Temporal Difference Learning

Temporal Difference Learning

- Break up the episode and learn directly from actual experience from the environment.
- TD is model free: no knowledge of transitions/rewards.
- Learn from incomplete episodes, by bootstrapping (i.e. update our guess of the value function from a certain step on).

MC and TD

- Goal: Given π, learn q_π online from experience.
- Incremental every-visit Monte Carlo
  - Update estimate of the value $Q(S_t, A_t)$ towards **actual** return G_t

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(G_t - Q(S_t, A_t))$$

- TD(0):
  - Update estimate of the value $V(S_t)$ towards the **estimated** return $Q(S_t, A_t)$ by:

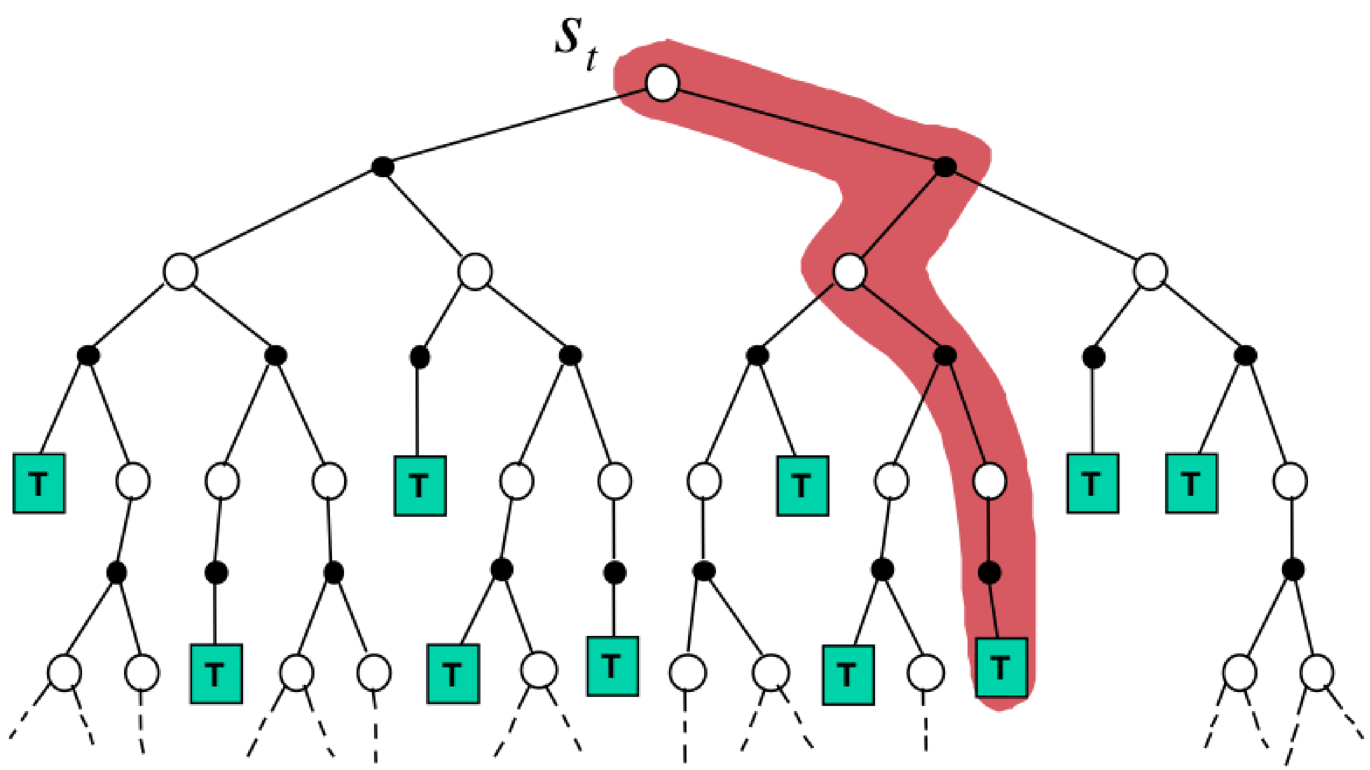
$$Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$$

MC vs TD

- TD can learn before the final episode, unlike MC.
- TD works in non-terminating environments, unlike MC.
- TD exploits the Markov property (hence more effective in Markov environments).
- MC does not exploit the Markov property (more effective in non-Markov environments).

