Probabilistic Decision Making

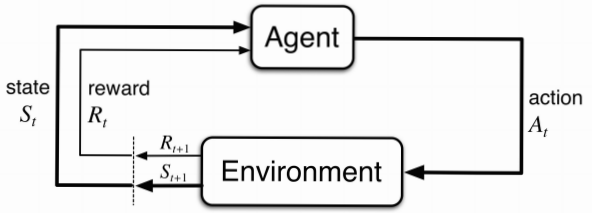
Lecture 16

Reinforcement Learning 3

Robert Peharz

Institute of Machine Learning and Neural Computation
Graz University of Technology
Winter Term 2025/26

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: Policy Improvement, Policy Iteration, Value Iteration

greedy policy given current policy π

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

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

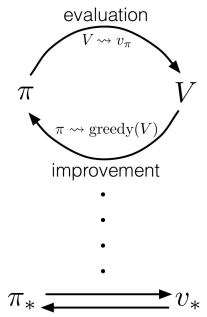
$$q^*(s,a) := \max_\pi q_\pi(s,a)$$

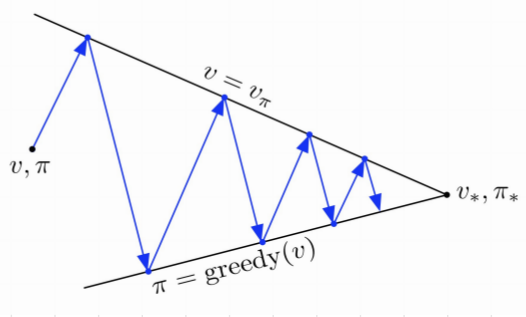
Policy Improvement Theorem

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

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 — ∞

So far: Known MDP Dynamics

Planning

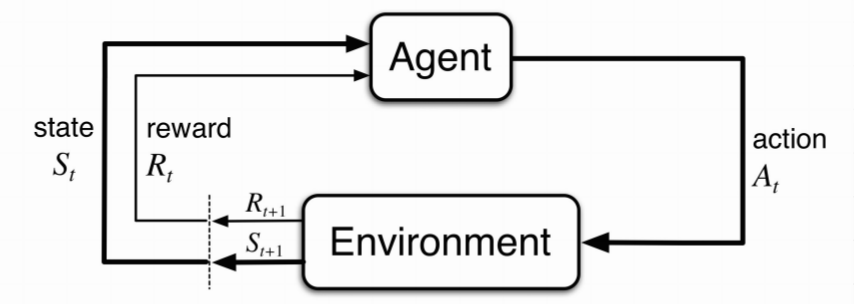
- so far, we computing value functions (v, q) and using them to learn optimal policies
- here, we always assumed that we have access to MDP
- this scenario is called planning (known MDP)

Today: Unknown MDP Dynamics

Reinforcement Learning

- now we assume that we can only interact with the MDP
- in particular, we assume we can sample from the dynamics $p(s', r | s, a)$
- like in planning, we have two core problems
  - policy evaluation
  - policy improvement

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 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 $S_0, A_0, R_1, \ldots, S_{T-2}, A_{T-2}, R_{T-1}, S_{T-1}, A_{T-1}, R_T$ (sub-episodes)
- $G \leftarrow 0$
- 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

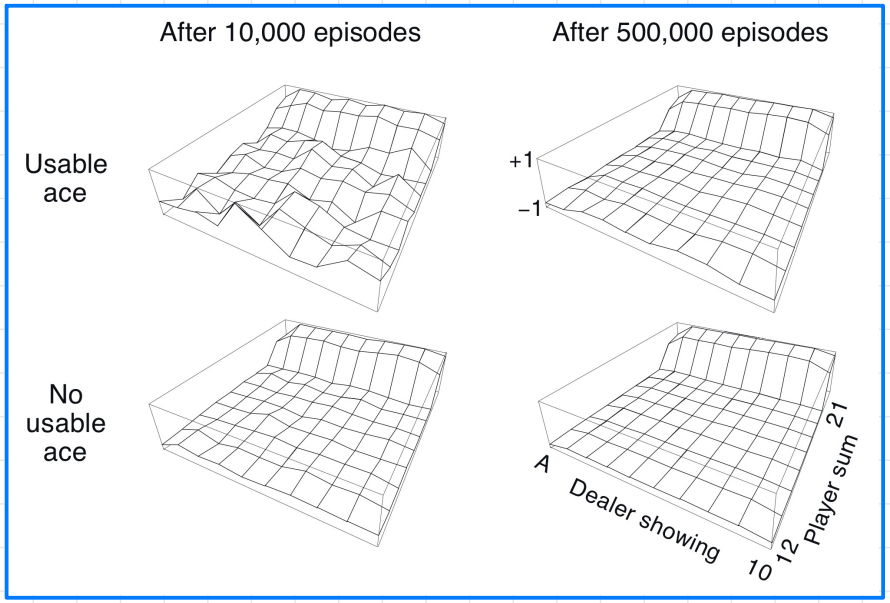
(Sutton & Barto, p. 93)

- 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

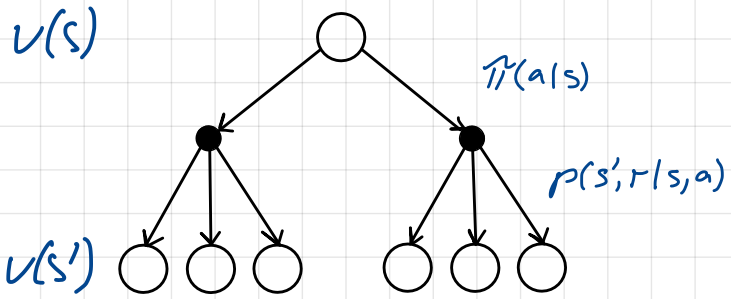
- easy to simulate
- Monte Carlo results ⟶



