Lecture 5: Model-free RL (Part II)

Max(Chong) Li

Outline

- TD Learning
- TD Control
- Summary: DP vs. TD

(Readings: Chapter 5.1, 5.2, 5.3.1, 5.3.2, 5.4.1 in RL-CPS book)

*Some materials are modified from David Silver's RL lecture notes

Outline

- TD Learning
- TD Control
- Summary: DP vs. TD

MC Learning (Refresher)

- Update $V(s)$ incrementally after episode $S_1, A_1, R_2, ..., S_T$
  For each state S_t with return G_t

$$N(S_t) \leftarrow N(S_t) + 1$$

$$V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)}(G_t - V(S_t))$$

- In non-stationary problems, it can be useful to track a running mean, i.e. forget old episodes.

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

Temporal Difference (TD) Learning

- TD methods learn directly from episodes of experience
- TD is *model-free*: no knowledge of MDP transitions / rewards
- TD learns from *incomplete* episodes, by *bootstrapping*
- TD updates a guess towards a guess

TD Learning

Goal: learn v_π online from experience under policy π

Incremental every-visit Monte-Carlo

- Update value $V(S_t)$ toward *actual* return G_t

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

Simplest temporal-difference learning algorithm: TD(0)

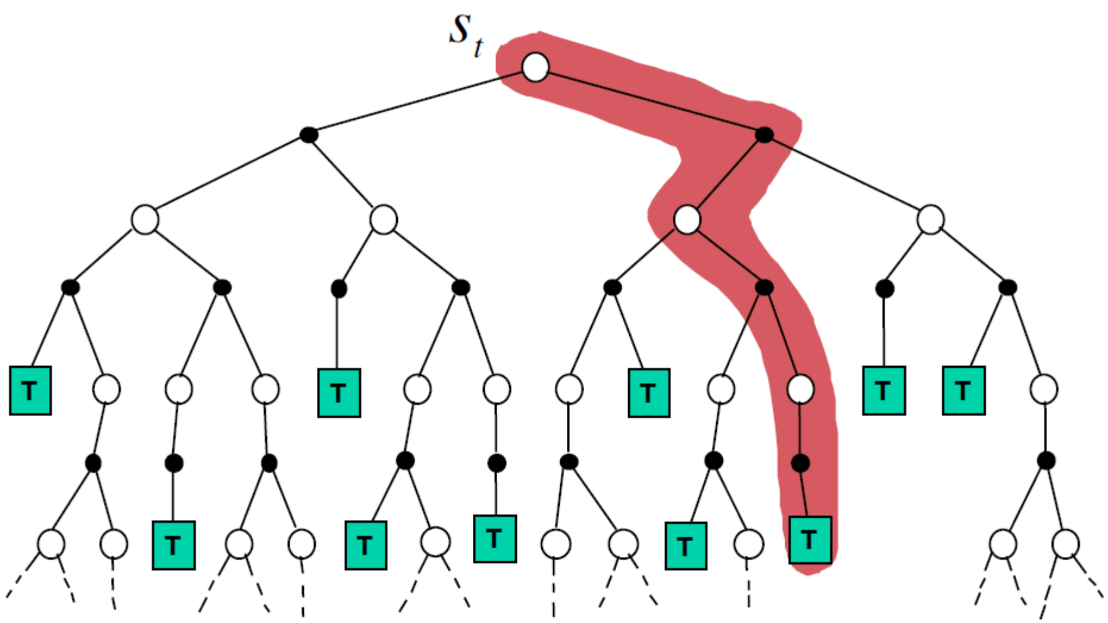
- Update value $V(S_t)$ toward *estimated* return $R_{t+1} + \gamma V(S_{t+1})$

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

- $R_{t+1} + \gamma V(S_{t+1})$ is called the *TD target*
- $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is called the *TD error*

