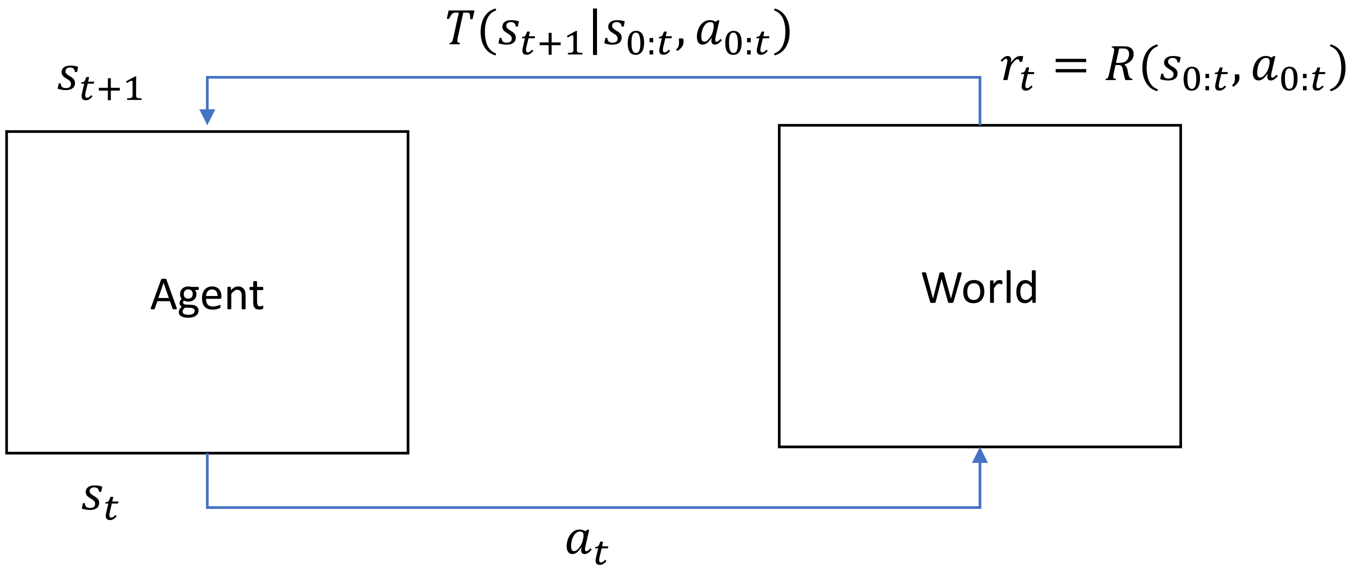
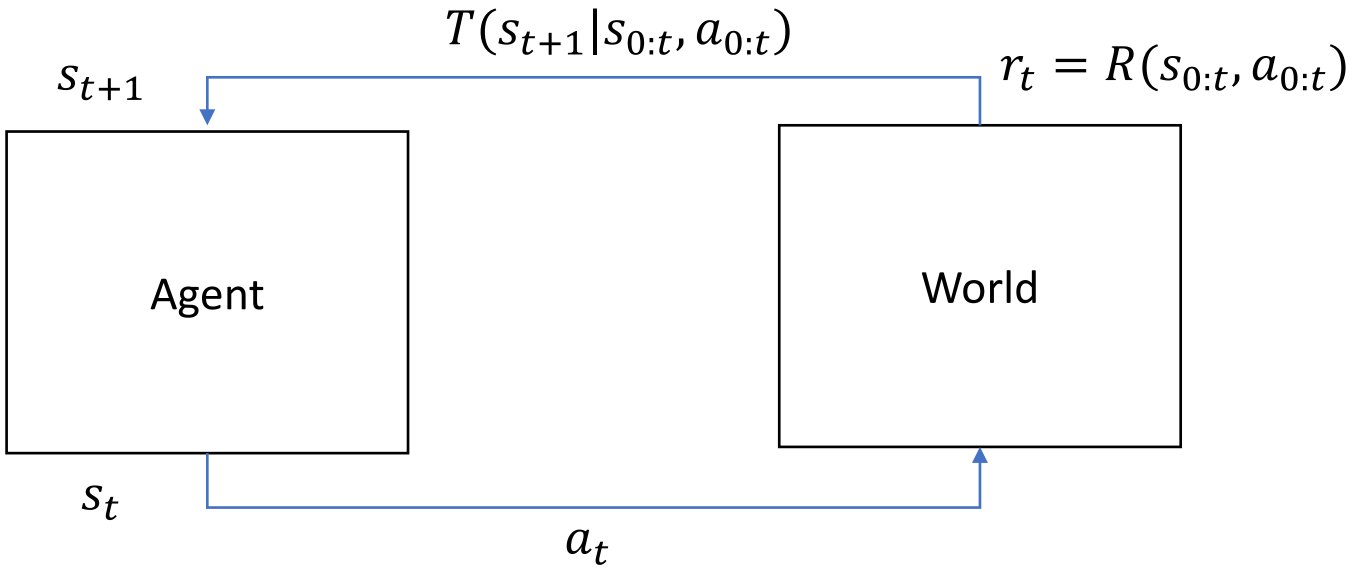
RL Overview

