Markov Decision Processes (MDPs)

Industrial AI Lab.

Prof. Seungchul Lee

Today

- Markov Chain
- Markov Reward Process
- Markov Decision Process

Markov Process

- A Markov process is a memoryless random process, i.e., a sequence of random states $s_1, s_2, \cdots$ with the Markov property

Definition

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

- $\mathcal{S}$ is a (finite) set of states
- $\mathcal{P}$ is a state transition probability matrix,
  $\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$

Markov Reward Process

Markov Chains with Rewards

- Suppose that each transition in a Markov chain is associated with a reward, r
- As the Markov chain proceeds from state to state, there is an associated sequence of rewards
- Discount factor γ

- Later, we will study dynamic programming and Markov decision theory ⇒ Markov Decision Process (MDP)
  - These topics include a *decision maker*, *policy maker*, or *control* that modify both the transition probabilities and the rewards at each trial of the Markov chain.

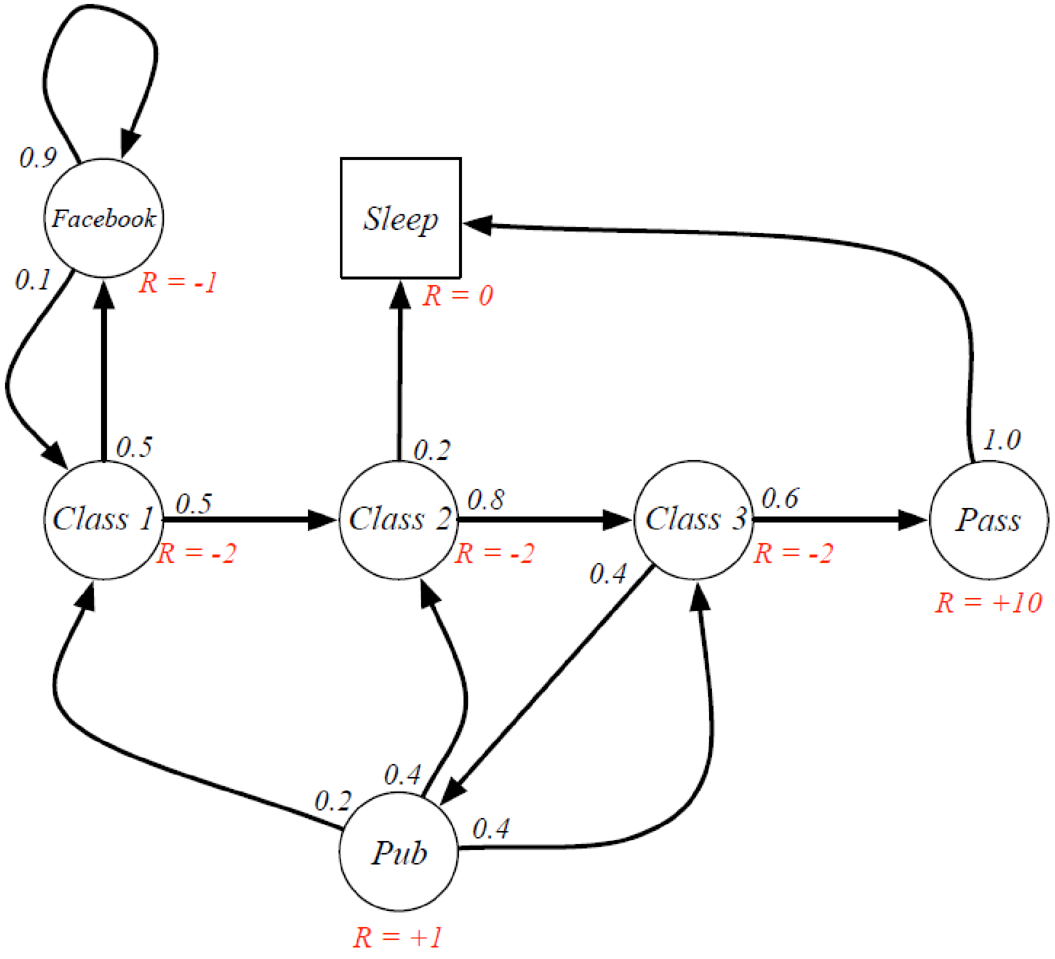
Markov Reward Process (MRP)

Definition

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

- $\mathcal{S}$ is a finite set of states
- $\mathcal{P}$ is a state transition probability matrix,
  $\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}_s = \mathbb{E}\left[R_{t+1} \mid S_t = s\right]$
- γ is a discount factor, $\gamma \in [0, 1]$

