Reinforcement *Learning*

by

Giovanni Stilo, PhD.

giovanni.stilo@univaq.it

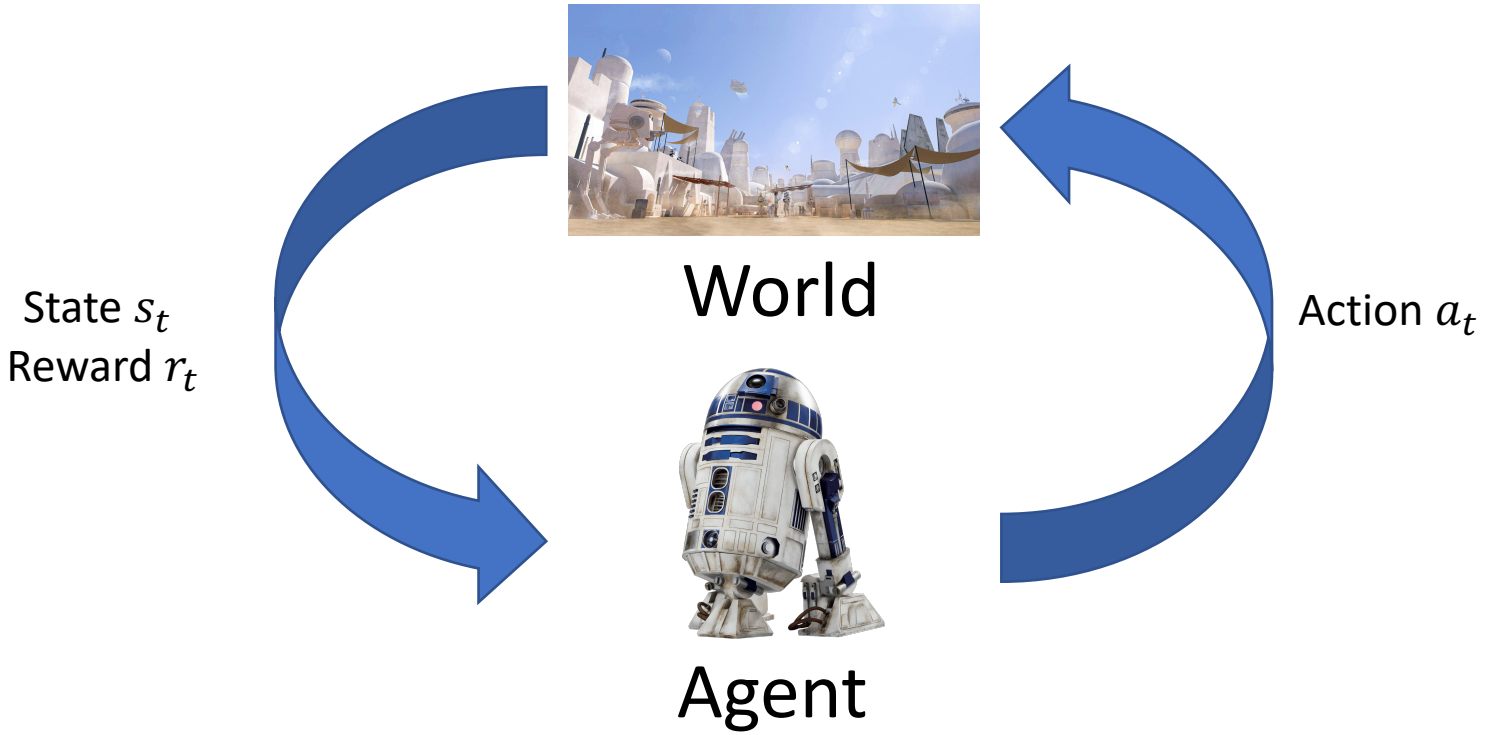
Last time Recap.

- **Model:**
  - Mathematical models of dynamics and reward
- **Policy:**
  - Function mapping agent's states to actions
- **Value function:**
  - future rewards from being in a state and/or action when following a policy

Outline

- Markov Processes
- Markov Reward Processes (MRPs)
- Markov Decision Processes (MDPs)

Full Observability: Markov Decision Process (MDP)



- MDPs can model a huge number of interesting problems and setting
  - Bandits: single state MDP
  - Optimal control mostly about continuous-state MDPs
  - Partially observable MDPs = MDP where state is history

Recall: Markov Assumption

- Information state: *sufficient statistic of history*
- State s_t is Markov if and only if:

  $p(s_{t+1} \mid s_t, a_t) = p(s_{t+1} \mid h_t, a_t)$

- Given present, the future is **independent** of the past (if you have *sufficient statistic of history)*;