Nishant Kumar

Setting



Setting



- What should our goal be?
- How do we measure it?
- How do we achieve it?

Definitions

- A single-agent RL system is defined by:
  - S is a set of states.
  - A is a set of actions.
  - R is a reward function that determines the immediate reward r_t received at time t.
  - T is a transition function that represents the stochasisity of the system.
  - γ is a discount factor between 0 and 1 (inclusive).
- An *agent* behaves according to a *policy* π, where π can be either:
  - a deterministic function dictating which action to take from which state, i.e. $\pi : S \rightarrow A$.
  - a stochastic function dictating with what probability to take an action from a state, i.e. $\pi : S \times A \rightarrow Pr[0, 1]$.
- A *trajectory* or *episode* τ is a sequence of (s_t, a_t, r_t, s_{t+1}) an agent experiences in one "run":

  $$\tau = s_0 \rightarrow a_0 \rightarrow r_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_{T-1}$$

  Where T denotes the *horizon* or length of the "run". Note that T could be ∞.

  Where $p(\tau)$ represents the probability of trajectory τ. Note that if rewards are deterministic, then the r_i's are not needed in $p(\tau)$. Note that $p(\tau)$ depends on π.
- Canonically, the goal of an agent starting in state s is to maximize its *expected sum of discounted rewards*:

  $$V^{\pi}(s_t) = \mathbb{E}_{\tau \sim p(\tau)}[\sum_{t'=t}^{T} \gamma^{t'-t} r_{t'} | s_t = s]$$

  $$\pi^* = \text{argmax}_{\pi} V^{\pi}(s_0)$$

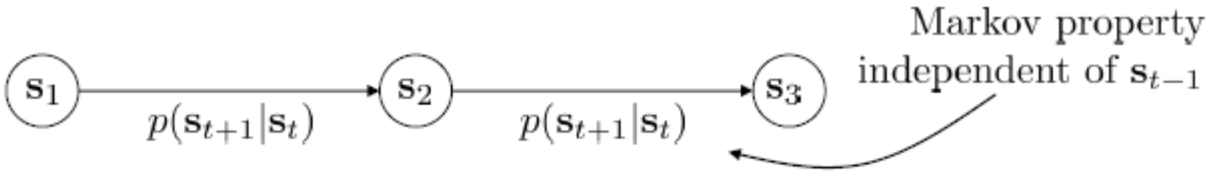
Definitions

Markov chain

$\mathcal{M} = \{\mathcal{S}, \mathcal{T}\}$

$\mathcal{S}$ – state space states $s \in \mathcal{S}$ (discrete or continuous)