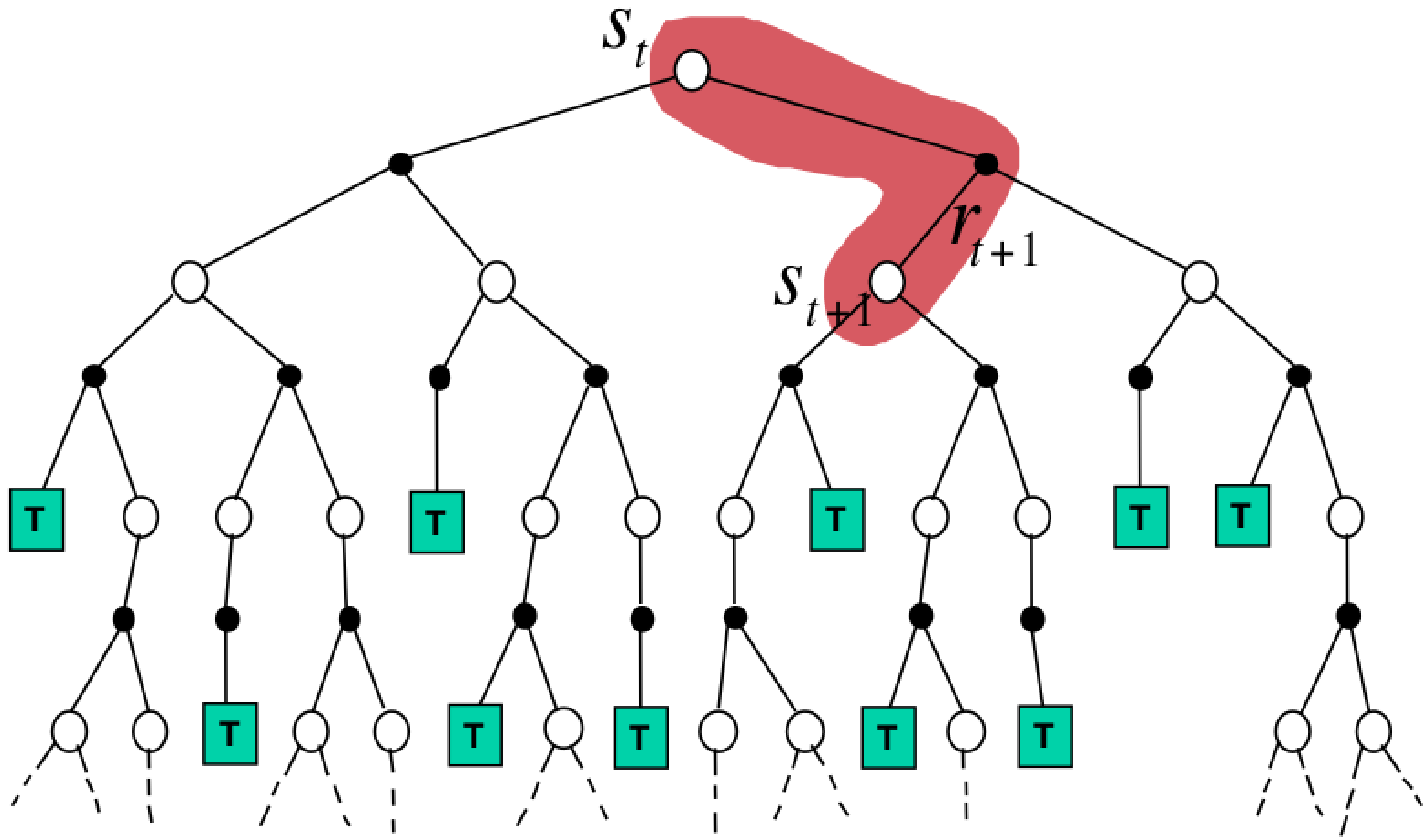
Monte-Carlo backup

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



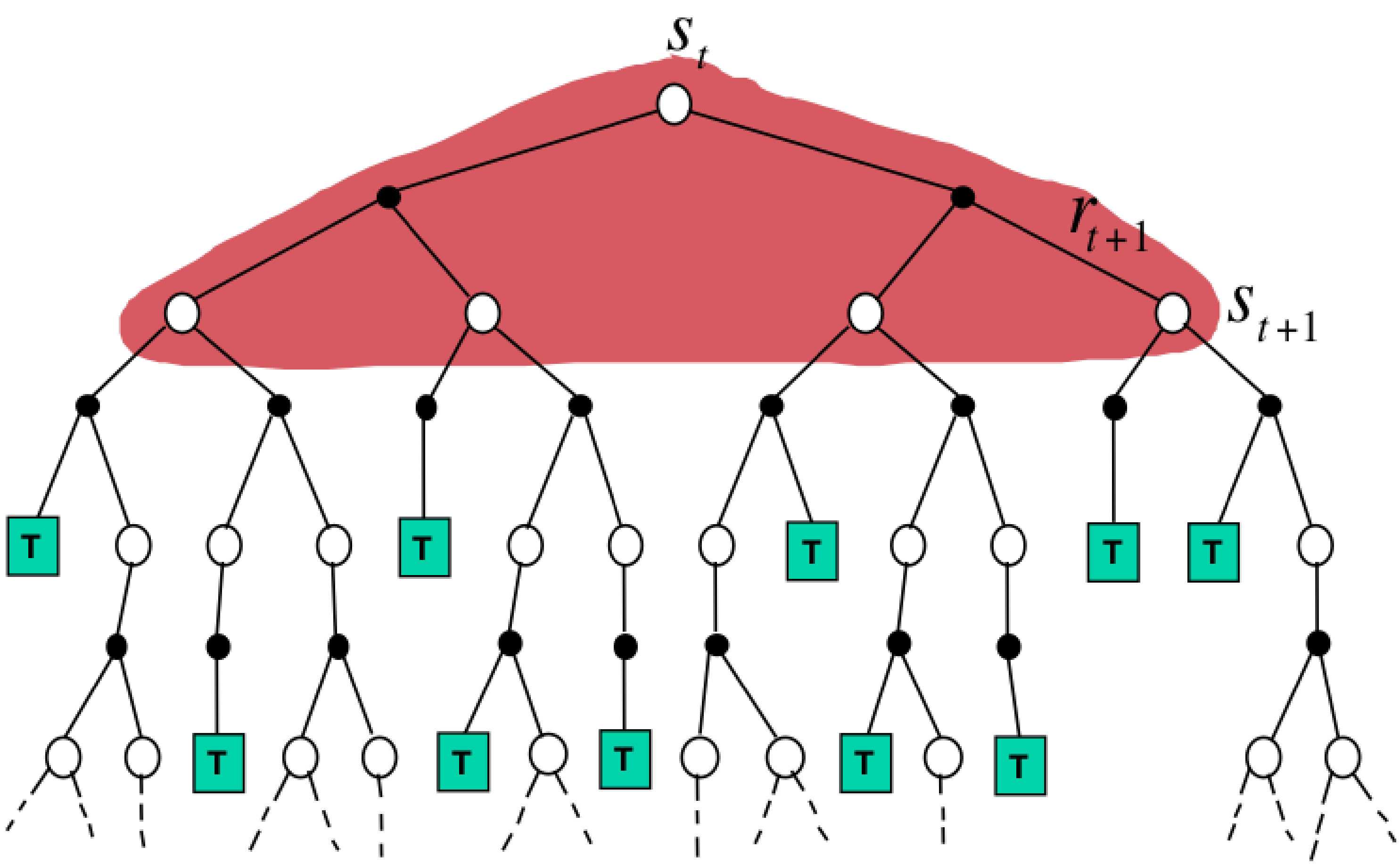
Temporal Difference backup

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



Dynamic programming backup

$$V(S_t) \leftarrow \mathbb{E}_\pi \left[R_{t+1} + \gamma V(S_{t+1})\right]$$



On and Off-Policy Learning

- **On-policy learning:**
  - "Learn on the job"
  - Learn about π by sampling experience from π.
- **Off-policy learning:**
  - "Shadow" someone.
  - Learn about policy π from experience sampled from policy π'.
  - Learn about *optimal* policy while following an *exploratory* policy.

