
# REINFORCEMENT LEARNING

## CP8319/CPS824
## Lecture 3

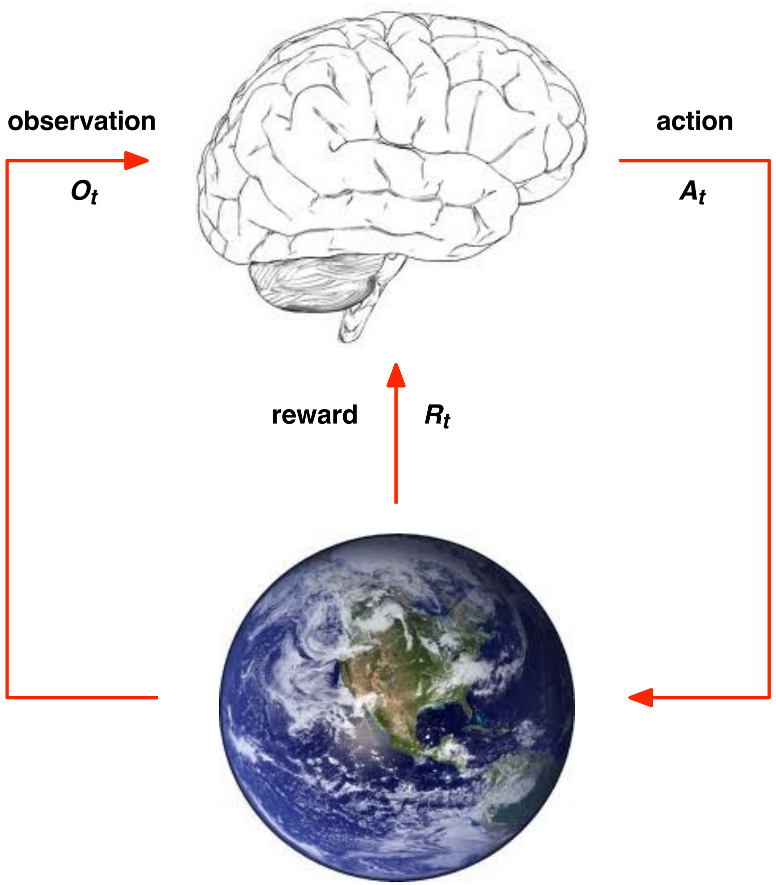
Instructor: Nariman Farsad

* 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. Markov Process
3. Markov Reward Process
4. Markov Decision Process

# RL: The Agent and the Environment




- At each step $t$ the agent:
  - Executes action $A_t$
  - Receives observation $O_t$
  - Receives scalar reward $R_t$
- The environment:
  - Receives action $A_t$
  - Emits observation $O_{t+1}$
  - Emits scalar reward $R_{t+1}$
- $t$ increments at env. step

# History and State

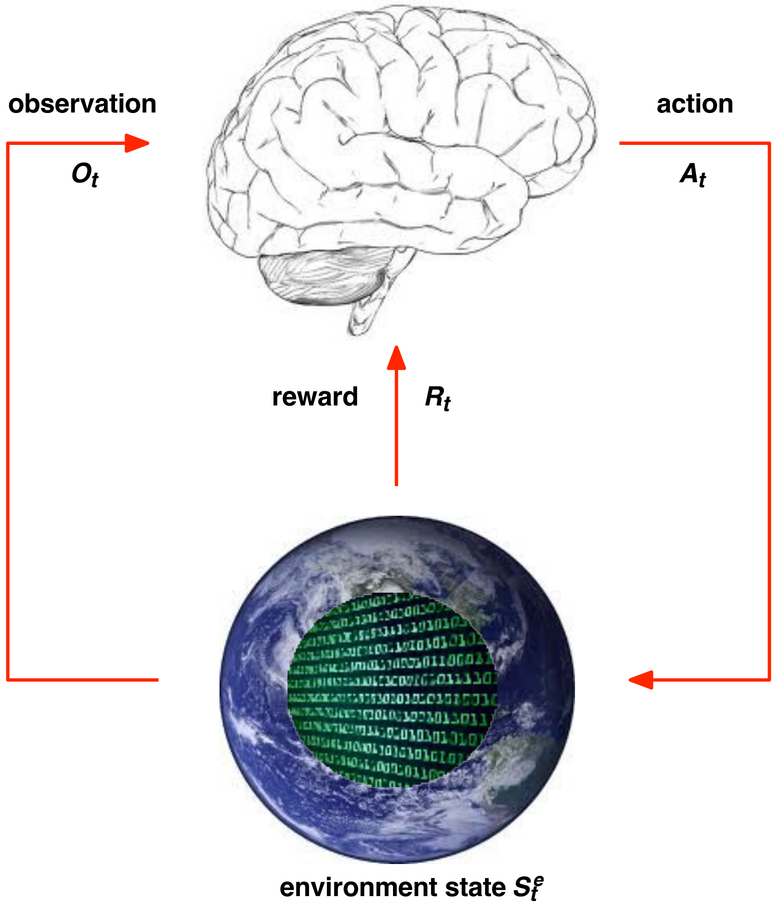
- The history is the sequence of observations, actions, rewards

