Reinforcement Learning

Lecture 5

Model-free Prediction

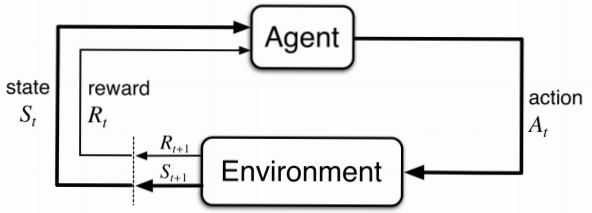
Monte Carlo

Temporal Differences

Robert Peharz

Institute of Theoretical Computer Science
Graz University of Technology
Winter Term 2023/24

Recap: Basic Notions in RL



Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, p)$

state space $\mathcal{S}$, action space $\mathcal{A}$, dynamics $p(s', r \mid s, a)$

Policy $\pi(a \mid s)$

Discounted Return

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad , \quad \gamma \in [0,1]$$

Value Function

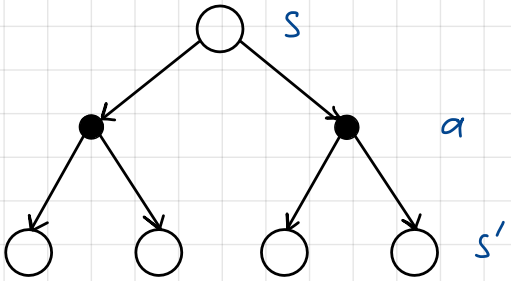
$$v_\pi(s) := \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right]$$

q-Function (state-action value function)

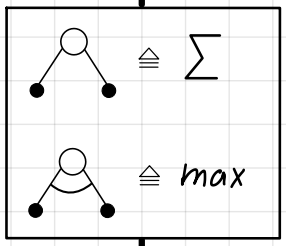
$$q_\pi(s,a) := \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right]$$

Recap: Bellman Equations, Backup Diagrams

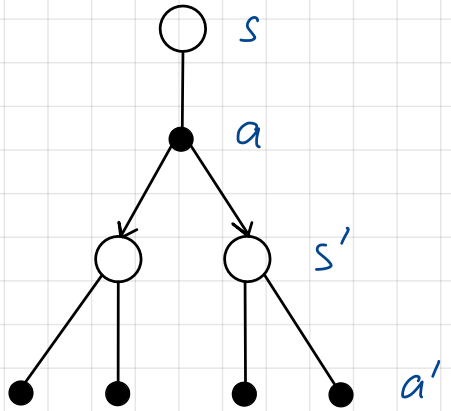
Bellman Expectation Equation in v



$$v_\pi(s) = r(s) + \gamma \sum_a \pi(a|s) \sum_{s'} p(s'|s,a)\, v_\pi(s')$$

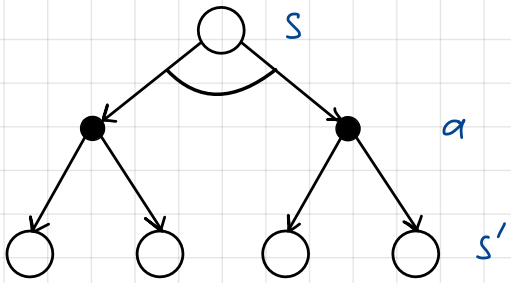


Bellman Expectation Equation in q



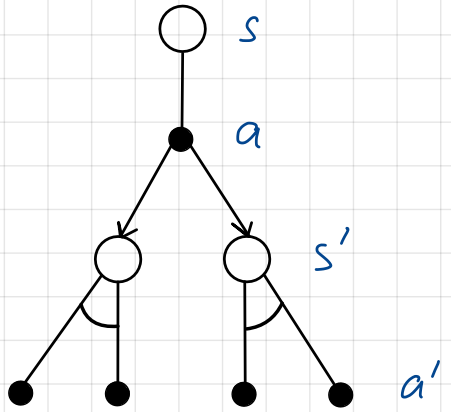
$$q_\pi(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \sum_{a'} \pi(a'|s')\, q_\pi(s',a')$$

Bellman Optimality Equation in v



$$v_*(s) = \max_a r(s,a) + \gamma \sum_{s'} p(s'|s,a)\, v_*(s')$$

Bellman Optimality Equation in q



$$q_*(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \max_{a'} q_*(s',a')$$

Recap: Policy Improvement, Policy Iteration, Value Iteration

greedy policy given current policy π

$$\pi'(a|s) = \begin{cases} 1 & \text{if } a = \arg\max_{a'} q_\pi(s,a') \\ 0 & \text{otherwise} \end{cases}$$

$$V^*(s) := \max_\pi V_\pi(s)$$

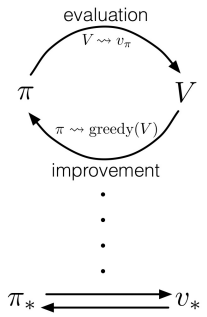
$$q^*(s,a) := \max_\pi q_\pi(s,a)$$

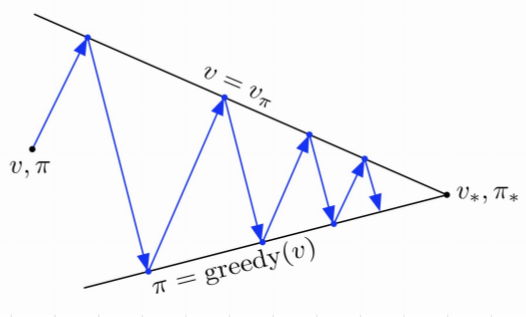
Policy Improvement Theorem

$$V_\pi(s) \leq V_{\pi'}(s) \quad \forall s \in \mathcal{S}$$

Theorem: Global Optimal Policy

For any initial policy π_0, policy iteration converges to an optimal policy with value function V^*





Value Iteration | **Policy Iteration with Truncated Policy Evaluation** | **Policy Iteration**

1 — number k of policy evaluation steps — ∞

Model-free Prediction, Monte Carlo

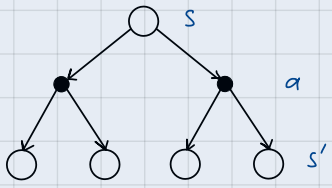
So far: Known MDP Dynamics

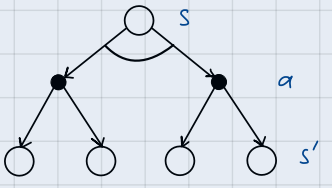
Planning

Bellman Expectation Equation

$p(s', r \mid s, a)$

Bellman Optimality Equation





$$v_\pi(s) = r(s) + \gamma \mathbb{E}[v_\pi(S_{t+1})]$$

$$v_\pi(s) = \max_a r(s,a) + \gamma \mathbb{E}[v_\pi(S_{t+1}) \mid a]$$

Policy Evaluation

(Generalized) Policy Iteration

Today: Unknown MDP Dynamics

Reinforcement Learning

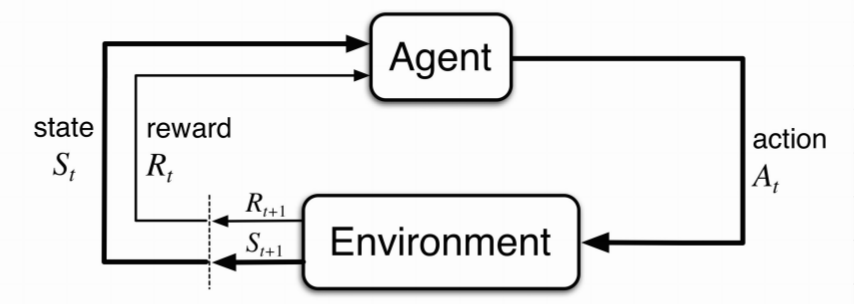
We can, however, interact with MDP and produce samples (experience)

