UC San Diego
JACOBS SCHOOL OF ENGINEERING

ECE/SIOC 228 Machine Learning for Physical Applications
Lecture 13: Markov Decision Process II

Yuanyuan Shi

Assistant Professor, ECE

University of California, San Diego

Outline

UC San Diego

- **Markov Decision Process**
- Value Iteration
- Policy Iteration

Markov Decision Process

UC San Diego

- A Markov Decision Process in a tuple $< S, A, P, R, \gamma >$
  - S is a finite set of states
  - A is a finite set of actions
  - P is a state transition probability matrix, $P_{ss'}^{a} = P[S_{t+1} = s' | S_t = s, A_t = a]$
  - R is a reward function, $R_s^a = R(R_{t+1} | S_t = s, A_t = a)$
  - γ is a discount factor $\gamma \in [0, 1]$

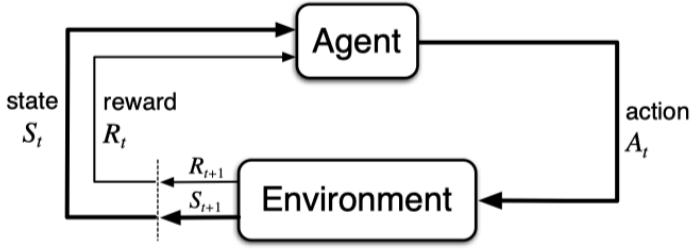
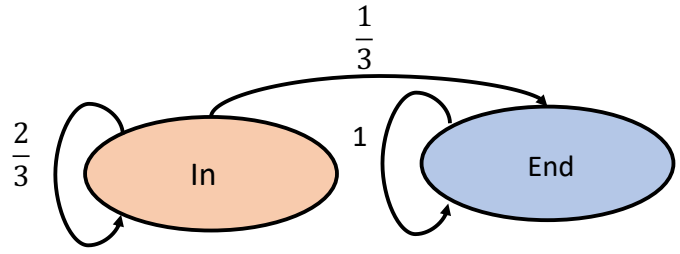


Figure 3.1: The agent–environment interaction in a Markov decision process.

Markov Decision Process Example

UC San Diego



Dice Game Markov Chain

For each round $r = 1,2, \ldots$

- You choose stay or quit
- If quit, you get $10 and the game ends
- If stay, you get $4 and then we roll a 6-sided dice
  - If the dice results in 1 or 2, game ends
  - Otherwise, game continues to the next round

State: in/end

action: stay, quit

reward: $R(in, quit) = \$10$, $R(in, stay) = \$4$

transition: $P_{ss'}^{a} = P(S'|S, a)$

Markov Decision Process Examples

UC San Diego

State: angle of joints, position of the manipulator, & derivatives (velocity and acceleration)

Action (control demand): torque or velocity commands

Reward:
- whether accomplishment a task.
- distance to the object

Policy

UC San Diego

- **A policy π** is a mapping from each state $s \in S$ to an action $a \in A$

$$\pi[a|s] = P[A_t = a|S_t = s]$$

- A policy full defines the behavior of an agent
- In MDP, policies only depend on the current state S_t, not the history
- Policy is stationary (time independent), $A_t \sim \pi(\cdot\,|S_t), \forall t > 0$
- Policy can be both stochastic and deterministic

Example: policy "stay" // policy "quit" // policy "50% stay, 50% quit"

Episode

UC San Diego

- **An episode** is a sequence of states, actions and rewards with a terminal state

$$S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}, R_{t+2}, S_{t+2}, A_{t+2}, R_{t+3}, S_{t+4}, \ldots..$$

Sample **episodes** starting from $S_1 = \text{In}$

- [In (stay), End] $4
- [In (stay), In (stay), In (stay), End] $12
- [In (stay), In (stay), End] $8
- [In (stay), In (stay), In (stay), In (stay), End] $16
- ……

- Episode with finite length (episodic) v.s. infinite length (continuing tasks)
- Episodes are independent from each other