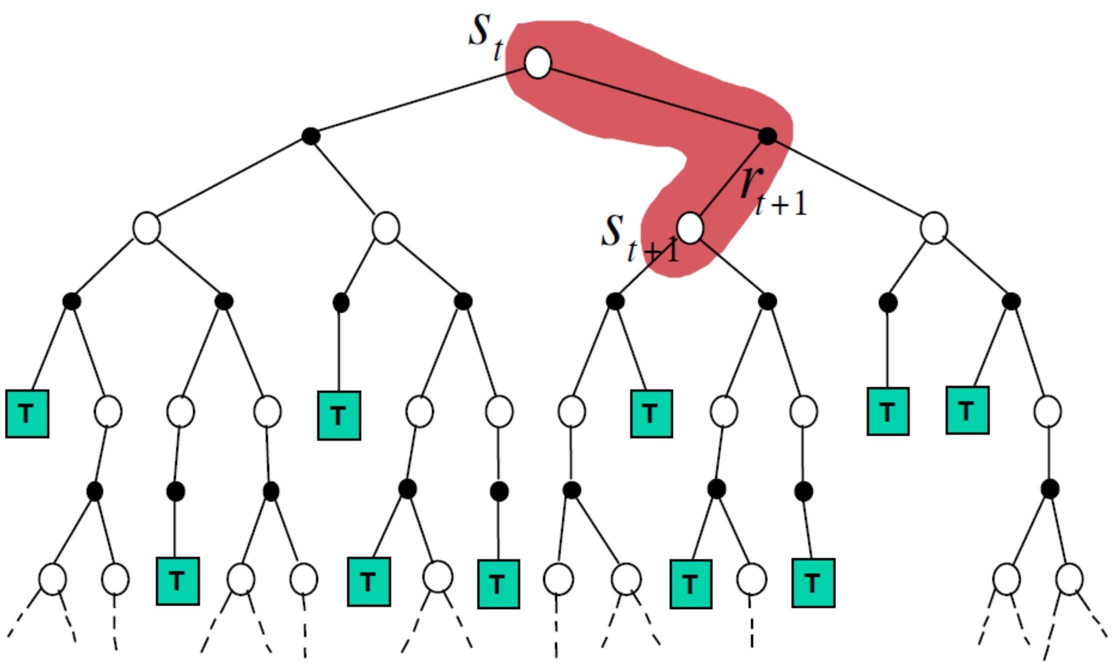
MC backup

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



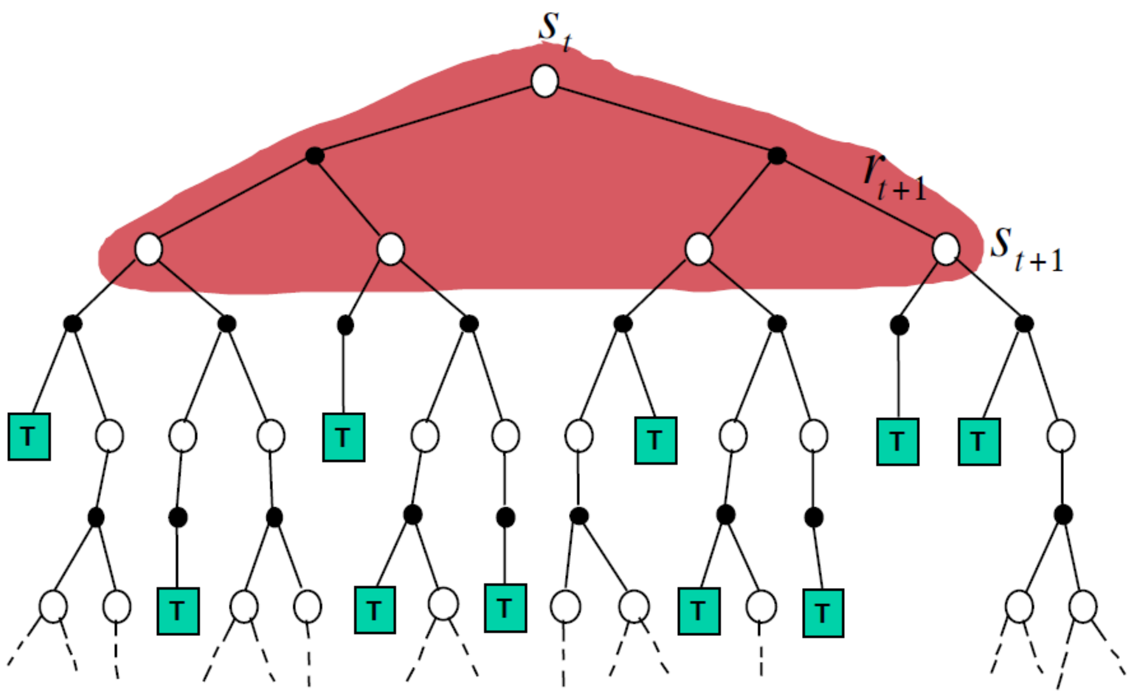
TD Backup

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



DP Backup

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



Bootstrapping & Sampling

Bootstrapping: update involves an estimate

- MC does not bootstrap
- DP bootstraps
- TD bootstraps

Sampling: update samples an expectation

- MC samples
- DP does not sample
- TD samples

MC vs. TD

TD can learn *before* knowing the final outcome

- TD can learn online after every step
- MC must wait until end of episode before return is known

