

Model-free prediction

Goal: Predict v_{π} when π is known, without knowing the MDP $(\mathcal{S}, \mathcal{R})$.

We observe episodes

$S_1, A_1, R_2, S_2, \dots, S_T$ terminal state
 episode

We want to compute $v_{\pi}(s) = \mathbb{E}_{\pi}(\mathbb{E}_t [f_t(s_t)])$

Monte Carlo policy evaluation

For each episode

→ G_t is calculated for each visited state

$$G_{T-1} = R_T$$

$$\text{for } t = T-1 \text{ to } 2$$

$$G_{t-1} = R_t + \gamma G_t$$

→ For each visited state s , and when increment a counter: $N(s) := N(s) + 1$
 increment total return: $T(s) := T(s) + g$

→ The value function is

$$V(s) = T(s) / N(s)$$

→ Monte Carlo every-visit algorithm

A variant is Monte Carlo first-visit.

Both algorithms converge towards $V_\pi(s)$ when $N(s) \rightarrow \infty$

Incremental mean

$$\mu = \langle x \rangle$$

$$\mu_k = \frac{1}{k} \sum_{i=1}^k x_i$$

$$\mu_k = \frac{1}{k} \left(\sum_{i=c}^{k-1} x_i + x_k \right)$$

$$= \frac{1}{k} ((k-1)\mu_{k-1} + x_k)$$

$$\mu_k = \mu_{k-1} + \frac{1}{k} (x_k - \mu_{k-1})$$
incremental mean

$$\mu_k = \mu_{k-1} + \alpha (x_k - \mu_{k-1})$$
running mean

Temporal difference (TD)

It is an alternative to Monte Carlo (MC) that only uses S_t and S_{t+1} .

$$\text{MC: } V(S_t) := V(S_t) + \alpha (G_t - V(S_t))$$

learning rate > 0 target output
error

running mean version of MC

$$TD: V(S_t) := V(S_t) + \alpha \left[\underbrace{R_{t+1} + \gamma V(S_{t+1})}_{\text{target}} - V(S_t) \right]$$

$\underbrace{(S_t, A_t, R_{t+1}, S_{t+1})}$

"Perfect target" would be:

$$\underbrace{V_\pi(S_t)}_{\text{TC approximates}} = \underbrace{R_{t+1} + \gamma V_\pi(S_{t+1})}_{\text{TD approximates this with 1 sample}}$$

$V_\pi(S_t)$ approximates thes with 1 sample

TD approximates this with $V(S_{t+1}) \approx V_\pi(S_{t+1})$

Comparison between TC and TD

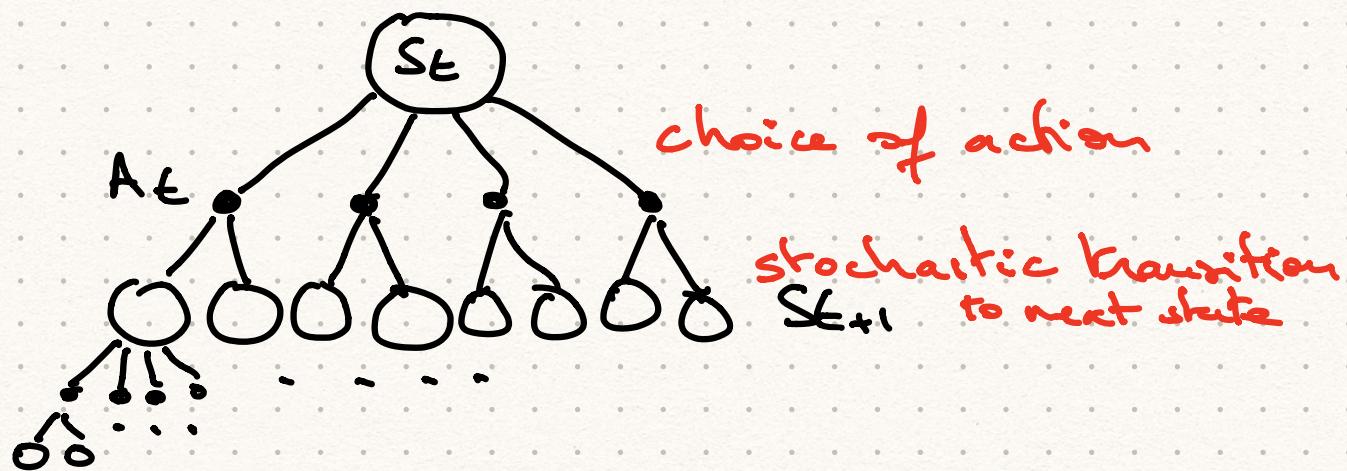
Both converge towards $V_\pi(s)$ (theorems)