Return

UC San Diego

- **The return G_t** is the total discounted reward from time step t

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- The discount $\gamma \in [0, 1]$ is the present value of future rewards
- The value of receiving reward R after $k + 1$ time-steps is $\gamma^k R$
- This values immediate reward above delayed reward
  - γ close to 0 leads to "greedy" evaluation (only care about current step)
  - γ close to 1 leads to "far-sighted" evaluation

Return

UC San Diego

- **The return G_t** is the total discounted reward from time step t

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

For each round r = 1,2, …

- You choose stay or quit
- If quit, you get $10 and the game ends
- If stay, you get $4 and then we roll a 6-sided dice
  - If the dice results in 1 or 2, game ends
  - Otherwise, game continues to the next round

save for the future

Discount $\gamma = 1$: [In (stay), In (stay), In (stay), In (stay), End] $4 + 4 + 4 + 4 = \$16$

Discount $\gamma = 0$: [In (stay), In (stay), In (stay), In (stay), End] $4 + 0 \cdot 4 + 0 \cdot 4 + 0 \cdot 4 = \$4$ ← live in the moment

Discount $\gamma = 1/2$: [In (stay), In (stay), In (stay), In (stay), End] $4 + 0.5 \cdot 4 + 0.5^2 \cdot 4 + 0.5^3 \cdot 4 = \$7.5$

balanced life

Why Discount?

UC San Diego

- Mathematically convenient to use discount rewards
- Animal/human behavior shows preference for immediate reward
- If the reward is financial, immediate rewards may earn more interest than delayed rewards
- Sometimes, it is possible to use undiscounted reward if all possible episodes are finite
- ……

Value function

UC San Diego

The **state-value function $v_\pi(s)$ of a policy π** in an MDP is the expected return starting from state s, and then following policy π

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$$

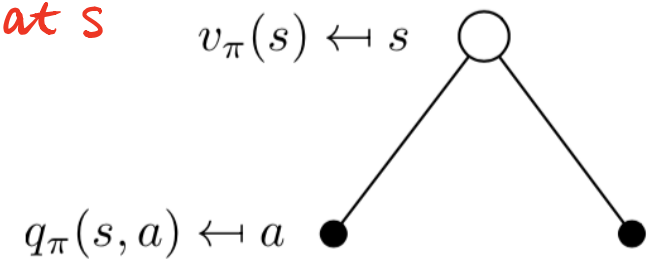
The **state-action value function $q_\pi(s, a)$ of a policy π** (sometimes also refer to as action value function) is the expected return starting from state s, taking action a, and then following policy π

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$$

- Example. 2 actions at s

$\pi(a_1|s) = 0.5$

$\pi(a_2|s) = 0.5$



$V_\pi(s) = 0.5 \cdot q_\pi(s, a_1)$

$+0.5\, q_\pi(s, a_2)$

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a)$$

$= \sum_{a \in A} \pi(a|s)\, q_\pi(s,a)$

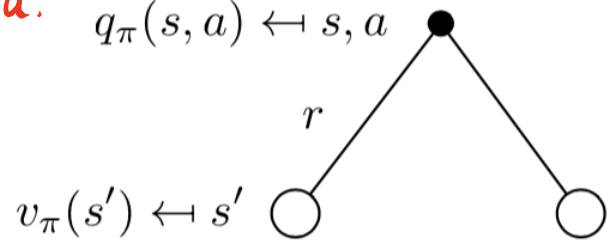
$= E_{a \sim \pi(a|s)}[\, q_\pi(s,a)]$

Value function

UC San Diego

Example:

At state S, take action a.



$P(S'=1 \mid S,a) = 0.8$

$P(S'=2 \mid S,a) = 0.2$

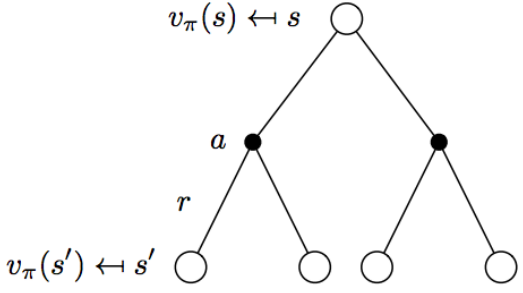
$q_\pi(S,a) = R_S^a + \gamma[0.8\, V_\pi(1) + 0.2\, V_\pi(2)]$