$$H_t = O_1, R_1, A_1, ..., A_{t-1}, O_t, R_t$$

- i.e. all observable variables up to time *t*
- i.e. the sensorimotor stream of a robot or embodied agent What happens next depends on the history:
  - The agent selects actions
  - The environment selects observations/rewards
- The State is the information used to determine what happens next Formally, state is a function of the history:

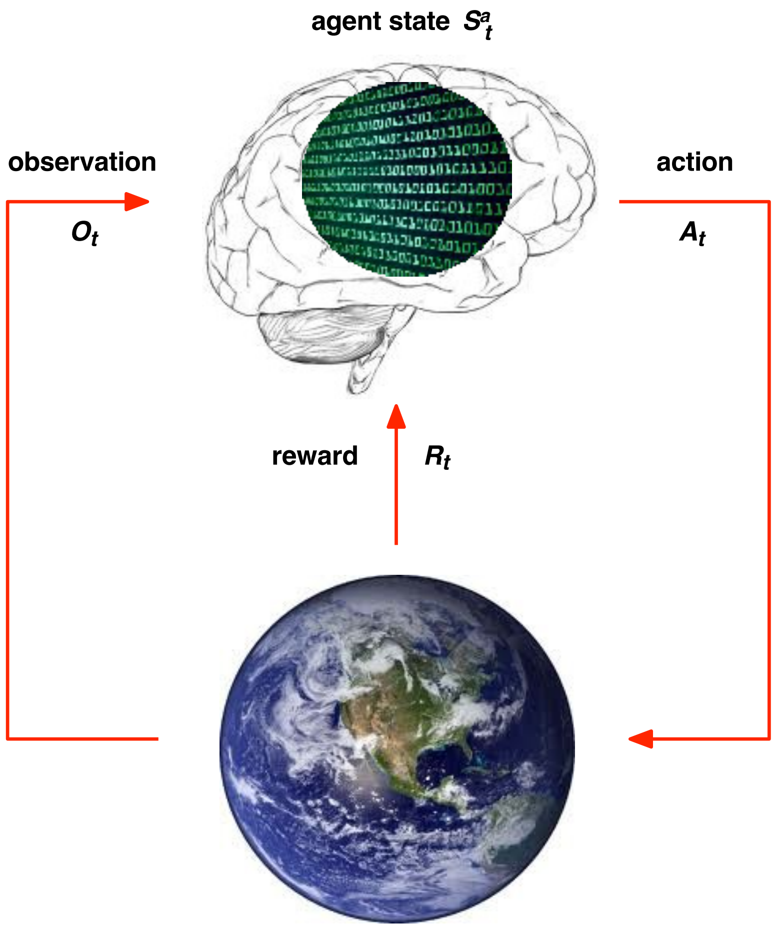
$$S_t = f(H_t)$$

# Environment State




- The environment state $S_t^e$ is the environment's private representation
- The environment uses the state to pick the next observation/reward
- The environment state is not usually visible to the agent directly
- Even when $S_t^e$ is the visible it may contain irrelevant information

# Agent State




- The agent state $S_t^a$ is the agent's internal representation
- i.e. whatever information the agent uses to pick the next action
- i.e. it is the information used by reinforcement learning algorithms
- It can be any function of history:

$$S_t^a = f\,(H_t)$$

# Today's Agenda

1. Last Lecture Review
2. Markov Process
3. Markov Reward Process
4. Markov Decision Process

# Markov Assumption

"The future is independent of the past given the present"

**Definition**

A state $S_t$ is *Markov* if and only if

$$P(S_{t+1} \mid S_t) = P(S_{t+1} \mid S_1, \ldots, S_t)$$

- The state captures all relevant information from the history
- Once the state is known, the history may be thrown away
- i.e. The state is a sufficient statistic of the future

# State Transition Matrix

For a Markov state $s$ and successor state $s'$, the *state transition probability* is defined by

$$P_{s,s'} = P(S_{t+1} = s' \mid S_t = s)$$

State transition matrix $\mathbf{P}$ defines transition probabilities from all states $s$ to all successor states $s'$,

$$\mathbf{P} = \begin{array}{cc} & \text{To } s' \\ \text{From } s & \begin{pmatrix} P_{1,1} & \cdots & P_{1,n} \\ \vdots & \ddots & \vdots \\ P_{n,1} & \cdots & P_{n,n} \end{pmatrix} \end{array}$$

Note that each row of the matrix sums to 1.

