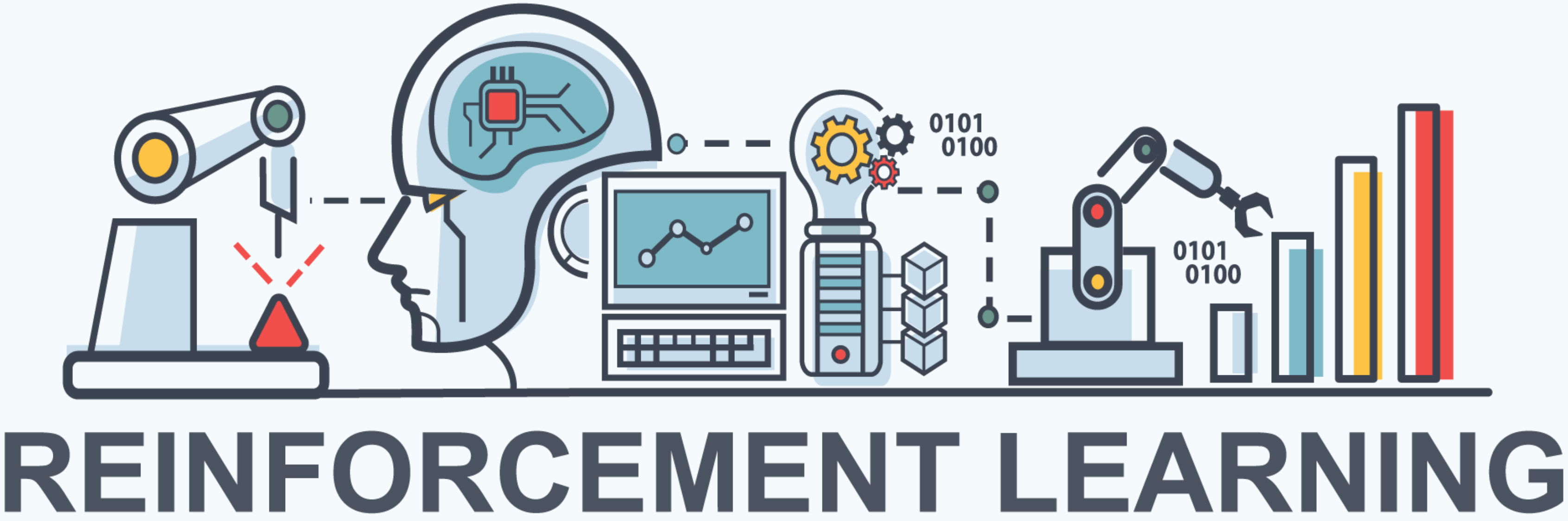


CP8319/CPS824
Lecture 7

Instructor: Nariman Farsad

* Some of the slides in this deck are adopted from courses offered David Silver, Emma Brunskill, and Sergey Levine.

Today's Agenda

1. **Last Lecture Review**

2. Model Free Policy Evaluation (Monte Carlo)

3. Model Free Policy Evaluation (Temporal Difference)

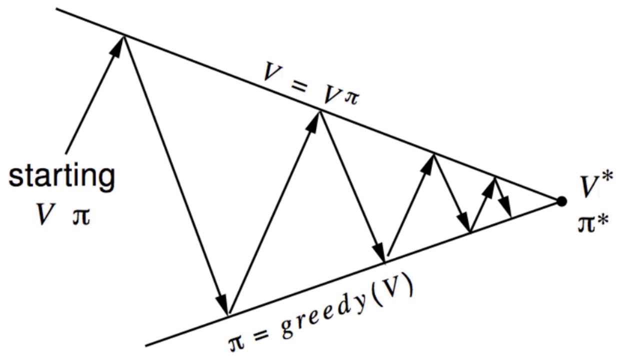
Policy Iteration

Set $i = 0$

Initialize $\pi_0(s)$ randomly for all states s

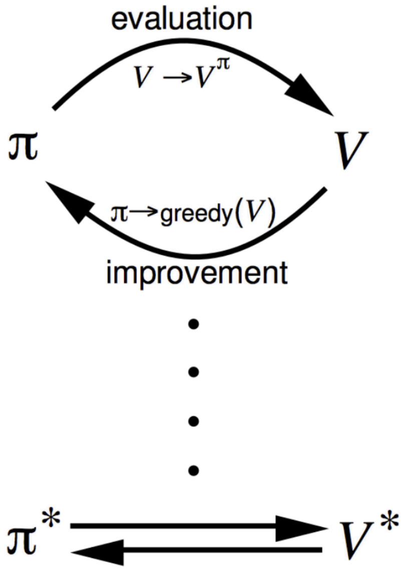
While $i == 0$ or $\| \pi_i - \pi_{i-1} \|_1 > 0$ (L1-norm, measures if the policy changed for any state):