$= R_S^a + \gamma \sum_{S' \in S} P_{SS'}^a V_\pi(S')$ $\quad q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$

$= R_S^a + \gamma\, E_{S' \sim P(S' \mid S,a)}[V_\pi(S')]$

Value function

UC San Diego



$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

$q_\pi(s,a)$

$$= E_{a \sim \pi(a|s)} \left[R_s^a + \gamma E_{s' \sim P_{ss'}^a} v_\pi(s') \right]$$

Value function

UC San Diego

The **state-value function $v_\pi(s)$ of a policy π** in an MDP is the expected return starting from state s, and then following policy π

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$$

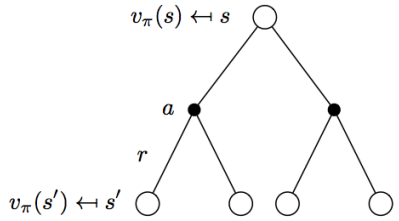
The **state-action value function $q_\pi(s, a)$ of a policy π** (sometimes also refer to as action value function) is the expected return starting from state s, taking action a, and then following policy π

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$$

Policy Evaluation

UC San Diego

- **Given a policy π, how can we evaluate it, i.e., compute $v_\pi(s)$ and $q_\pi(s,a)$?**
- Solve a linear system



$$v_\pi = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi v_\pi$$

where $R^\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|s)\, R_s^a$, $\mathcal{P}^\pi(s'|s) = \sum_{a\in\mathcal{A}} \pi(a|s)\, P(s'|s,a)$

$$\begin{bmatrix} v_\pi(1) \\ v_\pi(2) \\ \vdots \\ v_\pi(n) \end{bmatrix} = \begin{bmatrix} R^\pi(1) \\ R^\pi(2) \\ \vdots \\ R^\pi(n) \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}^\pi_{11} & \mathcal{P}^\pi_{12} & \cdots & \mathcal{P}^\pi_{1n} \\ \mathcal{P}^\pi_{21} & \mathcal{P}^\pi_{22} & \cdots & \mathcal{P}^\pi_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{P}^\pi_{n1} & \mathcal{P}^\pi_{n2} & \cdots & \mathcal{P}^\pi_{nn} \end{bmatrix} \begin{bmatrix} v_\pi(1) \\ v_\pi(2) \\ \vdots \\ v_\pi(n) \end{bmatrix}$$

$$v_\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

$= \sum_{a\in A} \pi(a|s)\, R_s^a + \gamma \sum_{a\in A} \pi(a|s) \sum_{s'\in S} P_{ss'}^a V_\pi(s')$

Policy Evaluation

UC San Diego

- The state value function of a policy π satisfies the following equation:

$$v_\pi = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi v_\pi$$

- We can solve it the above linear equation directly,

$$(I - \gamma \mathcal{P}^\pi) v_\pi = \mathcal{R}^\pi$$

$$v_\pi = (I - \gamma \mathcal{P}^\pi)^{-1} \mathcal{R}^\pi$$

- Direct solution only possible for small MDPs. High computational complexity when state space is large
- **Iterative methods for large MDPs,**

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

$$v_\pi^{(k+1)} = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi v_\pi^{(\mathrm{k})}, \quad v_\pi^{(1)} \to v_\pi^{(2)} \to \cdots \to v_\pi \text{ (convergence)}$$

Summary

- Markov Decision Process $< S, A, P, R, \gamma >$
  - State
  - Action
  - Transition Probability
  - Reward (and discount factor)
- Policy
- Return
- Value function: state value function $v_\pi(s)$, action value function $q_\pi(s, a)$

Outline

UC San Diego

- Markov Decision Process
- **Value Iteration**
- Policy Iteration

Optimal Value Function

UC San Diego

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

$$v_*(s) = \max_\pi v_\pi(s)$$

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

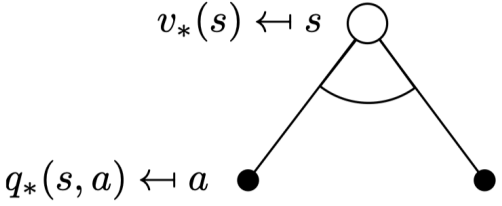
$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

