CSCI 699: Robotic Perception

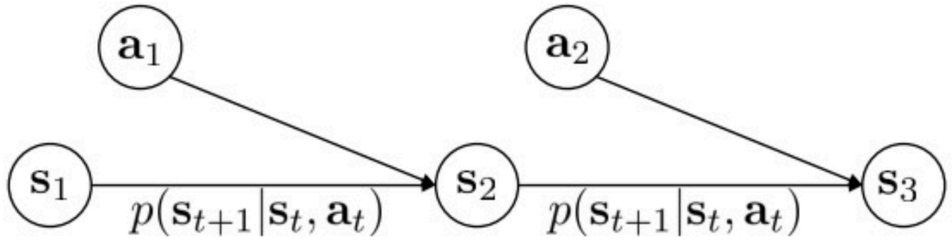
Yue Wang

October 1st, 2025

Notations, MDP, POMDP, return, value functions

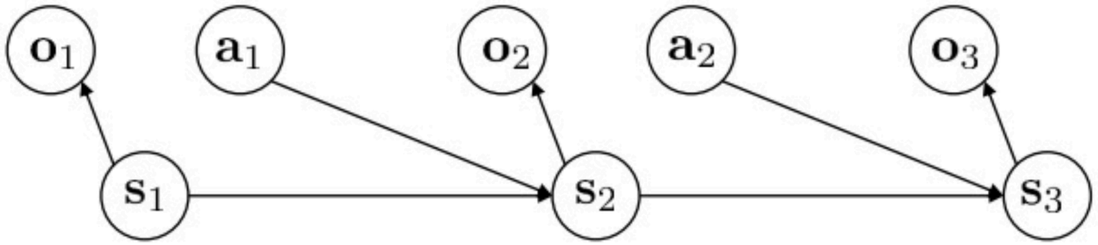
MDP

- Definitions
  - S is the state space. $s_t \in S$ is the state at time step t.
  - A is the action space. $a_t \in A$ is the action at time step t.
  - O is the observation space. $o_t \in O$ is the observation at time step t.
  - p is the one-step transition probability (a.k.a, dynamics): $s_{t+1} \sim p(\cdot \mid s_t, a_t)$.
  - h is the observation model: $o_t \sim h(\cdot \mid s_t)$
  - $r : S \times A \to \mathbb{R}$ is the reward function.
- A Markov decision process (MDP) is a tuple (S, A, p, r)
  - Goal: learn a policy $\pi_\theta(a_t \mid s_t)$



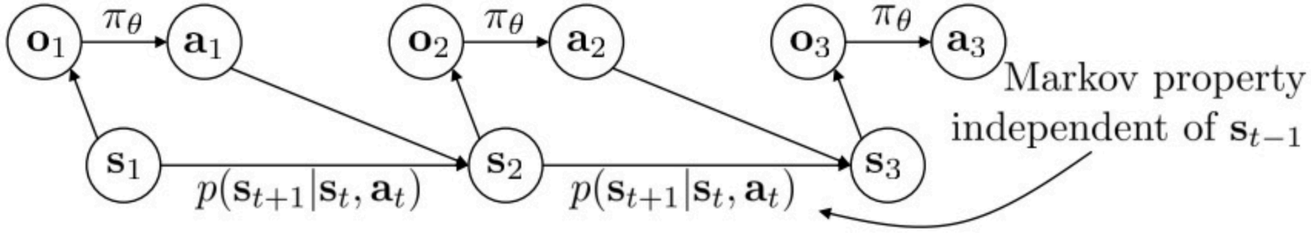
POMDP

- Definitions
  - S is the state space. $s_t \in S$ is the state at time step t.
  - A is the action space. $a_t \in A$ is the action at time step t.
  - O is the observation space. $o_t \in O$ is the observation at time step t.
  - p is the one-step transition probability (a.k.a, dynamics): $s_{t+1} \sim p(\cdot \mid s_t, a_t)$.
  - h is the observation model: $o_t \sim h(\cdot \mid s_t)$
  - $r : S \times A \to \mathbb{R}$ is the reward function.
- A partially observed Markov decision process (POMDP) is a tuple (S, A, O, p, h, r)
  - Goal: learn a policy $\pi_\theta(a_t \mid o_t)$



