Model-free prediction

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

We observe episodes

$S_1, A_1, R_2, S_2, \ldots S_T$ ← terminal state

episode

We want to compute $\boxed{v_\pi(s) = \mathbb{E}_\pi(G_t \mid S_t = s)}$

Monte Carlo policy evaluation

For each episode

→ G_t is calculated for each visited state

$G_{T-1} = R_T$

for $t = T-1$ to 2

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

→ For each visited state s, and during

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 $N(s) \to \infty$

Incremental mean

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

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

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

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

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

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

Temporal difference (TD)

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

MC: $V(S_t) := V(S_t) + \alpha\left(G_t - V(S_t) \right)$ running mean version of MC

learning rate > 0 ; target ; output ; error

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

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

"Perfect target" would be:

$$v_\pi(S_t) = R_{t+1} + \gamma v_\pi(S_{t+1})$$

MC approximates this with 1 sample

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

Comparaison between MC and TD

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

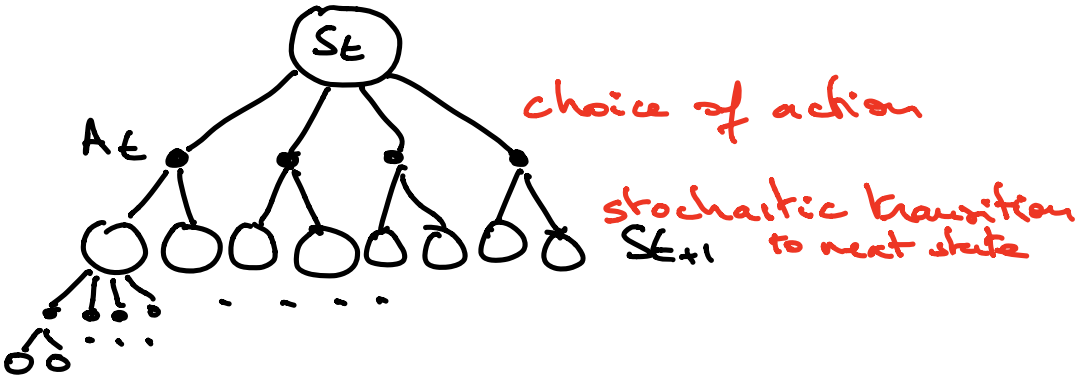
- MC is episodic and we need to wait until the end of an episode
- TD only depends on $S_t, A_t, R_{t+1}, S_{t+1}$ (not episodic, and update of $V(S_t)$ can be performed at time t).

- MC use 1 sample
  → no bias, high variance ($S_t \to S_T$)
- TD approximate v_π
  → bias, less variance (randomness from 1 transition $S_t \to S_{t+1}$)

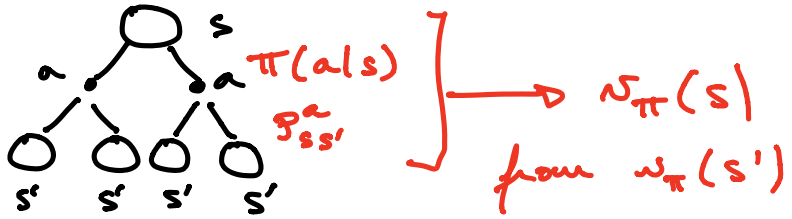
In practice:

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

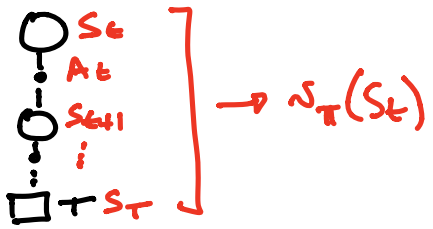
Graphical interpretation



DP (dynamic programming)
we exploit $\mathcal{P}^a_{ss'}$, $\mathcal{R}^a_s$

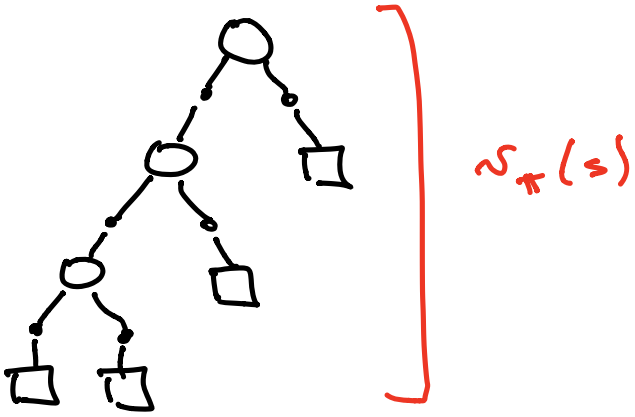


MC



TD

Exhaustive search



n-step temporal difference

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

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

$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^n V(S_{t+n})$

λ-trace TD

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

this an average of the $G_t^{(n)}$ with weights proportional to λ^n.

λ plays a role similar to discount

$\lambda \in [0, 1]$

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

$\lambda = 1 \longrightarrow$ similar to MC

$\lambda \in]0, 1[$

Model-free control

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

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

off-policy: learn π^* while implementing π

Bellman equations:

$$\pi^*(s) = \arg\max_a \left[\underbrace{R_s}_{?} + \underbrace{P^a_{ss'}}_{?} V_*(s') \right]$$

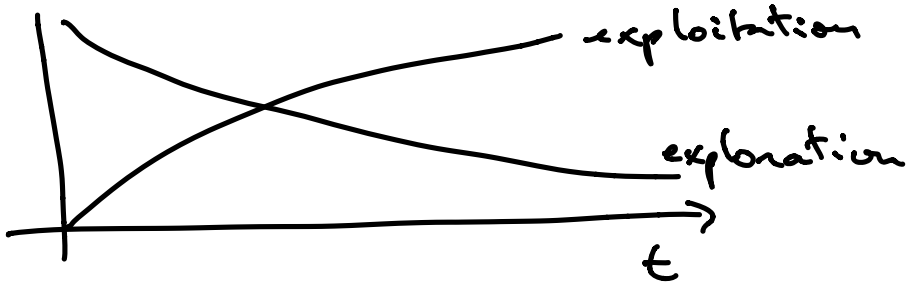
$$\pi^*(s) = \arg\max_a \underbrace{q_*(s,a)}_{\text{action-value function}}$$

$$q_*(s,a) = R_{t+1} + \gamma \sum_{s'} \underbrace{P^a_{ss'}}_{\text{unknown but estimated from samples of } s'} \max_{a'} q_*(s',a')$$

On-policy algorithms

Balance between two conflicting objectives:

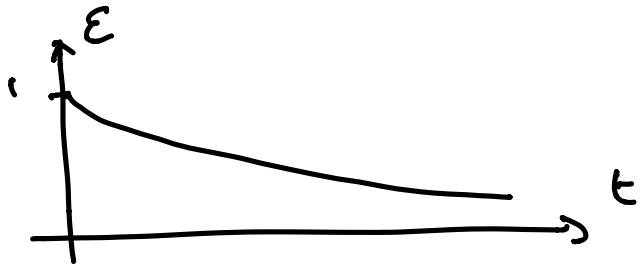
- exploration: sampling sufficiently all states to estimate their value
- exploitation: choosing actions that seem optimal



ε - greedy policy

Policy:

- with probability ε : random action
- ——— $(1-\varepsilon)$: greedy action $\left| \underset{a}{\text{argmax}}\, Q(s,a) \right.$



Monte Carlo control with ε-greedy policy

k : number of the episode

$\varepsilon = \frac{1}{k}$ ⟶ ε - greedy policy (with Q)

At the end of the episode

⟶ calculate all G_t

⟶ for each visited state

increment counter $N(S_t, A_t) \mathrel{+}= 1$

increment Q function

$$Q(S_t, A_t) := Q(S_t, A_t) + \frac{1}{N(S_t, A_t)} \left(G_t - Q(S_t, A_t) \right)$$

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

SARSA

TD algorithm for $Q(s,a)$

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

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

$\Big(\text{target} - Q(S,A)\Big)$: error

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

off-policy : Q-learning

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

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

Q-learning is off-policy : its learns the optimal policy even if the policy implemented is not optimal.

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

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