Temporal Difference Prediction

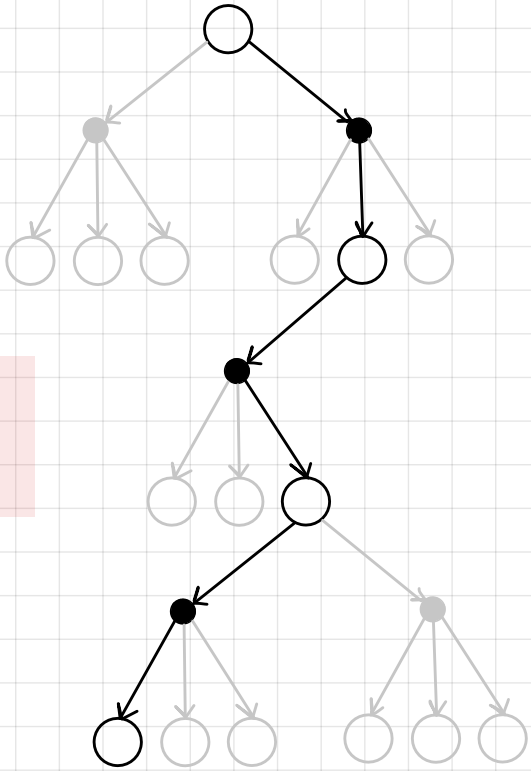
Unifying Planning and Monte Carlo?

Planning (Dynamic Programming)



- one step look-ahead considers all options
- 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"

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 !

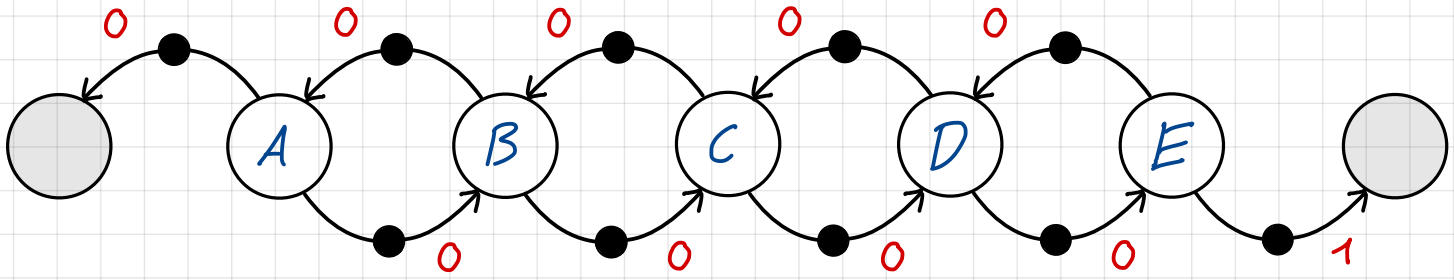
Intuitively, TD aims to establish Bellman expectation equation, at which point $V = V_\pi$. This indeed happens in the infinite data limit if stepsize α_t is set correctly:

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

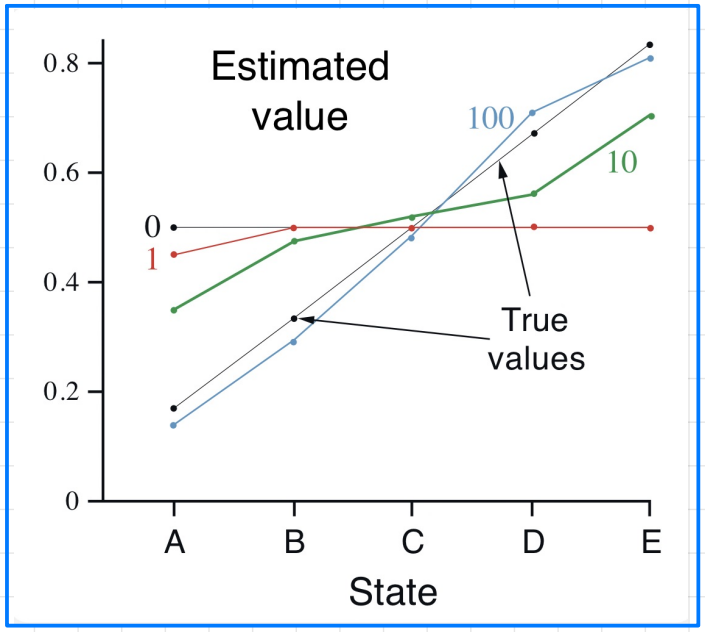
(standard conditions from stochastic optimization, cf. Robins-Monroe, 1951)

Monte Carlo vs. TD

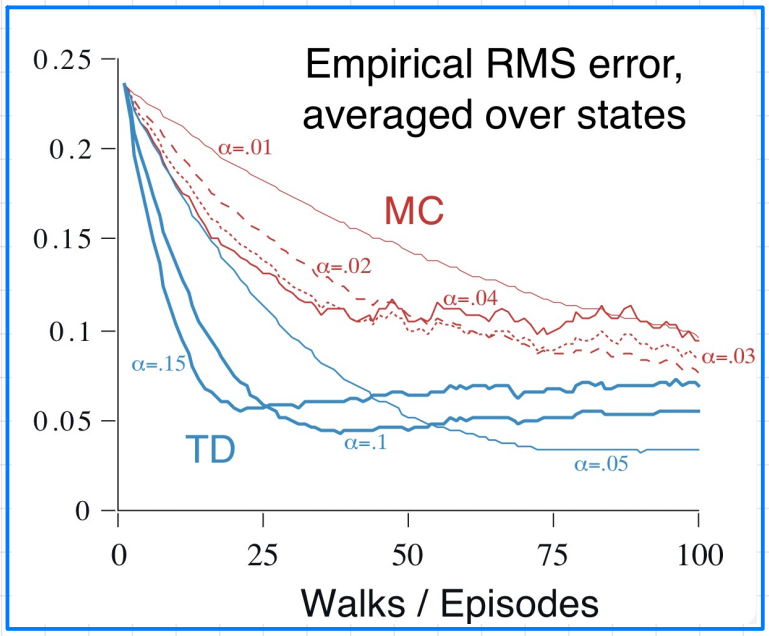
(Sutton & Barto, p.125)



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



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

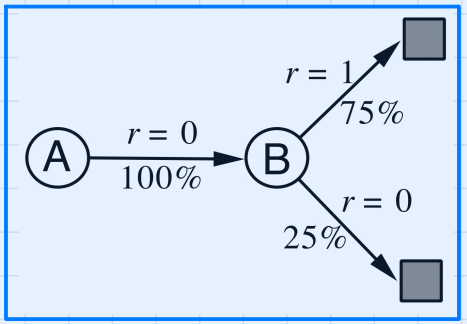
(Sutton & Barto, p.127)

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 $v(A) = \frac{3}{4}$.

