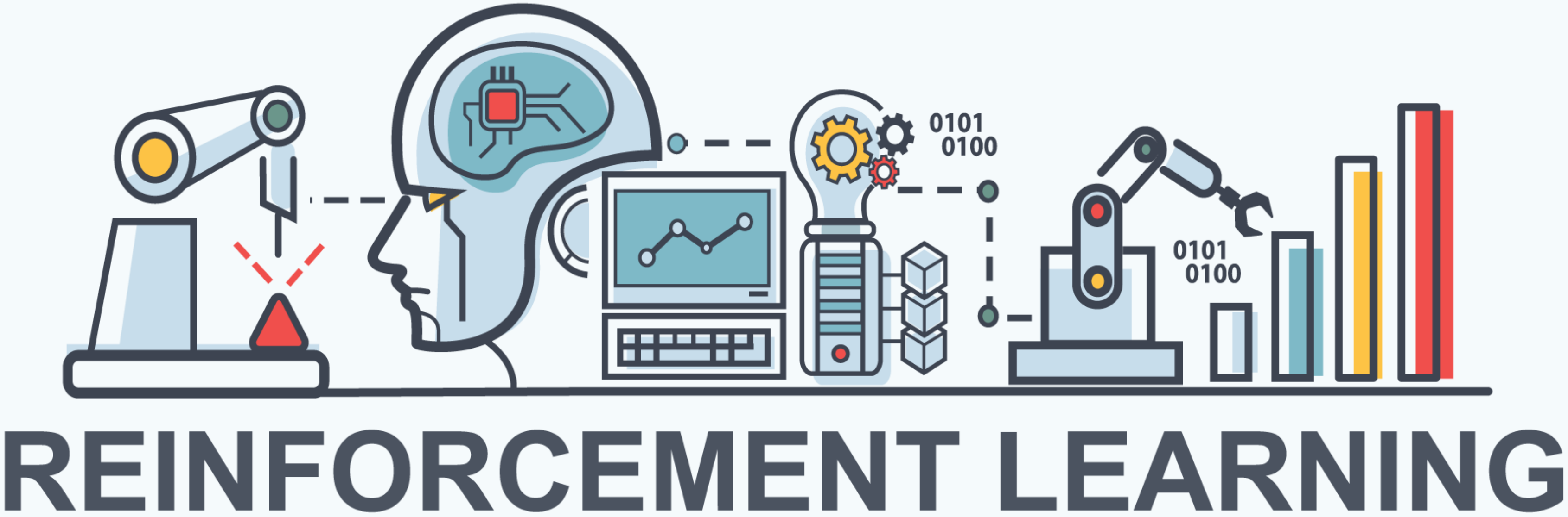


CP8319/CPS824
Lecture 6

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. Value Iteration

3. Model Free RL

Markov Decision Process (MDP)

A Markov decision process (MDP) is a Markov reward process with decisions/actions

Definition

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

- $\mathcal{S}$ is a (finite) set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathbf{P}$ is dynamics/transition model for each action,

$$P_{s,s'}^{a} = P\left(S_{t+1} = s' \mid S_t = s, A_t = a\right)$$

- R is the reward function, $R(s, a) = \mathbb{E}[r_t \mid S_t = s, A_t = a]$
- γ is a discount factor, $\gamma \in [0, 1]$

Optimal Value Function

Definition

The *optimal state-value function* $v^*(s)$ is the maximum value function over all policies

$$v^*(s) = \max_{\pi} v^{\pi}(s)$$

The *optimal action-value function* $Q^*(s,a)$ is the maximum action-value function over all policies

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a)$$

- The optimal value function specifies the best possible performance in the MDP.
- An MDP is “solved” when we know the optimal value fn.