$\mathcal{T}$ – transition operator $p(s_{t+1}|s_t)$



Definitions

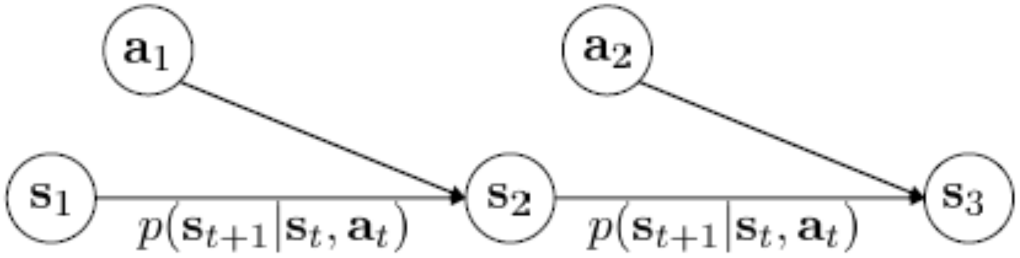
Markov decision process $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, r\}$

$\mathcal{S}$ – state space states $s \in \mathcal{S}$ (discrete or continuous)

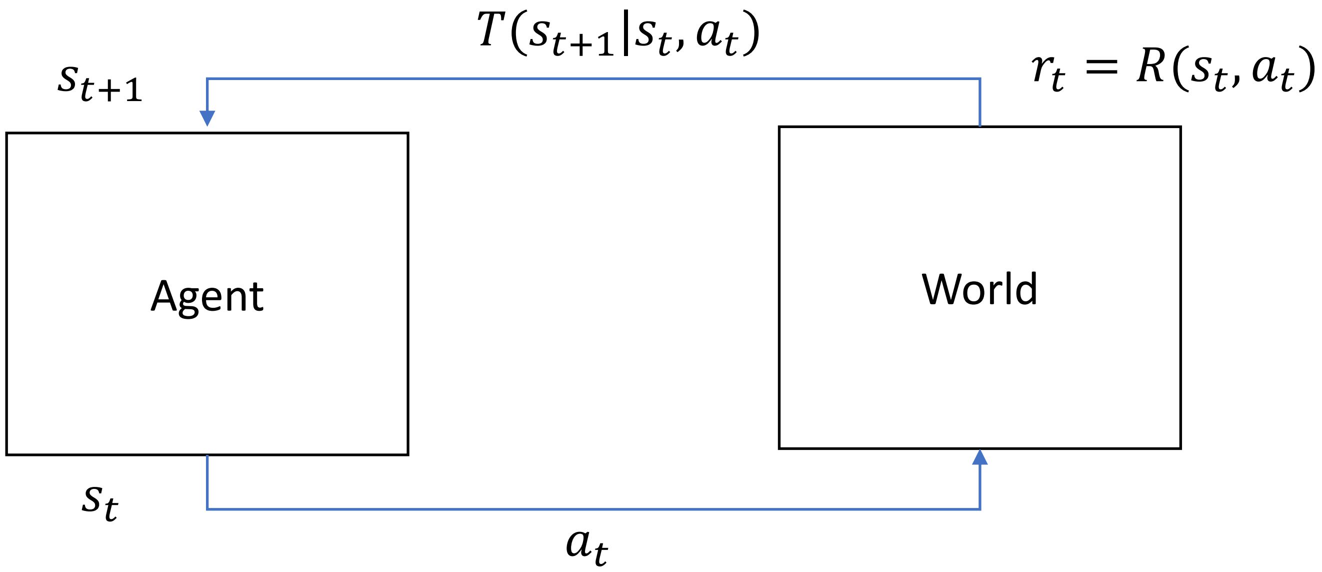
$\mathcal{A}$ – action space actions $a \in \mathcal{A}$ (discrete or continuous)

$\mathcal{T}$ – transition operator (now a tensor!)