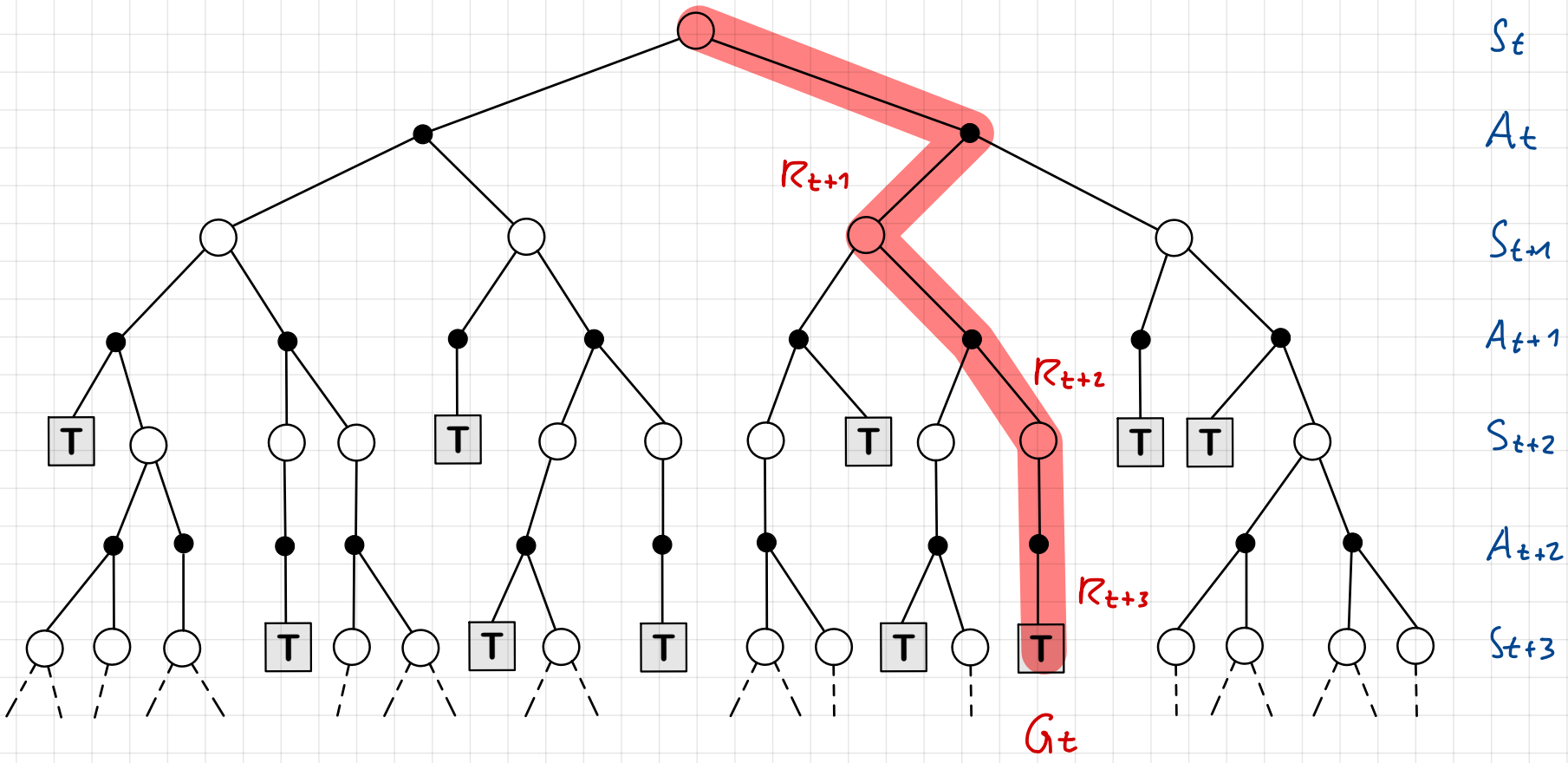
TD is **<u>equivalent</u>** to
1) fitting the MDP with <u>maximum likelihood</u>
2) solving MDP with Dynamic Programming



