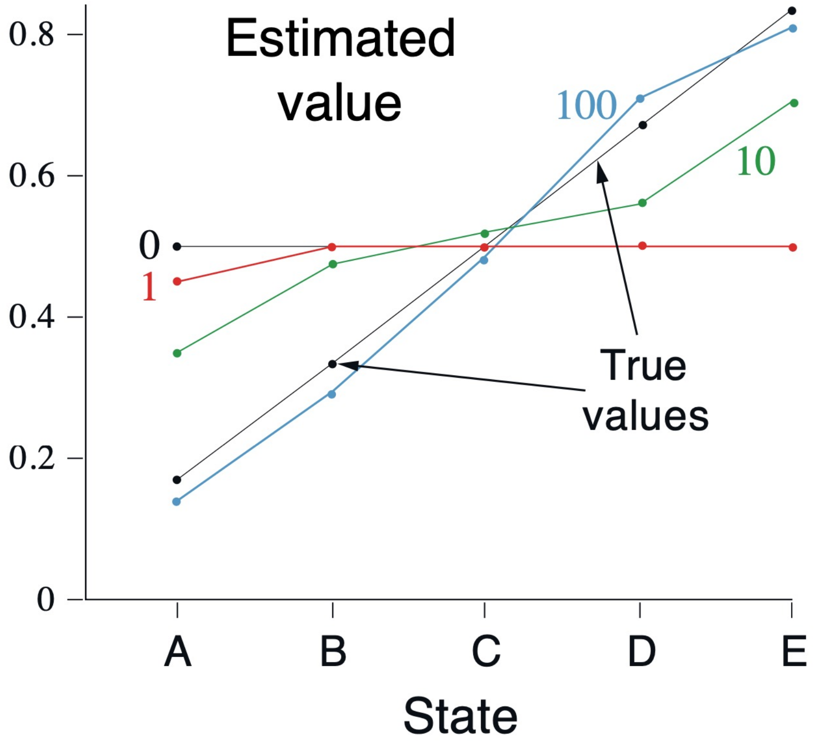
- Return $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$ is an unbiased estimate of the v_π
- True TD target $R_{t+1} + \gamma v_\pi(S_{t+1})$ is unbiased estimate of v_π
- The TD target $R_{t+1} + \gamma V(S_{t+1})$ is a biased estimate of v_π
- TD target is much lower variance than the 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 (2/3)

- MC has high variance, zero bias
  - Good convergence properties
  - 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 $v_\pi(s)$
  - More sensitive to initial value

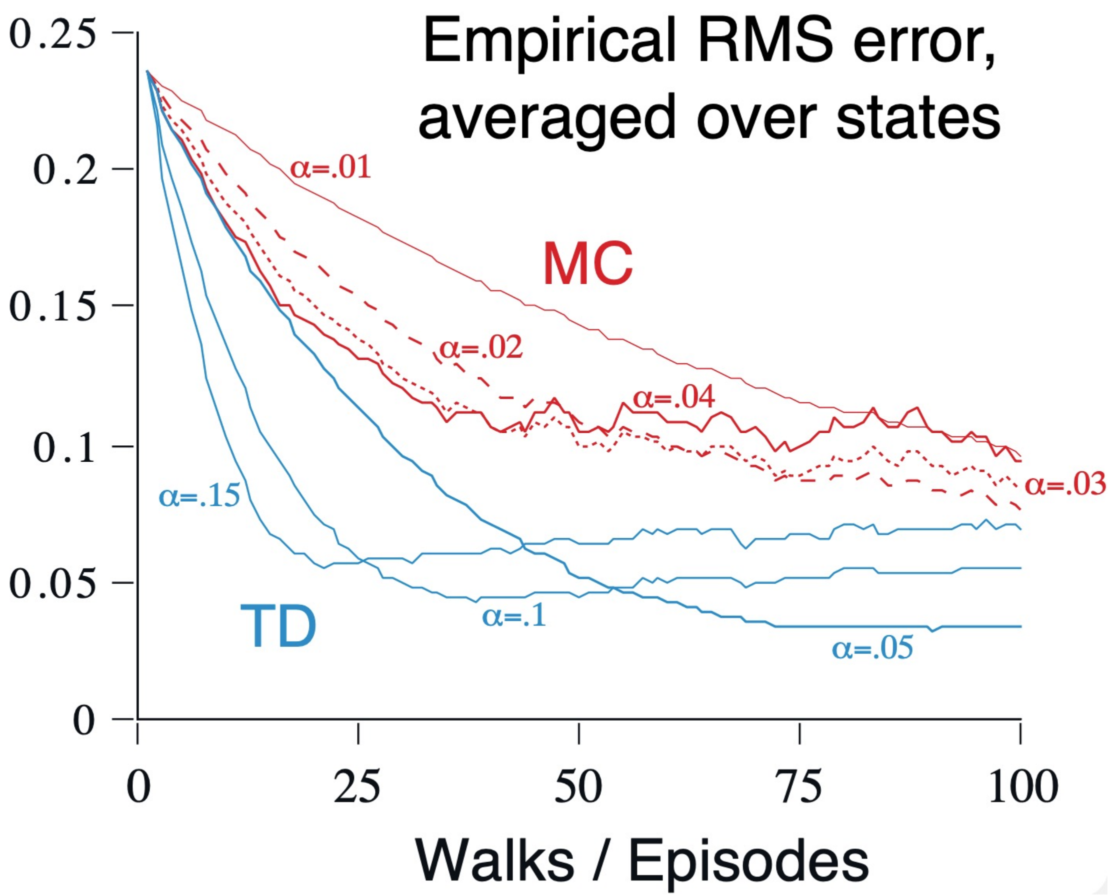
Example: Random Walk (5-states)





