Reinforcement Learning

Lecture 5: Monte-Carlo & Temporal-Difference Learning

S. M. Ahsan Kazmi

Recap

Last lecture:

- Planning via DP for a known MDP
  - Policy Evaluation
  - Policy Iteration
  - Value Iteration
  - Extensions (Dynamic Programming)

This lecture:

- Model-free prediction to estimate values in an unknown MDP
  - Monte-Carlo Learning
  - Temporal-Difference Learning

Recap: Markov decision process (MDP)

- A Markov decision process (MDP) is a Markov reward process with decisions.
- It is an environment in which all states are Markov.

Definition

A *Markov Decision Process* is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{P}$ is a state transition probability matrix,
  $\mathcal{P}_{ss'}^{a} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s, A_t = a\right]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}_s^a = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$
- γ is a discount factor $\gamma \in [0, 1]$.

Monte-Carlo Learning

Monte-Carlo Learning

- MC methods learn directly from episodes of experience
- MC is model-free: no knowledge of MDP transitions/rewards
- MC learns from complete episodes: no bootstrapping
- MC uses the simplest possible idea: value = mean return
    - Caveat: can only apply MC to episodic MDPs
- All episodes must terminate

Monte-Carlo Policy Evaluation

Goal: learn v_π from episodes of experience under policy π

$$S_1, A_1, R_2, \ldots, S_k \sim \pi$$

The **return** is the total discounted reward (for an episode ending at time $T > t$):

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-t-1} R_T$$

The value function is the expected return:

$$v_\pi(s) = \mathbb{E}\left[G_t \mid S_t = s, \pi\right]$$

We can just use **sample average** return instead of **expected** return

We call this **Monte Carlo policy evaluation**

First-Visit Monte-Carlo Policy Evaluation

- To evaluate state s
- The first time-step t that state s is visited in an episode
- Increment counter $N(s) \leftarrow N(s) + 1$
- Increment total return $S(s) \leftarrow S(s) + G_t$
- Value is estimated by mean return $V(s) = S(s)/N(s)$
- By law of large numbers, $V(s) \rightarrow v_\pi(s)$ as $N(s) \rightarrow \infty$

First-Visit Monte-Carlo Policy Evaluation

Initialize:

$\pi \leftarrow$ policy to be evaluated

$V \leftarrow$ an arbitrary state-value function

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Repeat forever:

Generate an episode using π

For each state s appearing in the episode:

$G \leftarrow$ return following the first occurrence of s

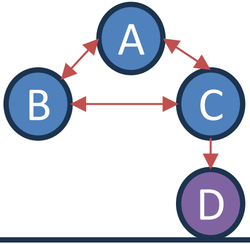
Append G to $Returns(s)$

$V(s) \leftarrow \text{average}(Returns(s))$

Every-Visit Monte-Carlo Policy Evaluation

- To evaluate state s
- Every time-step t that state s is visited in an episode
- Increment counter $N(s) \leftarrow N(s) + 1$
- Increment total return $S(s) \leftarrow S(s) + G_t$
- Value is estimated by mean return $V(s) = S(s)/N(s)$
- Again, $V(s) \rightarrow v_\pi(s)$ as $N(s) \rightarrow \infty$

Example



- 4 states MC: A, B, C, and a terminal state D. Assume the policy is random. The transition reward is -1.
  - Episode 1: A → B → C → D
  - Episode 2: A → C → B → C → D

- First-Visit Monte Carlo
  - Episode 1:
    - V(A) = -3
    - V(B) = -2
    - V(C) = -1
    - V(D) = 0 (terminal)
  - Episode 2:
    - V(A) = ((-3) + (-4)) / 2 = -3.5
    - V(B) = ((-2) + (-2)) / 2 = -2.0
    - V(C) = ((-1) + (-3)) / 2 = -2.0

- Every Visit Monte Carlo
  - Episode 1:
    - V(A) = -3
    - V(B) = -2
    - V(C) = -1
    - V(D) = 0 (terminal)
  - Episode 2:
    - V(A) = ((-3) + (-4)) / 2 = -3.5
    - V(B) = ((-2) + (-2)) / 2 = -2.0
    - V(C) = (-1 + -3 + -1) / 3 = -1.67

Incremental Monte-Carlo Updates

- 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.