- An MDP is "solved" when we know the optimal value function
- Notice if we have $q_*(s,a)$, we can obtain the optimal policy π_*: $\pi_*(s) = \arg\max_{a \in \mathcal{A}} q_*(s,a), \forall s \in S$
- **Value iteration and policy iteration are two ways to "solve" the MDP, a.k.a. find the optimal policy (or equivalently, find the optimal value function)**

Bellman Optimality Equations

UC San Diego

- The optimal value functions are recursively related by the Bellman optimality equations:

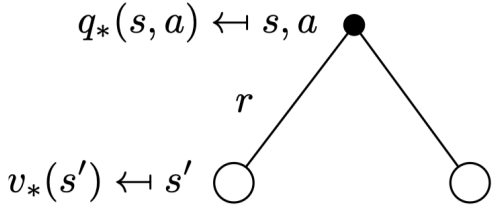


$$v_*(s) = \max_a q_*(s,a)$$

Bellman Optimality Equations

UC San Diego

- The optimal value functions are recursively related by the Bellman optimality equations:

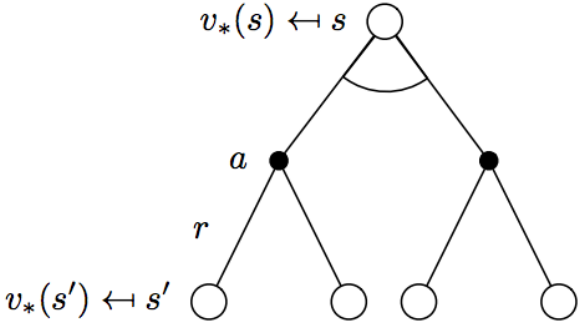


$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

Bellman Optimality Equations

UC San Diego

- The optimal value functions are recursively related by the Bellman optimality equations:

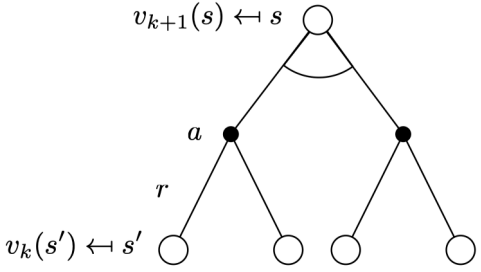


$$V_*(s) = \max_a \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_*(s') \right)$$

Value Iteration

UC San Diego

- Goal: find the optimal value function
- Solution: iterative application of the Bellman optimality equation
- $v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_*$



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

Convergence of Value Iteration

UC San Diego

Theorem. Value iteration converges to v_*. At convergence,

$$\forall s \in S, v_*(s) = \max_{a \in \mathcal{A}} \left(R_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s') \right)$$

- Proof is left as Homework (HW2 Q3)
- By first showing the Bellman optimality operator $Bell(v) = \max_{a \in \mathcal{A}} \left(R_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v(s') \right)$ is a contraction mapping.
- Show the fixed point is unique.
- We can also derive the optimal policy from the optimal value function v_*

Illustration of Value Iteration: Deterministic Environment

UC San Diego

- Simple grid world with a "goal state" with reward 10 and a "bad state" with reward -10. Move 1 step incurs reward -1.
- Both the "goal state" and "bad state" are terminal state, value of terminal state equal to 0 (future expected return equal to 0).
- Actions at each state: *up, down, left, right*. Taking an action that would bump into a wall leaves agent where it is.
- Discount factor $\gamma = 0.9$

