Reinforcement Learning (RL)

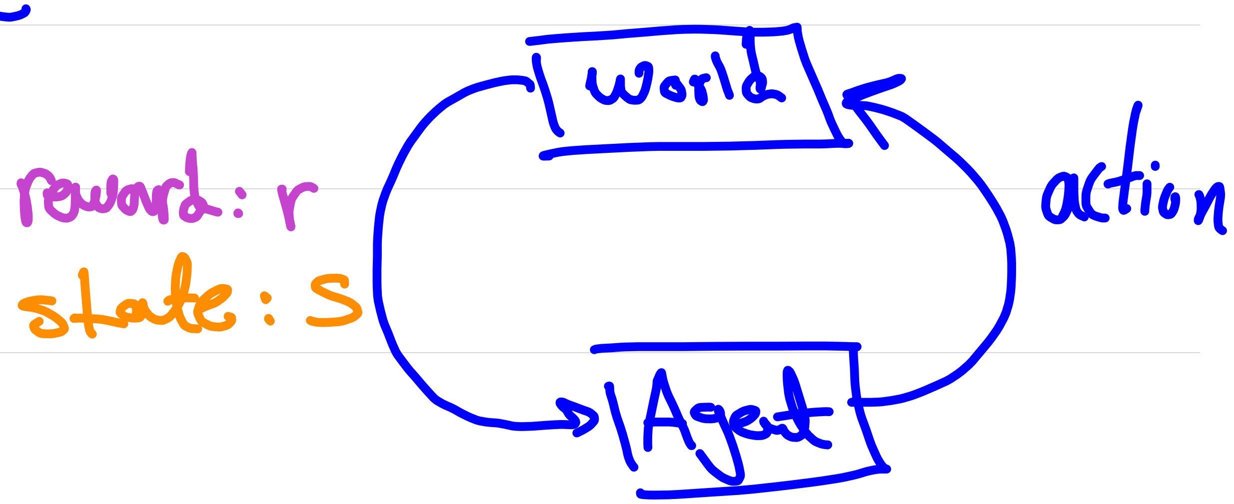
- Outline:
  - MDP V.S. RL
  - Active RL V.S. Passive RL
  - Active RL
    - Model based Learning
    - Model Free Learning
      - Direct Evaluation
      - Temporal Difference Learning
      - Q-learning

RL: Unknown Transition and Reward Model

- We saw how MDP can be solved
- What if the environment model is not entirely known?
  - How can we find a policy?
- Here comes RL
  - Learning what?
    - The MDP model OR some parts of the model (e.g., value functions or Q-functions which are enough to find the optimal policy
  - Learning from what?
    - From samples (aka episodes) of the process (i.e. transition of states)

Reinforcement Learning (RL)

- We still assume a MDP
  - (S, A, T, R, γ)
- We are still looking for a good policy
- We have a new challenge
  - We don't know T or R
    - Must explore new states and actions to observe environment
    - Basic Idea: Learn how to maximize expected rewards based on observed samples of transitions



MDP vs. RL

- MDP
  - Solving MDP is an offline problem.
    - The model is Known
  - The goal is to find a policy that Collects maximum utility
- Reinforcement Learning
  - RL is an online problem.
    - The model is not Known.
  - The goal is to perform actions to learn the model and Collect rewards

Passive RL vs. Active RL

- Passive RL:
  - How to Learn from already given experiences
    - Similar to supervise learning: learns from already given labeled datapoints
- Active RL:
  - How to collect new experience and learn from them

Passive RL

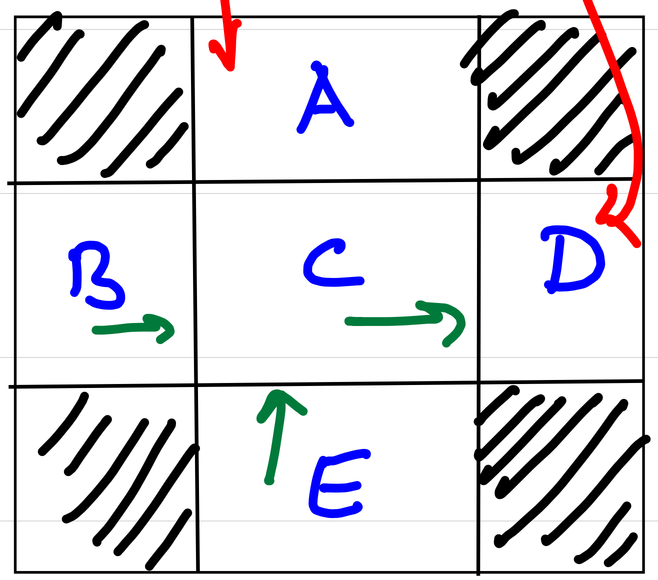
- Approaches to Passive RL
  - Model-Based Learning
    - Learn the MDP Dynamics (i.e. T) and Reward (i.e. R) from Samples
    - Then, Solve the MDP (i.e., use Policy iteration or value iteration)
  - Model-Free Learning
    - Directly learns $V(s)$ or $Q(s,a)$ from experience
    - Uses $V(s)$ or $Q(s,a)$ to make decisions