$$\mathcal{T}_{i,j,k} = p(s_{t+1} = i | s_t = j, a_t = k)$$



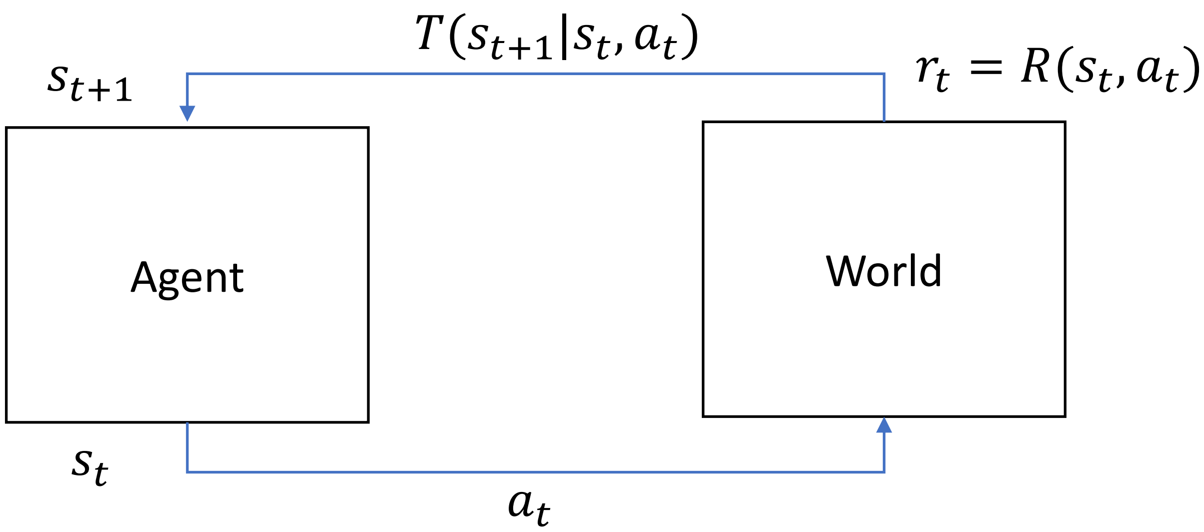
Setting, as an MDP

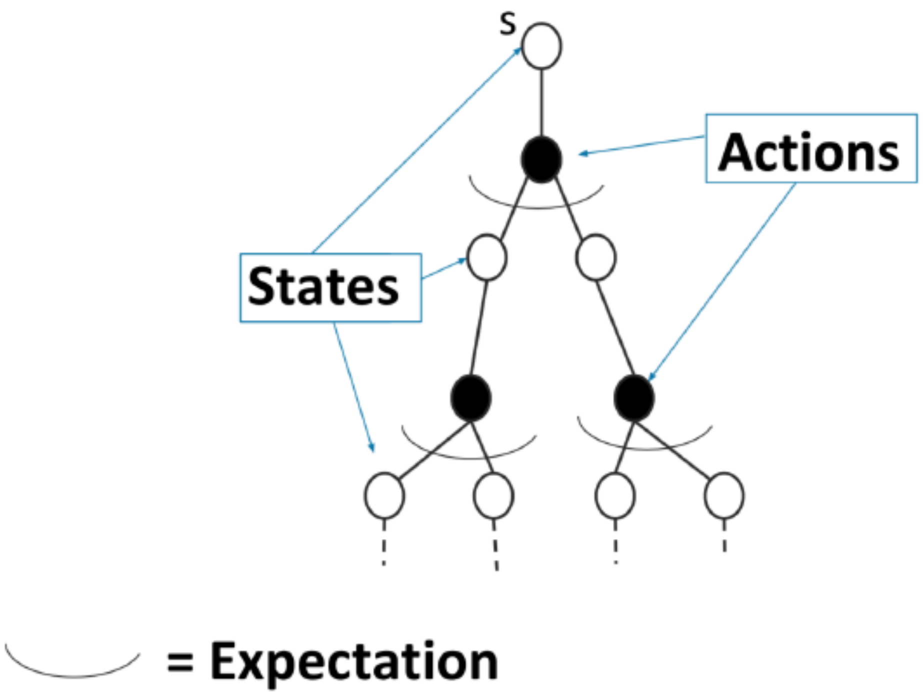


Definitions

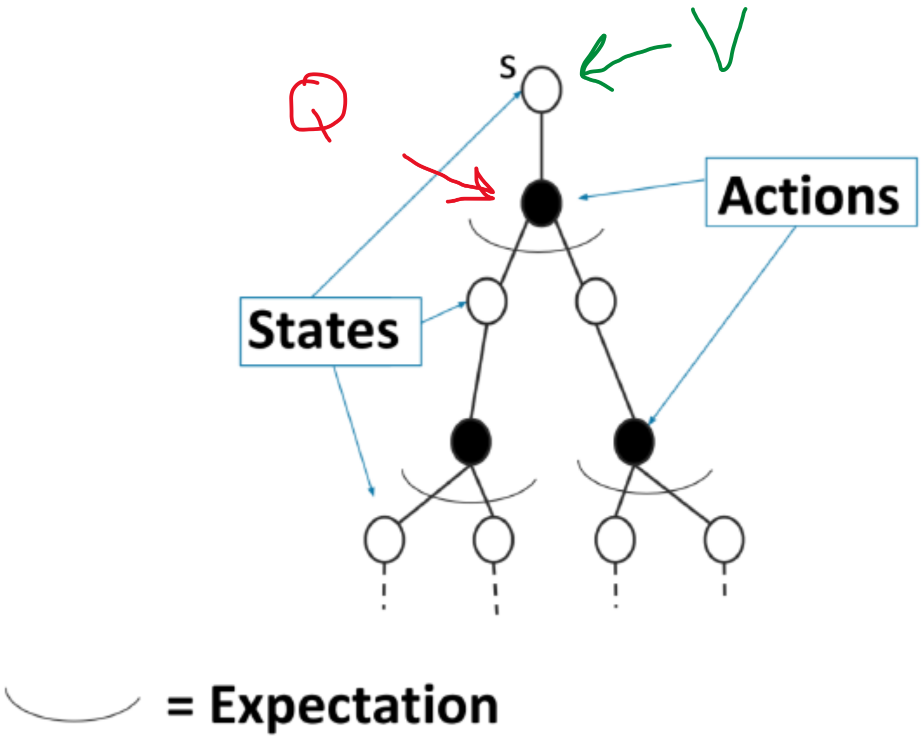
- An MDP is defined by $\langle S, A, R, T, \gamma \rangle$, where:
  - S is a set of states.
  - A is a set of actions.
  - R is a reward function and $R(s, a)$ (or alternatively $R(s, a, s'))$ is the immediate reward received from being in state s and taking action a.