Optimal Policy

Define a partial ordering over policies

$$\pi \geq \pi' \; if \; v^{\pi}(s) \geq v^{\pi'}(s), \forall s$$

Theorem

For any Markov Decision Process

- *There exists an optimal policy π^* that is better than or equal to all other policies, $\pi^* \geq \pi, \forall \pi$*
- *All optimal policies achieve the optimal value function,* $v^{\pi^*}(s) = v^*(s)$, or $\pi^* = \underset{\pi}{\operatorname{argmax}}\, v^{\pi}(s)$
- *All optimal policies achieve the optimal action-value function,* $Q^{\pi^*}(s,a) = Q^*(s,a)$

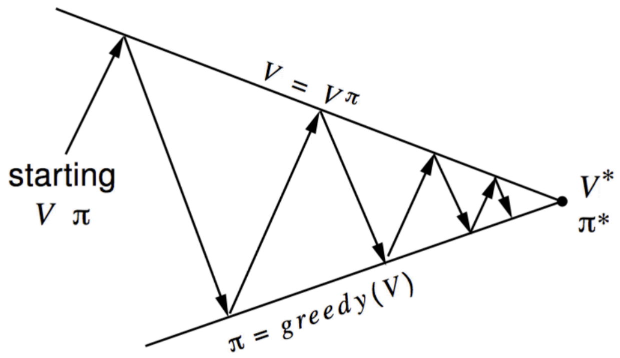
Policy Iteration

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**
- $i = i + 1$

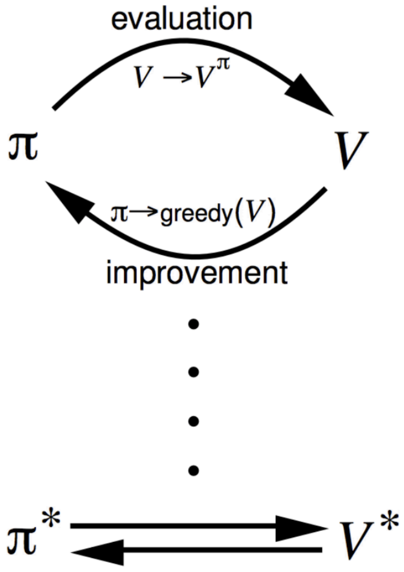


Policy evaluation Estimate v_π

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 $s \in \mathcal{S}$:

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

This is known as Bellman expectation backup

Policy Improvement

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

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

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)

Today’s Agenda

1. Last Lecture Review
2. **Value Iteration**
3. Model Free RL

Policy and Value Iteration

Policy iteration computes optimal value and policy
- Assumes for a given policy we know the value function (over infinite horizon)

Value iteration is another technique. Idea:
- Maintains optimal value of starting in a state s if have a finite number of steps k left in the episode
- Iterate to consider longer and longer episodes

Policy of Optimality

Any optimal policy can be subdivided into two components:

An optimal first action a^*

Followed by an optimal policy from successor state s'

Theorem (Principle of Optimality)

A policy $\pi(a|s)$ *achieves the optimal value from state* s, $v^\pi(s) = v^*(s)$ *if and only if* :

for any state s' *reachable from* s, π *achieves the optimal value from state* s' , $v^\pi(s') = v^*(s')$

Bellman Equation and Bellman Backup Operators

Value function of a policy must satisfy the Bellman equation

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

Bellman backup operator $\mathfrak{B}$:

- Applied to a value function
- Returns a new value function
- Improves the value if possible

$$\mathfrak{B}(v(s)) \xleftarrow{\text{short form}} \mathfrak{B}v(s) = \max_{a \in \mathcal{A}} \left(R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P_{s,s'}^{a} v(s') \right)$$

$\mathfrak{B}v$ yields a value function over all states s

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,

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

Policy Iteration as Bellman Operation

- Bellman backup operator $\mathfrak{B}^{\pi}$ for a particular policy is defined as

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

- Policy evaluation amounts to computing the fixed point of $\mathfrak{B}^{\pi}$ (i.e., when the Bellman operation does not change the $v(s)$)