Model-based Learning

- Model-based learning:
  - Basic idea:
    - Learn an estimate MDP model (i.e. T) based on experience
    - Then, solve the approximate MDP
- Step 1: Learn empirical MDP model $T(s,a,s') = P(s'|s,a)$
  - Count outcomes s' for each s,a
    - use the count to estimate $\hat{T}(s,a,s') = \hat{P}(s'|s,a)$
- Similarly estimate $\hat{R}(s,a,s')$
- Step 2: Solve the learned MDP (e.g., with value iteration or policy iteration)

Example: Model-based Learning Example

Terminal states



Input policy π

(Assume $\gamma = 1$)

Episode 1:

B, east, C, −1,
C, east, D, −1,
D, exit, "end", +10

Episode 2

B, east, C, −1,
C, east, D, −1,
D, exit, "end", +10

Episode 3

E, east, C, −1,
C, east, D, −1,
D, exit, "end", +10

Episode 4

E, east, C, −1,
C, east, A, −1,
A, exit, "end", 10

Learned Model

$\hat{T}(B, east, C) = 1$

$\hat{T}(C, east, D) = 3/4$

$\hat{T}(C, east, A) =$

$\vdots$

$\hat{R}(B, east, C) =$

$\hat{R}(C, east, D) =$

$\hat{R}(C, east, A) =$

$\vdots$

Pros and Cons of Model-based Learning

- Pro:
  - Makes efficient use of experience (low sample complexity)
- Con:
  - May not scale to large state space
    - Learns model one state-action pair at a time
    - Cannot solve MDP for very large $|S|$
  - Much harder when the environment is partially observable

Basic Idea Behind Model-Free Learning

- To approximate expectations w.r.t. a distribution, we can either
  - Estimate the distribution from samples, then compute the expectation based on the estimated distribution
  - or, bypass the distribution and estimate the expectation from the samples directly

Let's See an Example

- Consider the task of estimating the expected age of UofT students: $\mathbb{E}[A]$
- If the probability distribution of A was known, we could find it easily: $\mathbb{E}[A] = \sum_a P(a) \cdot a$
- Without $P(A)$, we have to collect samples: $[a_1, a_2, \ldots, a_N]$
  - Model-Based approach:
    - Estimat $P(A)$ first: $\hat{\mathbb{P}}[A=a] = \frac{N_a}{N}$
    - Then we use $\hat{\mathbb{P}}$ to estimate $\mathbb{E}[A]$: $\hat{\mathbb{E}}[A] = \sum_a \hat{\mathbb{P}}[A=a] \cdot a$
  - Model-Free approach:
    - Use the samples to estimate $\mathbb{E}[A]$: $\frac{\sum_i a_i}{N}$

Basic Idea Behind Model-Free Learning

- In RL, our ultimate goal is to find an estimate of expected return in each state.
  - Model-based Learning: Estimate the probability distribution of return from the samples. then use it to calculate the expected return.
  - Model-Free: estimate the expected return directly from samples.

Passive RL

- Simplified Passive RL model
  - Input: stream of transitions produced by following some fixed policy $\pi(s)$
    - E.g., we are given the following episodes:

      $(s, \pi(s), s', r, \pi(s'), r', \ldots, end)$

      $(s, \pi(s), s'', r, \pi(s''), r'', \ldots, end)$

      $\vdots$
  - Output: estimate of the state values $\hat{V}_\pi(s)$
  - Note: we don't know T and R