  - T is a transition function and $T(s'|s, a)$ is the probability of transitioning to state s' after taking action a from state s.
  - γ is a discount factor between 0 and 1 (inclusive).

Setting, as a tree





Value Functions: V and Q



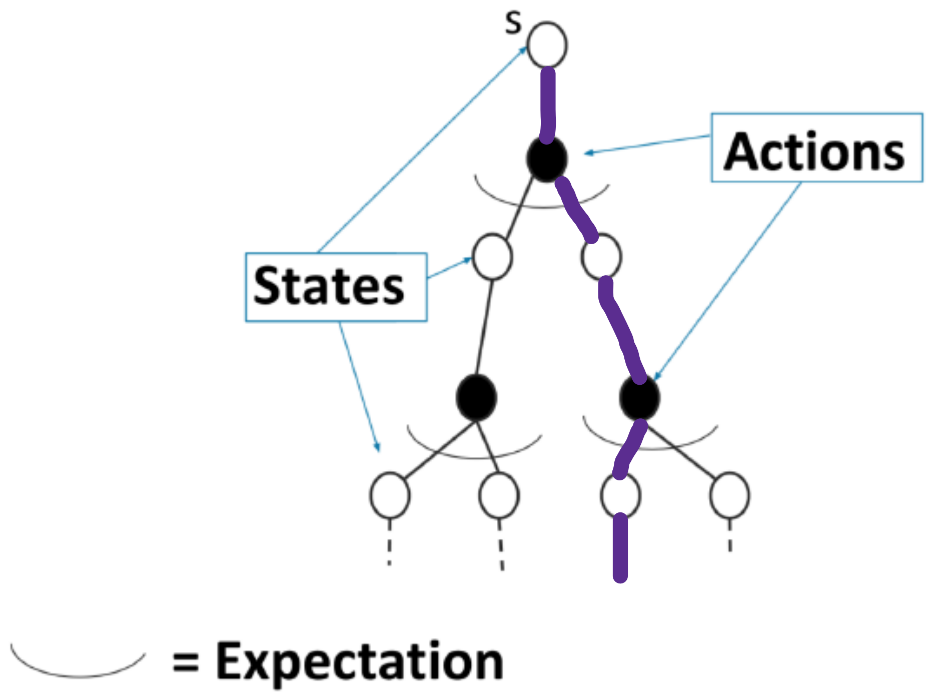
$$V^\pi(s_t) = \mathbb{E}_{\tau \sim p(\tau)}[\sum_{t'=t}^{T} \gamma^{t'-t} r_{t'} | s_t = s]$$
$$= Q^\pi(s_t, \pi(s_t))$$