# Markov Process

A Markov process is a memoryless random process, i.e. a sequence of random states $S_1, S_2, \ldots$ with the Markov property.

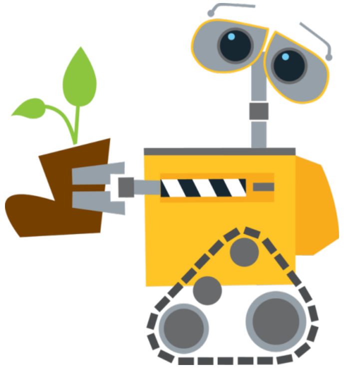
## Definition

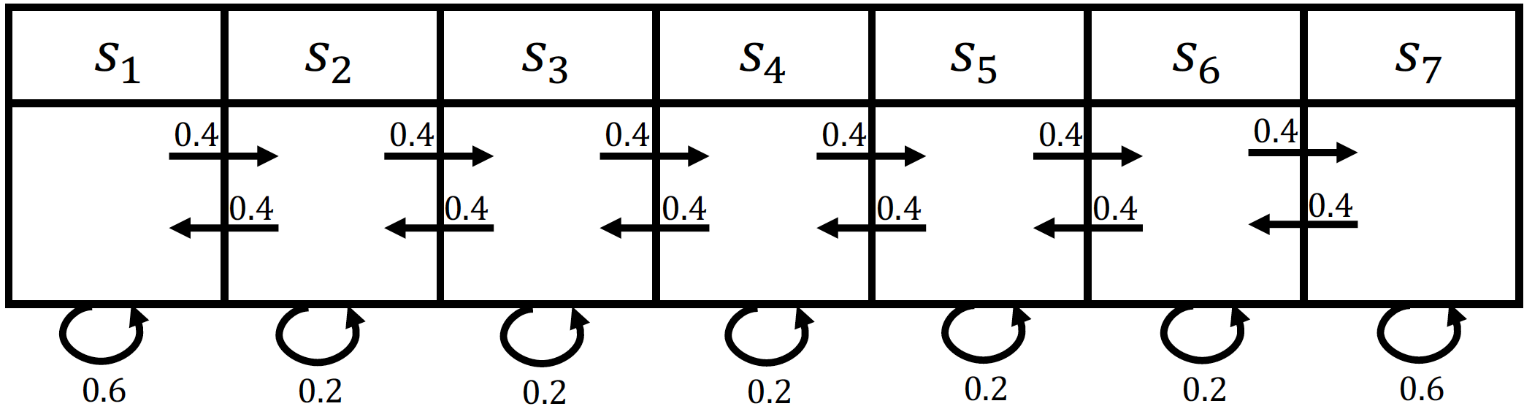
A *Markov Process* (or *Markov Chain*) is a tuple $\langle \mathcal{S}, \mathbf{P} \rangle$

- $\mathcal{S}$ is a (finite) set of states
- $\mathbf{P}$ is a state transition probability matrix,