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

Monte Carlo Update

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

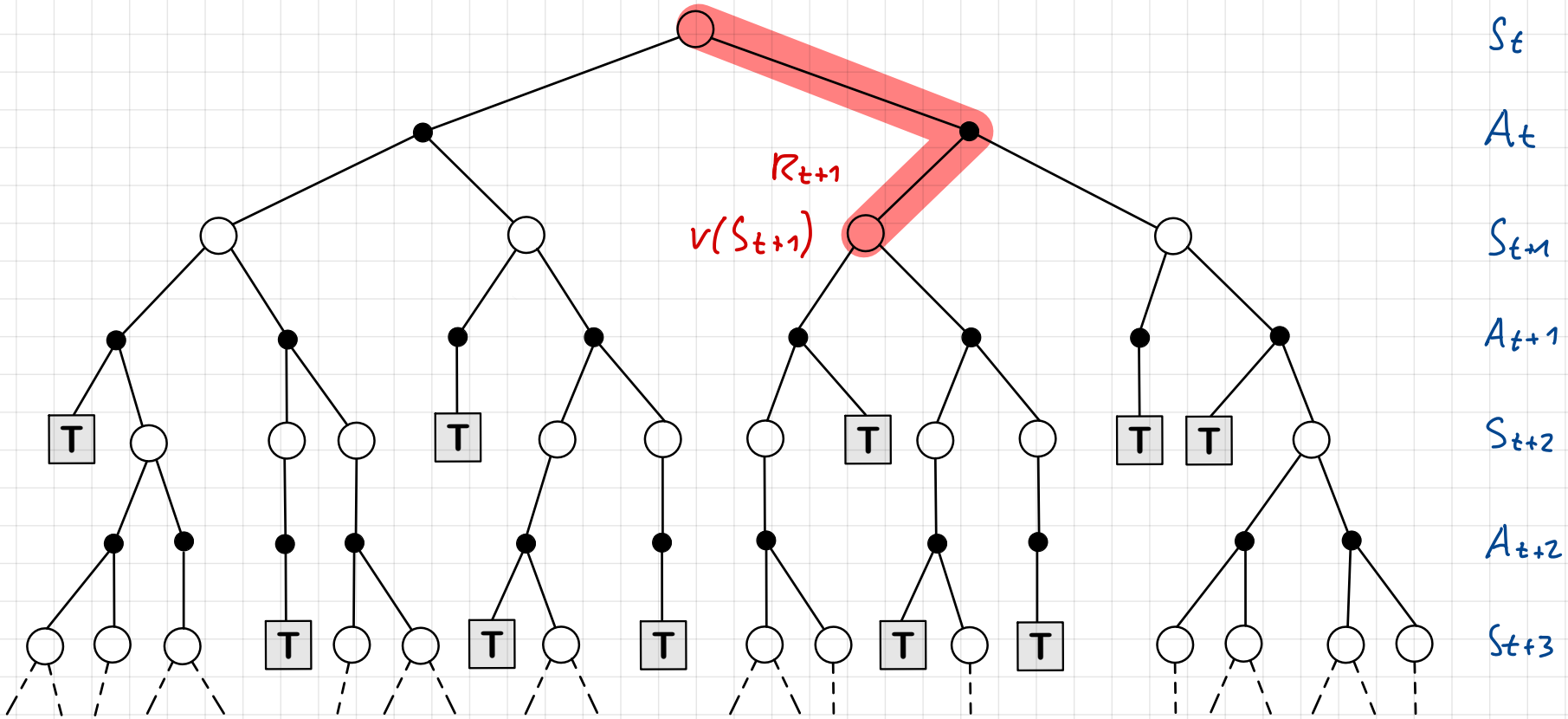


Adapted from D. Silver

TD Update

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

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

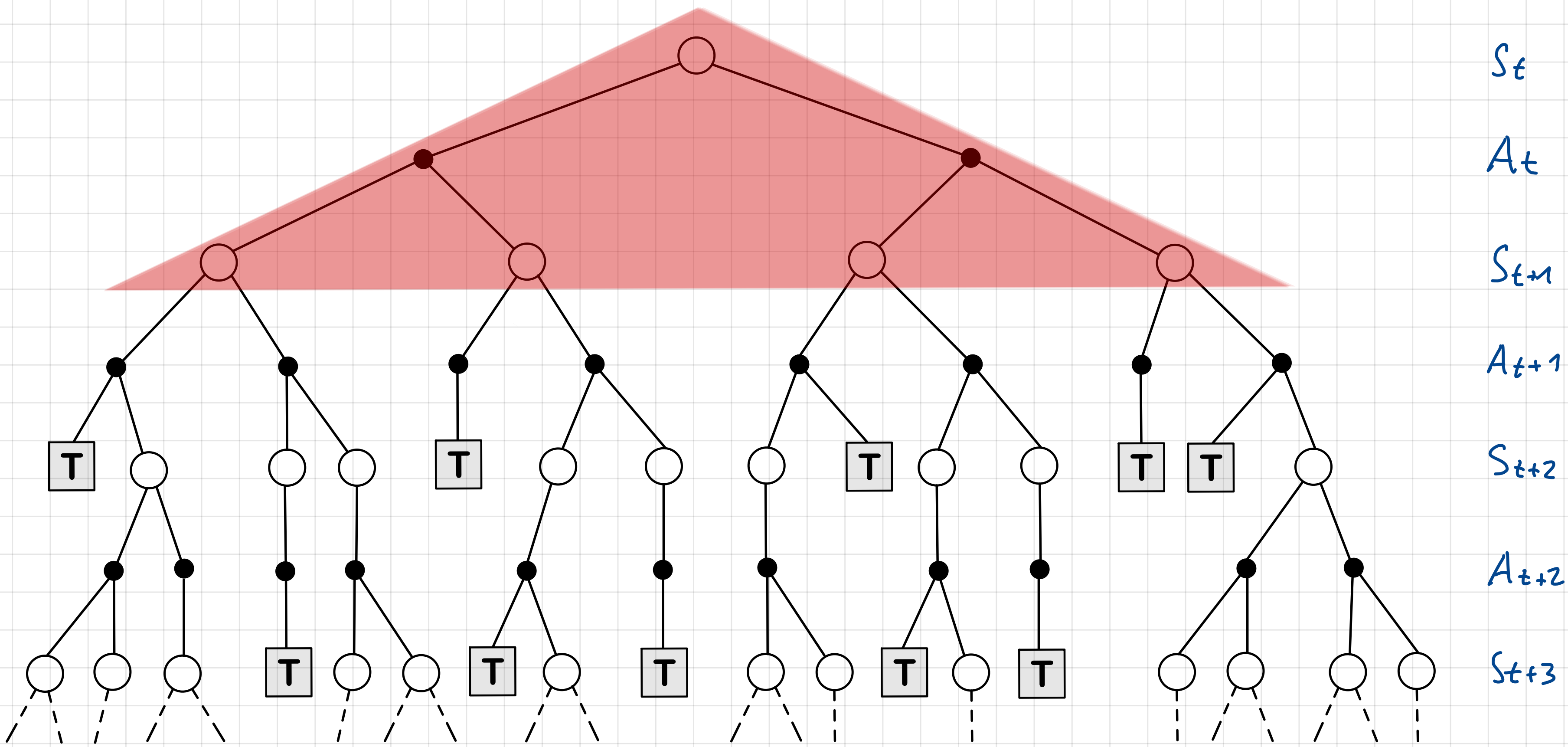


* don't confuse with bootstrap in statistics

Adapted from D. Silver

Dynamic Programming Update (Bellman)