Markov Process or Markov Chain

- Memoryless random process
  - Sequence of random states with Markov property

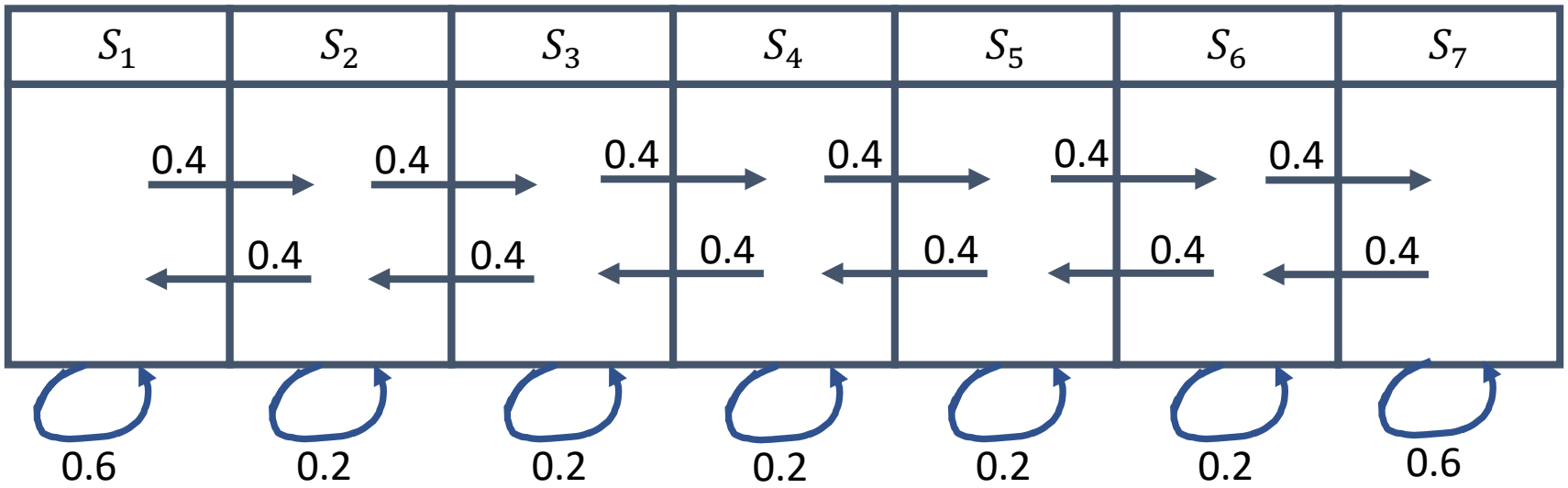
- Definition of Markov Process
  - *S* is a (finite) set of states *($s \in S$)*
  - *P* is dynamics/transition model that specifies

$$p(s_{t+1} = s' \mid s_t = s)$$

- **Note: no rewards, no actions**
- If finite number (N) of states, can express *P* as a matrix:

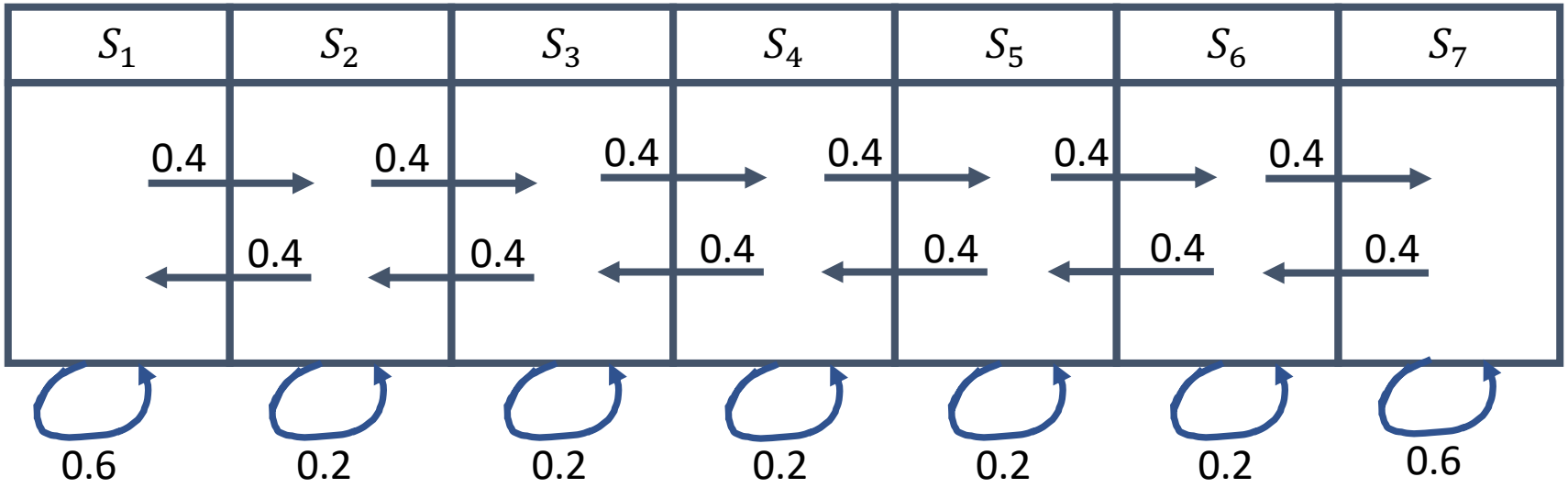
$$P = \begin{pmatrix} p(s_1|s_1) & p(s_2|s_1) & \cdots & p(s_N|s_1) \\ p(s_1|s_2) & p(s_2|s_2) & \cdots & p(s_N|s_2) \\ \vdots & \vdots & \ddots & \vdots \\ p(s_1|s_N) & p(s_2|s_N) & \cdots & p(s_N|s_N) \end{pmatrix}$$

Example: Mars Rover Markov Chain Transition Matrix, P



$$P = \begin{pmatrix} 0.6 & 0.4 & 0 & 0 & 0 & 0 & 0 \\ 0.4 & 0.2 & 0.4 & 0 & 0 & 0 & 0 \\ 0 & 0.4 & 0.2 & 0.4 & 0 & 0 & 0 \\ 0 & 0 & 0.4 & 0.2 & 0.4 & 0 & 0 \\ 0 & 0 & 0 & 0.4 & 0.2 & 0.4 & 0 \\ 0 & 0 & 0 & 0 & 0.4 & 0.2 & 0.4 \\ 0 & 0 & 0 & 0 & 0 & 0.4 & 0.6 \end{pmatrix}$$

Example: Mars Rover Markov Chain Episodes



- Example: Sample episodes starting from S4:
  - $s_4, s_5, s_6, s_7, s_7, s_7, \ldots$
  - $s_4, s_4, s_5, s_4, s_5, s_6, \ldots$
  - $s_4, s_3, s_2, \ldots$

Markov Reward Process (MRP)