- $v^{\pi_i} \leftarrow$ MDP value function **policy evaluation** of π_i (see slide 6 for formula)
- $\pi_{i+1} \leftarrow$ **Policy improvement**
- $i = i + 1$



Policy evaluation Estimate v_π
Iterative policy evaluation

Policy improvement Generate $\pi' \geq \pi$
Greedy policy improvement



MDP Policy Evaluation: Iterative Algorithm

Initialize $v_0(s) = 0$ for all s
For $k = 1$ until convergence:
    For all $s \in \mathcal{S}$:

$$v_{k+1}^{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P_{s,s'}^{a} v_k^{\pi}(s') \right) \longrightarrow$$

Action Chosen Randomly, e.g.:

- Flip a coin
- Go right if head
- Go left if tail

$$v_{k+1}^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} P_{s,s'}^{\pi(s)} v_k^{\pi}(s') \longrightarrow$$

One action is performed deterministically:

- Always go left

This is known as Bellman expectation backup

MDP Policy Evaluation: Assignment 1 Q2

Reward can be defined as:

$$R(s, s', a) = R(s, s', \pi(s))$$

Initialize $v_0(s) = 0$ for all s

For $k = 1$ until convergence:

For all $s \in \mathcal{S}$:

$$v_{k+1}^{\pi}(s) = \sum_{s' \in \mathcal{S}} P_{s,s'}^{\pi(s)} [R(s, s', \pi(s)) + \gamma v_k^{\pi}(s')]$$

```python
def policy_evaluation(P, nS, nA, policy, gamma=0.9, tol=1e-3):
    """Evaluate the value function from a given policy.

    Parameters
    ----------
    P, nS, nA, gamma:
        defined at beginning of file
    policy: np.array[nS]
        The policy to evaluate. Maps states to actions.
    tol: float
        Terminate policy evaluation when
            max |value_function(s) - prev_value_function(s)| < tol
    Returns
    -------
    value_function: np.ndarray[nS]
        The value function of the given policy, where value_function[s] is
        the value of state s
    """

    value_function = np.zeros(nS)
    print(P[1][policy[1]])
```

```
SFFF
FHFH
FFFH
HFFG
```

Deterministic

```
[(1.0, 0, 0.0, False)]
```

Stochastic

```
[(0.3333333333333333, 1, 0.0, False), (0.3333333333333333, 0, 0.0, False), (0.3333333333333333, 5, 0.0, True)]
```

Policy Improvement

Compute state-action value of a policy π_i
For $s \in \mathcal{S}$ and $a \in \mathcal{A}$:

- $Q^{\pi_i}(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P_{s,s'}^{a} v^{\pi_i}(s')$

Compute new policy π_{i+1}, for all $s \in \mathcal{S}$

- $\pi_{i+1}(s) = \arg\max_{a \in \mathcal{A}} Q^{\pi_i}(s,a)$

With probability 1 choose an action that maximizes Q (i.e., a deterministic policy)

Value Iteration

- Set k = 1
- Initialize $v_0(s) = 0$ for all states s
- Loop until [finite horizon, convergence]:
  - For each state s

$$v_{k+1}(s) = \max_{a \in \mathcal{A}} \left(R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P^a_{s,s'} v_k(s') \right)$$

  - View as Bellman backup on value function

$$v_{k+1} = \mathfrak{B} v_k$$

- To extract optimal policy if can act for k + 1 more steps,

$$\pi(s) = \arg\max_{a \in \mathcal{A}} \left(R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P^a_{s,s'} v_{k+1}(s') \right)$$

Today’s Agenda

1. Last Lecture Review

2. **Model Free Policy Evaluation (Monte Carlo)**

3. Model Free Policy Evaluation (Temporal Difference)

What we have learned up to now?