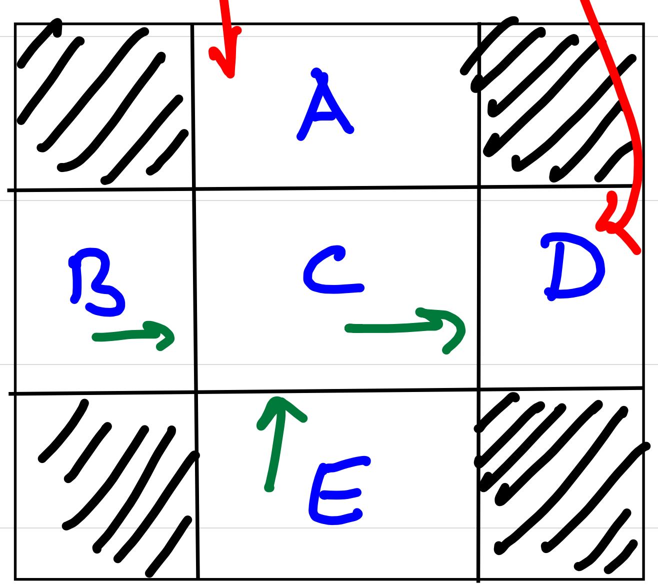
Direct Evaluation

- Consider the passive learning model described before, with the goal of estimating $V_\pi(s)$, i.e. expected total discounted reward from s onward.
- Direct Evaluation: It uses returns, the actual sums of discounted reward from s onward.
  - Average over multiple trials and visits to s.

This is also known as "direct utility estimation" or "Monte-Carlo evaluation"

Example: Direct Evaluation

Terminal states



Input policy π
(Assume $\gamma=1$)

$V_{\pi}(E) =$

Episode 1:

B, east, C, −1 ,
C, east, D, −1 ,
D, exit, "end", +10

Episode 2

B, east, C, −1 ,
C, east, D, −1 ,
D, exit, "end", +10

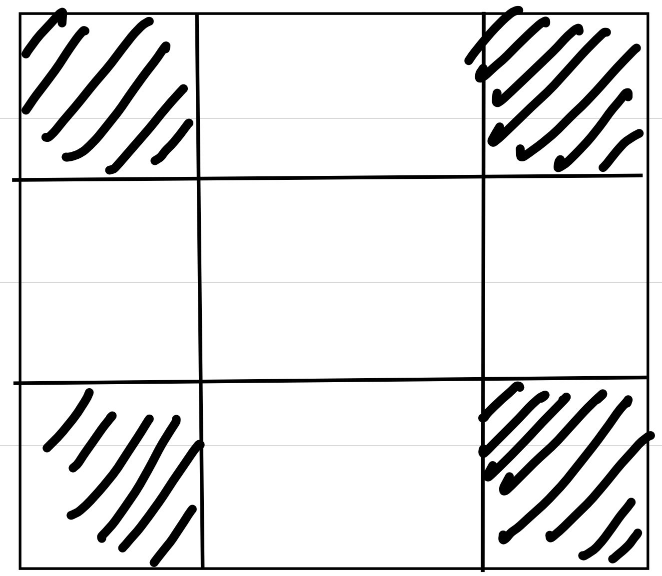
Episode 3

E, east, C, −1 ,
C, east, D, −1 ,
D, exit, "end", +10

Episode 4

E, east, C, −1 ,
C, east, A, −1 ,
A, exit, "end", −10

Output Values



Direct Evaluation Pros and Cons

- Pros:
  - It does not require any knowledge of T and R
  - It converges to the right answer in the limit.
- Cons:
  - It ignores information about state connections.
  - Each state must be learned separately
  - So, slow to learn.

How Can We Incorporate Information About state Connections?

- Direct Evaluation waits for each episode to end
  - Then it finds the sum of discounted return of state s for that episode
  - Then, it updates $\hat{V}_{\pi}(s)$
- why don't we use a similar approach to Policy Evaluatio?
  - Policy Evaluation uses the state Connection (by using Bellman equations):

$$V_{\pi,k+1}(s) \leftarrow \sum_{s'} T(s,\pi(s),s')\left[R(s,\pi(s),s') + \gamma V_{\pi,k}(s')\right]$$

  - Sadly, we don't have $R(\cdot)$ and $T(\cdot)$:(
    - Temporal Difference (TD) Learning to the rescue

Before TD, Let's See Some Naive Ideas

- Before we present TD, let's study some naive ideas to exploit state connections
  - Idea 1: Use actual samples to estimate the expectation
  - Idea 2: Update value of s after each transition s, a, s', r
- These two naïve ideas help us better understand the design principle behind TD

Idea 1

- Let's take a second look at the state value recursion relation:

- Hence, to estimate $V_\pi(s)$, we must estimate the expectation above.
  - Just like what we saw earlier, estimate the expected value of a random variable by finding average of its realizations

idea 1, cont'd

- Idea 1: use actual sample to estimate the expected return.