- To do policy evaluation, repeatedly apply operator until v stops changing

$$v^{\pi} = \mathfrak{B}^{\pi}\mathfrak{B}^{\pi}\mathfrak{B}^{\pi} \cdots \mathfrak{B}^{\pi} v$$

Policy Iteration Using Bellman Backup

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):

- $v^{\pi_i} = \mathfrak{B}^{\pi_i}\mathfrak{B}^{\pi_i}\mathfrak{B}^{\pi_i} \cdots \mathfrak{B}^{\pi_i} v$
- $\pi_{i+1}(s) = \arg\max_{a \in \mathcal{A}} \left(R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P^a_{s,s'} v^{\pi_i}(s') \right)$
- $i = i + 1$

Does Value Iteration Converge?

Contraction Operator:

- Let $\mathfrak{O}$ be an operator, and $||x||$ denote (any) norm of x
- If $||\mathfrak{O}x - \mathfrak{O}x'|| \leq ||x - x'||$, then $\mathfrak{O}$ is a contraction operator
  - That is the operator reduces the distance between x and x'

Does Value Iteration Converge?

- Yes, if discount factor $\gamma < 1$, or end up in a terminal state with probability 1
- Bellman backup is a contraction if discount factor, $\gamma < 1$
- If apply it to two different value functions, distance between value functions shrinks after applying Bellman equation to each

Proof that Bellman Backup is Contraction on v

Let $\|v - v'\| = \max_s |v(s) - v'(s)|$ be the infinity norm

$$\left\|\mathfrak{B}v_k - \mathfrak{B}v_j\right\| = \left\|\max_{a\in\mathcal{A}}\left(R(s,a) + \gamma \sum_{s'\in\mathcal{S}} P^a_{s,s'} v_k(s')\right) - \max_{a'\in\mathcal{A}}\left(R(s,a') + \gamma \sum_{s'\in\mathcal{S}} P^{a'}_{s,s'} v_j(s')\right)\right\|$$

$$\leq \max_{a\in\mathcal{A}}\left\|R(s,a) + \gamma \sum_{s'\in\mathcal{S}} P^a_{s,s'} v_k(s') - R(s,a) - \gamma \sum_{s'\in\mathcal{S}} P^a_{s,s'} v_j(s')\right\|$$

$$= \max_{a\in\mathcal{A}}\left\|\gamma \sum_{s'\in\mathcal{S}} P^a_{s,s'} (v_k(s') - v_j(s'))\right\|$$

$$\leq \max_{a\in\mathcal{A}}\left\|\gamma \sum_{s'\in\mathcal{S}} P^a_{s,s'} \|v_k - v_j\|\right\|$$

$$= \gamma \left\|v_k - v_j\right\|$$

Value and Policy Iteration Summary

Value iteration:

- Compute optimal value for horizon $= k$
  - Note this can be used to compute optimal policy if horizon $= k$
- Increment k

Policy iteration:

- Compute infinite horizon value of a policy
- Use to select another (better) policy
- Closely related to a very popular method in RL: policy gradient

Today’s Agenda

1. Last Lecture Review

2. Value Iteration

3. **Model Free RL**

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 first visits: $N(s) = N(s) + 1$
    - Increment total return $G(s) = G(s) + G_{i,t}$
    - Update estimate $v^\pi(s) = G(s)/N(s)$

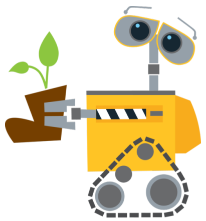
Every-Visit MC (On) Policy Evaluation

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 first 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

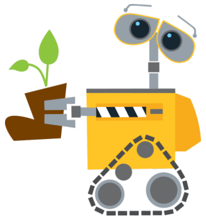
s_1	s_2	s_3	s_4	s_5	s_6	s_7
						

- 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?