- TC is episodic and we need to wait until the end of an episode
- TD only depends on $S_t, A_t, R_{t+1}, S_{t+1}$ (not episodic, and update of $V(S_t)$ can be performed at time t).
- TC uses 1 sample
 → no bias, high variance ($S_t \rightarrow S_+$)
- TD approximates V_π
 → bias, less variance (randomness from 1 transition $S_t \rightarrow S_{t+1}$)

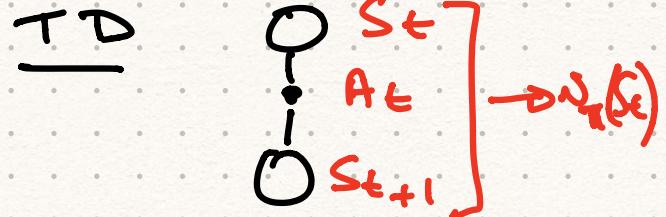
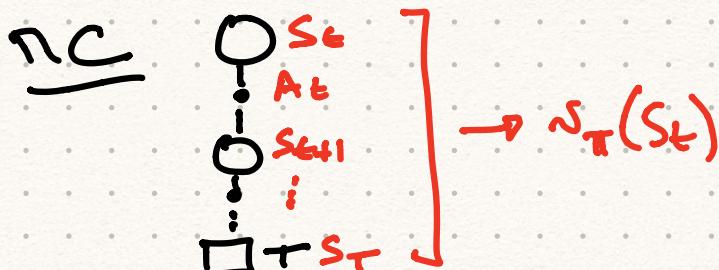
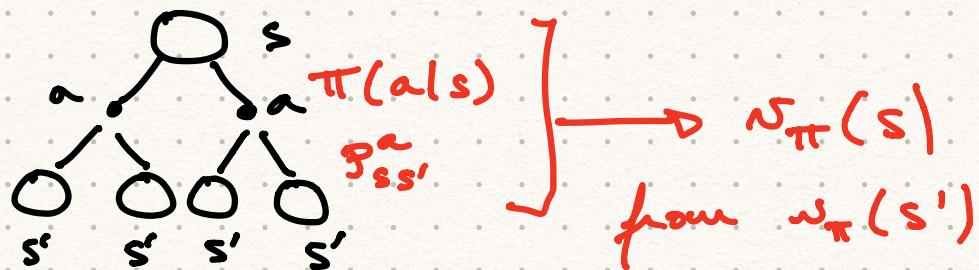
In practice:

- TD usually converge slightly faster but you need to tune α .
- TC is easier (no need to tune α) and convergence is ensured even w/ function approximation.

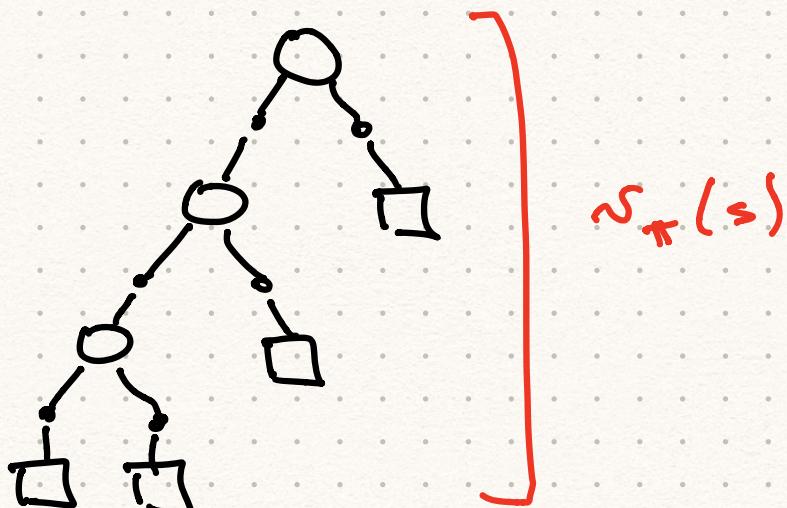
Graphical interpretation



DP (dynamic programming)
we exploit $P_{ss'}, R_s$



Exhaustive search



n-step temporal difference

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

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

$$G_E^{(n)} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{n-1} V(S_{t+n})$$

λ-trace TD

$$G_E^{\lambda} = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_E^{(n)} \quad (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} = 1$$

This is an average of the $G_E^{(n)}$ with weights proportional to λ^n .

λ plays a role similar to discount
 $\lambda \in [0, 1]$

$\lambda = 0 \rightarrow$ 1-step TD

$\lambda = 1 \rightarrow$ similar to MC

$\lambda \in]0, 1[$

No model-free control

Goal: Find $\pi^*(s)$ from observations.
(without knowing the TD π).