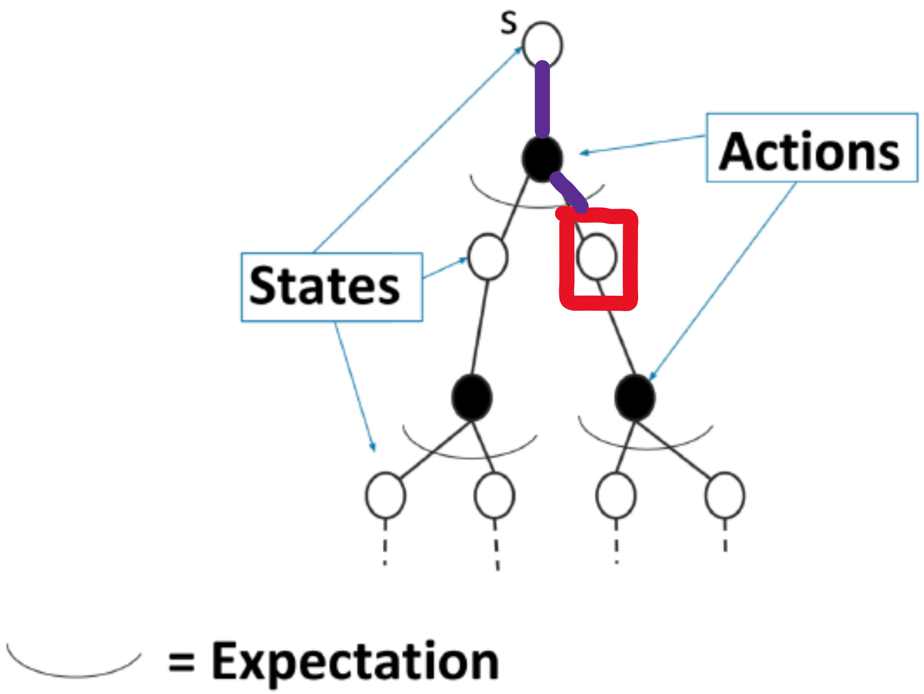
$$Q^\pi(s_t, a_t) = \mathbb{E}_{\tau \sim p(\tau)}[\sum_{t'=t}^{T} \gamma^{t'-t} r_{t'} | s_t = s, a_t = a]$$
$$= r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)}[V^\pi(s_{t+1})]$$

RL Algorithms (tabular)

	Known T and R	Unknown T and R
Evaluate policy	• Value Iteration / Dynamic Programming	• Monte Carlo Policy Evaluation • Temporal Difference Learning
Find optimal policy	• Value Iteration	• Monte Carlo Online Control • SARSA • Q-Learning

Key Idea: Bootstrapping





Value Iteration / DP (Policy Eval)

- Initialize $V_0^\pi(s) = 0$ for all s
- For $k = 1$ until convergence
  - For all s in S

$$V_k^\pi(s) = r(s, \pi(s)) + \gamma \sum_{s' \in S} p(s'|s, \pi(s)) V_{k-1}^\pi(s')$$

- $V_k^\pi(s)$ is exact value of k-horizon value of state s under policy π
- $V_k^\pi(s)$ is an estimate of infinite horizon value of state s under policy π

$$V^\pi(s) = \mathbb{E}_\pi[G_t|s_t = s] \approx \mathbb{E}_\pi[r_t + \gamma V_{k-1}|s_t = s]$$

Value Iteration (Find optimal)

- Set $k = 1$
- Initialize $V_0(s) = 0$ for all states s
- Loop until [finite horizon, convergence]:
  - For each state s

$$V_{k+1}(s) = \max_a R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V_k(s')$$