POMDP

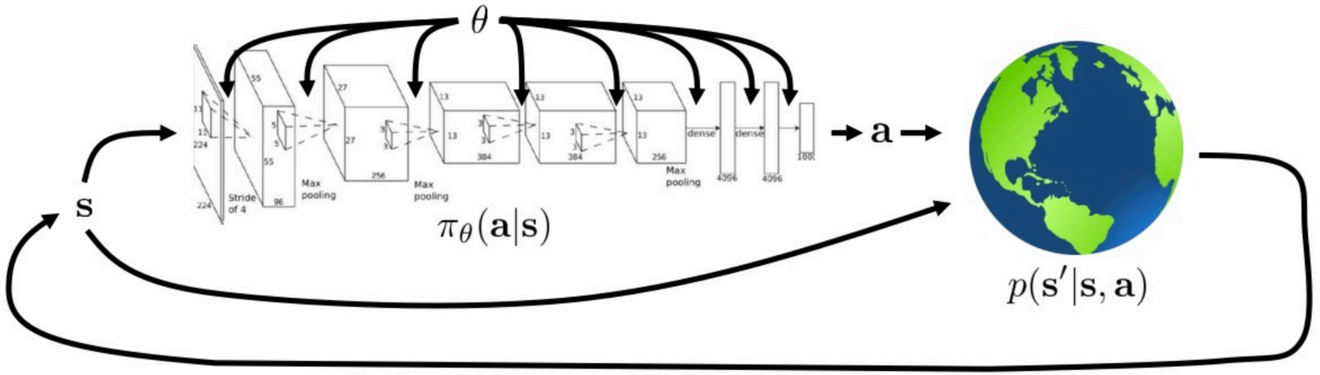
- Definitions
  - S is the state space. $s_t \in S$ is the state at time step t.
  - A is the action space. $a_t \in A$ is the action at time step t.
  - O is the observation space. $o_t \in O$ is the observation at time step t.
  - p is the one-step transition probability (a.k.a, dynamics):
    $s_{t+1} \sim p(\cdot \mid s_t, a_t)$.
  - h is the observation model: $o_t \sim h(\cdot \mid s_t)$
  - $r : S \times A \to \mathbb{R}$ is the reward function.
- Why Markov?
  - Allow us to throw away history once state is known!



Andrey Markov

The Goal of Reinforcement Learning and Control

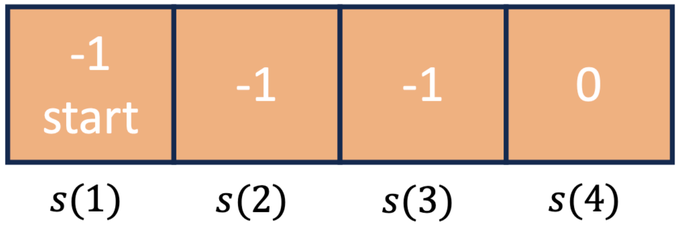
- Finite horizon case: T is finite
- Infinite horizon case: $T = \infty$
- The cumulative reward can be discounted: $\sum_t \gamma^t r(s_t, a_t)$ where $0 < \gamma \leq 1$
- Goal: find a policy to maximize the cumulative reward



$$\underbrace{p_\theta(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T)}_{\pi_\theta(\tau)} = p(\mathbf{s}_1) \prod_{t=1}^{T} \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

$$\theta^\star = \arg\max_\theta E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

Notion Explanation: 1D Grid World



- Four states: $s(1)$ (start), $s(2)$, $s(3)$, $s(4)$
- Three actions: left, right, stay
- $0 \leq \pi(a \mid s) \leq 1$ is π's probability of choosing a at state s
- Example (a smart deterministic policy): $\pi(a = \text{right} \mid s(2)) = 1$
- Example (a random policy): $\pi(a \mid s(2)) = 1/3$ for all actions
- $\pi(s)$: (slightly abuse the notation) A mapping from state to action!
- $0 \leq p(s' \mid s, a) \leq 1$ is probability of transiting to s' from s, with action a
- Example (deterministic system): $p(s(3) \mid s(2), a = \text{right}) = 1$
- Example (stochastic): $p(s(3) \mid s(2), a = \text{right}) = 0.9, p(s(2) \mid s(2), a = \text{right}) = 0.1$