[An Introduction to Reinforcement Learning, Sutton and Barto]

Example: Random Walk – MC vs TD



[An Introduction to Reinforcement Learning, Sutton and Barto]

Batch MC and TD

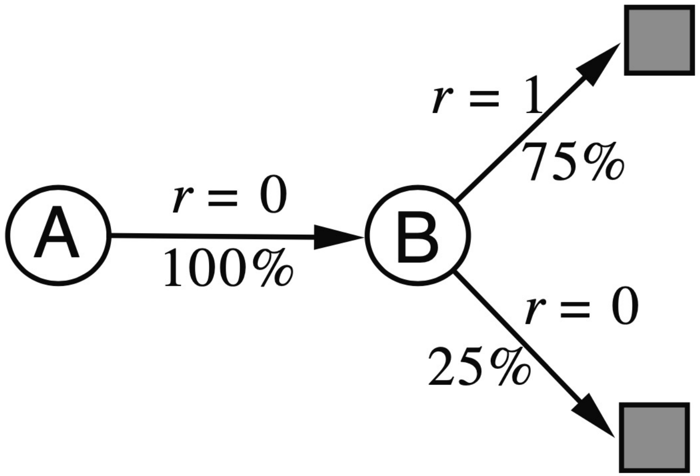
- MC and TD converge: $V(s) \rightarrow \mathrm{v}_\pi(s)$ as experience $\rightarrow \infty$
- But what about batch solution for finite experience?

$$
\begin{gathered}
s_1^1, a_1^1, r_2^1, \ldots s_{T_1}^1 \\
\vdots \\
s_1^K, a_1^K, r_2^K, \ldots, s_{T_K}^K
\end{gathered}
$$

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

Example: AB

- 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 $V(A), V(B)$?