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

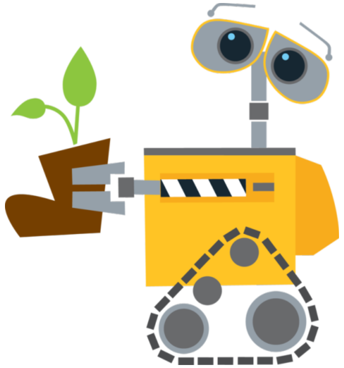
# Example: A Robot

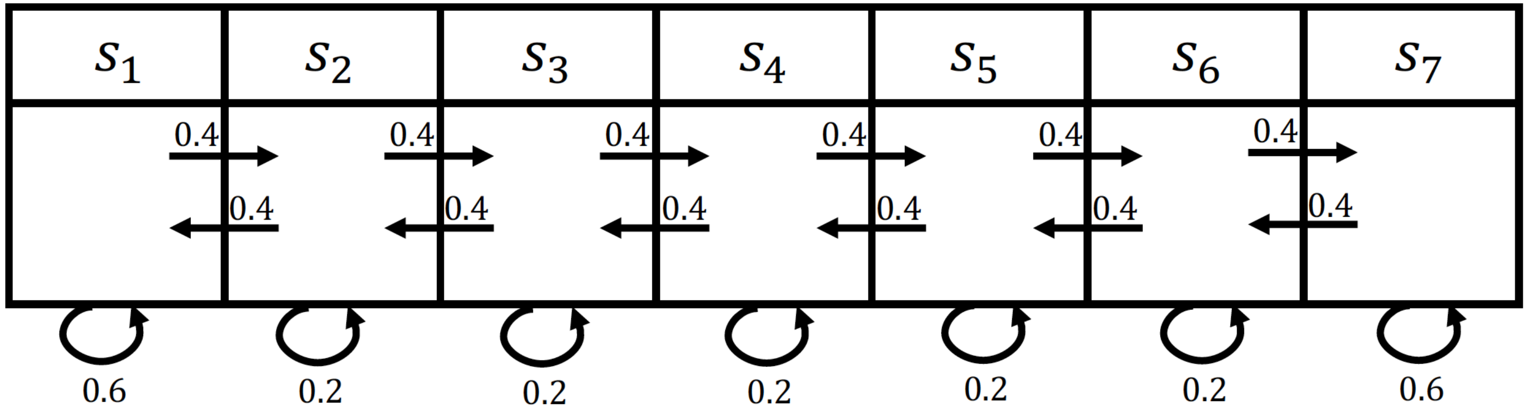





$$\mathbf{P} = \begin{pmatrix} 0.6 & 0.4 & 0 & 0 & 0 & 0 & 0 \\ 0.4 & 0.2 & 0.4 & 0 & 0 & 0 & 0 \\ 0 & 0.4 & 0.2 & 0.4 & 0 & 0 & 0 \\ 0 & 0 & 0.4 & 0.2 & 0.4 & 0 & 0 \\ 0 & 0 & 0 & 0.4 & 0.2 & 0.4 & 0 \\ 0 & 0 & 0 & 0 & 0.4 & 0.2 & 0.4 \\ 0 & 0 & 0 & 0 & 0 & 0.4 & 0.6 \end{pmatrix}$$

# Example: A Robot






Example: Sample episodes starting from $S_4$

- Episode 1: $s_4, s_5, s_6, s_7, s_7, s_7, \ldots$
- Episode 2: $s_4, s_4, s_5, s_4, s_5, s_6, \ldots$
- Episode 3: $s_4, s_3, s_2, s_1, \ldots$

# Today’s Agenda

1. Last Lecture Review
2. Markov Process
3. Markov Reward Process
4. Markov Decision Process

# Markov Reward Process

A Markov reward process is a Markov chain with values.

**Definition**

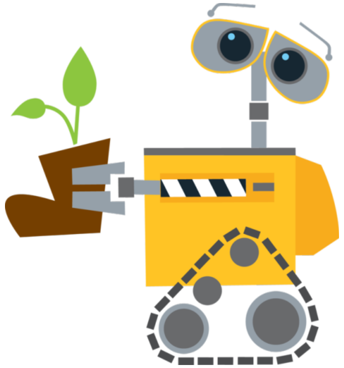
A *Markov Reward Process* is a tuple $\langle \mathcal{S}, \mathbf{P}, R, \gamma \rangle$

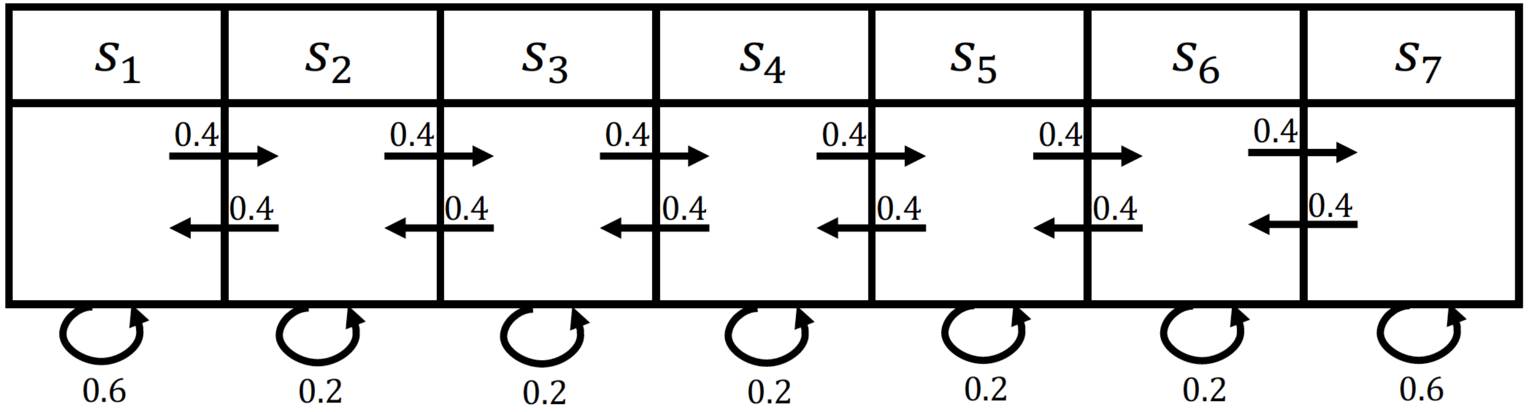
- $\mathcal{S}$ is a (finite) set of states
- $\mathbf{P}$ is a state transition probability matrix,