Model-free RL directly learn $\hat{\pi}$ (v_π, q_π) Today

Model-based RL learn (some aspect of) MDP as well

We start with prediction / policy evaluation (compute v_π / q_π, π fixed)

Model-free Policy Evaluation



- MDP dynamics $p(s', r \mid s, a)$ (environment) unknown
- Policy $\pi(a \mid s)$ given
- We can collect experience in the form of **episodes**

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \ldots, S_K \sim \pi, p$$

i.e., assume episodic tasks for now

How to compute the value function?

$$V_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right]$$

Monte Carlo (MC) Estimation

Let $\underline{X}$ be a random vector and $f: \underline{\mathcal{X}} \to \mathbb{R}$ be a function. (state space of $\underline{X}$)
Assume that we want to compute $\mathbb{E}[f(\underline{X})]$ and that we have N i.i.d. (independent and identically distributed) samples, i.e.

$$\underline{X}_i \sim p_{\underline{X}}, \quad 1 \leq i \leq N$$

The **Monte Carlo estimator** for $\mathbb{E}[f(\underline{X})]$ is defined as

$$\hat{E} := \frac{1}{N} \sum_{i=1}^{N} f(\underline{X}_i)$$

(Note that $\hat{E}$ is a random variable)

$\hat{E}$ is unbiased, i.e. $\mathbb{E}[\hat{E}] = \mathbb{E}[f(\underline{X})]$

Its variance is $\operatorname{var}(\hat{E}) = \frac{\operatorname{var}(f(\underline{X}))}{N}$

Thus, $\hat{E} \xrightarrow[N \to \infty]{} \mathbb{E}[f(\underline{X})]$

Estimating Area of Unit Circle with MC

- What is the area of the unit circle? Easy, $r^2\pi = \pi \approx 3.1415\ldots$
- One way to check this empirically is <u>Monte Carlo</u>:
  - define uniform distribution on unit square $[-1,1] \times [-1,1]$

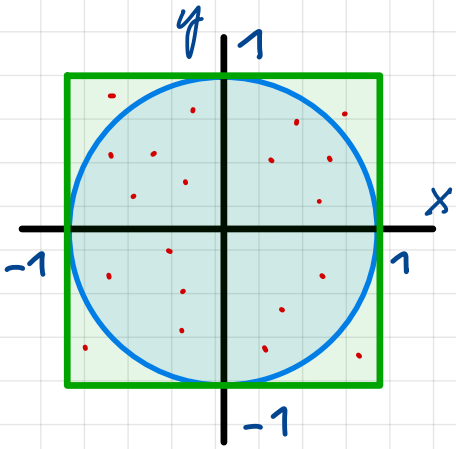
$$p(x,y) = \begin{cases} 1/4 & \text{if } |x| \le 1 \text{ and } |y| \le 1 \\ 0 & \text{o.w.} \end{cases}$$

  - area of unit circle:

indicator function $\mathbb{1}[a] = \begin{cases} 1 & \text{if } a \text{ is true} \\ 0 & \text{o.w.} \end{cases}$

$$\int_{-1}^{1}\int_{-1}^{1} \mathbb{1}[x^2+y^2 \le 1]\, dx\, dy = \iint 4\, p(x,y)\, \mathbb{1}[x^2+y^2 \le 1]\, dx\, dy$$

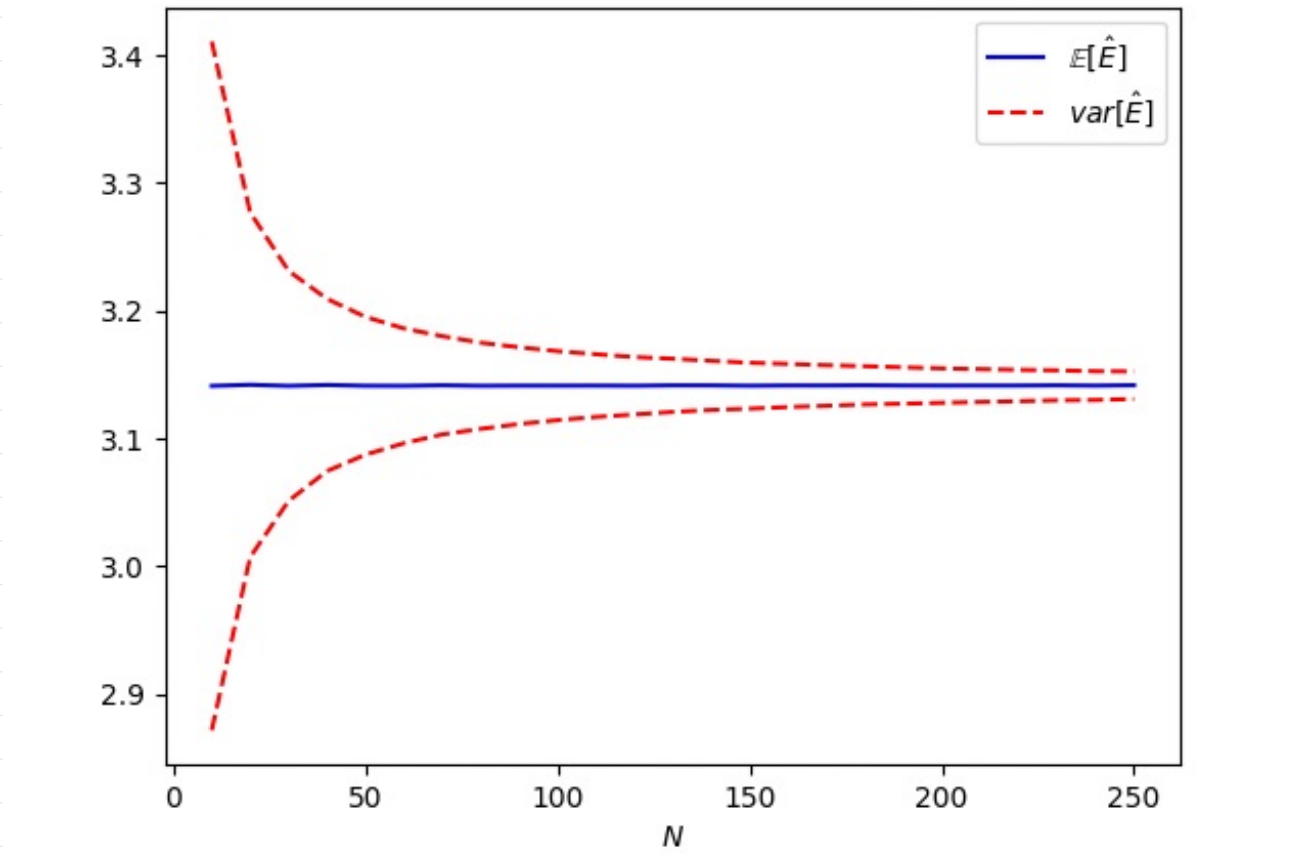
$$= 4\, \mathbb{E}[\mathbb{1}[x^2+y^2 \le 1]]$$