TD can learn *without* the final outcome

- TD can learn from incomplete sequences
- MC can only learn from complete sequences
- TD works in continuing (non-terminating) environments
- MC only works for episodic (terminating) environments

MC vs. TD

- Return $G_t = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{T-1} R_T$ is *unbiased* estimate of $v_\pi(S_t)$
- True TD target $R_{t+1} + \gamma v_\pi(S_{t+1})$ is *unbiased* estimate of $v_\pi(S_t)$
- TD target $R_{t+1} + \gamma V(S_{t+1})$ is *biased* estimate of $v_\pi(S_t)$
- TD target is much lower variance than the return:
  - Return depends on *many* random actions, transitions, rewards
  - TD target depends on *one* random action, transition, reward

MC vs. TD

MC has high variance, zero bias

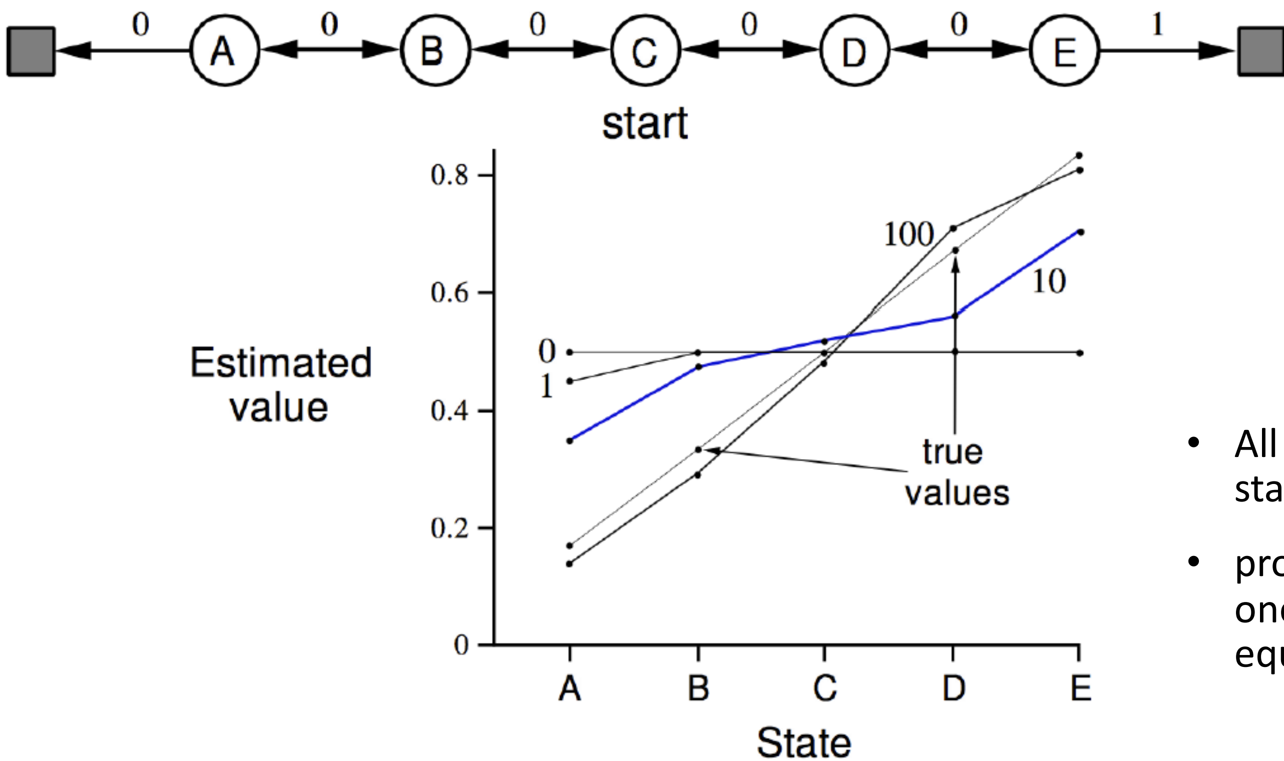
- Good convergence properties
- (even with function approximation)
- Not very sensitive to initial value
- Very simple to understand and use