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

- $R$ is the reward function, $R(\mathrm{s}) = \mathbb{E}[r_{t+1} | S_t = s]$
- $\gamma$ is a discount factor, $\gamma \in [0, 1]$

# Example: A Robot






Reward: +1 in $s_1$, +10 in $s_7$, 0 in all other states

# Horizon and Return

Definition of *horizon*, $H$

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

**Definition**

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

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{k=0}^{H} \gamma^k r_{t+k}$$

- The *discount* $\gamma \in [0, 1]$ is the present value of future rewards
- The value of receiving reward $r$ after $k$ time-steps is $\gamma^k r$
- This values immediate reward above delayed reward.
  - $\gamma$ close to 0 leads to "myopic" evaluation
  - $\gamma$ close to 1 leads to "far-sighted" evaluation

# Discount Value

- The *discount* $\gamma \in [0, 1]$ is the present value of future rewards
- The value of receiving reward $r$ after $k$ time-steps is $\gamma^k r$
- This values immediate reward above delayed reward.
  - $\gamma$ close to 0 leads to "myopic" evaluation
  - $\gamma$ close to 1 leads to "far-sighted" evaluation

$$G_t = \sum_{k=0}^{H} \gamma^k r_{t+k}$$

Most Markov reward and decision processes are discounted. Why?

- Mathematically convenient to discount rewards
- Avoids infinite returns in cyclic Markov processes
- Uncertainty about the future may not be fully represented
- If the reward is financial, immediate rewards may earn more than delayed rewards
- Animal/human behaviour shows preference for immediate reward
- It is sometimes possible to use *undiscounted* Markov reward processes (i.e. $\gamma = 1$), e.g. if all sequences terminate (i.e., have finite length).

# Value Function

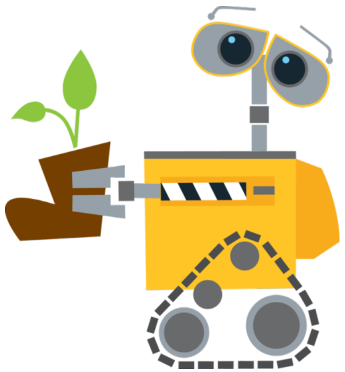
The value function $v(s)$ gives the long-term value of state $s$

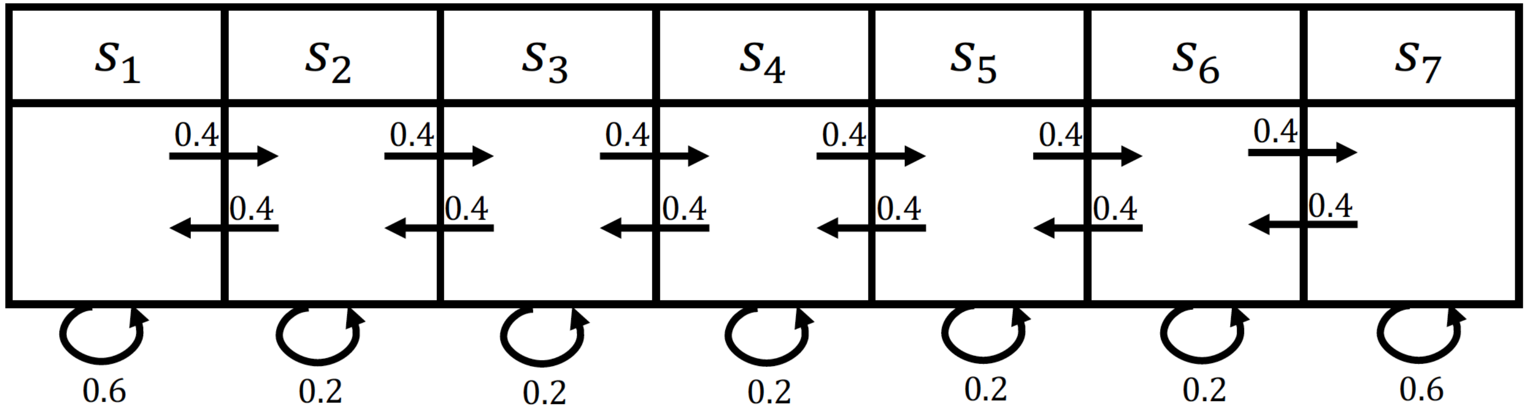
**Definition**

The *state value function* $v(s)$ of an MRP is the expected return starting from state $s$

