- **Feb 19, 2021 Notes: Temporal Difference**

  - Temporal difference: Constantly adjust value function to decrease error
    - ↪ Updates after every atomic experience
    - ↪ You do not have to update in the sequence you get data
      - ↪ You can replay experiences

  - TD w/ Function Approx.
    - ↪ Parameterize value function with weights w
    - ↪ Semi-gradient

  - In code: Input is not iterable of episodes, it is iterable on atomic experiences.
    - ↪ In MC we used "iterate_updates" but here we cannot use it.
      - ↪ Instead use "iterate.accumulate"
        - ↪ Inputs:
          1. Data structure you iterate over
          2. Function that takes
             - i. Current sum (or other function)
             - ii. Element in list we are looking at

             returns updated sum (or other function)
          3. Initial value of accumulator
        - ↪ You can accumulate any quantity over any data structure.
        - ↪ See slide 17 for code related to TD.

- Functional Programming: Code looks Clean / Simple

  ↳ Almost all bugs come from "imperative" style of programming with lots of loops.

- TD is the main way of implementing reinforcement learning

  ↳ Returns can be expressed in recursive manner (like DP)

  ↳ You can learn from experience, you don't need P_R (like MC)

  * Can overcome curse of dimensionality and curse of modeling

  * TD reflects the way humans learn in real life

    ↳ We make immediate updates (like TD), we don't wait and learn in bulk (as in MC)

- MC is unbiased estimate of VF, TD is biased

  ↳ But no free lunch:

    MC has high variance, TD has low variance

- Because MC is unbiased, it converges well even with function approx

- <u>Theorem:</u> Tabular TD prediction converges to true VF in the mean for constant α

  ↳ Requires Robbins-Monro learning rate schedule

$$\sum_{n=1}^{\infty} \alpha_n = \infty \quad \text{and} \quad \sum_{n=1}^{\infty} \alpha_n^2 < \infty$$

  ↳ TD forgets can more rapidly and can converge quickly

- Speed of Convergence of TD vs. MC:
  - Compare speed of convergence, efficiency in use of limited data
  - With constant α, TD performs better than MC
  - Compare TD and MC for different Markov Decision Processes using calculus
  - MC is sensitive to initial value, TD is not.
  - Try generating convergence graph

- So far we've focused on the mechanics of how TD and MC work.
  - Now we will talk about what it is learning?
    - Only have 10 episodes, but can replay experience = "Experience Replay." Randomly pull out traces for MC or atomic experiences in TD
      - What is learned by TD and MC is different.
      - TD pulls atomic experiences
      - MC is Simple Supervised learning
        - Gives you average return of each state
      - TD drives towards VF of MRP that is implied by experiences
        - Empirical Markov Reward Process
          - Get empirical transition probabilities
          - Takes advantage of Markov Properties
            - If process is not truely Markovian, MC is better
            - TD is not building MRP, but empirical MRP VF is the same as that constructed by TD

- MC is a Naive method, TD Calculates VF of an implied MRP

- We summarize MC, DP, TD in terms of whether they:

  ① Bootstrap : Update by VF utilize current or prior estimate of VF

  ② Experiencing : Interaction with actual or simulated environment

  ↳ learning vs. planning

* See slide 26 for characterizations

Not wide → • MC Backup is deep - Backup VF through entire episode

Not wide → • TD backup is shallow - Backup VF through one atomic experience

Very wide → • DP is shallow

- Asik : You always want to use DP whenever possible

  ↳ Approximate DP is a middle-ground between TD and RL

- Tabular n-step bootstrap :

$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_{t,n} - V(S_t; v) \right)$$

  ↳ n serves as a tuning knob between TD and MC

- λ-Return Prediction Algorithm ← This is an offline (batch update) algorithm

$$w_n = (1-\lambda)\,\lambda^{n-1} \quad \text{for all } n = 1, \dots, T-t-1$$

  ↳ Instead of $G_{t,n}$ a valid target is a weight-averaged target

  ↳ λ-return algorithm used as a knob between TD and MC

- Online algorithms update after every step

  ↳ We want an online algorithm that enables tuning from TD to MC

- **TD(λ) Algorithm:**
  - Slight detour: Eligibility Traces.
    - ↳ Consider events occurring in time.
      - ↳ Mechanism count which decays with time
  - Functions when you care about recent events more than older events
  - Eligibility Trace is a memory function for each State
    - ↳ At each time step, update memory function for all states.

$$E_t(s) = \gamma \lambda E_{t-1}(s) + \mathbb{I}[s_t = s]$$

  - See algebra proof on slide 39
    - ↳ Telescoping algorithm.
      - ↳ Try to write this out.

$$\underbrace{G_t^{(\lambda)} - V(s_t)}_{\text{Lambda return also update}} = \delta_t + \dots \text{ See slide 40.}$$

- When generalizing to function approx, eligibility traces for parameters, not States.
- Try out RL assignments from this week to build foundation.