Returns G_t

- Sum of future rewards: $G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$
- Episodic setting (finite horizon): $G_t = r_{t+1} + r_{t+2} + \ldots + r_T$
- The action sequence is $s_0, a_0, r_1, s_1, a_1, r_2, \ldots$
  - Some books/papers might use different index
- We will focus on the discounted infinite horizon case today

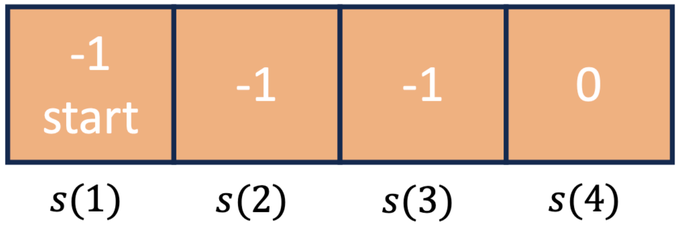
Returns G_t

- State-value function (V): the expected return starting from state s, and then following policy π: $V^{\pi}(s) = \mathbb{E}_{\pi}[G_0 \,|\, s_0 = s]$
- Action-value function (Q): the expected return starting from state s, taking action a, and then following policy π: $Q^{\pi}(s, a) = \mathbb{E}_{\pi}[G_0 \,|\, s_0 = s, a_0 = a]$
- Question: Why do we need both V and Q? What does $Q^{\pi}(s, a) > V^{\pi}(s)$ mean?

Optimal Value Functions

- Optimal V: $V^*(s) = \max_\pi V^\pi(s) = \max_\pi \mathrm{E}_\pi[G_0 \,|\, s_0 = s]$
- Optimal Q: $Q^*(s,a) = \max_\pi Q^\pi(s,a) = \max_\pi \mathrm{E}_\pi[G_0 \,|\, s_0 = s, a_0 = a]$
- Optimal policy: $\pi^*(s) = \mathrm{argmax}_\pi V^\pi(s)$
- Properties: $V^*(s) = \max_a Q^*(s,a)$ and $\pi^*(s) = \mathrm{argmax}_a Q^*(s,a)$
- If we know $Q^*(s,a)$, we immediately have the optimal policy!
  - If we only know $V^*(s)$, we need the dynamics to do one step lookahead to choose the optimal action

Example: 1D Grid World



- Four states: $s(1)$ (start), $s(2)$, $s(3)$, $s(4)$
- Three actions: left, right, stay
- Assume no discount and deterministic transitions
- $V^*(s(1)) = -3, V^*(s(2)) = -2, V^*(s(3)) = -1, V^*(s(4)) = 0$
- $Q^*(s(1), a = \text{stay}) = -4, Q^*(s(1), a = \text{right}) = -3$, so $\pi^*(s(1)) = \text{right}$

Policy Evaluation

Policy Evaluation

- Goal: Given some π (might not be optimal), compute its value functions: $V^{\pi}(s) = \mathbb{E}[G_0 \,|\, s_0 = s]$ and $Q^{\pi}(s, a) = \mathbb{E}_{\pi}[G_0 \,|\, s_0 = s, a_0 = a]$

- Recursive relationships for returns:
  - One-step reward + discounted next step's return

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} = r_{t+1} + \gamma(r_{t+2} + \gamma r_{t+3} \ldots) = r_{t+1} + \gamma G_{t+1}$$

Policy Evaluation

- Goal: Given some π (might not be optimal), compute its value functions: $V^\pi(s) = \mathbb{E}[G_0 \,|\, s_0 = s]$ and $Q^\pi(s,a) = \mathbb{E}_\pi[G_0 \,|\, s_0 = s, a_0 = a]$
- Leveraging the recursive relationship
- $\mathbb{E}_\pi[\dots]$ means averaging all randomness
  - Policy π is stochastic
  - Transition p is stochastic
- The third evaluation: we expand the expectation
- Simplify a bit (assume r is a deterministic function of state)

$$V^\pi(s) = \mathbb{E}_\pi[G_0 \,|\, s_0 = s]$$
$$= \mathbb{E}_\pi(r_1 + \gamma G_1 \,|\, s_0 = s]$$
$$= \sum_a \pi(a \,|\, s) \sum_{s',r} p(s', r \,|\, s, a)[r + \gamma V^\pi(s')]$$

$$V^\pi(s) = r(s) + \gamma \sum_a \pi(a \,|\, s) \sum_{s'} p(s' \,|\, s, a) V^\pi(s')$$