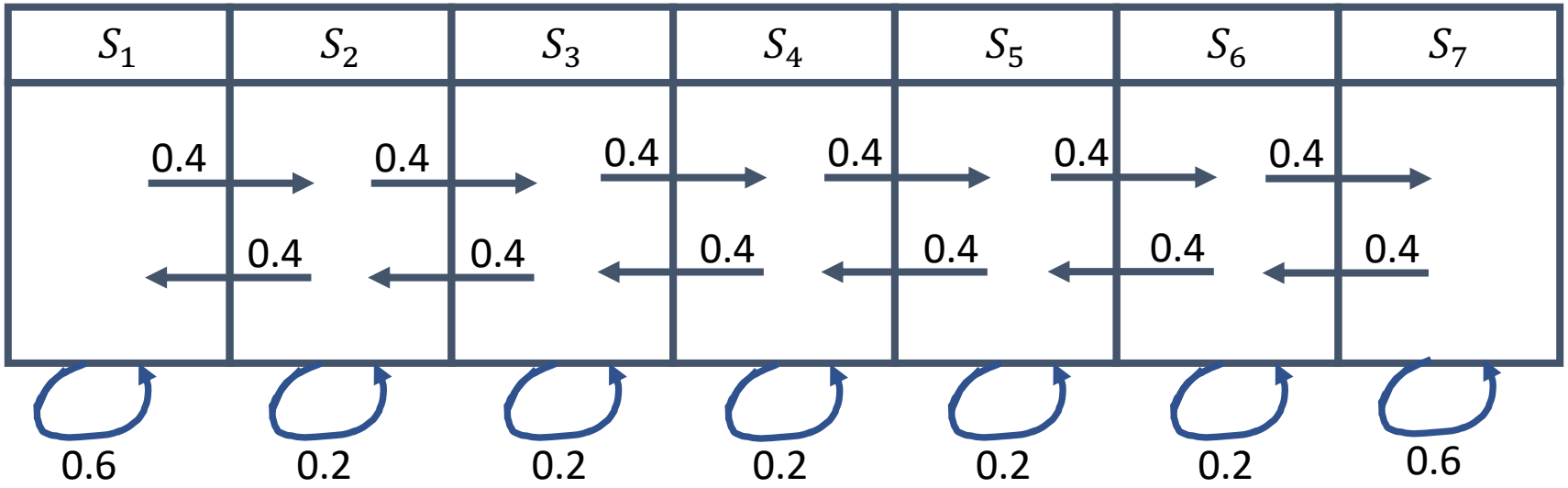
- Markov Reward Process is a Markov Chain + rewards

- Definition of Markov Reward Process (MRP):
  - S is a (finite) set of states *($s \in S$)*
  - P is dynamics/transition model that specifies *p($s_{t+1} = s' \mid s_t = s$)*
  - **R is a reward function *R($s_t = s$) = $E[\, r_t \mid s_t = s)$***
  - Discount factor $\gamma \in [0, 1]$

- **Note:** no actions
- $MRP = (S, P, R, \gamma)$

- If finite number (N) of states: *R can be expressed as a vector*

Example: Mars Rover MRP



- **Reward:** *+1 in S_1; +10 in S_7; 0 in all the other states*

Return & Value Function

- Definition of Horizon:
  - Number of time steps in each episode
  - Can be infinite
  - Otherwise called **finite** Markov reward process

- Definition of Return, G_t (for a MRP):
  - Discounted sum of rewards from time step ***t*** to horizon

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+1} + \gamma^3 r_{t+3} + \cdots$$

- Definition of State Value Function, V(s) (for a MRP)
  - Expected return from starting in state s:

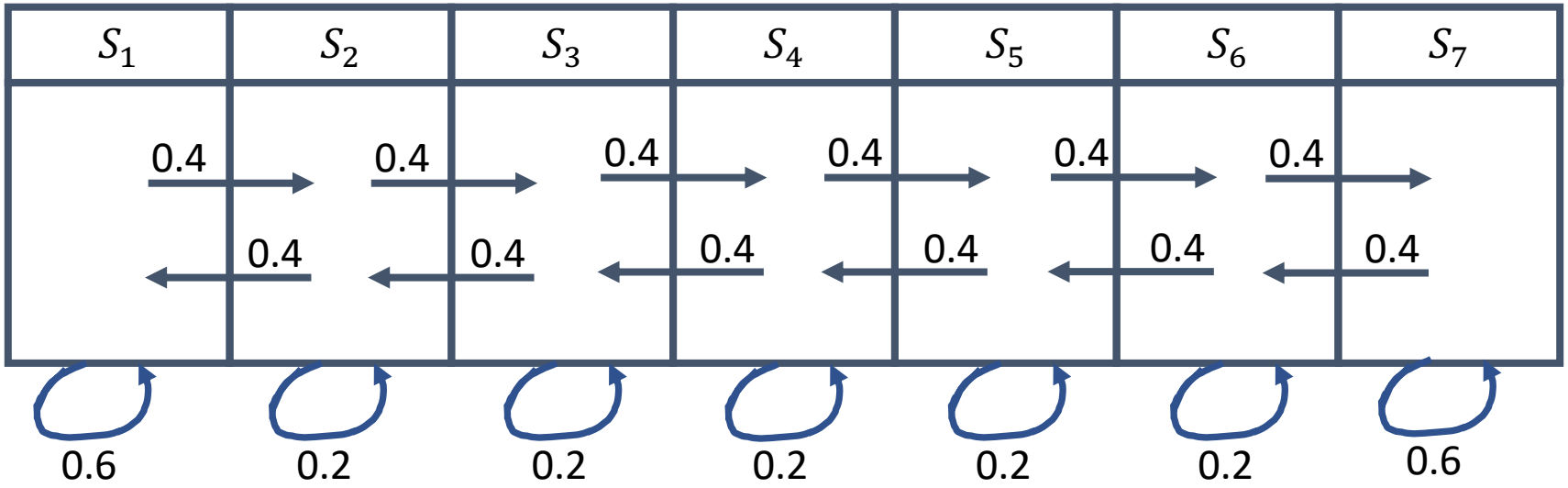
$$\mathrm{V(s)} = \mathbb{E}[\, G_t \mid s_t = s] = \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+1} + \gamma^3 r_{t+3} + \cdots \mid s_t = \mathrm{s}]$$