$$v(s) = \mathbb{E}[G_t \mid S_t = s]$$

# Example: A Robot






Reward: +1 in $s_1$, +10 in $s_7$, 0 in all other states

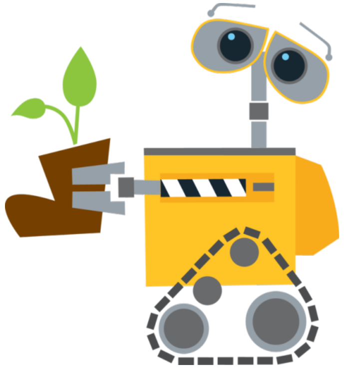
Sample returns for sample 4-step episodes, $\gamma = 1/2$

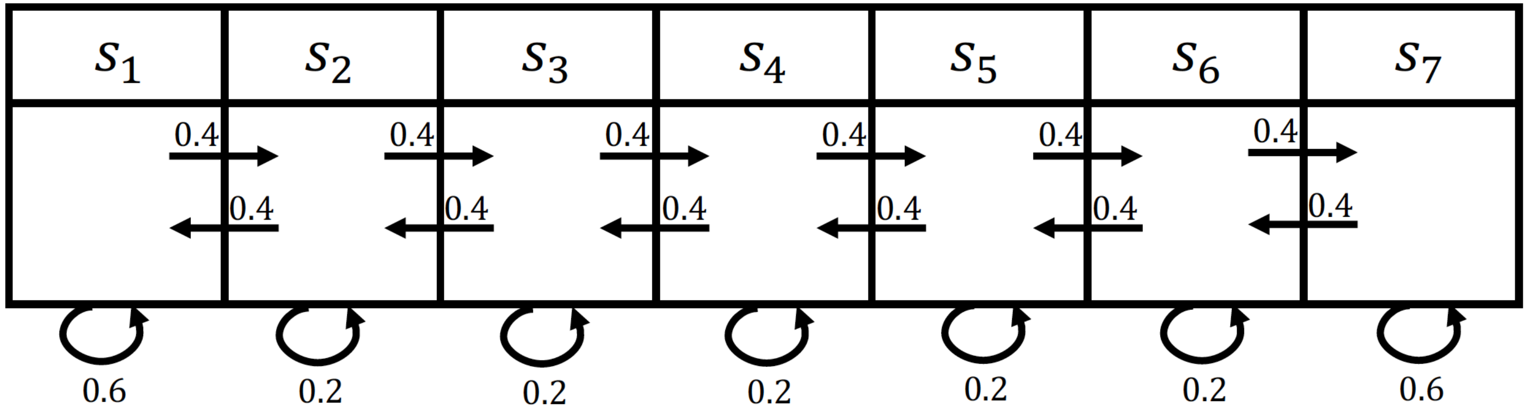
$s_4, s_5, s_6, s_7$:

$s_4, s_4, s_5, s_4$:

$s_4, s_3, s_2, s_1$:

# Example: A Robot






Reward: +1 in $s_1$, +10 in $s_7$, 0 in all other states

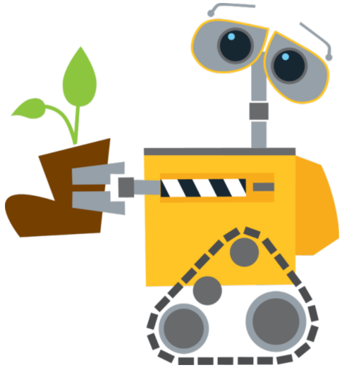
Sample returns for sample 4-step episodes, $\gamma = 1/2$