$|S| = 9$

$|A| = 4$

0	0	10
0	0	-10
0	0	0

Original reward function

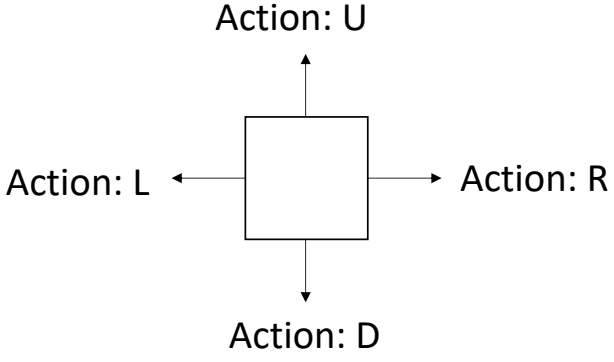


Illustration of Value Iteration

UC San Diego

0	0	0
0	0	0
0	0	0

Initialize value function: $v_0(s)$

-1	9	0
-1	-1	0
-1	-1	-1

Value function after 1 iteration: $v_1(s)$

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

Illustration of Value Iteration

UC San Diego

7.1	9	0
-1.9	7.1	0
-1.9	-1.9	-1.9

$v_2(s)$

7.1	9	0
5.39	7.1	0
-2.71	5.39	-2.71

$v_3(s)$

7.1	9	0
5.39	7.1	0
3.85	5.39	3.85

$v_4(s)$

7.1	9	0
5.39	7.1	0
3.85	5.39	3.85

$v_5(s)$

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

Illustration of Value Iteration

UC San Diego

$a = \text{argmax}_{a \in [U, D, L, R]}$

7.1	9	0
5.39	7.1	0
3.85	5.39	3.85

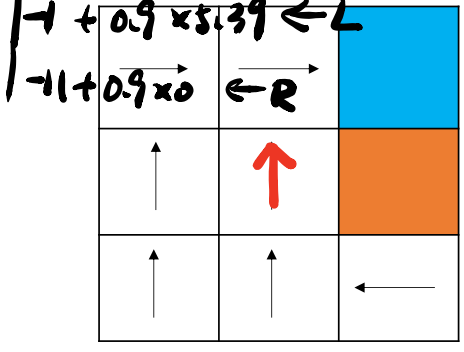
$v_5(s)$

$-1 + 0.9 \times 9 \leftarrow U$ ✓

$-1 + 0.9 \times 5.39 \leftarrow D$

$-1 + 0.9 \times 5.39 \leftarrow L$

$-1 + 0.9 \times 0 \leftarrow R$



Best policy based on $v_5(s)$

An optimal policy can be found by maximizing over $q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_*(s')$

$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \underset{a \in \mathcal{A}}{\text{argmax}}\, q_*(s,a) \\ 0 & \text{otherwise} \end{cases}$$

Outline

UC San Diego

- Markov Decision Process
- Value Iteration
- **Policy Iteration**

Policy Iteration

UC San Diego

- Given a policy π
  - Evaluate the policy $\pi \rightarrow v_\pi(s)$ $v_\pi = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi v_\pi$
  - Improve the policy by acting greedily with respect to v_π:

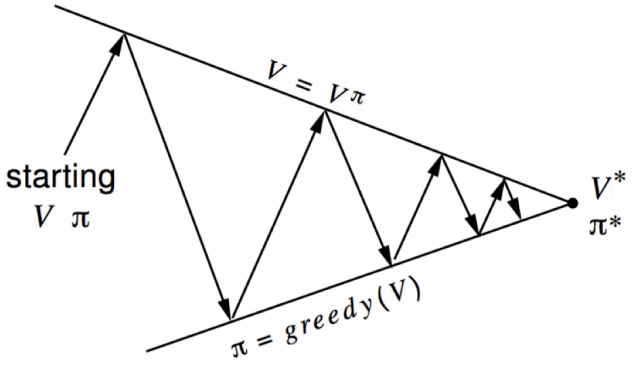
$$\pi'(s) \leftarrow \underset{a \in \mathcal{A}}{\text{argmax}} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in S} P(s'|s,a) v_\pi(s') \right) = \underset{a \in \mathcal{A}}{\text{argmax}}\, q_\pi(s,a)$$

Policy Iteration Algorithm:

1. Initialize policy π (e.g., randomly)
2. Perform policy evaluation
3. Perform policy improvement , obtain policy π'
4. If policy changed in last iteration, return to Step 2