On-Policy Learning: Sarsa

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Example: Frozen Lake

- https://gym.openai.com/evaluations/eval_HmYHTSY5QYOkl77yacPLkg

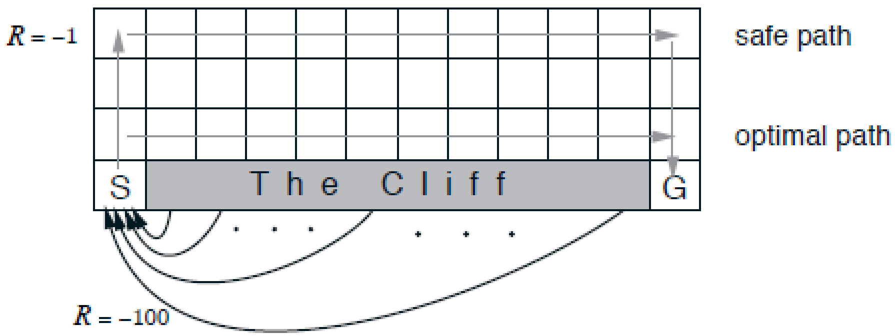
Off-Policy Learning: Q-Learning

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Repeat (for each step of episode):
        Choose A from S using policy derived from Q (e.g., ε-greedy)
        Take action A, observe R, S'
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
        $S \leftarrow S'$;
    until S is terminal

Example: Frozen Lake

- https://gym.openai.com/evaluations/eval_j8yzfQ4BQ9O6dI5Pr7SrtQ

Q-Learning vs Sarsa



Cliffworld - Results

Afterstates