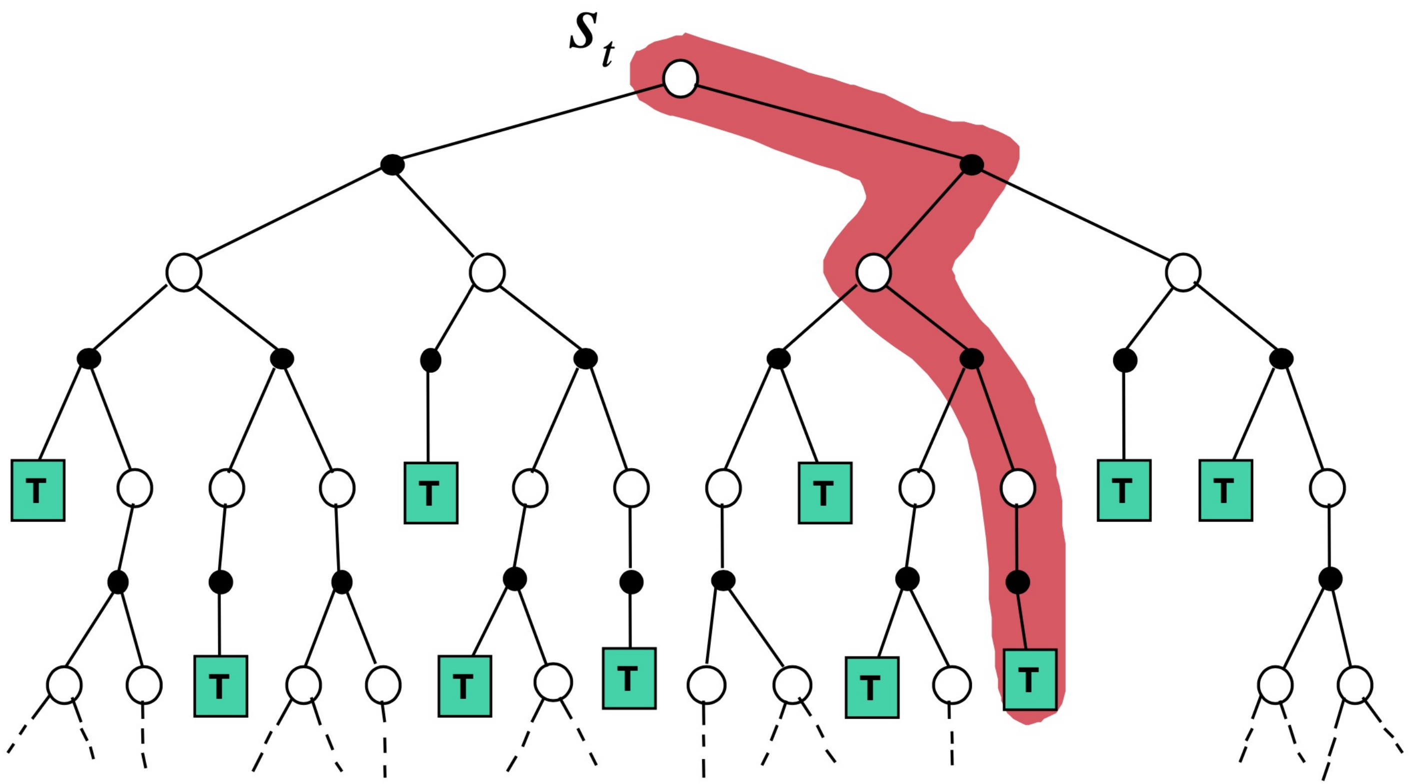
[An Introduction to Reinforcement Learning, Sutton and Barto]

Advantages and Disadvantages of MC vs. TD (3/3)