$s, \pi(s), s', \ldots\ldots$

$s, \pi(s), s'', \ldots\ldots$

$\vdots$

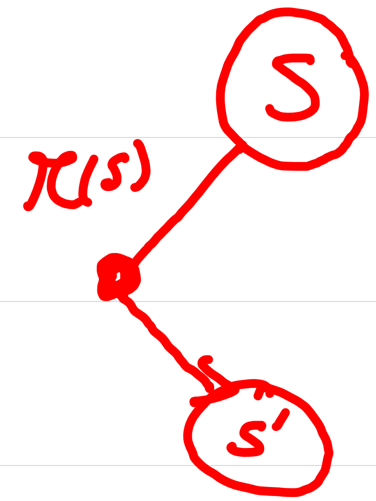
$s, \pi(s), s''', \ldots$

Idea 2

- Idea 1 didn't work, because we cannot generate N samples whenever we want.
- Instead, we should learn to use the possible sample that we may observe in an episode.
- Idea 2: update value of S after each observed transition $s, a, s', \dots$

[1] Upon seeing this transition, we have a sample:

Sample of $V(s)$:



Running Average

- How can we compute the average of $x_1, x_2, \ldots, x_n$ numbers?
  - Method 1: Add them up and divide by n.
  - Method 2: Keep a running average μ_K and a running count K.
    - $K=0$, $\mu_0 = 0$
    - $K=1$, $\mu_1 = (0 \times \mu_0 + x_1)/1 = x_1$
    - $K=2$, $\mu_2 = (1 \times \mu_1 + x_2)/2 = \frac{x_1 + x_2}{2}$
    - $K=3$, $\mu_3 = (2 \times \mu_2 + x_3)/3 = \frac{x_1 + x_2 + x_3}{3}$
    - General formula: $\mu_n = \left((n-1)\mu_{n-1} + x_n\right)/n$

$$= \left(1 - \frac{1}{n}\right)\mu_{n-1} + \frac{1}{n} x_n$$

Running Average

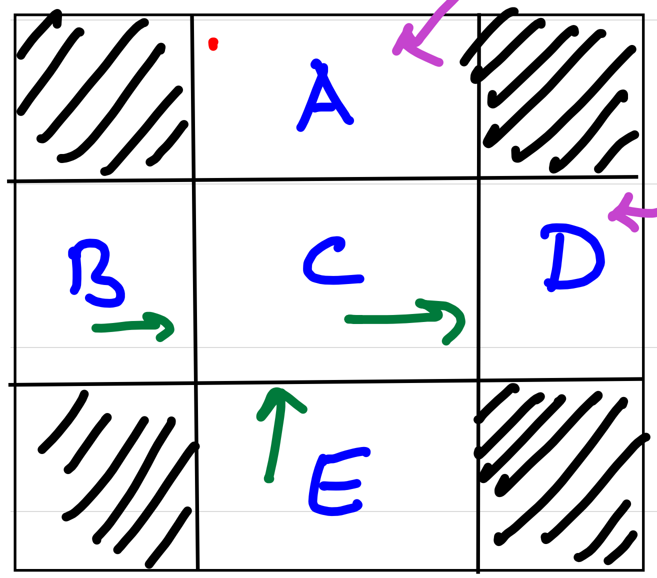
- What if we use a weighted averag with a fixed weight?
  - $\mu_n = (1-\alpha)\mu_{n-1} + \alpha x_n$
  - $\mu_n = \dfrac{x_n + (1-\alpha)x_{n-1} + (1-\alpha)^2 x_{n-2} + \cdots}{1 + (1-\alpha) + (1-\alpha)^2 + \cdots}$
- This way of running average makes recent samples more important.
- Also, it forgets about the past (distant values were wrong anyway)

Temporal Difference Learning

- TD updated the values by maintaining a running average
  -
  -
    -
- This is Temporal Difference Learning Rule.
  - $[\text{Sample} - V_\pi(s)]$ is the TD error.
  - α is the learning rate
  - We observe a sample, move $V_\pi(s)$ a little bit to make it more consistent with its neighbor $V_\pi(s')$