So far we have solved a *known* MDP, i.e., dynamics and the reward function are known

Moving forward:

- Estimate the value function of an *unknown* MDP

- Optimize the value function of an *unknown* MDP

Monte-Carlo Reinforcement Learning

- MC methods learn directly from episodes of experience
- MC is *model-free*: no knowledge of MDP transitions / rewards
  - The agent must still be able to act and experiment in environment
- MC learns from *complete* episodes
- MC idea: value = mean return ≈ average return across many episodes
- Caveat: can only apply MC to *episodic* MDPs
  - All episodes must terminate

Monte-Carlo (On) Policy Evaluation

- Aim: estimate $v^{\pi}(s)$ given episodes generated under policy π
  - e.g., $s_1, a_1, r_1, s_2, a_2, r_2, \ldots$ where the actions are sampled from π
- $G_t = rt + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$ under policy π
- $v^{\pi}(s) = \mathbb{E}_{\pi}[G_t \mid S_t = s]$
- Simple: Estimates expectation by empirical average (given episodes sampled from policy of interest)
- Updates *V* estimate using **sample** of return to approximate the expectation
- Does not assume Markov process
- Converges to true value under some (generally mild) assumptions

First Visit MC (On) Policy Evaluation

Initialize $N(s) = 0, G(s) = 0\ \forall s \in \mathcal{S}$

Loop:

- Sample episode i: $s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \ldots, s_{i,T_i}$
- Define $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \cdots + \gamma^{T-1} r_{i,T_i}$ as return from time step t onwards in i-th episode
- For each state s visited in episode i:
  - For first time t that state s is visited in episode i:
    - Increment counter of total visits: $N(s) = N(s) + 1$
    - Increment total return $G(s) = G(s) + G_{i,t}$
    - Update estimate $v^\pi(s) = G(s)/N(s)$

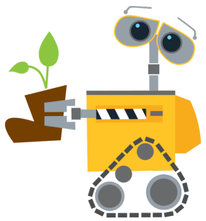
Every-Visit MC (On) Policy Evaluation

Initialize $N(s) = 0, G(s) = 0\ \forall s \in \mathcal{S}$

Loop:

- Sample episode i: $s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \dots, s_{i,T_i}$
- Define $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \cdots + \gamma^{T-1} r_{i,T_i}$ as return from time step t onwards in i-th episode
- For each state s visited in episode i:
  - For every time t that state s is visited in episode i:
    - Increment counter of total visits: $N(s) = N(s) + 1$
    - Increment total return $G(s) = G(s) + G_{i,t}$
    - Update estimate $v^\pi(s) = G(s)/N(s)$

Example: A Robot

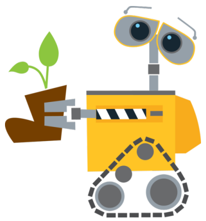
s_1	s_2	s_3	s_4	s_5	s_6	s_7
						

- R = [1 0 0 0 0 0 +10] for any action
- Sample episode = $(s_3, a_1, 0, s_2, a_1, 0, s_2, a_1, 0, s_1, a_1, 1,$ terminal)
- Let $\gamma = 1$

First visit MC estimate of v of each state after this episode?

Every visit MC estimates of s_2?

Example: A Robot

s_1	s_2	s_3	s_4	s_5	s_6	s_7
						

- R = [1 0 0 0 0 0 +10] for any action
- Sample episode = $(s_3, a_1, 0, s_2, a_1, 0, s_2, a_1, 0, s_1, a_1, 1,$ terminal)
- Let $\gamma\ = 0.9$ practice on your own

First visit MC estimate of v of each state after this episode?