TD has low variance, some bias

- Usually more efficient than MC
- TD(0) converges to $v_\pi(s)$
- (but not always with function approximation)
- More sensitive to initial value

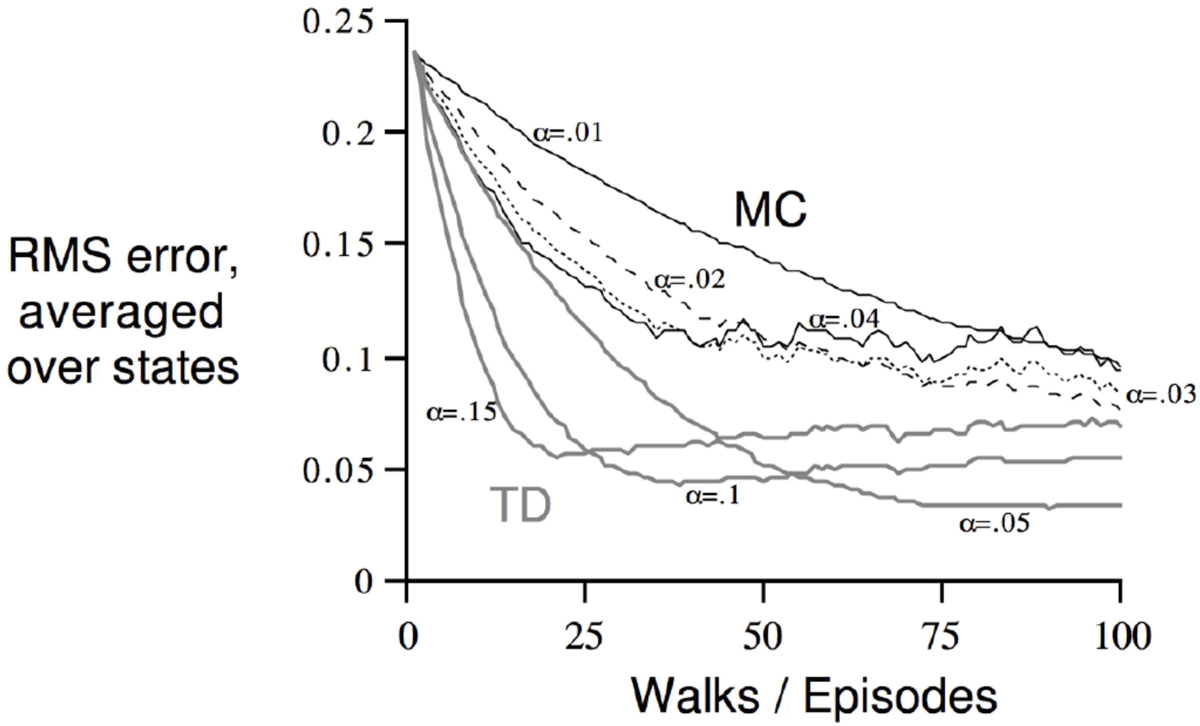
In practice, TD methods have usually been found to converge faster than constant- MC methods on stochastic tasks

Example: Random Walk



- All episodes start in the center state C.
- proceed either left or right by one state on each step, with equal probability.

Example: Random Walk



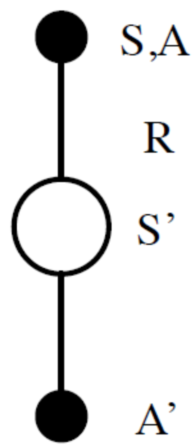
Outline

- TD Learning
- TD Control
- Summary: DP vs. TD

TD Control

- Temporal-difference (TD) learning has several advantages over Monte-Carlo (MC)
  - Lower variance
  - Online
  - Incomplete sequences
- Natural idea: use TD instead of MC in our control loop
  - Apply TD to $Q(S, A)$
  - Use ϵ-greedy policy improvement
  - Update every time-step