Example: TD

Terminal states with known values



Input policy π

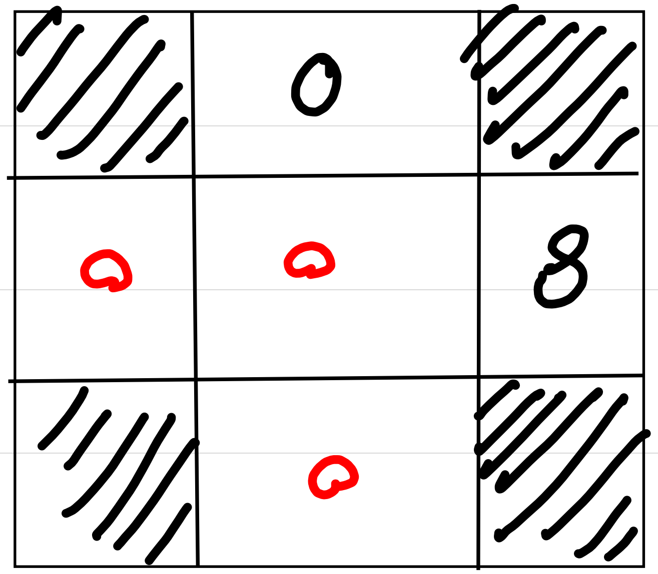
(Assume $\gamma = 1$)

($\alpha = 1/2$)

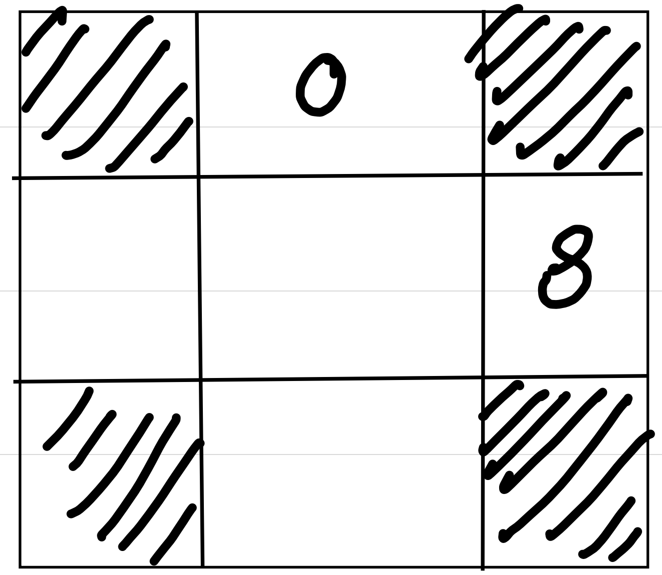
Episode 1:

B, east, C, −2,
C, east, D, −2,
D, exit, "end", +10

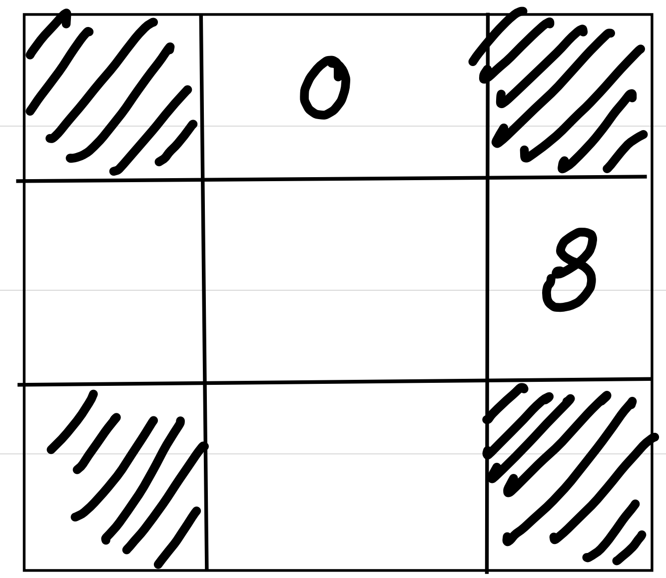
Initially set to zero



B, east, C, −2



C, east, D, −2



Problems with TD Value Learning

- TD value learning is a model-free way to do policy evaluation
  - ↳ mimicking Bellman updates with running sample averages.

- But we can't use the value function, or improve the policy without T.

$$Q(s,a) = \sum_{s'} T(s,a,s')\left[R(s,a,s') + \gamma V(s')\right]$$

- What can we do?
  - ↳ Learn q-values, not values
  - ↳ makes action selection model-free, too!