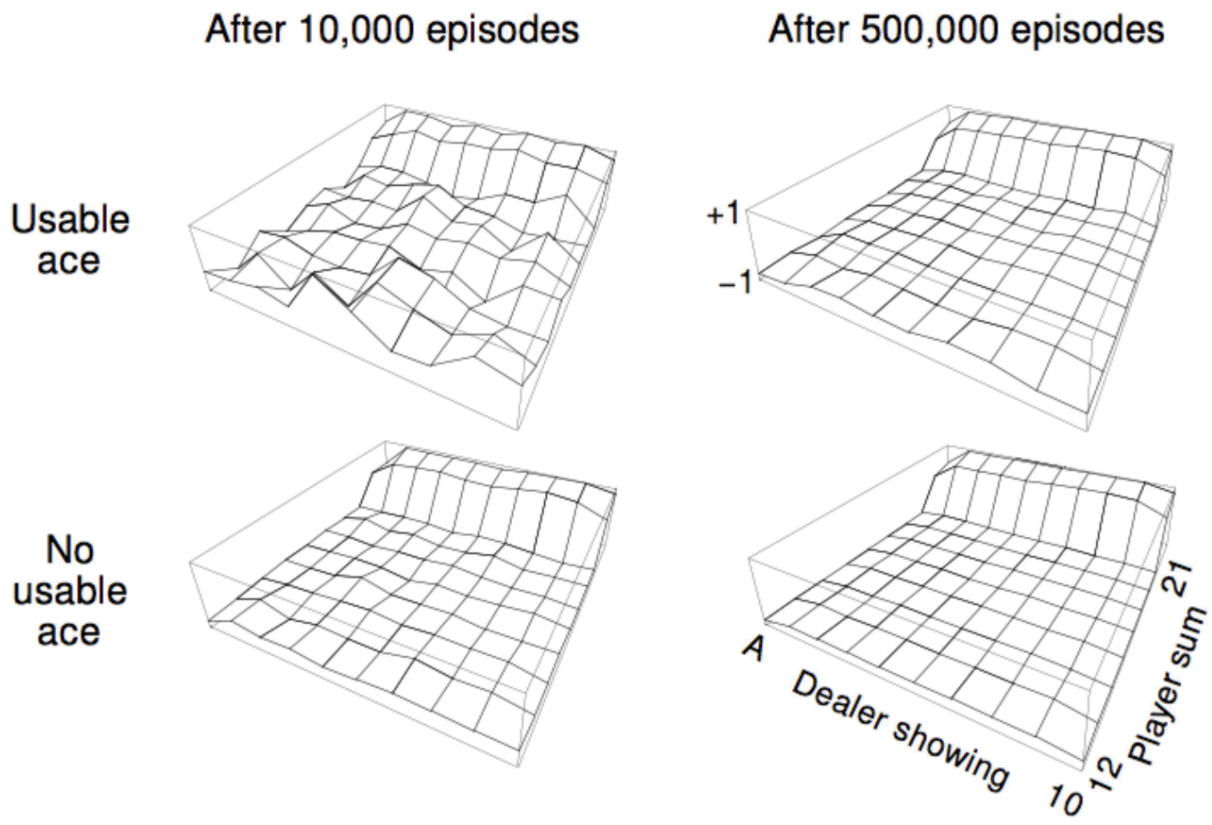
Every visit MC estimates of s_2?

Example: Blackjack

- States (200 of them): Current

  sum (12-21)
  - Dealer's showing card (ace-10)
  - Do I have a "useable" ace? (yes-no)
- Action stick: Stop receiving cards (and terminate)
- Action twist: Take another card (no replacement)
- Reward for stick:
  - +1 if sum of cards > sum of dealer cards
  - 0 if sum of cards = sum of dealer cards
  - -1 if sum of cards < sum of dealer cards
- Reward for twist:
  - -1 if sum of cards > 21 (and terminate)
  - 0 otherwise
- Transitions: automatically twist if sum of cards < 12

Example: Blackjack



Policy: stick if sum of cards ≥ 20, otherwise twist

Incremental Mean Calculation

The mean $\mu_1, \mu_2, \ldots$ of a sequence $x_1, x_2, \ldots$ can be computed incrementally,

$$\begin{aligned}
\mu_k &= \frac{1}{k} \sum_{j=1}^{k} x_j \\
&= \frac{1}{k} \left(x_k + \sum_{j=1}^{k-1} x_j \right) \\
&= \frac{1}{k} \left(x_k + (k-1)\mu_{k-1} \right) \\
&= \mu_{k-1} + \frac{1}{k} \left(x_k - \mu_{k-1} \right)
\end{aligned}$$

Incremental MC (On) Policy Evaluation

Initialize $N(s) = 0, G(s) = 0 \ \forall s \in \mathcal{S}$

Loop:

- Sample episode i: $s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \dots, s_{i,T_i}$
- Define $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \cdots + \gamma^{T-1} r_{i,T_i}$ as return from time step t onwards in i-th episode
- For each state s visited in episode i:
  - For every time t that state s is visited in episode i:
    - Increment counter of total visits: $N(s) = N(s) + 1$
    - Update estimate $v^\pi(s) = v^\pi(s) + \frac{1}{N(s)}(G_{i,t} - v^\pi(s))$