- 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 Update

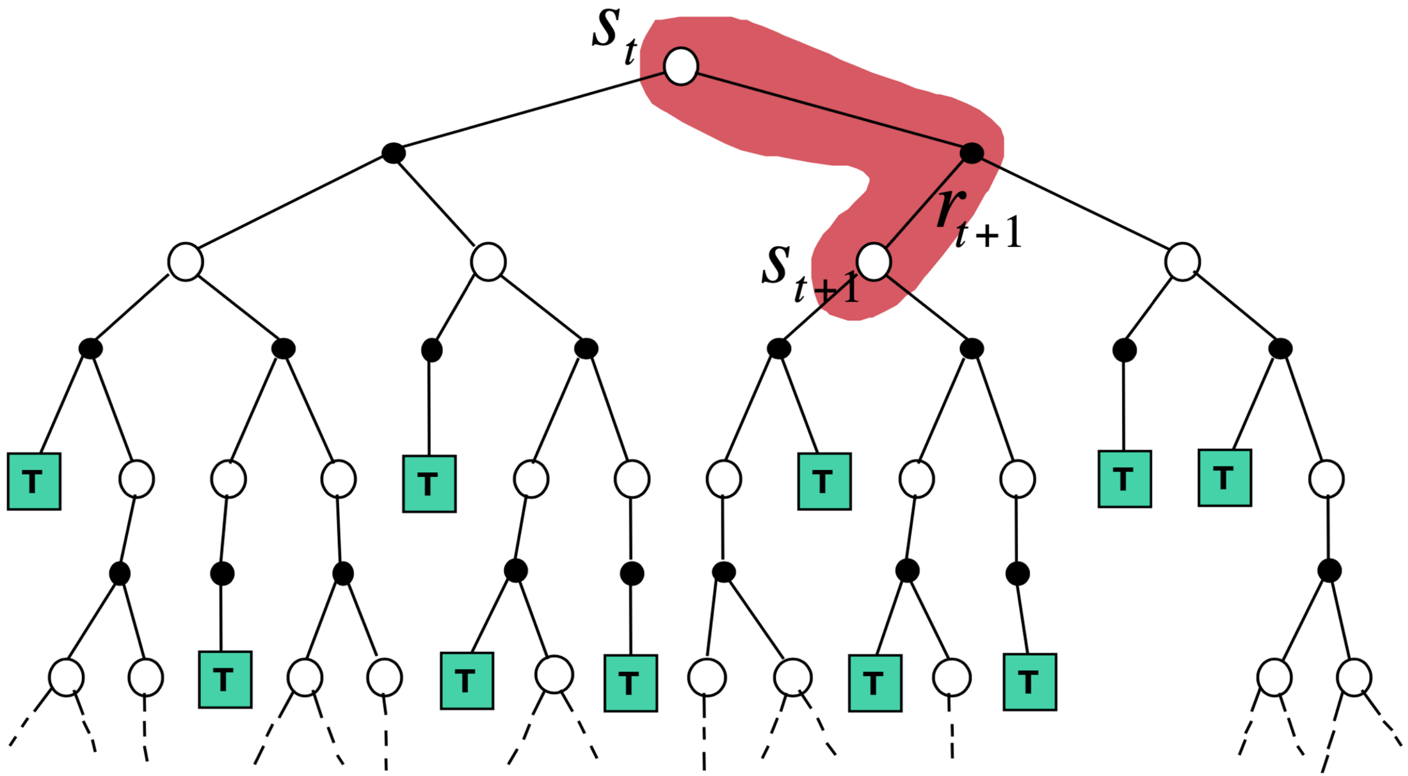
$$V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$$



[David Silver, IRL, UCL 2015]