  - simulate N draws (X_i, Y_i) from unit square
  - $\boxed{\pi \approx 4 \cdot \hat{E}} = 4 \cdot \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}[X_i^2 + Y_i^2 \le 1]$

Estimating Area of Unit Circle MC con't

Mean and variance of Monte Carlo estimator $4\hat{E}$



(estimated by 10^6 repeated Monte Carlo runs, for each N)

Monte Carlo

- ✓ unbiased
- ✓ asymptotically exact
- ✓ p_X not required
- ✓ f not (explicitly) required

} just needs to be simlated

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

- simulate many episodes
- compute G
- average
- efficiency: also use sub-episodes for respective starting states

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots, S_{T-1}, A_{T-1}, R_T$$

Monte Carlo Evaluation (Prediction)

estimate with Monte Carlo

$$V_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right]$$

MC Prediction

- Given: π
- Initialize empty list Returns(s) for all $s \in \mathcal{S}$
- Initialize $V(s) = 0$ for all $s \in \mathcal{S}$

repeat

- generate an episode $S_0, A_0, R_1, \ldots, S_{T-2}, A_{T-2}, R_{T-1}, S_{T-1}, A_{T-1}, R_T$
- $G \leftarrow 0$
- iterate $t = T-1, \ldots, 0$:
  - $G \leftarrow \gamma G + R_{t+1}$
  - append G to Returns(S_t)
  - $V(S_t)$ = average(Returns(S_t))

// this recursively computes $G_t = \sum_k \gamma^k R_{t+k+1}$ for each sub-episode

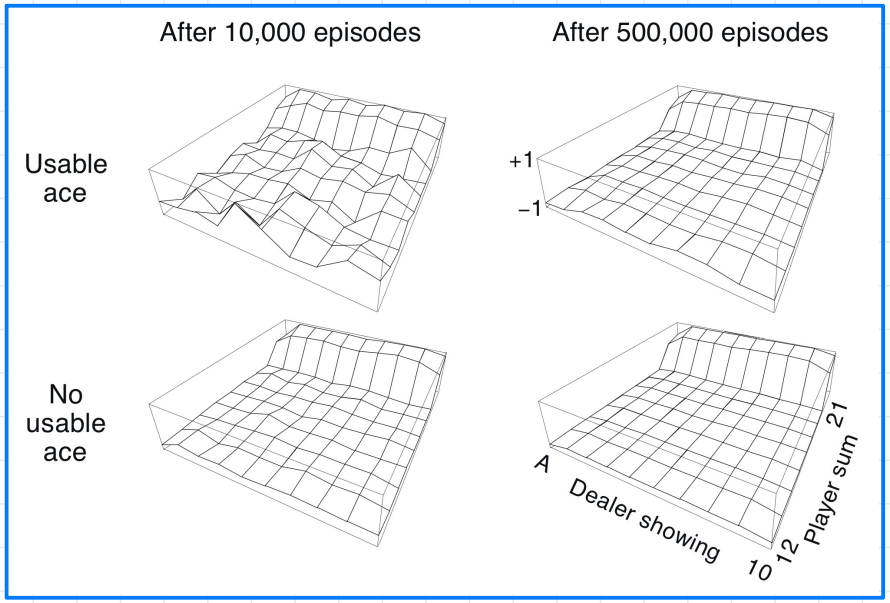
Black Jack

(Sutton & Barto, p. 93)

- you and the dealer get 2 cards
- one of the dealer's cards is face up (visible)
- you have the following options (actions)
  - **hit**: ask for another card
  - **stick**: stop and use your current hand
- if you get more than 21 points you are **bust** (you lose)
- dealer plays after you: they stick with 17 or higher
- win: $R = 1$, lose: $R = -1$, draw: $R = 0$
- assume policy: stick with 20 or 21, otherwise hit

2:	2 points
3:	3 points
⋮	
10:	10 points
face cards:	10 points
ace:	1 or 11 points

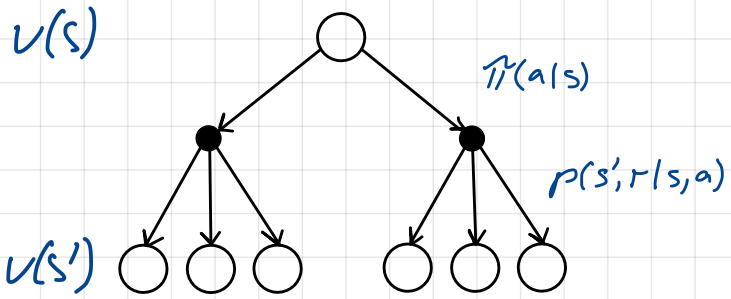
- easy to simulate
- Monte Carlo results ⟶



Temporal Difference Prediction

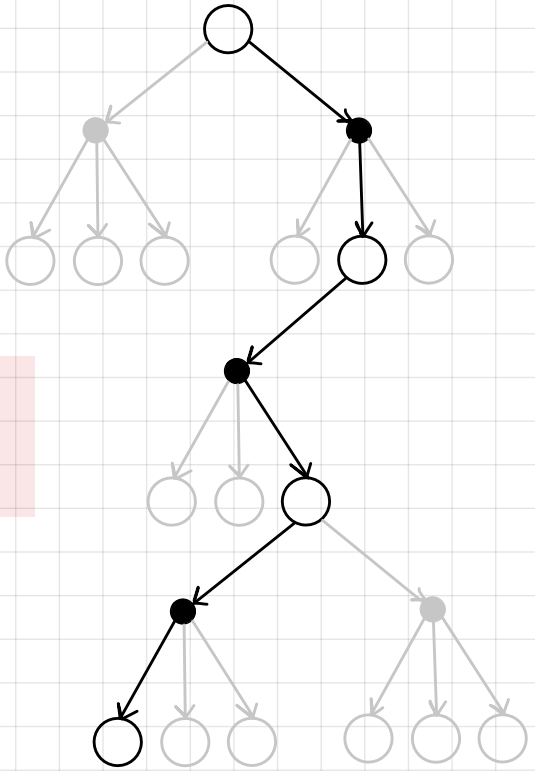
Unifying Planning and Monte Carlo?

Planning (Dynamic Programming)



- one step look-ahead considers all options
- $\mathbb{E}[\dots] = \sum_{a,s',r} \dots$
- uses current estimate $v(S_{t+1})$ $(q(S_{t+1}, A_{t+1}))$ as "backup" (Bellman equations)

Monte Carlo



- long range look-ahead
- considers one option for each t
- uses empirical G_t samples

What's in between?