Student MRP



0.9
Facebook
0.1
R = -1
Sleep
R = 0
0.5
0.2
1.0
Class 1
0.5
Class 2
0.8
Class 3
0.6
Pass
R = -2
R = -2
0.4
R = -2
R = +10
0.4
0.2
0.4
Pub
R = +1

Reward over Multiple Transitions

- Return

Definition

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

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

Value Function

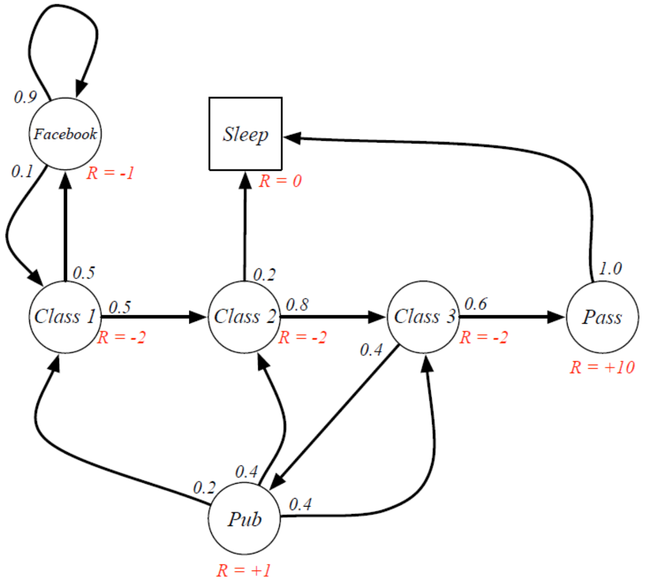
- The value function $v(s)$ gives the long-term value of state s