Temporal-Difference Update

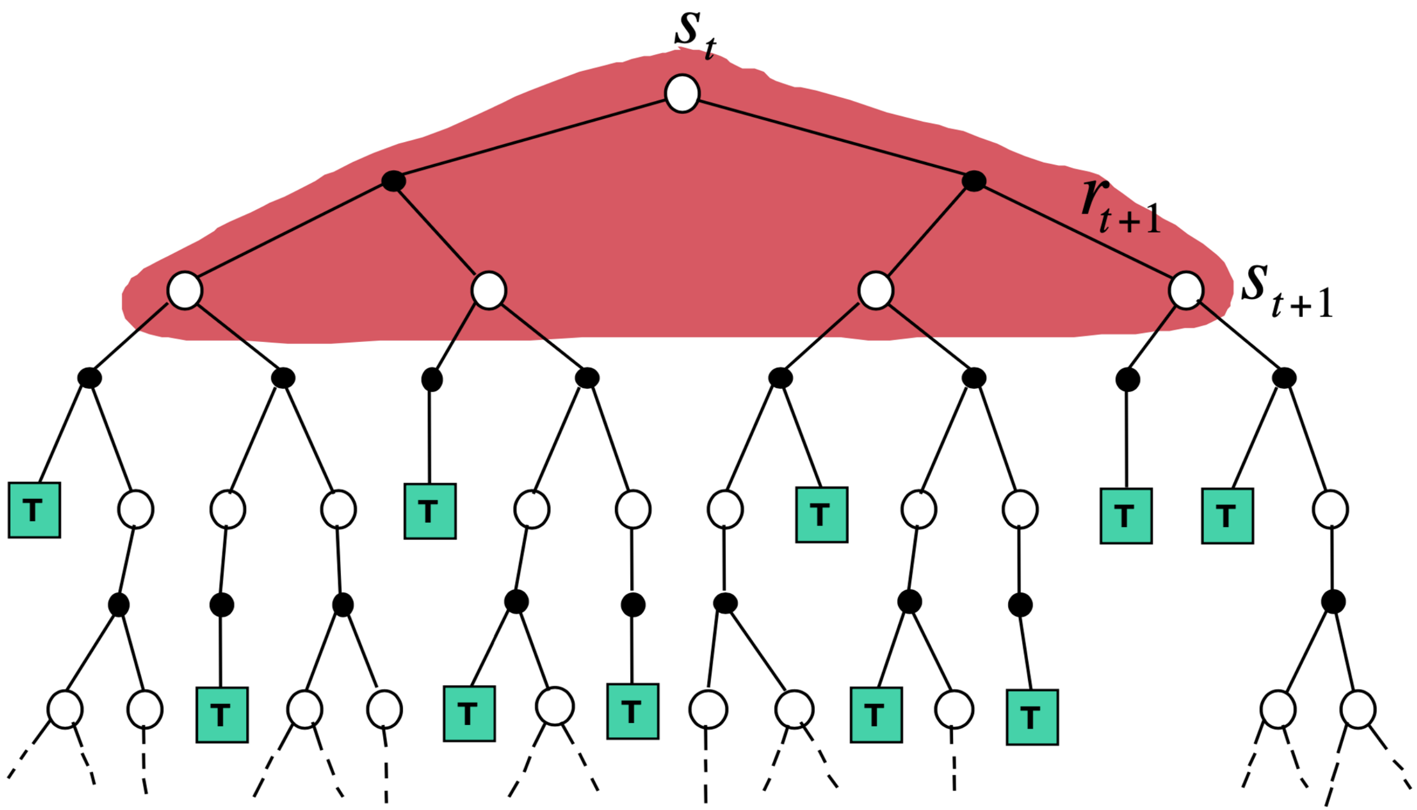
$$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$



[David Silver, IRL, UCL 2015]