Intermezzo: Online Monte Carlo

- Monte Carlo
  - maintain a set Returns(s) for each $s \in \mathcal{S}$
  - $v_\pi(s) = \mathbb{E}_\pi[G_t \mid s] \approx \text{average}(\text{Returns}(s))$
- Online Monte Carlo
  - $v(s) \leftarrow 0$, $N(s) \leftarrow 0$
  - whenever a new sample G_t becomes available for s:

    $N(s) \leftarrow N(s) + 1$

    $v(s) \leftarrow v(s) + \frac{1}{N(s)}(G_t - v(s))$
  - generalize this to a learning rule

    $v(s) \leftarrow v(s) + \alpha(\overbrace{G_t - v(s)}^{\text{"}\delta\text{"}})$

    α: step size; G_t: target; $v(s)$: current estimate
  - constant α: online learning, forgetting long past data

online (incremental) averages:

$$\bar{x}_N = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$= \frac{1}{N}\left(x_N + \sum_{i=1}^{N-1} x_i\right)$$

$$= \frac{1}{N}\left(x_N + (N-1)\,\bar{x}_{N-1}\right)$$

$$= \frac{1}{N}\left(x_N + N\bar{x}_{N-1} - \bar{x}_{N-1}\right)$$

$$= \bar{x}_{N-1} + \frac{1}{N}\left(x_N - \bar{x}_{N-1}\right)$$

Temporal Difference (TD) Prediction

$$V(S) \leftarrow V(S) + \alpha \left(G_t - V(S) \right)$$

target

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

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

$$\approx V(S')$$

Thus, could $R_{t+1} + \gamma V(S')$ be a good target? Still correct algorithm?

Temporal Difference (TD) Prediction

- Initialize $V(s)$ arbitrarily, except $V(s) = 0$ for terminal s
- for each episode
  - init s
  - repeat
    - take action $a \sim \pi(a|s)$, observe s', r
    - $V(s) \leftarrow V(s) + \alpha \left(r + \gamma V(s') - V(s) \right)$
    - $s \leftarrow s'$

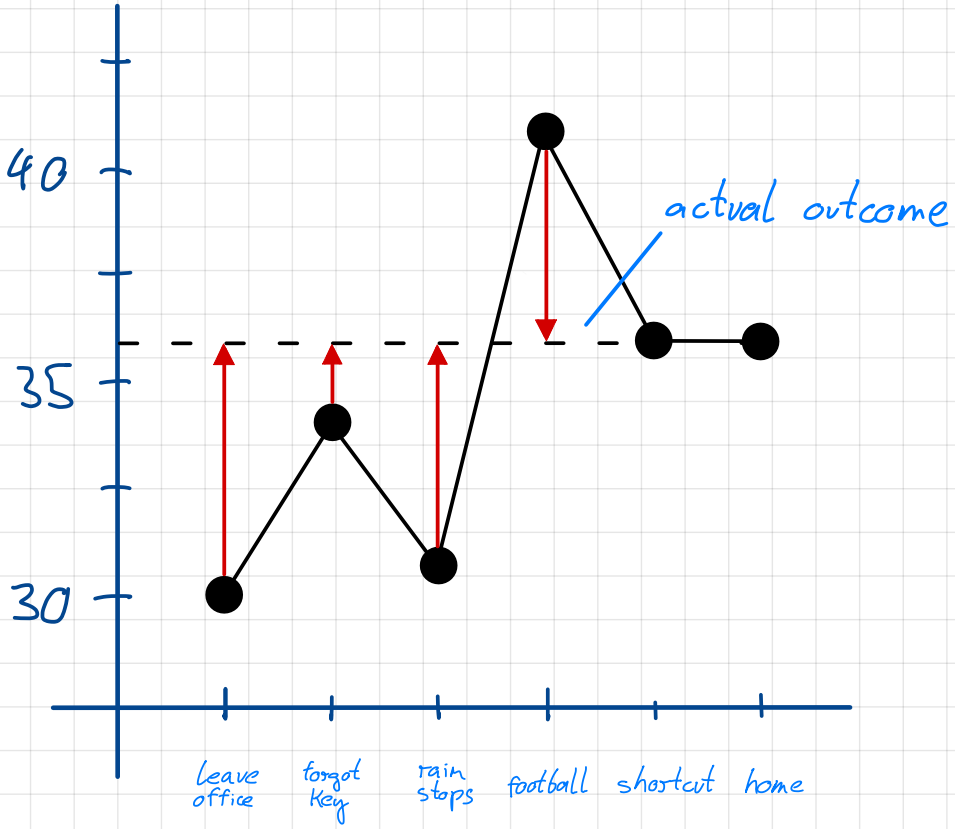
// for continuing task, this is one "infinite episode"

Cycling Home (from a Dutch University)

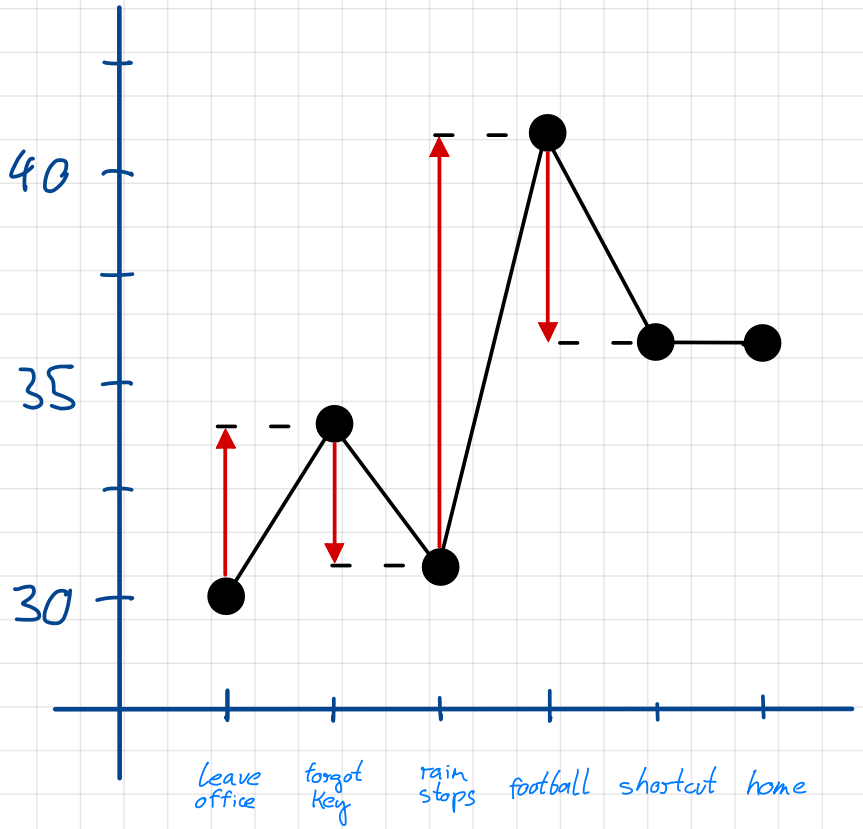
state	elapsed time	time updates	predicted total time
leaving office	0	0	30
forgot bike key on desk	2	+4	34
leave campus, rain stops	6	-3	31
local football club has won against Ajax Amsterdam: drunk fans block bike path	15	+10	41
construction of shortcut bike path completed	30	-5	36
arriving home	36	0	36

adapted from Sutton & Barto, p. 122

Cycling Home (from a Dutch University)



Updates via Monte Carlo



Updates via Temporal Differences

Soundness of TD

$$V(s) \leftarrow V(s) + \alpha \underbrace{(R_{t+1} + \gamma V(s') - V(s))}_{\delta_t : \text{TD error}}$$

Assume $\mathbb{E}[\delta_t \mid S_t = s] = 0$

Thus, $\mathbb{E}[R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \mid S_t = s] = 0$

$$\mathbb{E}[V(S_t) \mid S_t = s] = \mathbb{E}[R_{t+1} + \gamma V(S_{t+1}) \mid S_t = s]$$

$$V(s) = \mathbb{E}[R_{t+1} + \gamma V(S_{t+1}) \mid S_t = s]$$ Bellman expectation equation !