(coincident in the discrete case)

Discount Factor

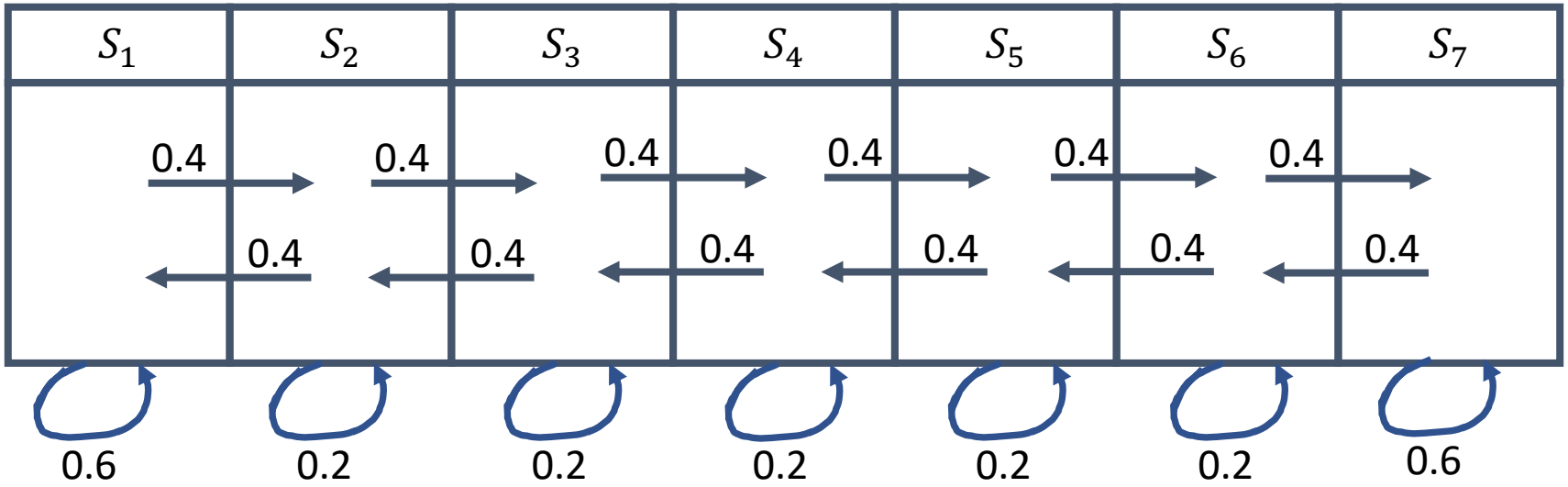
- Mathematically convenient (avoid infinite returns and values)
- Humans often act as if there's a discount factor < 1
- $\gamma = 0$: Only care about immediate reward
- $\gamma = 1$: Future reward is as beneficial as immediate reward
- If episode lengths are always finite, can use $\gamma = 1$

Example: Mars Rover MRP



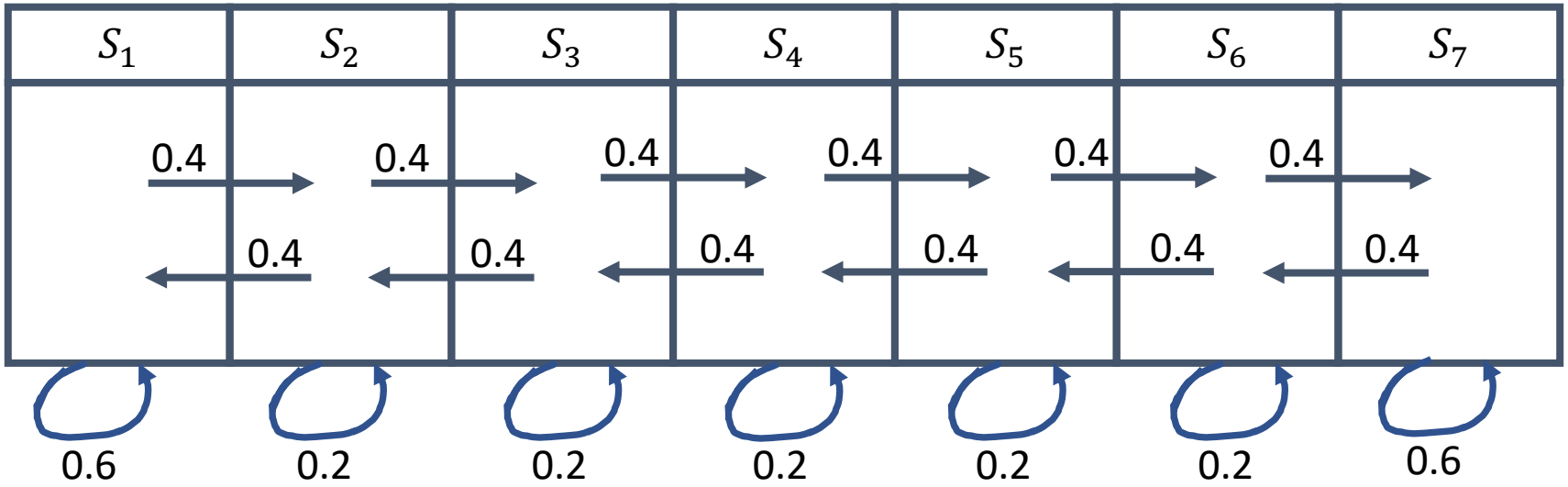
- **Reward:** *+1 in S_1; +10 in S_7; 0 in all the other states*
- **Returns** for 4-step episodes, $\gamma = \frac{1}{2}$
  - s_4, s_5, s_6, s_7: $0 + \frac{1}{2} \times 0 + \frac{1}{4} \times 0 + \frac{1}{8} \times 10 = 1.25$