Received a reward at state s

Sample action a

Enter state s'

Expected future rewards

Policy Evaluation

- Goal: Given some π (might not be optimal), compute its value functions: $V^{\pi}(s) = \mathbb{E}[G_0 \,|\, s_0 = s]$ and $Q^{\pi}(s,a) = \mathbb{E}_{\pi}[G_0 \,|\, s_0 = s, a_0 = a]$
- Simplify a bit (assume r is a deterministic function of state)

$$V^{\pi}(s) = \underbrace{r(s)}_{\text{Received a reward at state s}} + \gamma \sum_{a} \underbrace{\pi(a \,|\, s)}_{\text{Sample action a}} \sum_{s'} \underbrace{p(s' \,|\, s, a)}_{\text{Enter state s'}} \underbrace{V^{\pi}(s')}_{\text{Expected future rewards}}$$

- For Q:

$$Q^{\pi}(s,a) = \underbrace{r(s)}_{\text{Received a reward at state s}} + \gamma \sum_{s} \underbrace{p(s' \,|\, s, a)}_{\text{Enter state s'}} \sum_{a'} \underbrace{\pi(a' \,|\, s')}_{\text{Sample action a'}} \underbrace{Q^{\pi}(s', a')}_{\text{Expected future rewards}}$$

Policy Evaluation

- Goal: Given some π (might not be optimal), compute its value functions: $V^{\pi}(s) = \mathbb{E}[G_0 \,|\, s_0 = s]$ and $Q^{\pi}(s,a) = \mathbb{E}_{\pi}[G_0 \,|\, s_0 = s, a_0 = a]$
- Simplify a bit (assume r is a deterministic function of state)

$$V^{\pi}(s) = \underbrace{r(s)}_{\text{Received a reward at state s}} + \gamma \sum_{a} \underbrace{\pi(a \,|\, s)}_{\text{Sample action a}} \sum_{s'} \underbrace{p(s' \,|\, s, a)}_{\text{Enter state s'}} \underbrace{V^{\pi}(s')}_{\text{Expected future rewards}}$$

- For Q:

$$Q^{\pi}(s,a) = \underbrace{r(s)}_{\text{Received a reward at state s}} + \gamma \sum_{s} \underbrace{p(s' \,|\, s, a)}_{\text{Enter state s'}} \sum_{a'} \underbrace{\pi(a' \,|\, s')}_{\text{Sample action a'}} \underbrace{Q^{\pi}(s', a')}_{\text{Expected future rewards}}$$

Policy Evaluation

$$V^\pi(s) = r(s) + \gamma \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s, a) V^\pi(s')$$

Received a reward at state s

Sample action a

Enter state s′

Expected future rewards

- For tabular case (grid world), this is a set of linear equations! V^π is its unique solution
- Matrix form: $V^\pi = r + \gamma T^\pi V^\pi$
- Suppose there are $|S|$ states: $s_1, s_2, \ldots$
  - T^π is a $|S| \times |S|$-dim matrix whose (j, k) entry gives $p(s_k \mid s_j, a = \pi(s_j))$
  - T^π is often called a stochastic matrix (each row sums to 1, $T^\pi \mathbf{1} = \mathbf{1}$)
  - V^π is a $|S|$-dim vector whose (j) entry gives $V^\pi(s_j)$
  - r is a $|S|$-dim vector whose (j) entry gives $r(s_j)$
- Then we have a closed-form solution: $V^\pi = (I - \gamma T^\pi)^{-1} r$
- Or we can do iterative policy evaluation $V_{k+1} = r + \gamma T^\pi V_k$ until converge

Policy Evaluation Summary

- Goal: given some π (might not be optimal), compute its value functions

$$V^{\pi}(s) = \mathbb{E}[G_0 \,|\, s_0 = s] \qquad Q^{\pi}(s,a) = \mathbb{E}_{\pi}[G_0 \,|\, s_0 = s, a_0 = a]$$

$$V^{\pi}(s) = r(s) + \gamma \sum_{a} \pi(a \,|\, s) \sum_{s'} p(s' \,|\, s, a) V^{\pi}(s')$$

Received a reward at state s — Sample action a — Enter state s' — Expected future rewards