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

Summary

MC has several **advantages** over DP:

- Can learn directly from interaction with the environment
- No need for full models
- No need to learn about ALL states (no bootstrapping)
- Less harmed by violating Markov property
- MC methods provide an alternate policy evaluation process

Visit Chapter 5 for more details

Temporal-Difference Learning

Temporal-Difference Learning(TD)

- 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

MC and TD

- 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

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

- Simplest temporal-difference learning algorithm: TD(0)
  - Update value $V(S_t)$ toward *estimated* return $R_{t+1} + \gamma V(S_{t+1})$

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

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

TD target for prediction

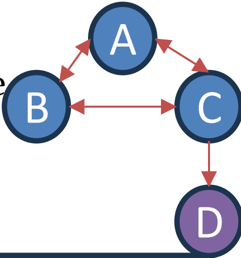
- The TD target: $R_{t+1} + \gamma V(S_{t+1})$
    - it is an estimate like MC target because it samples the expected value
    - it is an estimate like the DP target because it uses the current estimate of V

Algorithm

Input: the policy π to be evaluated
Initialize $V(s)$ arbitrarily (e.g., $V(s) = 0, \forall s \in \mathcal{S}^+$)
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    $A \leftarrow$ action given by π for S
    Take action A; observe reward, R, and next state, S'
    $V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$
    $S \leftarrow S'$
  until S is terminal

Figure 6.1: Tabular TD(0) for estimating v_π.

Example



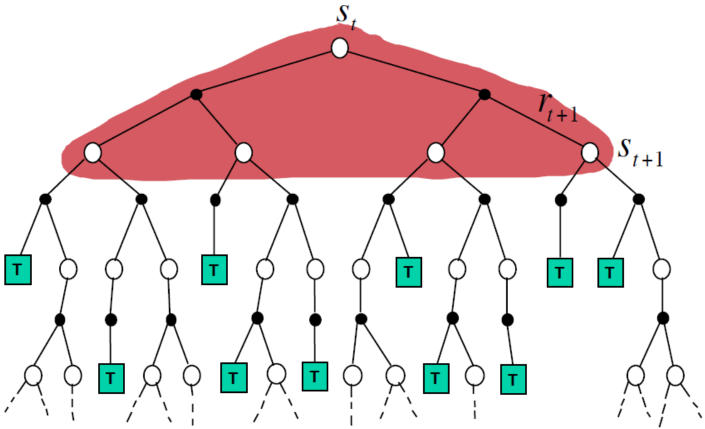
- 4 states MC: A, B, C, and a terminal state D. Assume the policy is random. The transition reward is -1, α=0.5, γ=1.
  - Episode 1: A → B → C → D
  - Episode 2: A → C → B → C → D

- TD learning
  - Episode 1:
    - $V(A) = 0+0.5(-1+1\times0-0)=-0.5$
    - $V(B) =0+0.5(-1+1\times0-0)=-0.5$
    - $V(C) =0+0.5(-1+1\times0-0)=-0.5$
    - $V(D) = 0$ (terminal)
  - Episode 2:
    - $V(A) =-0.5+0.5(-1+1\times(-0.5)-(-0.5))=-1.0$
    - $V(C) =-0.5+0.5(-1+1\times(-0.5)-(-0.5))=-1.0$
    - $V(B) = -0.5+0.5(-1+1\times(-1.0)-(-0.5))=-1.0$
    - $V(C) =-1.0+0.5(-1+1\times0-(-1.0))=-1.5$