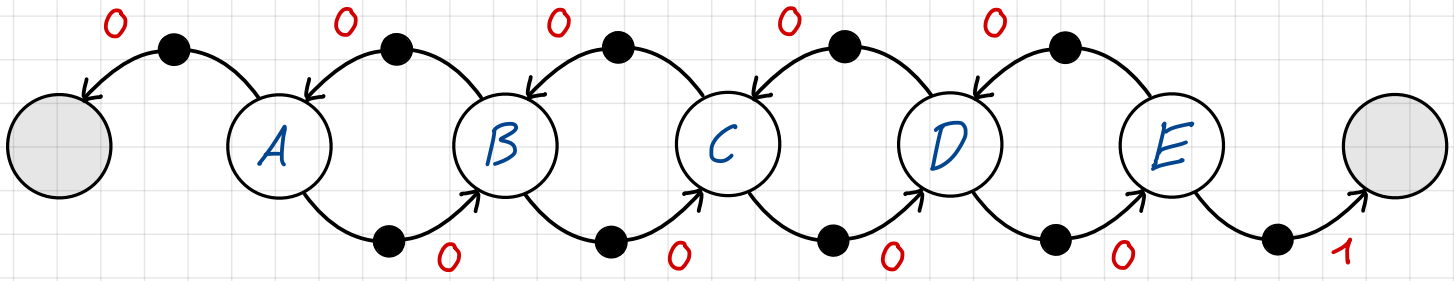
Intuitively, TD aims to establish Bellman expectation equation, at which point $V = V_\pi$. This indeed happens in the infinite data limit if stepsize α_t is set correctly:

$$\sum_{t=0}^{\infty} \alpha_t = \infty, \quad \sum_{t=0}^{\infty} \alpha_t^2 < \infty$$ e.g. $\alpha_t = \frac{1}{t}$

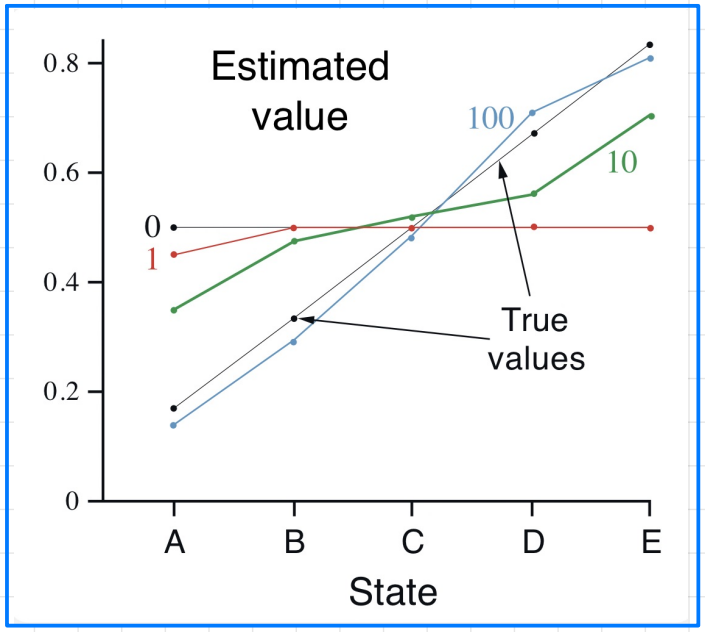
(standard conditions from stochastic optimization, cf. Robins-Monroe, 1951)

Monte Carlo vs. TD

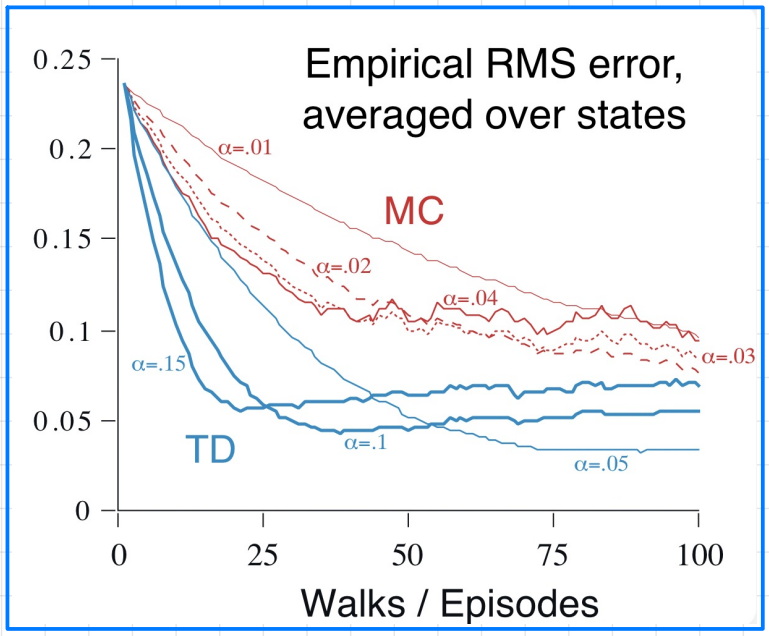
(Sutton & Barto, p.125)



$\tilde{\pi}(a|s) = 0.5$



Value function learned by TD



MC vs. TD

Monte Carlo vs. TD II

(Sutton & Barto, p. 127)

How do MC and TD compare on finite data?
Consider the following data for state and reward (ignoring actions)

1: A, 0, B, 0
2: B, 1
3: B, 1
4: B, 0
5: B, 1
6: B, 1
7: B, 1
8: B, 1

How would you estimate $v(A)$ and $v(B)$?

Assume undiscounted return.

Monte Carlo vs. TD II

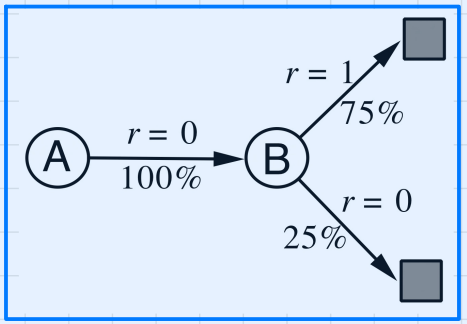
(Sutton & Barto, p.127)

For B, it is sensible to assume $v(B) = \frac{6}{8} = \frac{3}{4}$, since in 6 out of 8 cases when being in state B we got 1 reward, 0 otherwise.