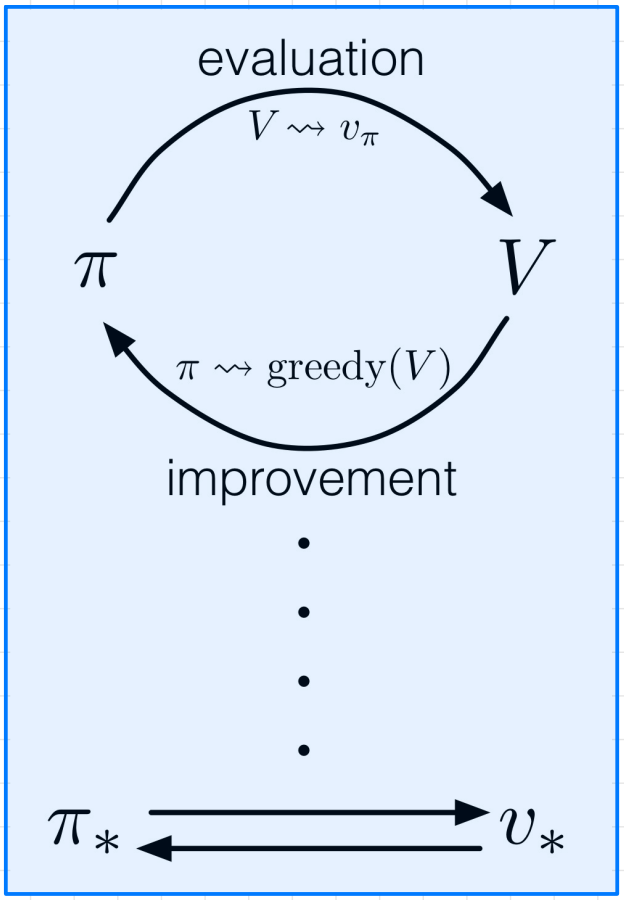
$$V(s) \leftarrow \mathbb{E}[R_{t+1} + \gamma V(S_{t+1})]$$



Adapted from D. Silver

Model-free Control (Policy Improvement)

Generalised Policy Iteration (GPI)



- Alternating between (approximately) computing v_π and (approximately) computing greedy(v_π)
- Worked for model-based RL
- We know already two model-free techniques to compute v_π, MC and TD
- Essentially, we will plug them into GPI
- There are two subtleties, however

Image: Sutton & Barto

Subtlety 1: We need q

- Policy improvement:

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

For planning, it was enough to estimate $V_{\hat{\pi}}(s)$, since

$$q_{\hat{\pi}}(s,a) = \underbrace{r(s,a) + \gamma \sum_{s'} p(s'|s,a)}_{\text{MDP, now unknown}} V_{\hat{\pi}}(s')$$

- Thus, we need to directly estimate $q_{\hat{\pi}}$
- No problem, MC and TD carry over to (s,a) pairs

Basically, the same updates

- Monte Carlo

$$v(s) \leftarrow v(s) + \frac{1}{N(s)}(G_t - v(s))$$

$$q(s,a) \leftarrow q(s,a) + \frac{1}{N(s,a)}(G_t - q(s,a))$$

- Temporal Differences (TD)