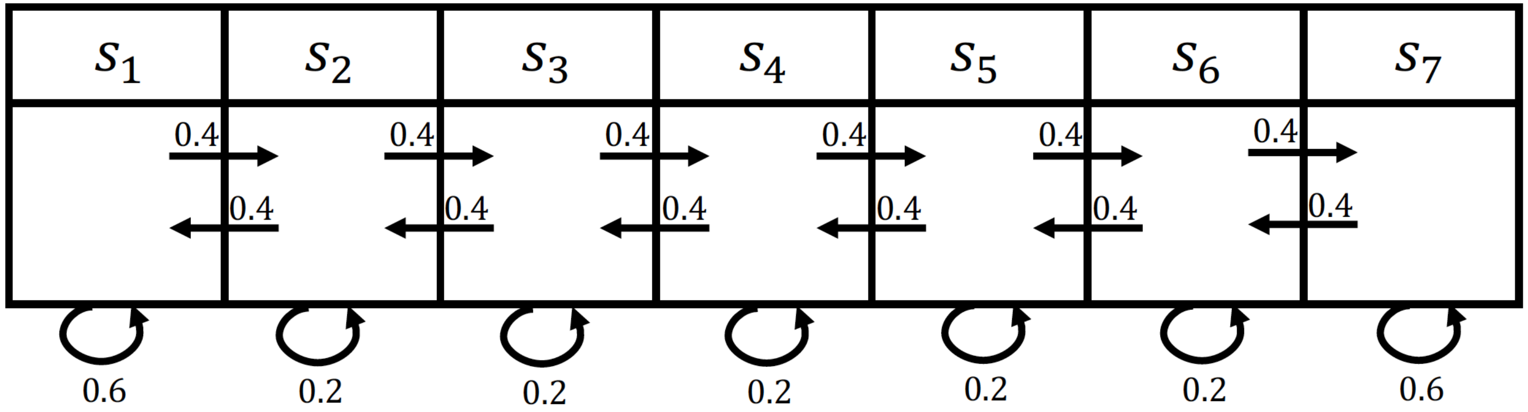
$s_4, s_5, s_6, s_7$: $0 + \frac{1}{2} \times 0 + \frac{1}{4} \times 0 + \frac{1}{8} \times 10 = 1.25$

$s_4, s_4, s_5, s_4$: $0 + \frac{1}{2} \times 0 + \frac{1}{4} \times 0 + \frac{1}{8} \times 0 = 0$

$s_4, s_3, s_2, s_1$: $0 + \frac{1}{2} \times 0 + \frac{1}{4} \times 0 + \frac{1}{8} \times 1 = 0.125$

# Example: A Robot






Reward: +1 in $s_1$, +10 in $s_7$, 0 in all other states

What about the value function?

$$V = [1.53\ 0.37\ 0.13\ 0.22\ 0.85\ 3.59\ 15.31]$$

# Computing the Value of MRP: Simulation

Could estimate by simulation (Monte Carlo)

- Generate a large number of episodes
- Average returns
- Concentration inequalities bound how quickly average concentrates to expected value
- Requires no assumption of Markov structure

# Computing the Value of MRP: Bellman Equation

$$
\begin{aligned}
v(s) &= \mathbb{E}[G_t | S_t = s] \\
&= \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots | S_t = s] \\
&= \mathbb{E}[r_t | S_t = s] + \mathbb{E}[\gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots | S_t = s] \\
&= \mathbb{E}[r_t | S_t = s] + \gamma \mathbb{E}[r_{t+1} + \gamma r_{t+2} + \cdots | S_t = s] \\
&= \mathbb{E}[r_t | S_t = s] + \gamma \mathbb{E}[G_{t+1} | S_t = s] \\
&= \mathbb{E}[r_t | S_t = s] + \gamma \mathbb{E}[v(S_{t+1}) | S_t = s]
\end{aligned}
$$

# Computing the Value of MRP: Bellman Equation

$$v(s) = \mathbb{E}[r_t | S_t = s] + \gamma \mathbb{E}[v(S_{t+1}) | S_t = s]$$

$$= \underbrace{R(s)}_{\text{immediate reward}} + \underbrace{\gamma \sum_{s' \in \mathcal{S}} P_{s,s'} v(s')}_{\text{Discounted sum of future rewards}}$$

The value function can be decomposed into two parts:

- immediate reward $R_t$
- discounted value of successor state $\gamma v(S_{t+1})$

# Computing the Value of MRP: Bellman Equation

For finite state MRP, we can express $V(s)$ using a matrix equation

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

$$\mathbf{v} = \mathbf{r} + \gamma \mathbf{P} \mathbf{v}$$

# Computing the Value of MRP: Bellman Equation

For finite state MRP, we can express $V(s)$ using a matrix equation

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

$$\mathbf{v} = \mathbf{r} + \gamma \mathbf{P} \mathbf{v}$$

$$\mathbf{v} - \gamma \mathbf{P} \mathbf{v} = \mathbf{r}$$

$$(\mathbf{I} - \gamma \mathbf{P}) \mathbf{v} = \mathbf{r}$$

$$\mathbf{v} = (\mathbf{I} - \gamma \mathbf{P})^{-1} \mathbf{r}$$

Solving directly requires taking a matrix inverse $\sim O(N^3)$

# Computing the Value of MRP: Iterative Algorithm

Use *dynamic programming Value Iteration*

- Initialize $v_0(s) = 0 \; \forall s \in \mathcal{S}$
- For $k = 1$ until convergence (i.e., iterations):
  - For all $s \in \mathcal{S}$:
    - $v_k(s) = R(s) + \gamma \sum_{s' \in \mathcal{S}} P_{s,s'} v_{k-1}(s')$