There are <u>two meaningful answers for A</u>

1) <u>the "Monte Carlo way"</u>
   once observed A and got $G = 0$, thus, estimate $v(A) = 0$.
2) <u>the "TD way"</u>
   whenever we were in A, we got 0 reward and went to B, which we estimated as $v(B) = \frac{3}{4}$. Thus, estimate $v(A) = \frac{3}{4}$.

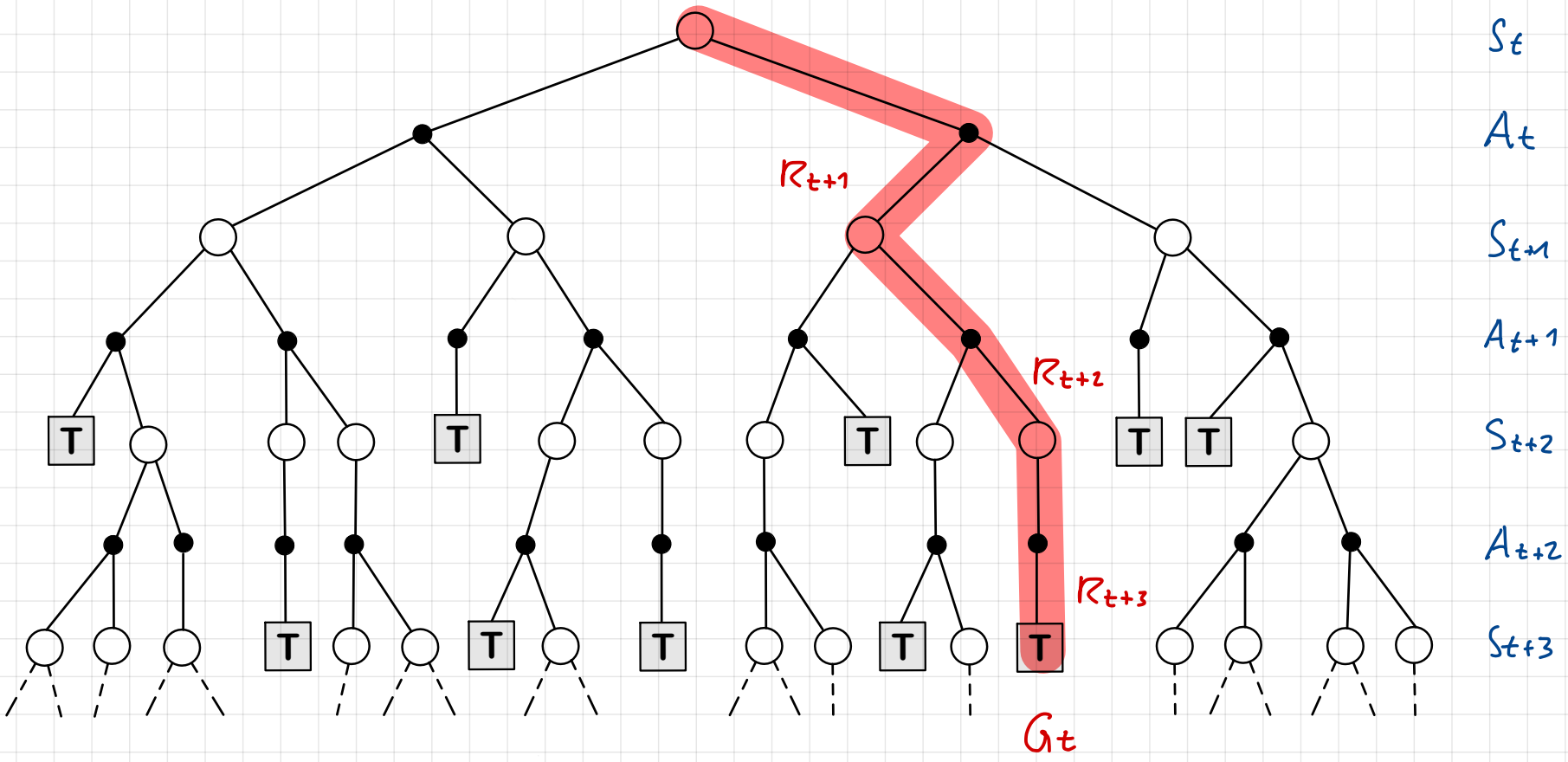
TD is **<u>equivalent</u>** to
1) fitting the MDP with <u>maximum likelihood</u>
2) solving MDP with Dynamic Programming



MC <u>does not exploit the MDP structure</u>.
In fact, MC <u>still works when **Markov property is violated**!</u>