Definition

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

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

Student MRP Returns

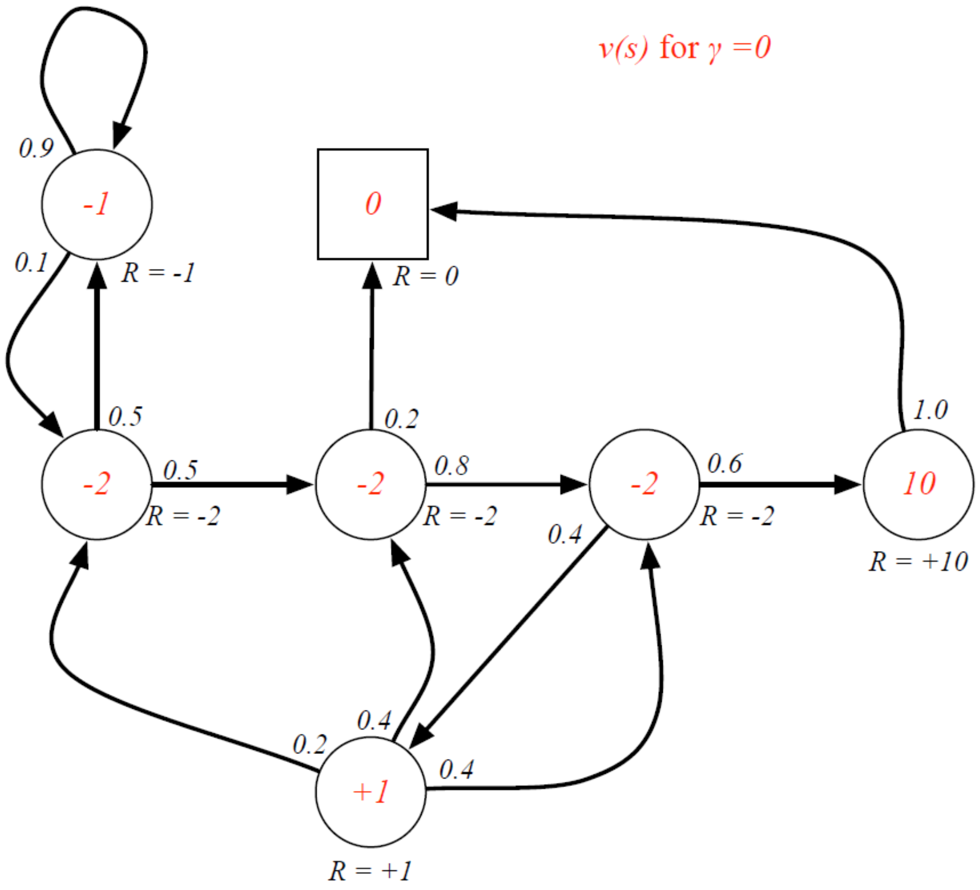


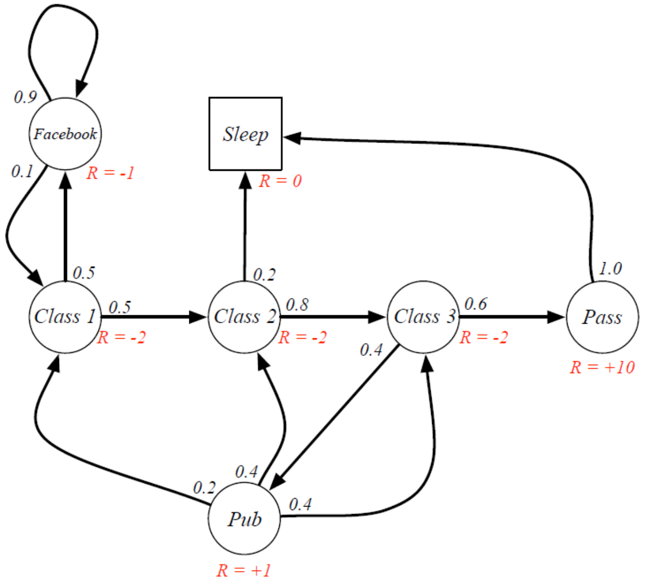
Sample returns for Student MRP:
Starting from $S_1 = \text{C1}$ with $\gamma = \frac{1}{2}$

$$G_1 = R_2 + \gamma R_3 + \ldots + \gamma^{T-2} R_T$$

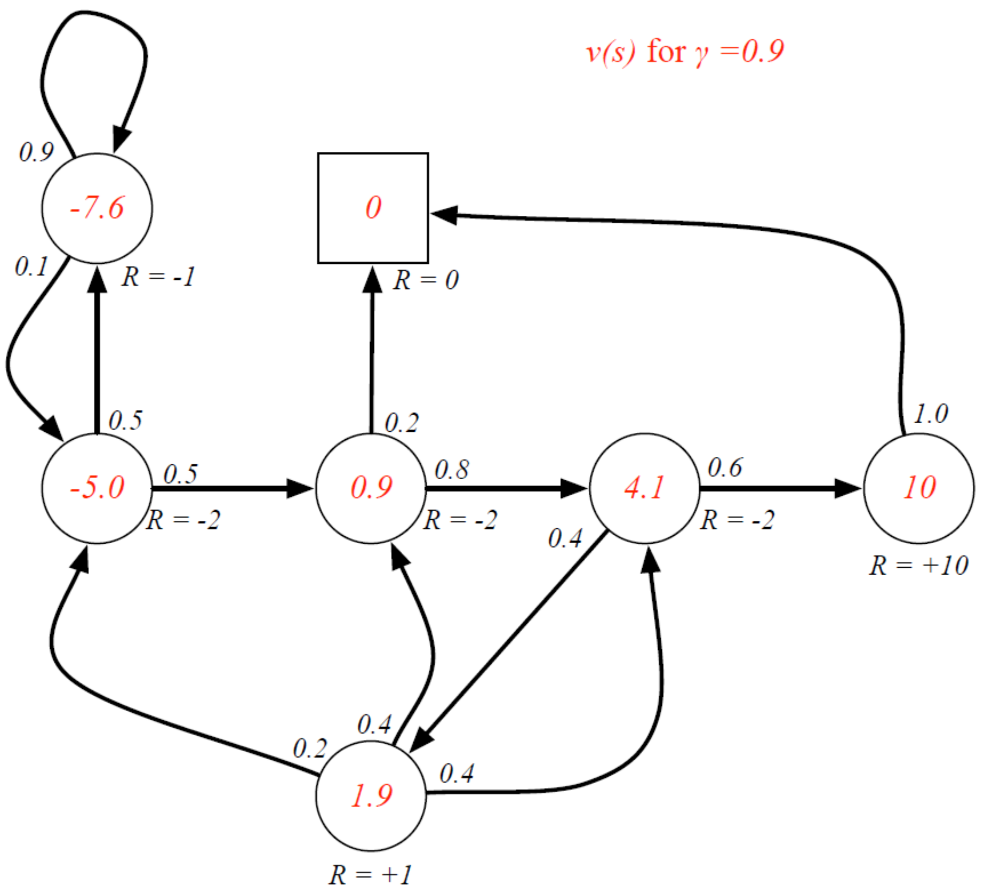
C1 C2 C3 Pass Sleep	$v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 10 * \frac{1}{8}$	$= -2.25$
C1 FB FB C1 C2 Sleep	$v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16}$	$= -3.125$
C1 C2 C3 Pub C2 C3 Pass Sleep	$v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 1 * \frac{1}{8} - 2 * \frac{1}{16} \ldots$	$= -3.41$
C1 FB FB C1 C2 C3 Pub C1 ...	$v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16} \ldots$	$= -3.20$
FB FB FB C1 C2 C3 Pub C2 Sleep		