  - Equivalently, in Bellman backup notation

$$V_{k+1} = BV_k$$

- To extract optimal policy if can act for $k+1$ more steps,

$$\pi(s) = \arg\max_a R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V_{k+1}(s')$$

Monte Carlo Policy Eval

Initialize $N(s) = 0,\ G(s) = 0\ \forall s \in S$
Loop

- Sample episode $i = s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \ldots, s_{i,T_i}$
- Define $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \cdots \gamma^{T_i - 1} r_{i,T_i}$ as return from time step t onwards in ith episode
- For each state s visited in episode i
  - For **first** time t that state s is visited in episode i
    - Increment counter of total first visits: $N(s) = N(s) + 1$
    - Increment total return $G(s) = G(s) + G_{i,t}$
    - Update estimate $V^\pi(s) = G(s)/N(s)$

Monte Carlo Policy Eval (alt.)

Initialize $N(s) = 0,\ G(s) = 0\ \forall s \in S$

Loop

- Sample episode $i = s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \ldots, s_{i,T_i}$
- Define $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \cdots \gamma^{T_i - 1} r_{i,T_i}$ as return from time step t onwards in ith episode
- For state s visited at time step t in episode i
  - Increment counter of total first visits: $N(s) = N(s) + 1$
  - Update estimate

$$V^{\pi}(s) = V^{\pi}(s) + \alpha(G_{i,t} - V^{\pi}(s))$$

- $\alpha = \frac{1}{N(s)}$: identical to every visit MC
- $\alpha > \frac{1}{N(s)}$: forget older data, helpful for non-stationary domains

Temporal Difference Learning

Input: α

Initialize $V^{\pi}(s) = 0, \forall s \in S$

Loop

- Sample **tuple** (s_t, a_t, r_t, s_{t+1})
- $V^{\pi}(s_t) = V^{\pi}(s_t) + \alpha(\underbrace{[r_t + \gamma V^{\pi}(s_{t+1})]}_{\text{TD target}} - V^{\pi}(s_t))$

Monte Carlo Online Control

Initialize $Q(s, a) = 0, N(s, a) = 0 \; \forall(s, a)$, Set $\epsilon = 1$, $k = 1$
$\pi_k = \epsilon$-greedy(Q) // Create initial ϵ-greedy policy
loop
    Sample k-th episode $(s_{k,1}, a_{k,1}, r_{k,1}, s_{k,2}, \ldots, s_{k,T})$ given π_k
    $G_{k,t} = r_{k,t} + \gamma r_{k,t+1} + \gamma^2 r_{k,t+2} + \cdots \gamma^{T_i - 1} r_{k,T_i}$
    for $t = 1, \ldots, T$ **do**
        if First visit to (s, a) in episode k **then**
            $N(s, a) = N(s, a) + 1$
            $Q(s_t, a_t) = Q(s_t, a_t) + \frac{1}{N(s,a)}(G_{k,t} - Q(s_t, a_t))$
        end if
    end for
    $k = k + 1$, $\epsilon = 1/k$
    $\pi_k = \epsilon$-greedy(Q) // Policy improvement
end loop

SARSA

Set initial ϵ-greedy policy π, $t = 0$, initial state $s_t = s_0$
Take $a_t \sim \pi(s_t)$ // Sample action from policy
Observe (r_t, s_{t+1})
loop
    Take action $a_{t+1} \sim \pi(s_{t+1})$
    Observe (r_{t+1}, s_{t+2})
    $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
    $\pi(s_t) = \arg\max_a Q(s_t, a)$ w.prob $1 - \epsilon$, else random
    $t = t + 1$
end loop

Q-Learning

Initialize $Q(s, a), \forall s \in S, a \in A$ $t = 0$, initial state $s_t = s_0$

Set π_b to be ϵ-greedy w.r.t. Q

loop

  Take $a_t \sim \pi_b(s_t)$ // Sample action from policy

  Observe (r_t, s_{t+1})

  $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \arg\max_a Q(s_{t_1}, a) - Q(s_t, a_t))$

  $\pi(s_t) = \arg\max_a Q(s_t, a)$ w.prob $1 - \epsilon$, else random

  $t = t + 1$

end loop