Policy Iteration

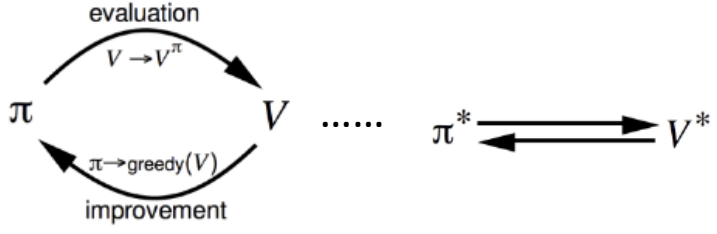
UC San Diego



Policy evaluation Estimate v_π
Iterative policy evaluation

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

- Convergence property of policy iteration: $\pi \rightarrow \pi^*$
- Proof involves showing that each policy evaluation is a contraction and policy must improve each step, or be optimal policy (policy improvement theorem)



Policy Improvement Theorem

UC San Diego

- Consider a deterministic policy, $a = \pi(s)$
- We can improve the policy by acting greedily

$$\pi'(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, q_\pi(s, a)$$

- This improves the value from any state s over one step,

$$q_\pi\big(s, \pi'(s)\big) = \max_{a \in \mathcal{A}} q_\pi(s, a) \geq q_\pi\big(s, \pi(s)\big) = v_\pi(s)$$

- It therefore improves the value function,

$$\begin{aligned} v_\pi(s) \leq q_\pi(s, \pi'(s)) &= \mathbb{E}_{p,\pi'}\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s\right] \\ &\leq \mathbb{E}_{p,\pi'}\left[R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) \mid S_t = s\right] \\ &\leq \mathbb{E}_{p,\pi'}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 q_\pi(S_{t+2}, \pi'(S_{t+2})) \mid S_t = s\right] \\ &\leq \mathbb{E}_{p,\pi'}\left[R_{t+1} + \gamma R_{t+2} + \cdots \mid S_t = s\right] = v_{\pi'}(s) \end{aligned}$$

Policy Improvement Theorem

UC San Diego

- If improvements stop,

$$q_\pi\big(s, \pi'(s)\big) = \max_{a \in \mathcal{A}} q_\pi(s, a) = q_\pi\big(s, \pi(s)\big) = v_\pi(s)$$

- The Bellman optimality equation has been satisfied,

$$v_\pi(s) = \operatorname*{argmax}_{a \in \mathcal{A}} q_\pi(s, a)$$

- Therefore, $v_\pi(s) = v_*(s)$ for all $s \in S$

- So π is am optimal policy

Illustration of Policy Iteration

UC San Diego

0	0	10
0	0	-10
0	0	0

Original reward function

- Initialize with a policy $\pi(s) = [\mathrm{U}, \mathrm{D}, \mathrm{L}, \mathrm{R}]$ with prob 0.25

Illustration of Policy Iteration

UC San Diego

-5.78 →	-1.97 →	0
-7.7 ↑	-7.69 ↑	0
-8.62 ↑	-8.93 ↑	-10.02 ←

v_π of the initial random policy

- Initialize with a policy $\pi(s) = [\mathrm{U}, \mathrm{D}, \mathrm{L}, \mathrm{R}]$ with prob 0.25
- Perform policy evaluation of policy π
- Perform policy improvement

$$\pi'(s) \leftarrow \underset{a \in \mathcal{A}}{\operatorname{argmax}} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in S} P(s'|s,a) v_\pi(s') \right)$$

- **Find the optimal policy in 1 iteration!**
- Can also compute optimal value function by running policy evaluation of π_*

7.1	9	0
5.39	7.1	0
3.85	5.39	3.85

v_{π_*}

Policy Iteration or Value Iteration?

UC San Diego

- Policy iteration requires fewer iterations that value iteration, but each iteration requires solving a linear system instead of just applying Bellman operator

- For small MDPs, policy iteration is often faster

- For MDPs with large state spaces, solving for v_π explicitly would involve solving a large system of linear equations, and could be difficult. In these problems, value iteration may be preferred.

What is next?

UC San Diego

Today's lecture

- **Planning** in Markov Decision Process (offline)
  - Have the environment model (reward, transition)
  - Find the optimal policy via value iteration and policy iteration

Next lectures

- **Reinforcement Learning** in Markov Decision Process (online)
  - Don't know how the environment works
  - Find the optimal policy by interacting with the environment (taking actions and collect reward)