Example: Mars Rover MRP



- **Reward:** *+1 in S_1; +10 in S_7; 0 in all the other states*
- **Returns** for 4-step episodes, $\gamma = \frac{1}{2}$
  - s_4, s_5, s_6, s_7: $0 + \frac{1}{2} \times 0 + \frac{1}{4} \times 0 + \frac{1}{8} \times 10 = 1.25$
  - s_4, s_4, s_5, s_4: $0 + \frac{1}{2} \times 0 + \frac{1}{4} \times 0 + \frac{1}{8} \times 0 = 0$
  - s_4, s_3, s_2, s_1: $0 + \frac{1}{2} \times 0 + \frac{1}{4} \times 0 + \frac{1}{8} \times 1 = 0.125$

Example: Mars Rover MRP



- **Reward:** *+1 in S_1; +10 in S_7; 0 in all the other states*
- **Returns** for 4-step episodes, $\gamma = \frac{1}{2}$
  - s_4, s_5, s_6, s_7: $0 + \frac{1}{2}\times 0 + \frac{1}{4}\times 0 + \frac{1}{8}\times 10 = 1.25$
  - s_4, s_4, s_5, s_4: $0 + \frac{1}{2}\times 0 + \frac{1}{4}\times 0 + \frac{1}{8}\times 0 = 0$
  - s_4, s_3, s_2, s_1: $0 + \frac{1}{2}\times 0 + \frac{1}{4}\times 0 + \frac{1}{8}\times 1 = 0.125$
- Value Function: Expected return from starting in state s:

$$\mathrm{V(s)} = \mathbb{E}[\, G_t \mid s_t = s] = \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+1} + \gamma^3 r_{t+3} + \cdots \mid s_t = \mathrm{s}]$$

V = [1.53, 0.37, 0.13, 0.22, 0.85, 3.59, 15.31]

Computing the Value of a Markov Reward Process

- $\mathbf{V}(\boldsymbol{s}) = \mathbb{E}[\boldsymbol{G_t}|\boldsymbol{S_t} = \boldsymbol{s}] = \mathbb{E}\left[\boldsymbol{R_t} + \boldsymbol{\gamma R_{t+1}} + \boldsymbol{\gamma^2 R_{t+2}} + \boldsymbol{\gamma^3 R_{t+3}} + \cdots | \boldsymbol{S_t} = \boldsymbol{s}\right]$

- Could be estimated by simulation:
  - Generate a large number of episodes
  - Average returns
  - Convergence speed of approximately $1/\sqrt{n}$
    (n number of episodes)
  - Simulation requires **no assumption** of Markov structure

Bellman Equation for MRPs

In MRP the value function can be decomposed into two parts:

- immediate reward R_t
- discounted future rewards:

- $\mathrm{V(s)} = \mathbb{E}[G_t | S_t = s]$

$$= \mathbb{E}\left[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \cdots \mid S_t = s\right]$$

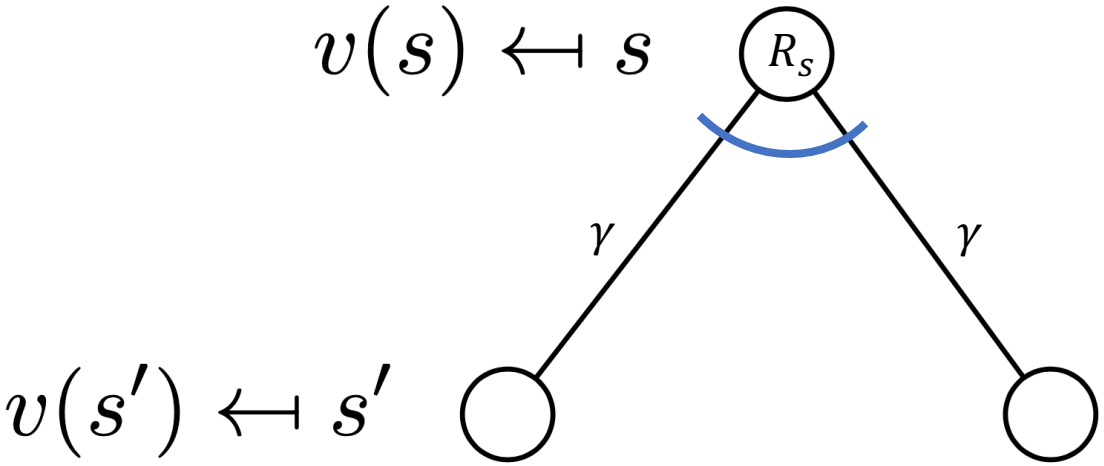
$$= \mathbb{E}\left[R_t + \gamma (R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots) \mid S_t = s\right]$$

$$= \mathbb{E}\left[R_t + \gamma G_{t+1} \mid S_t = s\right]$$

$$= \mathbb{E}\left[R_t + \gamma V(S_{t+1}) \mid S_t = s\right]$$

Bellman Equation for MRPs

$$\mathbb{E}\left[R_t + \gamma V(S_{t+1}) \middle| S_t = s\right]$$



$$V(S) = R_s + \gamma \sum_{s' \in S} P(S'|S)\, V(S')$$

Computing the Value of a Markov Reward Process

- Could estimate by simulation but the Markov property yields additional structure.
- MRP **value function** can be written:
- $$V(s) = \underbrace{R(s)}_{Immediate\ reward} + \underbrace{\gamma \sum_{\{s' \in S\}} P(s'|s)V(s')}_{\substack{Discounted\ sum\ of \\ future\ reward}}$$