Computational complexity: for each iteration $O(N^2)$, where $N = |\mathcal{S}|$

# Today's Agenda

1. Last Lecture Review
2. Markov Process
3. Markov Reward Process
4. Markov Decision Process

# Markov Decision Process (MDP)

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

**Definition**

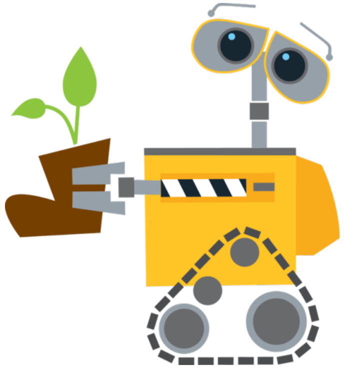
A *Markov Decision Process* is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathbf{P}, R, \gamma \rangle$

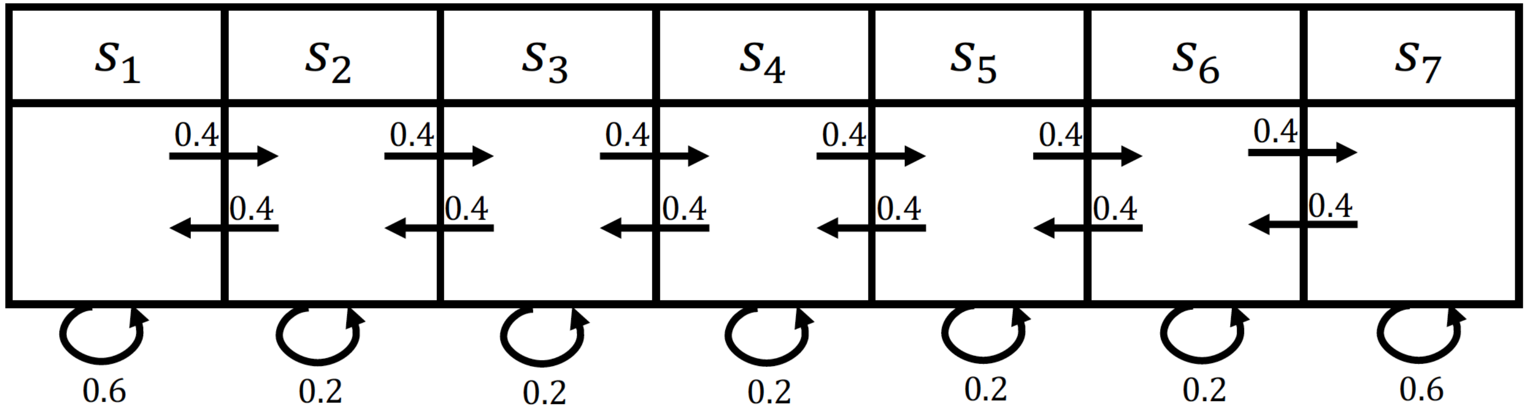
- $\mathcal{S}$ is a (finite) set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathbf{P}$ is dynamics/transition model for each action,

$$P^{a}_{s,s'} = P\,(St_{+1} = s' | S_t = s, A_t = a)$$

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

# Example: A Robot






What are these two deterministic actions?

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

# MDP: Policy

**Definition**

A *policy* $\pi$ is a distribution over actions given states,

$$\pi(a|s) = P\ (A_t = a|S_t = s)$$

- A policy fully defines the behaviour of an agent
- MDP policies depend on the current state (not the history)
- i.e. Policies are *stationary* (time-independent),

$$A_t \sim \pi(\cdot\,|S_t), \forall t > 0$$

# MDP Given a Policy

- Given an MDP $M = \langle \mathcal{S}, \mathcal{A}, \mathbf{P}, R, \gamma \rangle$ and a policy $\pi$
- The state sequence $S_1, S_2, \ldots$ is a Markov process $\langle \mathcal{S}, \mathbf{P}^{\pi} \rangle$
- The state and reward sequence $S_1, R_2, S_2, \ldots$ is a Markov reward process $\langle \mathcal{S}, \mathbf{P}^{\pi}, R^{\pi}, \gamma \rangle$

With

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

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