$$v(s) \leftarrow v(s) + \alpha(R_{t+1} + \gamma v(s') - v(s))$$

$$q(s,a) \leftarrow q(s,a) + \alpha(R_{t+1} + \gamma q(s',a') - q(s,a))$$

- But, more expensive

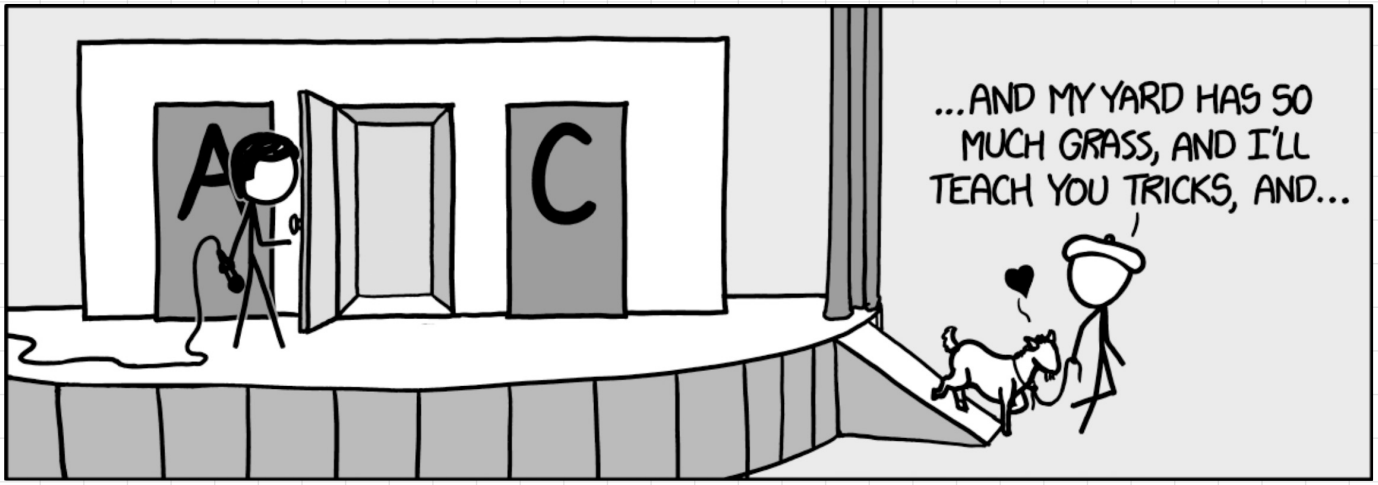
v_π ... $|S|$ values

q_π ... $|S| \times |A|$ values

$$\begin{pmatrix} v_\pi(s_1) \\ v_\pi(s_2) \\ \vdots \\ v_\pi(s_N) \end{pmatrix} \qquad \begin{pmatrix} q_\pi(s_1,a_1) & \dots & q_\pi(s_1,a_M) \\ \vdots & \ddots & \vdots \\ q_\pi(s_N,a_1) & \dots & q_\pi(s_N,a_M) \end{pmatrix}$$

Subtlety 2: Exploration

Image: xkcd.com



- You have the choice between doors A, B, and C.
  - you pick door A and get 5 reward
  - you pick door B and get 0 reward
  - you pick door C and get 1 reward
  - door A ⟶ 3 reward
  - door A ⟶ 4 reward
  - door A ⟶ 7 reward

- Is door A the best? ⟶ **Exploration vs. Exploitation**

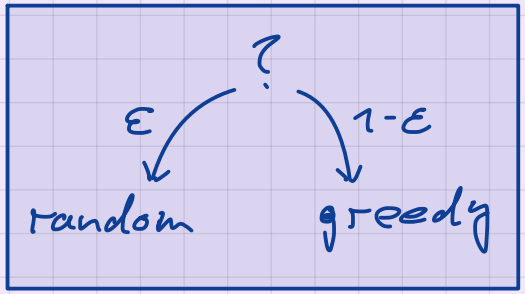
Fundamental trade-off in RL

Epsilon Greedy Policies

Main way to incorporate exploration:

ε-soft Policy: $\tilde{\pi}(a|s) \geq \varepsilon \quad \forall s, a$

ε-greedy Policy: With probability ε, take random action (uniformly), otherwise take greedy action



$$\tilde{\pi}(a|s) = \begin{cases} 1-\varepsilon + \frac{\varepsilon}{|\mathcal{A}(s)|} & \text{if } a = \arg\max_{a'} q(s,a') \\ \frac{\varepsilon}{|\mathcal{A}(s)|} & \text{o.w.} \end{cases}$$

- ε-greedy: the "hardest" $\frac{\varepsilon}{|\mathcal{A}(s)|}$-soft policy
- for $\varepsilon = 0$: greedy policy

Monte Carlo Control

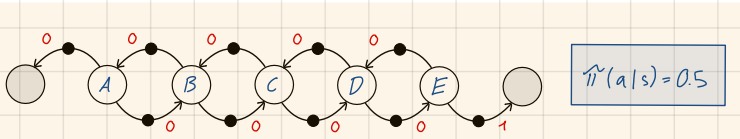
- initialize $q(s, a)$ arbitrarily $\forall s \in \mathcal{S}, a \in \mathcal{A}$
- $N(s, a) \leftarrow 0 \quad \forall s \in \mathcal{S}, a \in \mathcal{A}$

- repeat
  - generate episode using ε-greedy (q)

    $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$
  - $G \leftarrow 0$
  - for $t = T-1, \ldots, 0$:
    - $G \leftarrow R_{t+1} + \gamma G$
    - $N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$
    - $q(S_t, A_t) \leftarrow q(S_t, A_t) + \frac{1}{N(S_t, A_t)} \left(G - q(S_t, A_t) \right)$

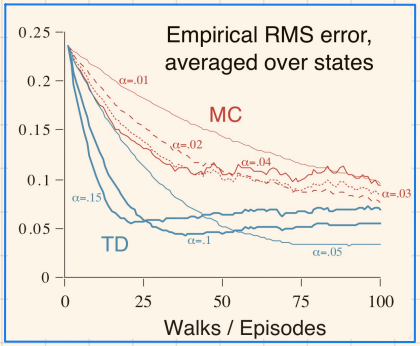
Converges to optimal policy, if ε is annealed to 0 slowly enough, e.g. $O(1/\text{iteration count})$

Sarsa: TD Control



TD update for evaluation:

$$V(S) \leftarrow V(S) + \alpha\left(R_{t+1} + \gamma V(S') - V(S)\right)$$

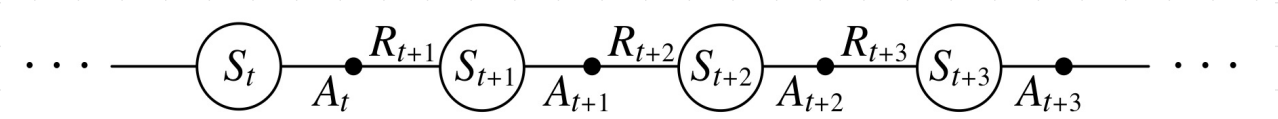


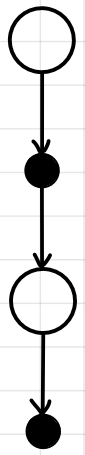
For control, we need TD update for q:

$$q(S,A) \leftarrow q(S,A) + \alpha\left(R_{t+1} + \gamma q(S',A') - q(S,A)\right)$$

This rule is applied to episodes generated by current $\hat{\pi}$ (on-policy), using each consecutive quintuple $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$

SARSA





Images: Sutton & Barto

Sarsa: TD Control

Let $\varepsilon > 0$, $\alpha \in (0, 1]$ // hyperparameters for ε-greedy, stepsize