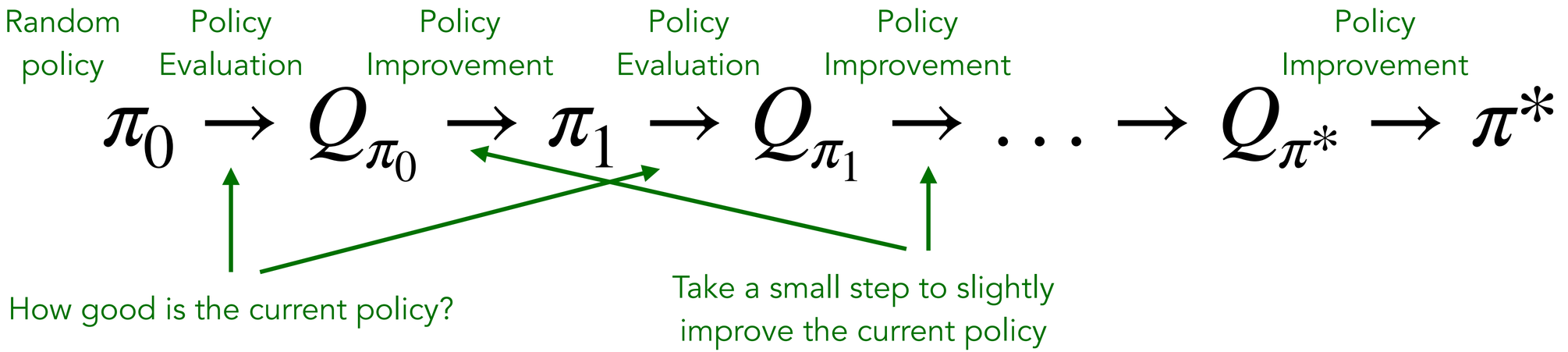
$$Q^{\pi}(s,a) = r(s) + \gamma \sum_{s} p(s' \,|\, s, a) \sum_{a'} \pi(a' \,|\, s') Q^{\pi}(s', a')$$

Received a reward at state s — Enter state s' — Sample action a' — Expected future rewards

- Using the above recursive relationship (Bellman equation)
- Known as "dynamic programming"
- Convergence guarantee.
- Require knowing dynamics (transition probability).

Policy Improvement

- Goal: iteratively improve the policy



- An intuitive idea: $\pi'(s) = \text{argmax}_a Q^\pi(s, a)$
  - I.e., "greedily" update policy to optimize previous policy's Q values
- Does it work?
- Yes! Theorem: if $Q^\pi(s, \pi'(s)) \geq V^\pi(s), \forall s$, then $V^{\pi'}(s) \geq V^\pi(s), \forall s$

Policy Improvement

- An intuitive idea: $\pi'(s) = \text{argmax}_a Q^\pi(s, a)$
  - I.e., "greedily" update policy to optimize previous policy's Q values
- When does $\pi(s) = \text{argmax}_a Q^\pi(s, a)$, policy is unchanged?
- Bellman optimality equations!

$$V^*(s) = \max_a Q^*(s, a)$$

$$\pi^*(s) = \arg\max_a Q^*(s, a)$$

  - Using the principle of optimality

$$V^*(s) = \max_a \sum_{s', r} p(s', r \mid s, a) \left[r + \gamma \cdot V^*(s') \right]$$

$$Q^*(s, a) = \sum_{s', r} p(s', r \mid s, a) \left[r + \gamma \cdot \max_{a'} Q^*(s', a') \right]$$

Policy Iteration using V

- Can also do Q value function (policy evaluation is more complex but improvement is easier)

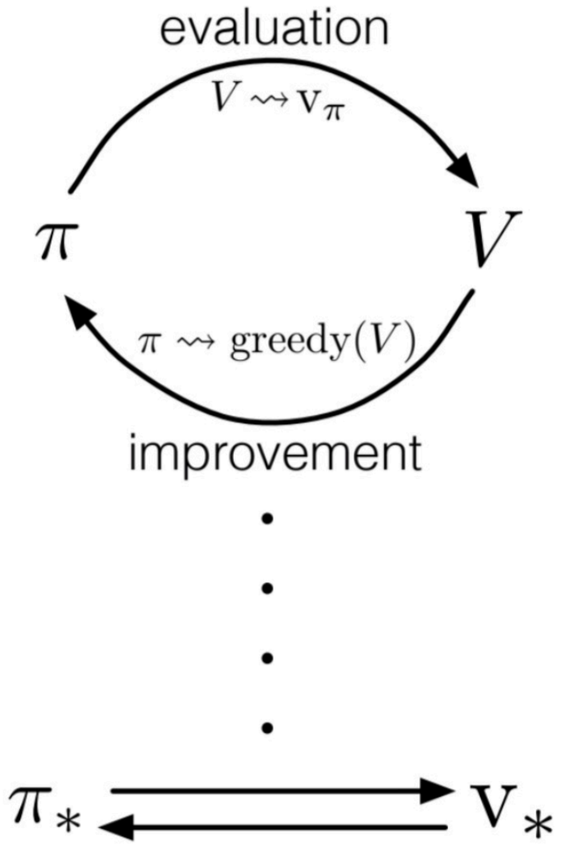
1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