Matrix Form of Bellman Equation for MRP

$$V(s) = \underbrace{R(s)}_{immediate\ reward} + \underbrace{\gamma \sum_{\{s' \in S\}} P(s'|s)V(s')}_{discounted\ sum\ of\ future\ reward}$$

$$\begin{bmatrix} V(s_1) \\ V(s_2) \\ \vdots \\ V(s_N) \end{bmatrix} = \begin{bmatrix} R(s_1) \\ R(s_2) \\ \vdots \\ R(s_N) \end{bmatrix} + \gamma \begin{bmatrix} p(s_1|s_1) \cdots & p(s_N|s_1) \\ p(s_1|s_2) \cdots & p(s_N|s_2) \\ \vdots \quad \ddots & \vdots \\ p(s_1|s_N) \cdots & p(s_N|s_N) \end{bmatrix} \begin{bmatrix} V(s_1) \\ V(s_2) \\ \vdots \\ V(s_N) \end{bmatrix}$$

$$V = R + \gamma PV$$

Matrix Form of Bellman Equation for MRP

$$V(s) = \underbrace{R(s)}_{immediate\ reward} + \underbrace{\gamma \sum_{\{s' \in S\}} P(s'|s)V(s')}_{\substack{discounted\ sum\ of \\ future\ reward}}$$

$$\begin{bmatrix} V(s_1) \\ V(s_2) \\ \vdots \\ V(s_N) \end{bmatrix} = \begin{bmatrix} R(s_1) \\ R(s_2) \\ \vdots \\ R(s_N) \end{bmatrix} + \gamma \begin{bmatrix} p(s_1|s_1) \cdots & p(s_N|s_1) \\ p(s_1|s_2) \cdots & p(s_N|s_2) \\ \vdots \quad \ddots & \vdots \\ p(s_1|s_N) \cdots & p(s_N|s_N) \end{bmatrix} \begin{bmatrix} V(s_1) \\ V(s_2) \\ \vdots \\ V(s_N) \end{bmatrix}$$

$$V = R + \gamma PV;$$

$$V - \gamma PV = R;$$

$$(I - \gamma P)V = R;$$

$$V = (I - \gamma P)^{-1} R;$$

- Can be solved directly: requires taking a matrix inverse $\sim O(N^3)$

Iterative Algorithm for Computing Value of a MRP

- The Dynamic Programming (Iterative);
- Initialize $V_0(s) = 0 \; \forall \; s \in S$;
- For $k = 1$ until **convergence**:
  - $\forall s \in S$

$$V_k(s) = R(s) + \gamma \sum_{\{s' \in S\}} P(s'|s) \, V_{k-1}(s')$$

  - Equivalent to do:

$$V_k = R + \gamma P V_{k-1};$$

- Computational complexity: $O(|S|^2)$ for each iteration $(|S| = N)$

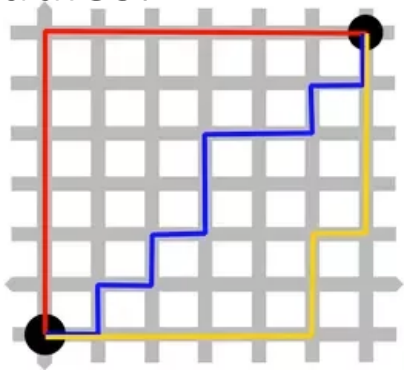
What converge mean in practice?

Recap Norms

- Euclidean norm is a special case of the p-norms:

$$\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}}$$

- Two important cases:
  - Infinity norm, or Chebyshev norm is the maximum element:
    - $\|\boldsymbol{x}\|_\infty = \max_i |x_i|$
  - L_1 norm, or Manhattan norm is the sum of modules:
    - $\|\boldsymbol{x}\|_1 = \sum_{i=1}^{n} |x_i|$



Markov Decision Process (MDP)

- Markov Decision Process is Markov Reward Process + **actions**

- Definition of Markov Decision Process (MDP):
  - S is a (finite) set of states *($s \in S$)*
  - **A is a (finite) set of actions *($a \in A$)***
  - P is dynamics/transition model **for each action**, that specifies
    $$P(s_{t+1} = s' \mid s_t = s, a_t = a)$$
  - R is a reward function **for each action** :
    $$\mathrm{R}(s_t = s, a_t = a) = E[\, r_t \mid s_t = s, a_t = a]$$
  - Discount factor $\gamma \in [0, 1]$

- MDP is a tuple: (S, A, P, R, γ)

NOTE: Reward is sometimes defined as a function of the current state, or as a function of the (state, action, next state) tuple. We will assume reward is a function of **state and action.**

Example: Mars Rover MDP (2 deterministic actions)

S_1	S_2	S_3	S_4	S_5	S_6	S_7
						

$$P(s'|s,a_1) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \qquad P(s'|s,a_2) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Which kind of Actions we have here?

MDP Policies

- Policy **fully defines** what **action** to **take** in each state:
  - Can be *deterministic* or *stochastic*;

- MDP policies depend on the current state(not the history?):
  - Policies are stationary (time-independent):

$$a_t \sim \pi(\cdot \mid S_t), \forall t > 0$$

- A policy π is a distribution over actions given states*:*

Policy: $\pi(a|s) = P(a_t = a \mid s_t = s)$

MDP + Policy

MDP + $\pi(a|s)$?

MDP + Policy

MDP + $\pi(a|s)$ = Markov Reward Process

- Precisely, it is the MRP $(S, R_\pi, P_\pi, \gamma)$, where:

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

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

- Implies **we can** use same techniques to **evaluate** the value of a policy for a MDP as we could to compute the value of a MRP, defined by R^π and P^π .