Monte Carlo Backup

$$V(S) \leftarrow V(S) + \alpha \left(G_t - V(S) \right)$$

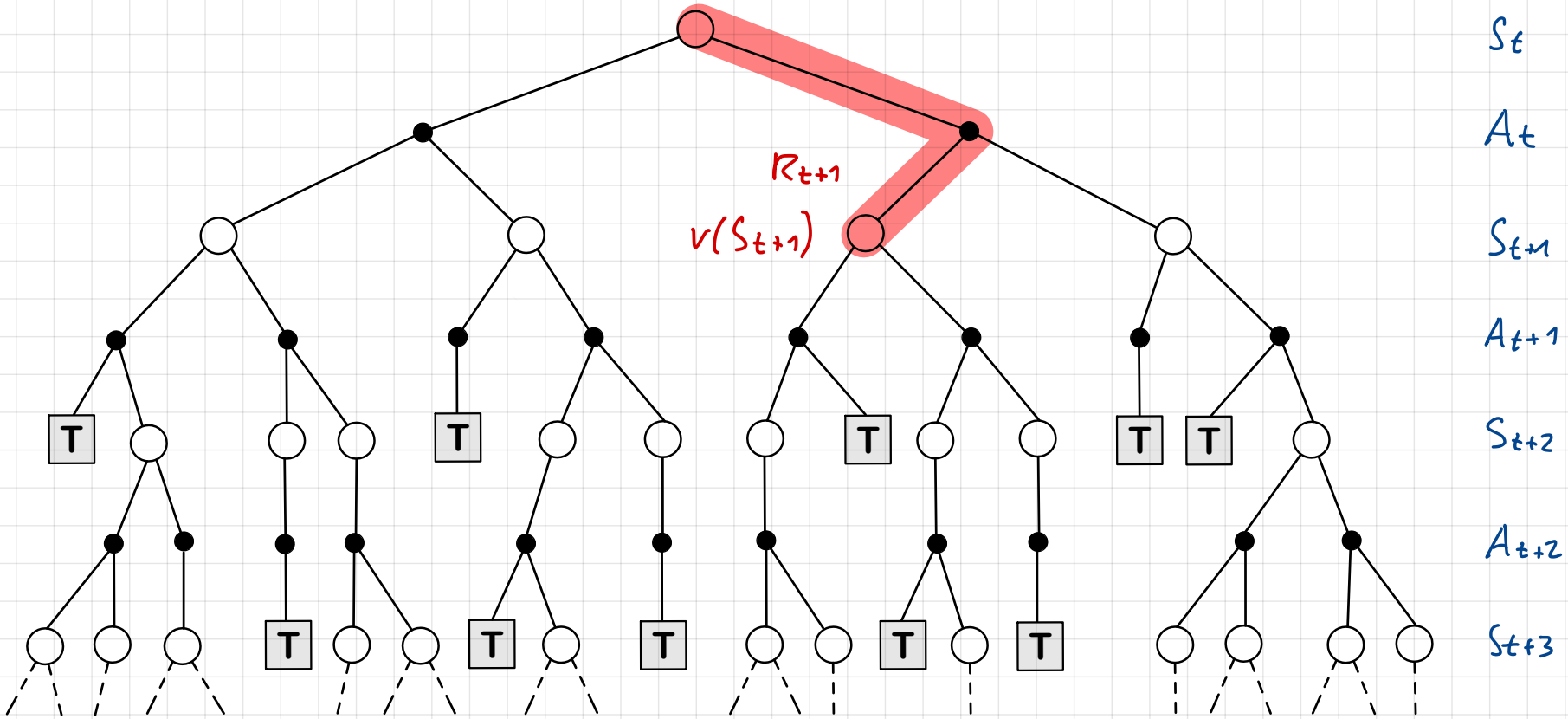


Adapted from D. Silver

TD Backup

uses current estimate v as part of target: "bootstrapping" *

$$v(s) \leftarrow v(s) + \alpha \left(R_{t+1} + \gamma v(S_{t+1}) - v(s) \right)$$



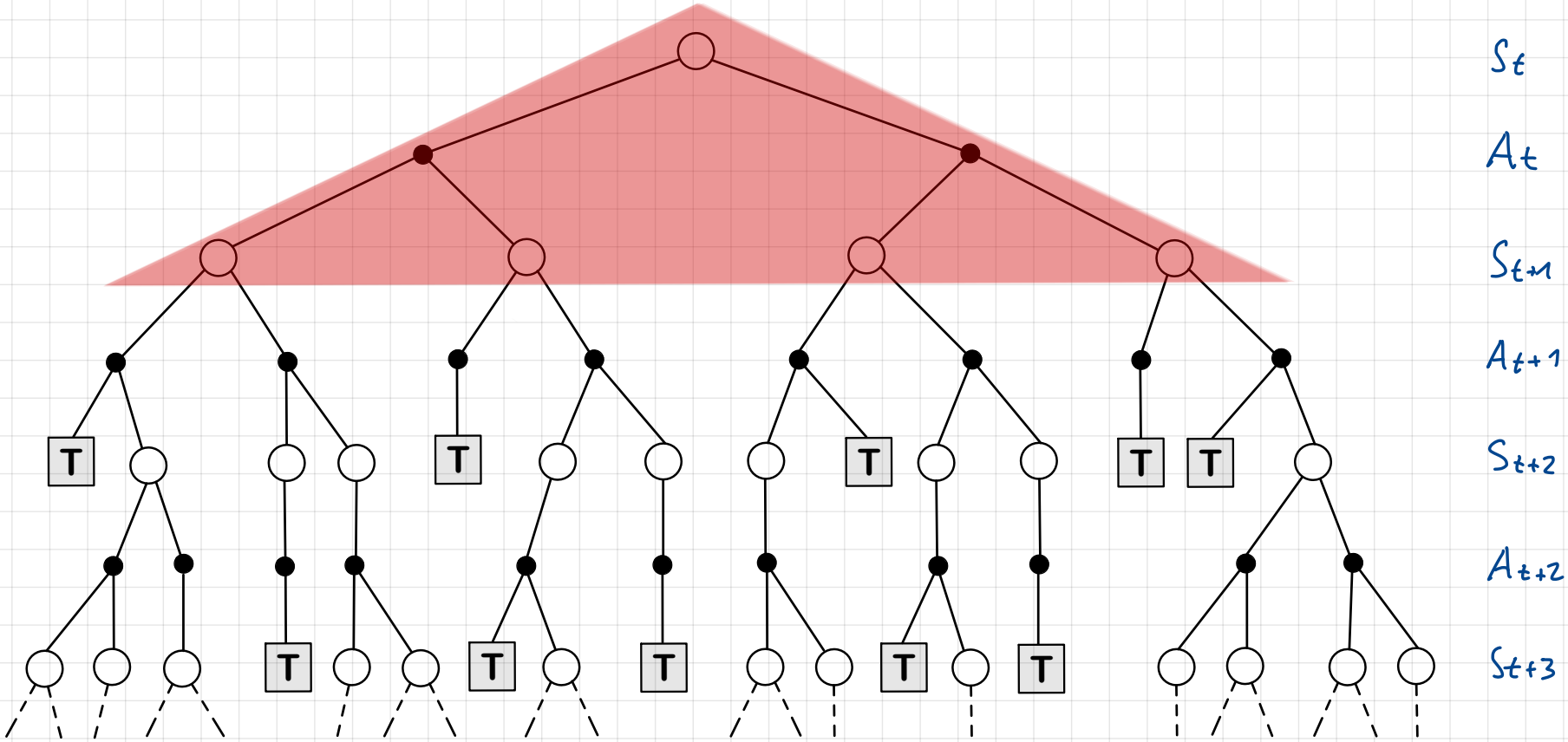
* don't confuse with bootstrap in statistics

Adapted from D. Silver

Dynamical Programming Backup

bootstrapping

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



Adapted from D. Silver

Properties of Prediction Algorithms

	allows for continuing tasks	bootstraps (uses current V as proxy)	model-free	samples	look-ahead
Dynamic Programming	✓	✓			1
Monte Carlo			✓	✓	∞
Temporal Differences	✓	✓	✓	✓	1

N-step Time Differences

We can easily come up with schemes in between TD and MC.

General sampling-based update rule

$$V(s) \leftarrow V(s) + \alpha \left(\underbrace{\hat{G}}_{\text{target}} - V(s) \right)$$

(1-step) TD: $\hat{G} = R_{t+1} + \gamma V(S_{t+1})$

2-step TD: $\hat{G} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2})$

3-step TD: $\hat{G} = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 V(S_{t+3})$

$\vdots$

∞-step TD (MC) $\hat{G} = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \ldots = G_t$