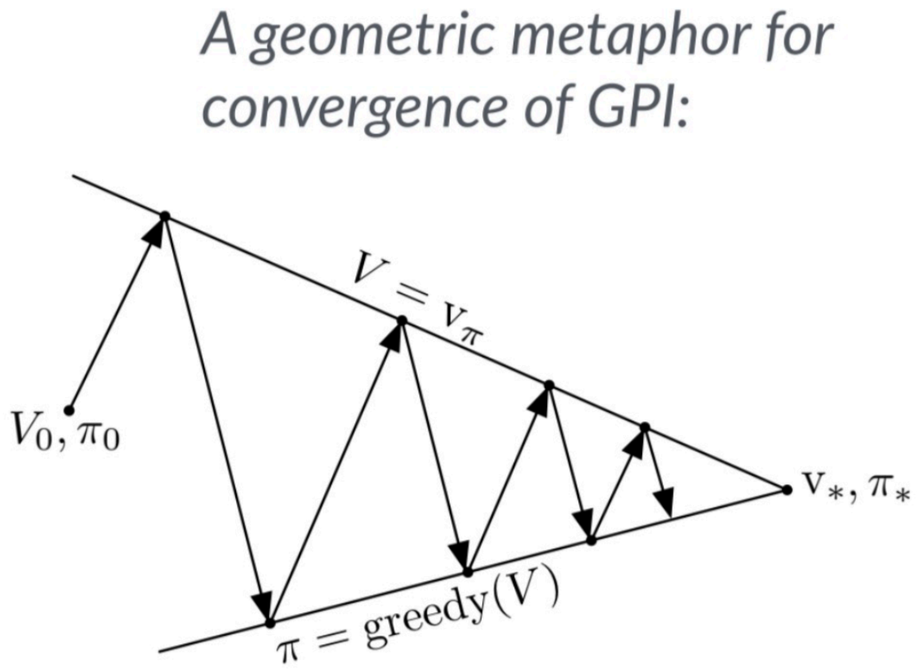
2. Policy Evaluation
   Loop:
      $\Delta \leftarrow 0$
      Loop for each $s \in \mathcal{S}$:
         $v \leftarrow V(s)$
         $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s)) \left[r + \gamma V(s') \right]$
         $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
      old-action $\leftarrow \pi(s)$
      $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma V(s') \right]$
      If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

General Policy Iteration

- Any interleaving of policy evaluation and policy improvement
- (Almost) All RL methods are a form of GPI





Value Iteration

- Policy iteration involves an iterative policy evaluation
- Can we remove it?
- Yes! Think about Bellman optimality equation:

$$V^*(s) = \max_a Q^*(s,a) = \max_a \sum_{s'} p(s', r \mid s, a)\,[\, r + \gamma \cdot V^*(s')\,]$$

- Idea: directly update the value function!

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $v \leftarrow V(s)$
| $V(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma V(s')\right]$
| $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \arg\max_a \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma V(s')\right]$

Summary

- Bellman optimality equations are nonlinear
- In tabular case with known dynamics, we can do DP-based policy improvement
- In general RL settings, approximately solve the Bellman optimality equations!

$$V^*(s) = \max_a \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma \cdot V^*(s') \right]$$

$$Q^*(s, a) = \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma \cdot \max_{a'} Q^*(s', a') \right]$$

Continuous state/action space (need function approximation for value functions)

Unknown dynamics (must be approximated, usually using sampling)

Max over continuous action space

Acknowledgement

- Many contents are taken from UT Austin CS391, CMU 16-831