Dynamic Programming Update

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

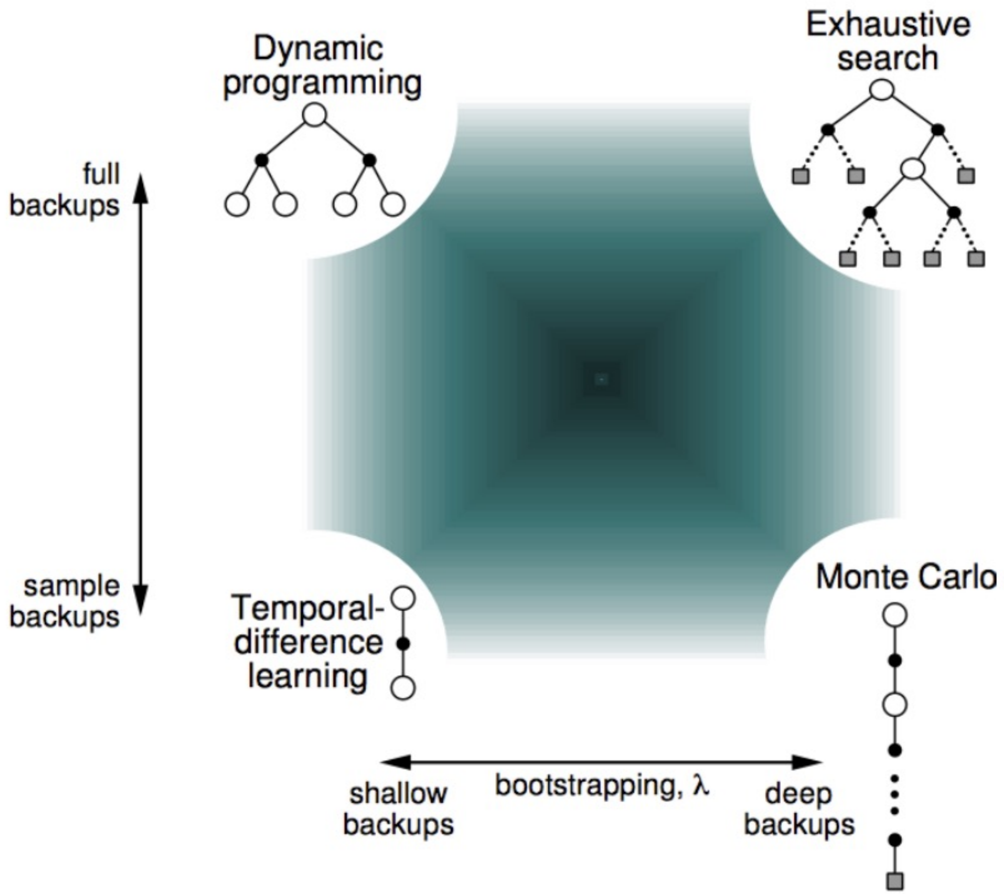


[David Silver, IRL, UCL 2015]

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

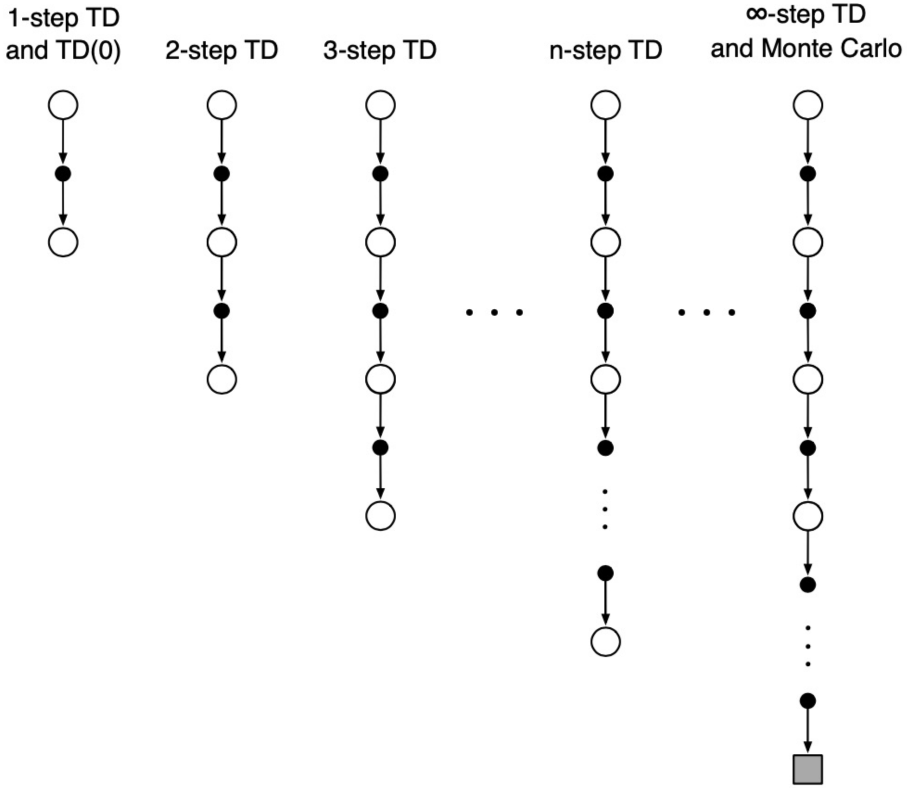


[David Silver, IRL, UCL 2015]

n-Step TD

n-Step Prediction

- Instead of just looking one step in the future, let's look n steps:



[An Introduction to Reinforcement Learning, Sutton and Barto]