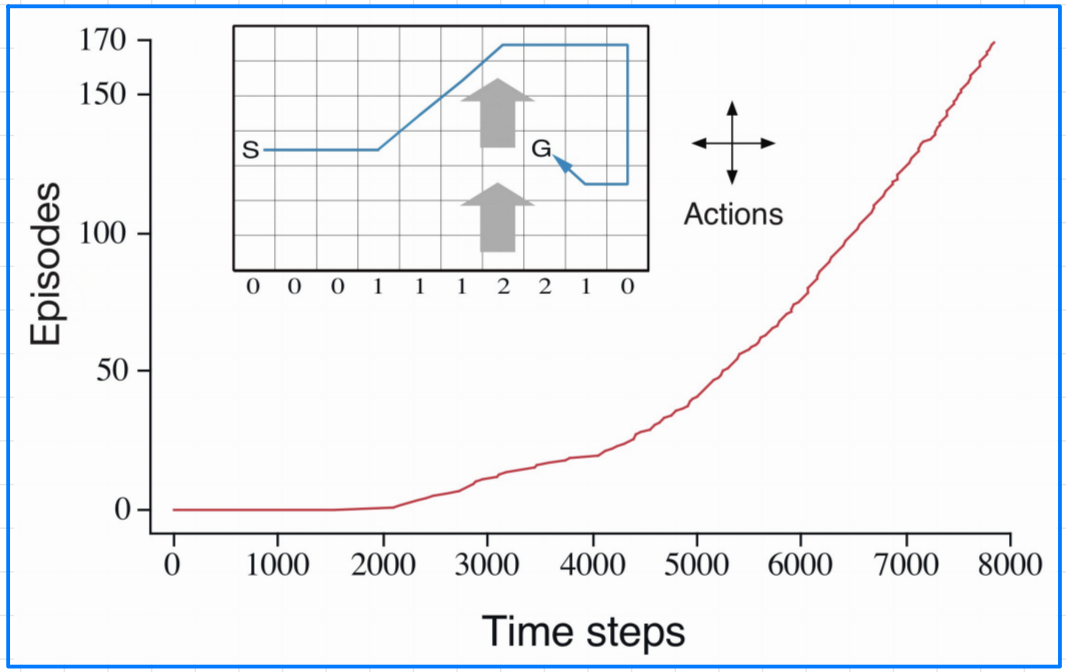
Initialize $q(s, a)$ arbitrarily, except $q(s, a) = 0$ for terminal s

- repeat: // for continuing tasks => one repetition
  - init S
  - generate A from ε-greedy$(q(S, \cdot))$
  - repeat until S is terminal: // forever for continuing task
    - take action A
    - observe R, S'
    - generate A' from ε-greedy$(q(S', \cdot))$
    - $q(S, A) \leftarrow q(S, A) + \alpha \left(R + \gamma q(S', A') - q(S, A) \right)$
    - $S, A \leftarrow S', A'$

Converges to optimal policy, if ε is annealed to 0 slowly enough, e.g. $O(1/\text{iteration count})$

Windy Grid World with Sarsa

- dedicated start state S and goal G
- wind in the center, displacing agent by 1 or 2 cells
- Sarsa with $\alpha = 0.5$ and $\varepsilon = 0.1$
- over time, agent learns to reach goal quicker



read: more completed episodes in fewer time steps

- Monte Carlo would learn very slowly, since it waits until end of episode before update — this takes very long in this grid world!

Image: Sutton & Barto

On Policy vs. Off Policy

Exploration

Exploitation

On-policy methods

"Learning on the job"

need to find a compromise

e.g exploring starts, ϵ-greedy

Off-policy methods

"Looking over the shoulder"

separate the problem

Behavior policy b

generates behavior, data

Target policy π

learns from b's data

Q-learning: Off-Policy TD Control

[Watkins 1989]

Sarsa: $q(s,a) \leftarrow q(s,a) + \alpha\big(R_{t+1} + \gamma\, q(S_{t+1}, A_{t+1}) - q(s,a)\big)$

$\mathbb{E}[\delta_t] = 0 \triangleq$ Bellman expectation equation

Q-learning $q(s,a) \leftarrow q(s,a) + \alpha\big(R_{t+1} + \gamma \max_{a'} q(S_{t+1}, a') - q(s,a)\big)$

$\mathbb{E}[\delta_t] = 0 \triangleq$ Bellman optimality equation

Let $\varepsilon > 0$, $\alpha \in (0,1]$ // hyperparameters for ε-greedy, stepsize
Initialize $q(s,a)$ arbitrarily, except $q(s,a) = 0$ for terminal s
- repeat:
  - init S
  - repeat until S is terminal:
    - take action A from behavioral policy, e.g. ε-greedy$(q(S,\cdot))$
    - observe S', R
    - $q(S,A) \leftarrow q(S,A) + \alpha\big(R + \gamma \max_{a'} q(S',a') - q(S,A)\big)$
    - $S \leftarrow S'$

Off-policy: learn (hard) greedy policy while following ε-greedy policy.
ε can be constant (no GLIE). Convergence to q^* for $\sum_k \alpha_k = \infty$, $\sum_k \alpha_k^2 < \infty$.

The Cliff

- grid world with starting state and goal
- the grey area (the cliff) yields large penalty ($R=-100$) and transition to the start
- when running Sarsa and Q-learning with constant $\varepsilon = 0.1$, one will learn the optimal path and the other the safer path.
- which algorithm will learn which path?