On-policy: learn policy π while implementing the same policy π (that changes through time)

Off-policy: learn π^* while implementing π

Bellman equations:

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} \left[\underbrace{R_s}_{?} + \underbrace{\gamma \sum_{s'} P_{ss'}^a \pi(s')}_{?} \right]$$

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} q^*(s, a)$$

action-value function

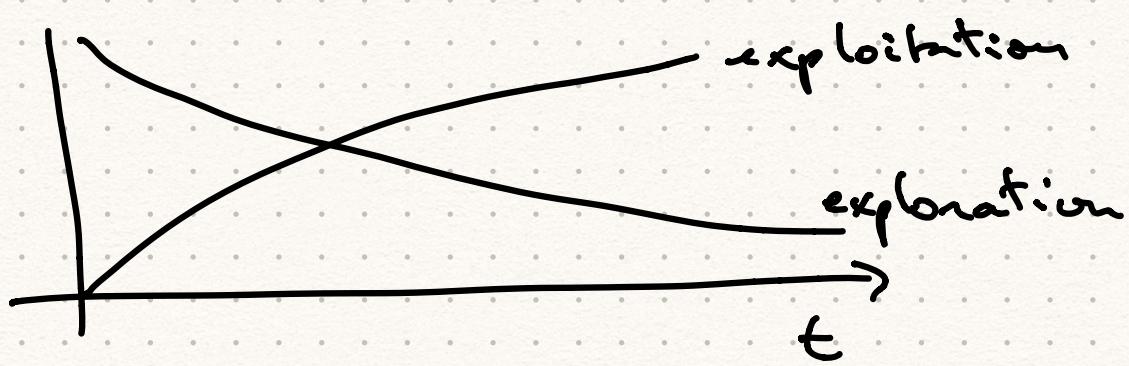
$$q^*(s, a) = R_{t+1} + \gamma \sum_{s'} \underbrace{P_{ss'}^a}_{?} \max_{a'} q^*(s', a')$$

unknown but estimated from sample of s'

On-policy algorithms

Balance between two conflicting objectives:

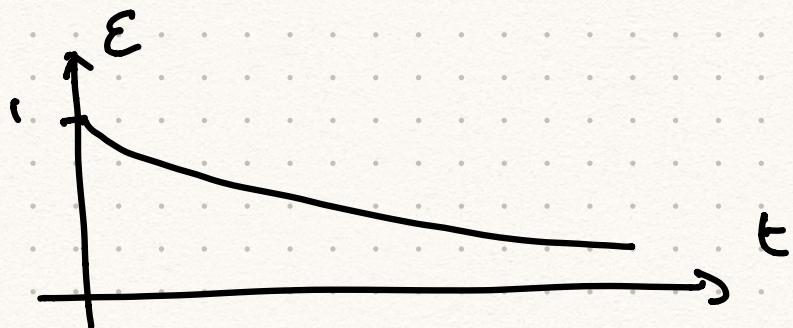
- exploration: sampling sufficiently all states to estimate their values
- exploitation: choosing actions that seem optimal



ϵ -greedy policy

Policy :

- with probability ϵ : random action
- $\frac{1-\epsilon}{\epsilon}$: greedy action
| argmax_a Q(s,a)



Monte Carlo control with ϵ -greedy policy

K : number of the episode

$$\epsilon = \frac{1}{K} \rightarrow \epsilon\text{-greedy policy (with Q)}$$

At the end of the episode

→ calculate all G_E

→ for each visited state

increment counter $N(S_E, A_E) += 1$

increment Q function

$$Q(S_E, A_E) := Q(S_E, A_E) + \frac{1}{N(S_E, A_E)} (G_E - Q(S_E, A_E))$$

This π_C ϵ -greedy algorithm converges towards π^* , $q_*(s, a)$ (theorem).

SARSA

TD algorithm for $Q(s, a)$

$$(S, A, R, S', A') \leftarrow (S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$$

$$Q(S, A) := Q(S, A) + \alpha \underbrace{(R + \gamma Q(S', A') - Q(S, A))}_{\text{target error}}$$

together with an ϵ -greedy policy that exploits $Q(S, A)$ and $\epsilon \xrightarrow[t \rightarrow \infty]{} 0$

Off-policy : Q-learning

Q-learning is similar to SARSA but does not use $A' = \pi(S'; \epsilon, Q)$

$$Q(S, A) := Q(S, A) + \alpha \underbrace{\left[R + \gamma \max_{a'} Q(S', a') - Q(S, A) \right]}_{\text{target}}$$

Q-learning is off-policy : it learns the optimal policy even if the policy implement is not optimal.

In practice, we can use an ϵ -greedy policy . but we don't require $\epsilon \xrightarrow[t \rightarrow \infty]{} 0$

Theorem: SARSA and Q-learning both converge towards q^* and π^* .