Rule: $V(s)=V(s)+\alpha(R(s')+\gamma V(s')-V(s))$

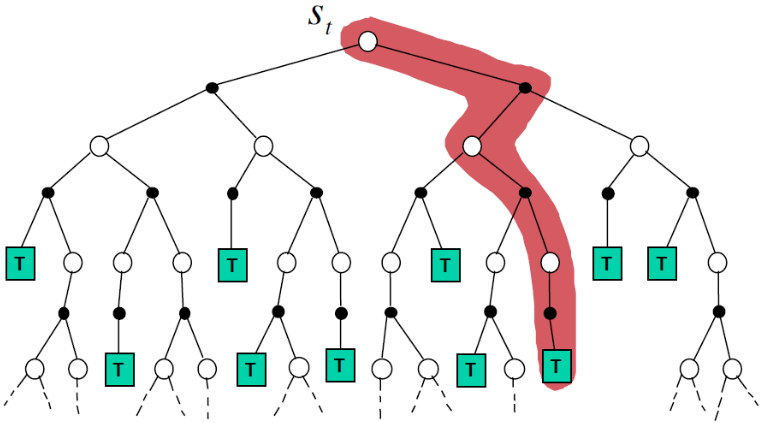
Dynamic Programming Backup

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



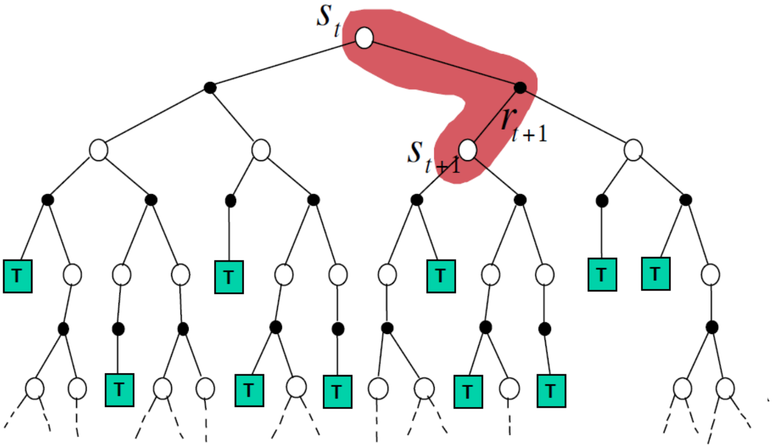
Monte-Carlo Backup

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



Temporal-Difference Backup

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



Example: Driving Home

- Consider driving home:
    - Each day you drive home
    - Your goal is to try and predict how long it will take at particular stages
    - When you the leave office you note the time, day, & other relevant info

- Consider the policy evaluation or prediction task

Example: Driving Home

State	*Elapsed Time (minutes)*	*Predicted Time to Go*	*Predicted Total Time*
leaving office, friday at 6	0	30	30
reach car, raining	5	35	40
exiting highway	20	15	35
2ndary road, behind truck	30	10	40
entering home street	40	3	43
arrive home	43	0	43

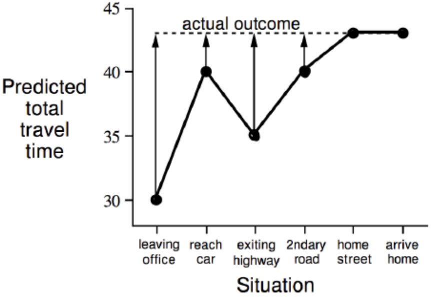
Driving home as an RL problem

State	Elapsed Time (minutes)	R	V(s) Predicted Time to Go	V(office) Predicted Total Time
leaving office, friday at 6	0	5	30	30
reach car, raining	5	15	35	40
exiting highway	20	10	15	35
2ndary road, behind truck	30	10	10	40
entering home street	40	3	3	43
arrive home	43		0	43

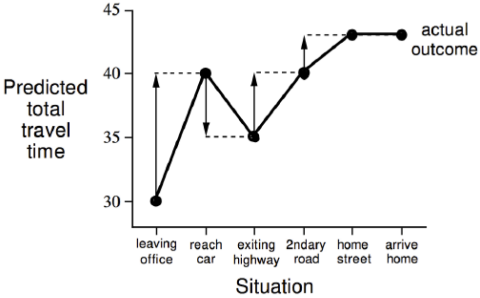
- **Goal:** update the prediction of total time leaving from office, while driving home
    - With MC we would need to wait for a termination — until we get home — then calculate Gt for each step of the episode, then apply our updates