MDP Policy Evaluation, Iterative Algorithm

- Initialize $V_0(s) = 0 \; \forall \; s \in S$;
- For $k = 1$ until **convergence**:
  - $\forall s \in S$

$$V_k^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} P(s'|s, \pi(s)) V_{k-1}^{\pi}(s')$$

- This is called the **Bellman backup** for a particular policy

Policy Evaluation: Example & Check Your Understanding

S_1	S_2	S_3	S_4	S_5	S_6	S_7
						

- Two deterministic Actions: a_1, a_2
- **Reward:** *+1 in S_1; +10 in S_7; 0 in all the other states*
- **Policy** $\forall s$: $\pi(s) = a_1$;
- $\gamma = 0$;
- **Iteratively:**
  - $V_k^{\pi}(s) = r(s, \pi(s)) + \gamma \sum_{s' \in S} p(s'|s, \pi(s))\, V_{k-1}^{\pi}(s')$

$$V^{\pi} = ?$$

Policy Evaluation: Example & Check Your Understanding

S_1	S_2	S_3	S_4	S_5	S_6	S_7
						

- Two deterministic Actions: a_1, a_2
- **Reward:** *+1 in S_1; +10 in S_7; 0 in all the other states*
- **Policy** $\forall s$: $\pi(s) = a_1$;
- $\boldsymbol{\gamma = 0}$;
- Iterative:
  - $V_k^{\pi}(s) = r(s, \pi(s)) + \gamma \sum_{s' \in S} p(s'|s, \pi(s))\, V_{k-1}^{\pi}(s')$

$$V^{\pi} = [1, 0, 0, 0, 0, 0, 10]$$

Practice: MDP 1 Iteration of Policy Evaluation, Mars Rover Example

- **Dynamics**: $p(s_6|s_6, a_1) = 0.5, \;\; p(s_7|s_6, a_1) = 0.5, \ldots$

- **Reward:** for all actions, *+1 in S_1; +10 in S_7; 0 otherwise;*

- **Policy**: $\forall s\colon \pi(s) = a_1$;
- $\boldsymbol{V_k} = [1, 0, 0, 0, 0, 0, 10]$,
- $\boldsymbol{k} = 1$,
- $\boldsymbol{\gamma} = 0.5$;
  - $\forall s \in S$:

$$V_k^{\pi}(s) = r(s, \pi(s)) + \gamma \sum_{s' \in S} p(s'|s, \pi(s)) V_{k-1}^{\pi}(s')$$

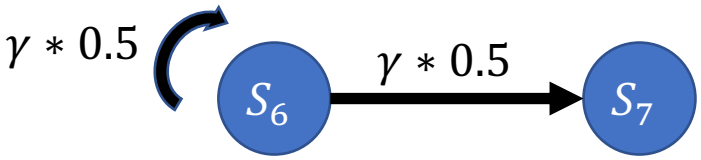
  - Compute the first Iteration only for $V_2(s_6)$

Practice: MDP 1 Iteration of Policy Evaluation, Mars Rover Example

- **Dynamics**: $p(s_6|s_6, a_1) = 0.5, \quad p(s_7|s_6, a_1) = 0.5, \ldots$
- **Reward:** for all actions, *+1 in S_1; +10 in S_7; 0 otherwise;*
- **Policy**: $\forall s$: $\pi(s) = a_1$; $\boldsymbol{V_k^\pi} = [1, 0, 0, 0, 0, \mathbf{0}, \mathbf{10}], \boldsymbol{k} = 1, \ \boldsymbol{\gamma} = \mathbf{0.5}$;
  - $\forall s \in S$:

$$V_k^\pi(s) = r(s, \pi(s)) + \gamma \sum_{s' \in S} p(s'|s, \pi(s)) V_{k-1}^\pi(s')$$

- $V_{k+1}(s_6) = r(s_6, a_1) + \gamma * 0.5 * V_k(s_6) + \gamma * 0.5 * V_k(s_7)$
- $V_{k+1}(s_6) = 0 + 0.5 * 0.5 * 0 + 0.5 * 0.5 * 10$
- $V_{k+1}(s_6)$=2.5



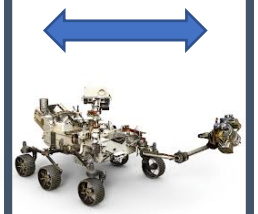
MDP Control

- Compute/Learn the optimal policy:

$$\pi^*(S) = arg\ max_\pi\ V^\pi(S)$$

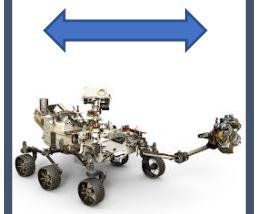
- There exists a **unique optimal value function**;

Check Your Understanding

S_1	S_2	S_3	S_4	S_5	S_6	S_7
						

- **7 discrete states** (location of rover)
- **2 actions:** a_1 or a_2
- Q_0: *How many deterministic policies are there?*
- Q_1 : *Is the optimal policy for a MDP always unique?*

Check Your Understanding

S_1	S_2	S_3	S_4	S_5	S_6	S_7
						

- **7 discrete states** (location of rover)
- **2 actions:** Left or Right
- Q_0: *How many deterministic policies are there?*

$$|A|^{|S|} = 2^7$$

- Q_1 : *Is the optimal policy for a MDP always unique?*

No, there may be two(or more) actions that have the same optimal value function (not visible in this example)

MDP Control Problem!

- Compute/Learn the optimal policy:

$$\pi^*(S) = arg\ max_\pi\ V^\pi(S)$$

MDP Control

- Compute/Learn the optimal policy:

$$\pi^*(S) = arg\ max_\pi V^\pi(S)$$

- **Has been proved:**
  - There exists a **unique optimal value function**;
  - **Optimal policy** for a MDP in an infinite horizon problem (agents act forever) is:
    - Deterministic;
    - Stationary (does not depend on time step);
    - Unique? Not necessarily, may have state-actions with identical optimal values;

Policy Search

- Compute/Learn the optimal policy:

$$\pi^*(S) = arg\ max_\pi\ V^\pi(S)$$

- We know that the number of deterministic policies is $|A|^{|S|}$
- **One option is searching:**
  - Compute all of them and pick the maximal one.
- Policy iteration is generally more efficient solution than enumeration

MDP Policy Iteration (PI)

- Set $i = 0$
  (we want to index the policies that we compute)
- Initialize $\pi_0(s)$ randomly for all states $s \in S$;
- While $i == 0$ or $\| \pi_i - \pi_{i-1} \|_1 > 0$:
  - $V^{\pi_i} \leftarrow$ MDP V function policy evaluation of π_i;
  - $\pi_{i+1} \leftarrow$ Policy improvement; (new part)
  - $i = i + 1$;