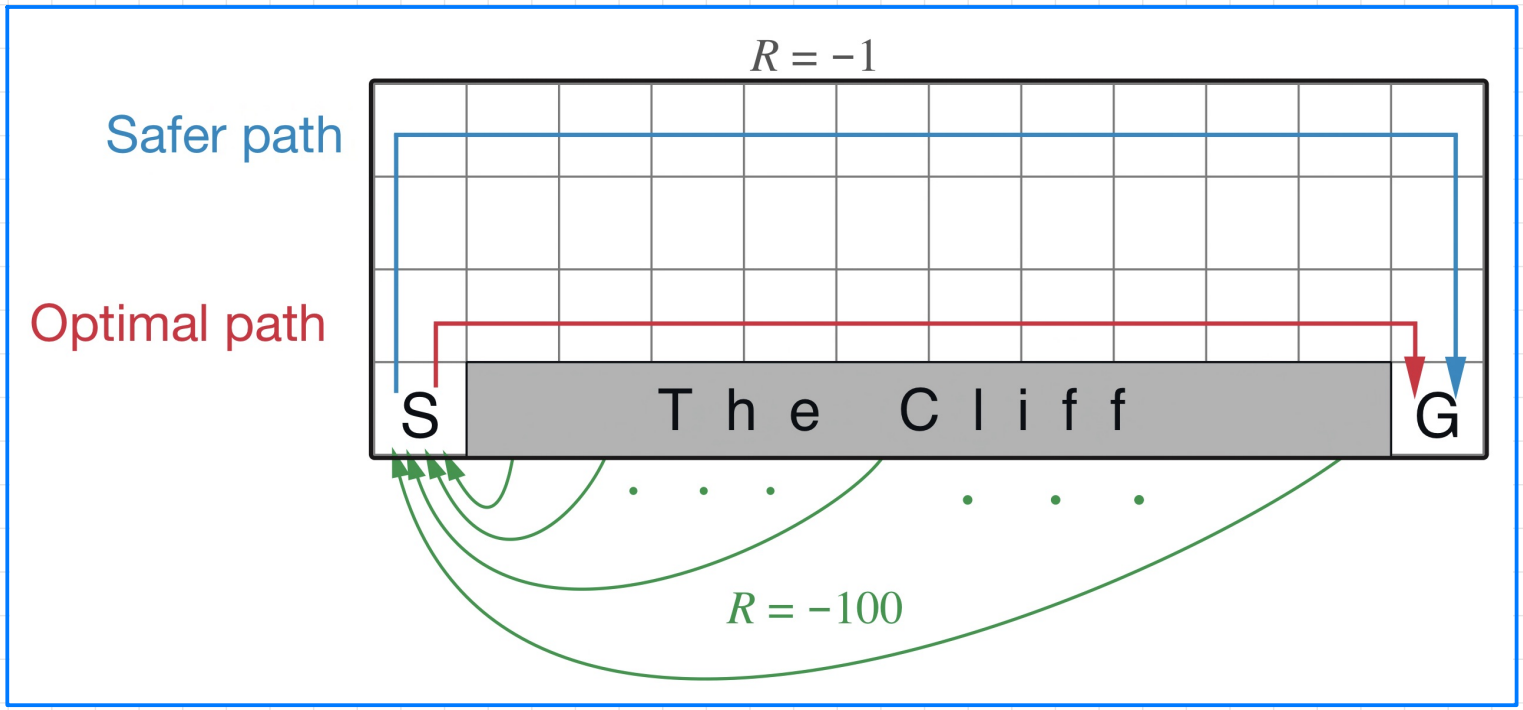


Image: Sutton & Barto

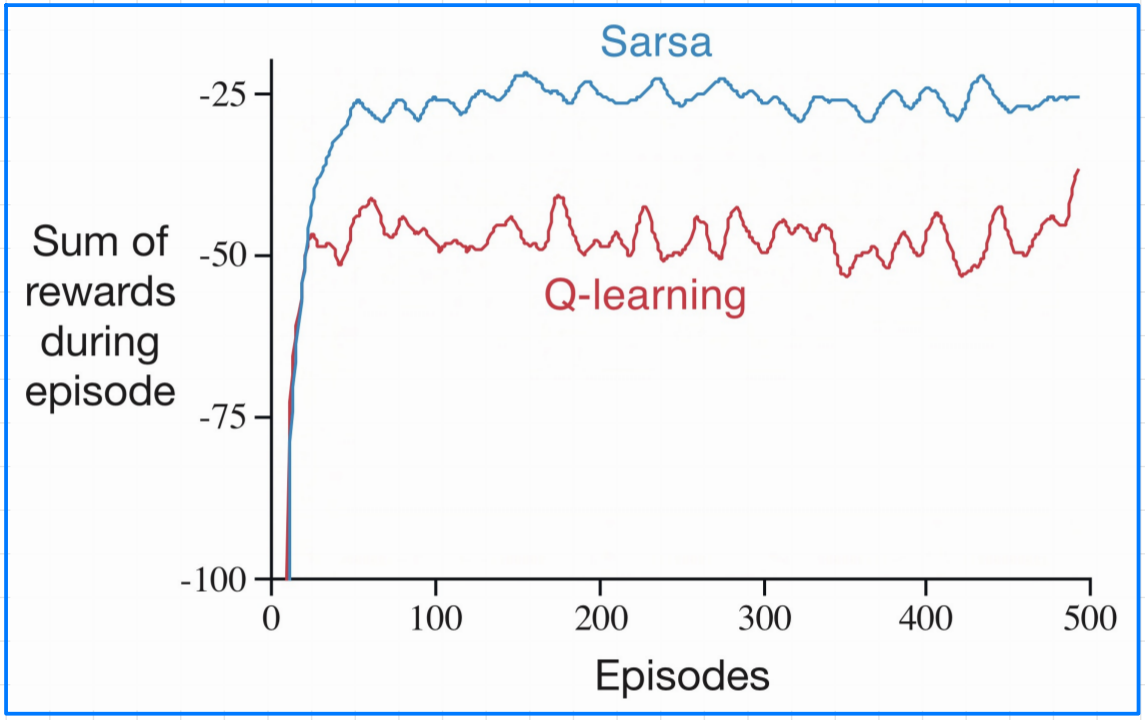
The Cliff cont'd

Q-learning: optimal path

Sarsa: safer path

Sarsa is on-policy: return of behavior policy is maximized

Q-learning is off-policy: return of behavior policy not a goal per se



If $\varepsilon = 1/k$ (GLIE)

- Sarsa converges to π^*
- Q-learning achieves optimal online returns

Image: Sutton & Barto

Summary Reinforcement Learning