SARSA



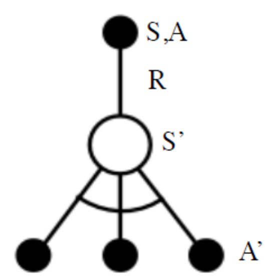
$$Q(S, A) \leftarrow Q(S, A) + \alpha \left(R + \gamma Q(S', A') - Q(S, A) \right)$$

SARSA for On-policy Control

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

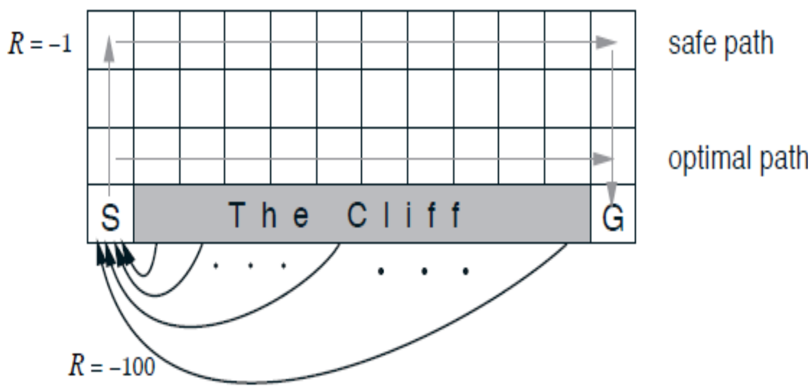
Q-Learning

- One of the most important breakthroughs in RL
- An Off-policy TD control algorithm (Watkins, 1989), why off-policy?



$$Q(S, A) \leftarrow Q(S, A) + \alpha \left(R + \gamma \max_{a'} Q(S', a') - Q(S, A) \right)$$

Example: Cliff

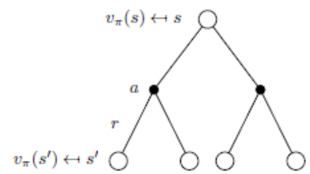
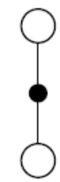
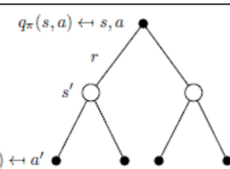
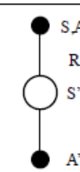
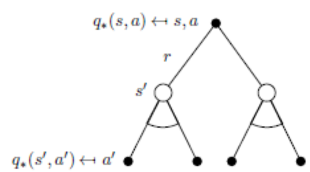
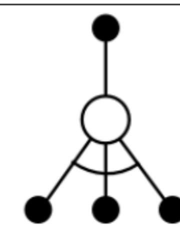


- Ɛ = 0.1, Q-learning is worse than Sarsa, why?
- If Ɛ -> 0, Q-learning v.s. Sarsa ?

Outline

- TD Learning
- TD Control
- Summary: DP vs. TD

Summary: DP vs. TD

	Full Backup (DP)	*Sample Backup (TD)*
Bellman Expectation Equation for $v_\pi(s)$	 Iterative Policy Evaluation	 TD Learning
Bellman Expectation Equation for $q_\pi(s, a)$	 Q-Policy Iteration	 Sarsa
Bellman Optimality Equation for $q_*(s, a)$	 Q-Value Iteration	 Q-Learning

Summary: DP vs. TD

Full Backup (DP)	*Sample Backup (TD)*
Iterative Policy Evaluation	TD Learning
$V(s) \leftarrow \mathbb{E}\left[R + \gamma V(S') \mid s\right]$	$V(S) \xleftarrow{\alpha} R + \gamma V(S')$
Q-Policy Iteration	Sarsa
$Q(s,a) \leftarrow \mathbb{E}\left[R + \gamma Q(S',A') \mid s,a\right]$	$Q(S,A) \xleftarrow{\alpha} R + \gamma Q(S',A')$
Q-Value Iteration	Q-Learning
$Q(s,a) \leftarrow \mathbb{E}\left[R + \gamma \max_{a' \in \mathcal{A}} Q(S',a') \mid s,a\right]$	$Q(S,A) \xleftarrow{\alpha} R + \gamma \max_{a' \in \mathcal{A}} Q(S',a')$

where $x \xleftarrow{\alpha} y \equiv x \leftarrow x + \alpha(y - x)$