New Definition: State-Action the Q-Value

- The value function for a given policy π is $V^{\pi}(s)$
- State-action value of a policy (Q-Value):

$$Q^{\pi}(s,a) = R(s,\boldsymbol{a}) + \gamma \sum_{s' \in S} P(s'|s,\boldsymbol{a}) V^{\boldsymbol{\pi}}(s')$$

- Take the action **a**, get the reward $R(s,\boldsymbol{a})$, then follow the policy $\boldsymbol{\pi}$ after the first transition $P(s'|s,\boldsymbol{a})$;

Policy Improvement Step

- Compute all the state-action values (q-values) of a policy π_i
  - $\forall\, s \in S \; and \; \forall \boldsymbol{a} \in A$:

$$Q^{\boldsymbol{\pi_i}}(s, \boldsymbol{a}) = R(s, \boldsymbol{a}) + \gamma \sum_{s' \in S} P(s'|s, \boldsymbol{a}) V^{\boldsymbol{\pi_i}}(s')$$

- Compute/update the new policy π_{i+1} , $\forall\, s \in S$:

$$\pi_{i+1}(S) = arg\, max_a\, Q^{\pi_i}(s, a)$$

$$max_a\, Q^{\pi_i}(s, a) \;\geq Q^{\pi_i}(s, \pi_i(s))$$

- Note: $\pi_{i+1} \neq \pi_i$ if there is a strict inequality for at least one state.

MDP Policy Iteration (PI)

- Set $i = 0$
- Initialize $\pi_0(s)$ randomly for all states $s \in S$;
- While $i == 0$ or $\| \pi_i - \pi_{i-1} \|_1 > 0$:
  - $V^{\pi_i} \leftarrow$ MDP V function policy **evaluation** of π_i;
  - $\pi_{i+1} \leftarrow$ Policy **improvement**;
  - $i = i + 1$;

Delving Deeper Into Policy Improvement Step

- Compute state−action value of a policy π_i
  - $\forall\, s \in S \; and \; \forall a \in A$:

$$Q^{\pi_i}(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^{\pi_i}(s')$$

$$arg\, max_a\, Q^{\boldsymbol{\pi_i}}(s, \boldsymbol{a}) \geq R(s, \boldsymbol{\pi\_i(s)}) + \gamma \sum_{s' \in S} P(s'|s, \boldsymbol{\pi\_i(s)}) V^{\pi_i}(s') = V^{\pi_i}(s)$$

- Compute new policy π_{i+1} , $\forall\, s \in S$:

$$\pi_{i+1}(S) = arg\, max_a\, Q^{\pi_i}(s,a)$$

$$max_a\, Q^{\pi_i}(s,a) \;\geq Q^{\pi_i}(s, \pi_i(s))$$

- Suppose we take $\pi_{i+1}(s)$ for **one action**, then follow π_i forever (by definition of Q-Function)
  - Our expected sum of rewards is **at least as good** as if we had always followed π_i
- But new proposed policy is to always follow π_{i+1} !

Monotonic Improvement in Policy

- **Definition:**

$$V^{\pi_1} \geq V^{\pi_2} \Leftrightarrow \forall s \in S\colon\ V^{\pi_1}(s) \geq V^{\pi_2}(s)$$

- **Proposition:**
  - $V^{\pi_{i+1}} \geq V^{\pi_i}$ with strict inequality if π_i is suboptimal,

    where π_{i+1} is the new policy obtained from policy improvement of π_i

- **Proof:**
  - See Sutton & Barto Reinforcement Learning: An Introduction . Ch 4.2

Policy Iteration (PI): Check Your Understanding

- Set $i = 0$;
- Initialize $\pi_0(s)$ randomly for all states $s \in S$;
- While $i == 0$ or $\| \pi_i - \pi_{i-1} \|_1 > 0$:
  - $V^{\pi_i} \leftarrow$ MDP V function policy **evaluation** of π_i;
  - $\pi_{i+1} \leftarrow$ Policy **improvement**;
  - $i = i + 1$;
- **Q0: If policy doesn't change, can it ever change again?**
- **Q1: Is there a maximum number of iterations of policy iteration?**

Policy Iteration (PI): Check Your Understanding

- If $\boldsymbol{\pi_{i+1} = \pi_i} \longrightarrow \forall\, s \in S\colon\ \pi_{i+1}(s) = \pi_i(s)$
- Then we know that $\forall s \in S\colon \quad Q^{\pi_{i+1}}(s,a) = Q^{\pi_i}(s,a)$
- Recall policy improvement step:

$$Q^{\pi_i}(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^{\pi_i}(s')$$

$$\pi_{i+1}(S) = arg\ max_a\ Q^{\pi_i}(s,a)$$

$$\pi_{i+2}(S) = arg\ max_a\ Q^{\pi_{i+1}}(s,a) = arg\ max_a\ Q^{\pi_i}(s,a)$$

- **Therefore policy cannot ever change again**

Policy Iteration (PI): Check Your Understanding

- Set $i = 0$
- Initialize $\pi_0(s)$ randomly for all states $s \in S$;
- While $i == 0$ or $\| \pi_i - \pi_{i-1} \|_1 > 0$:
  - $V^{\pi_i} \leftarrow$ MDP V function policy **evaluation** of π_i;
  - $\pi_{i+1} \leftarrow$ Policy **improvement**;
  - $i = i + 1$;

- **If policy doesn't change, can it ever change again?**

  NO

- **Is there a maximum number of iterations of policy iteration?**

  $|A|^{|S|}$: since that is the maximum number of policies.
  Note that the policy improvement step is monotonically improving, each policy can only appear in one round of policy iteration unless it is an optimal policy.

Summary

- Markov Processes
- Markov Reward Processes (MRPs)
- Markov Decision Processes (MDPs)
- MDP Optimal Policy/Control