Driving home as an RL problem

Changes recommended by Monte Carlo methods (α=1)



Changes recommended by TD methods (α=1)



Advantages and Disadvantages of MC vs. TD

TD can learn before knowing the final outcome

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

TD can learn without the outcome

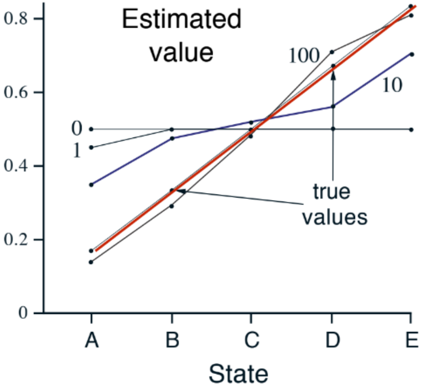
- TD can learn from incomplete sequences
- MC can only learn from complete sequences
- TD works in continuing (non-terminating) environments
- MC only works for episodic (terminating) environments

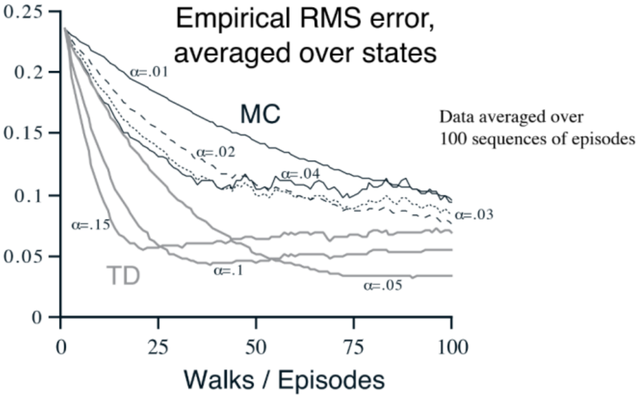
Both MC and TD converge (under certain assumptions), but which is faster?

Random Walk Example

- C is start state, episodic, undiscounted γ = 1
- π is left or right with equal probability in all states
- termination at either end
- rewards +1 on **right** termination, 0 otherwise
- what does $v_\pi(s)$ tell us?
  - probability of termination on right side from each state, under random policy
  - what is v_π =[A B C D E]?
    - v_π = [1/6 2/6 3/6 4/6 5/6]
- Initialize V(s) = 0.5 $\forall\ s \in \mathcal{S}$

TD and MC





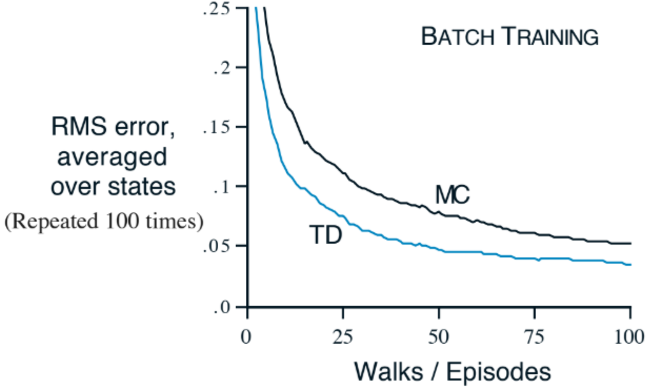
$$V(S_t) \leftarrow V(S_t) + \alpha\Big[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\Big]$$

Batch MC and TD

- Batch Updating: train completely on a finite amount of data
    - e.g., train repeatedly on 10 episodes until convergence.
- Compute updates according to TD or MC, but only update estimates after each complete pass through the data.
- For any finite Markov prediction task, under batch updating, TD converges for sufficiently small α.
- Constant-α MC also converges under these conditions, but to a different answer!

Random Walk under Batch Updating

- After each new episode, all episodes seen so far are treated as a batch
- This growing batch is repeatedly processed by TD and MC until convergence



End of lecture