N-step Time Differences

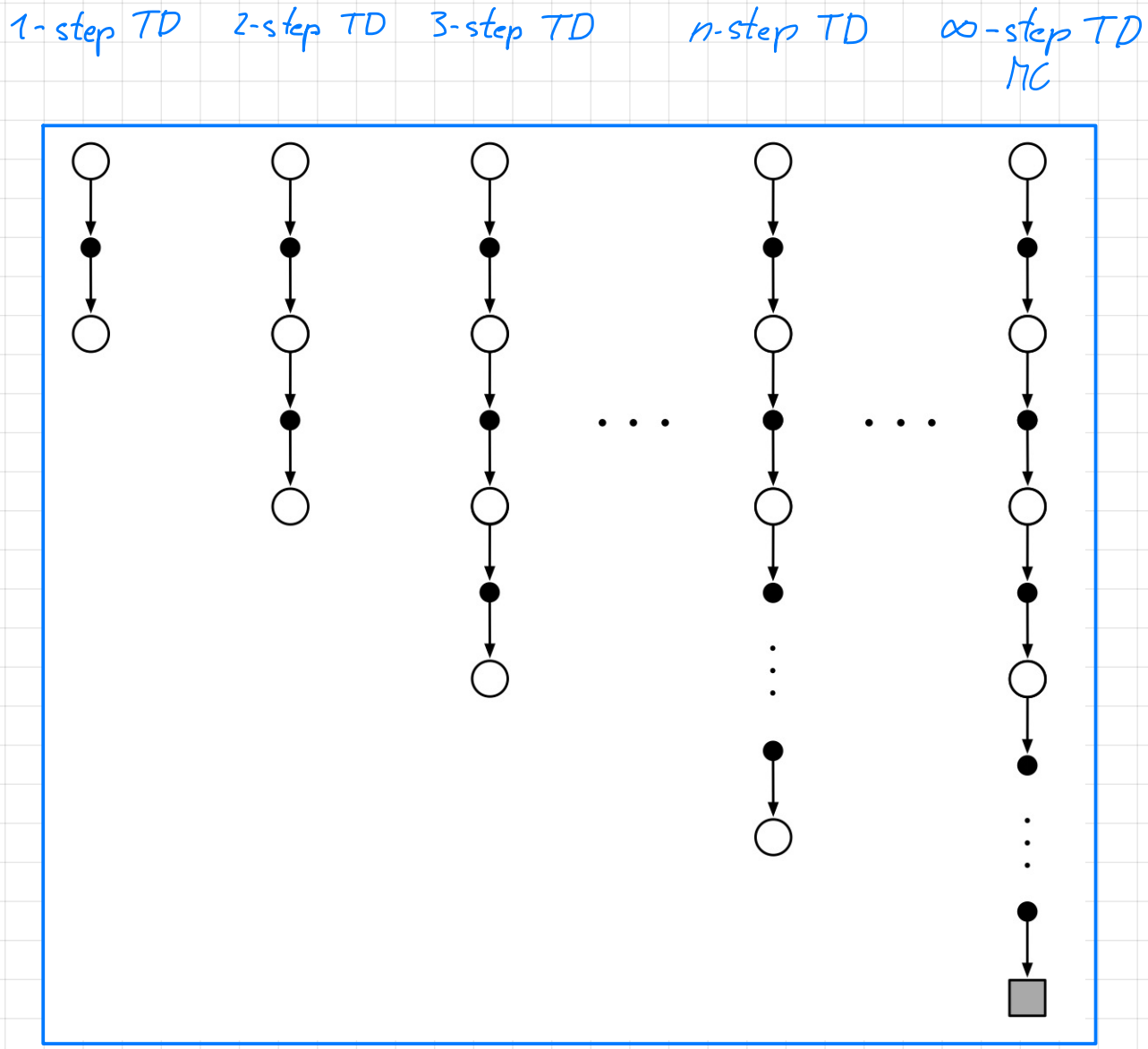


Image: Sutton & Barto

Unified View of RL Algorithms

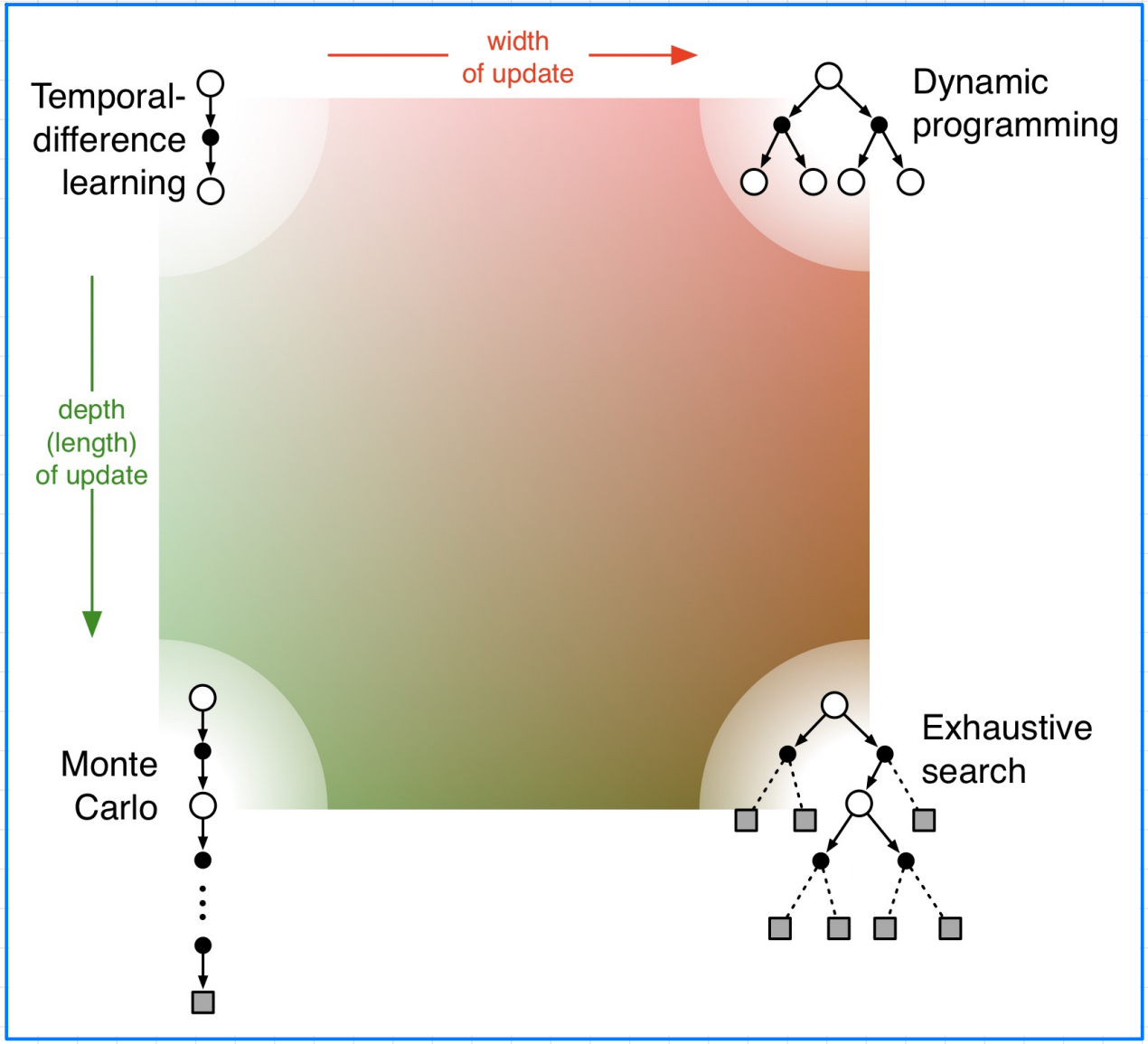


Image: Sutton & Barto