Incremental MC (On) Policy Evaluation

Initialize $N(s) = 0, G(s) = 0 \ \forall s \in \mathcal{S}$

Loop:

- Sample episode i: $s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \dots, s_{i,T_i}$
- Define $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \cdots + \gamma^{T-1} r_{i,T_i}$ as return from time step t onwards in i-th episode
- For each state s visited in episode i:
  - For every time t that state s is visited in episode i:
    - Increment counter of total visits: $N(s) = N(s) + 1$
    - Update estimate $v^\pi(s) = v^\pi(s) + \alpha(G_{i,t} - v^\pi(s))$

$\alpha = \frac{1}{N(s)}$: Identical to every visit MC

$\alpha > \frac{1}{N(s)}$: forget older data, helpful for non-stationary domains

MC Summary

- Generally high variance estimator
  - Reducing variance can require a lot of data
  - In cases where data is very hard or expensive to acquire, or the stakes are high, MC may be impractical
- Requires episodic settings
  - Episode must end before data from episode can be used to update v

Today’s Agenda

1. Last Lecture Review

2. Model Free Policy Evaluation (Monte Carlo)

3. **Model Free Policy Evaluation (Temporal Difference)**

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 Policy Evaluation

- Aim: estimate $v^\pi(s)$ given episodes generated under policy π
- $G_t = rt + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$ under policy π
- $v^\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$
- Recall Bellman operator (if know MDP models):

$$\mathfrak{B}^\pi v(s) = R(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} P_{s,s'}^{\pi(s)} v(s')$$

- In incremental every-visit MC, update estimate using:

$$v^\pi(s_t) = v^\pi(s_t) + \alpha(G_{i,t} - v^\pi(s_t))$$

- Insight: have an estimate of v, use to estimate expected return

$$v^\pi(s_t) = v^\pi(s_t) + \alpha([r_t + \gamma v^\pi(s_{t+1})] - v^\pi(s_t))$$

TD(0) Policy Evaluation

- Aim: estimate $v^\pi(s)$ given episodes generated under policy π
- $G_t = rt + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$ under policy π
- $v^\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$
- Simplest temporal-difference learning algorithm: TD(0)
  - Update value $v^\pi(s_t)$ toward estimated return $r_t + \gamma v^\pi(s_{t+1})$

$$v^\pi(s_t) = v^\pi(s_t) + \alpha([r_t + \gamma v^\pi(s_{t+1})] - v^\pi(s_t))$$

- $r_t + \gamma v^\pi(s_{t+1})$ is called the ***TD target***
- $\delta_t = r_t + \gamma v^\pi(s_{t+1}) - v^\pi(s_t)$ is called the ***TD error***
- Can immediately update value estimate after (s, a, r, s') tuple
- Don't need episodic setting

TD(0) Policy Evaluation Algorithm

Input: α
Initialize $v^{\pi}(s) = 0, \forall s \in \mathcal{S}$
Loop

- Sample tuple (s_t, a_t, r_t, s_{t+1})
- $v^{\pi}(s_t) = v^{\pi}(s_t) + \alpha([r_t + \gamma v^{\pi}(s_{t+1})] - v^{\pi}(s_t))$

TD(0) Policy Evaluation Algorithm Example

Input: α
Initialize $v^{\pi}(s) = 0, \forall s \in \mathcal{S}$
Loop
- Sample tuple (s_t, a_t, r_t, s_{t+1})
- $v^{\pi}(s_t) = v^{\pi}(s_t) + \alpha([r_t + \gamma v^{\pi}(s_{t+1})] - v^{\pi}(s_t))$

s_1	s_2	s_3	s_4	s_5	s_6	s_7
						

- R = [1 0 0 0 0 0 +10] for any action
- $\pi(s) = a_1 \; \forall s, \gamma = 1$. Any action from s1 and s7 terminates episode
- Sample episode = $(s_3, a_1, 0, s_2, a_1, 0, s_2, a_1, 0, s_1, a_1, 1,$ terminal)
- First/every visit MC estimate of v of each state? [1,1,1,0,0,0,0]
- TD estimate of all states (init at 0) with $\alpha = 1$?

Next time… Model free control