n-Step Return

- Consider the following n-step returns for n=1,2,…,∞
  - $n = 1$ – TD

$$G_{t:t+1} = R_{t+1} + \gamma V(S_{t+1})$$

  - $n = 2$

$$G_{t:t+2} = R_{t+1} + \gamma R_{t+1} + \gamma^2 V(S_{t+2})$$

  - $n = \infty$ - MC

$$G_{t:t+\infty} = R_{t+1} + \gamma R_{t+1} + \ \ldots$$

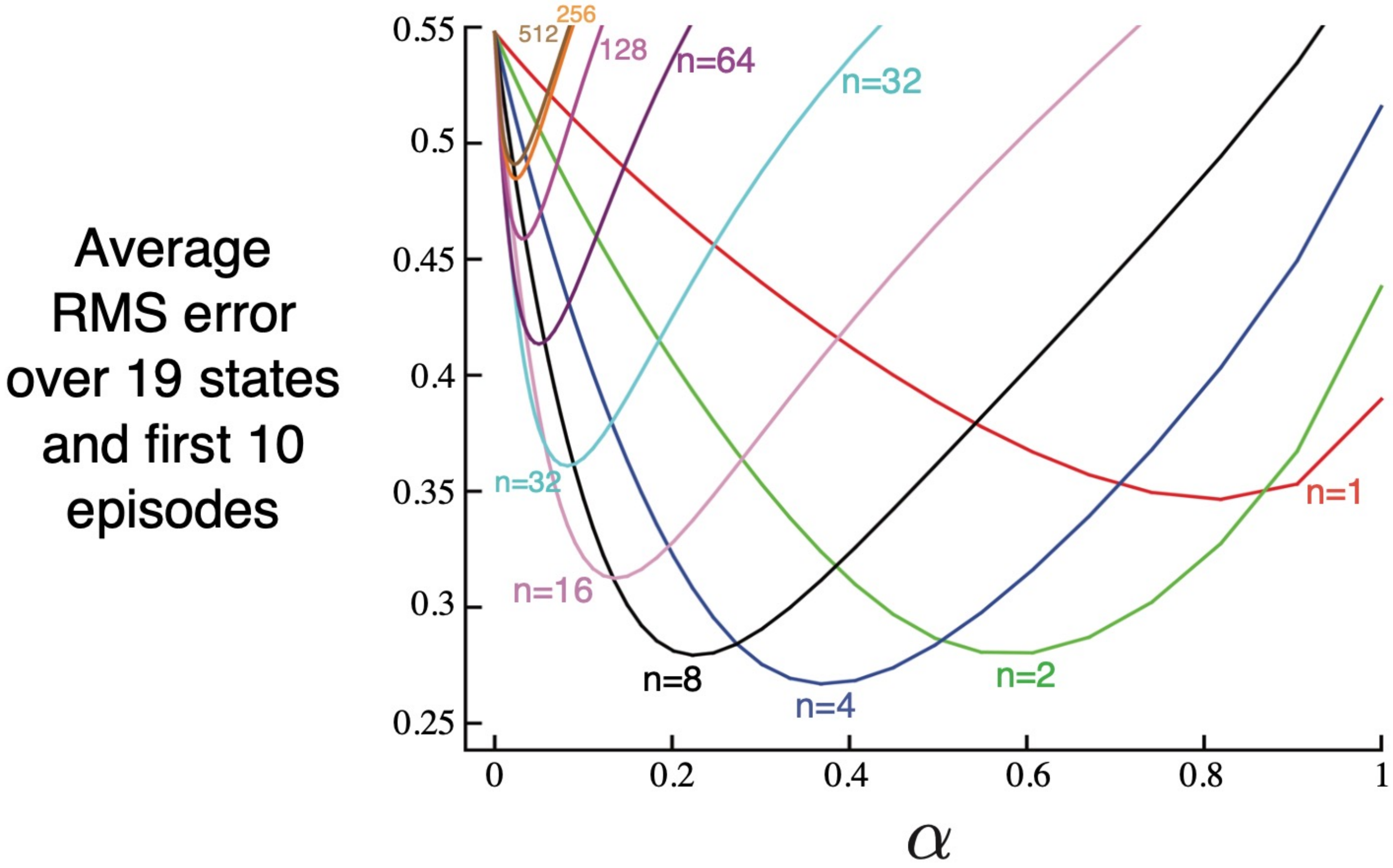
- We can define the n-step return

$$G_{t:t+n} = R_{t+1} + \gamma R_{t+1} + \cdots + \gamma^n V(S_{t+n})$$

- n-step temporal-difference learning

$$V(S_t) \leftarrow V(S_t) + \alpha(G_{t:t+n} - V(S_t))$$

Example: Random Walk (19-states)



Summary

- We discussed about three methods that can be used for the estimation of the value function given a policy
- MC is learning using sampling by averaging the return of many episodes
- TD is learning using bootstrapping and can use unfinished episodes
- n-step TD is lying at the space between TD and MC