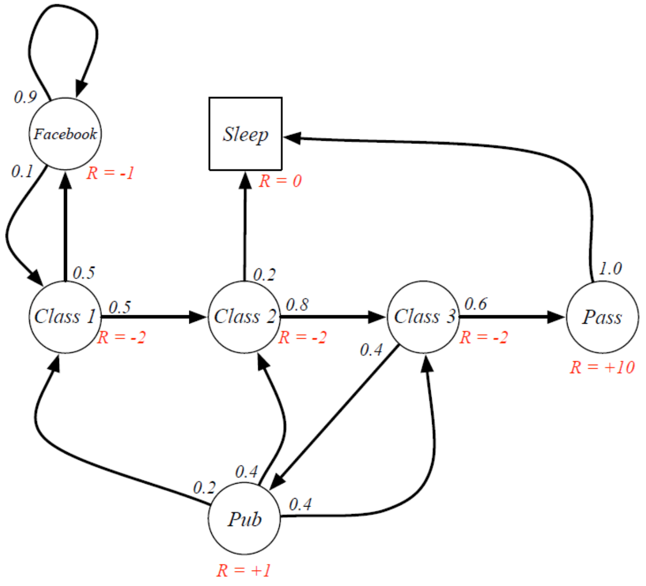
State-Value Function for Student MRP (1)



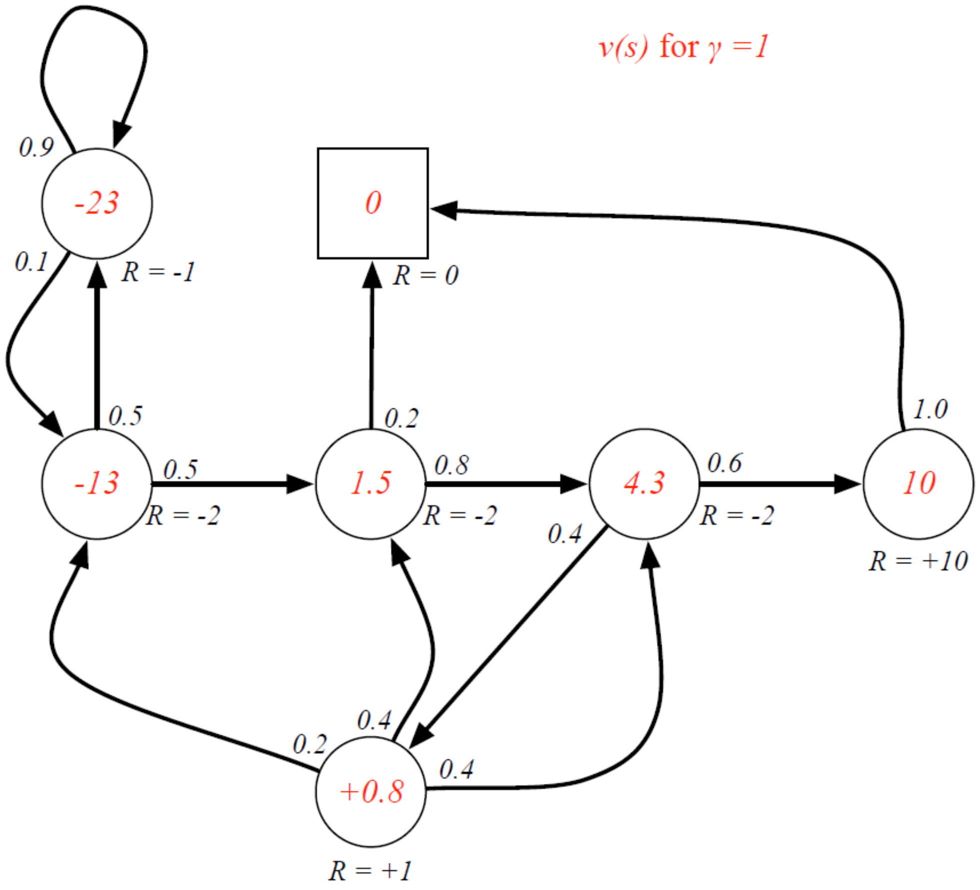


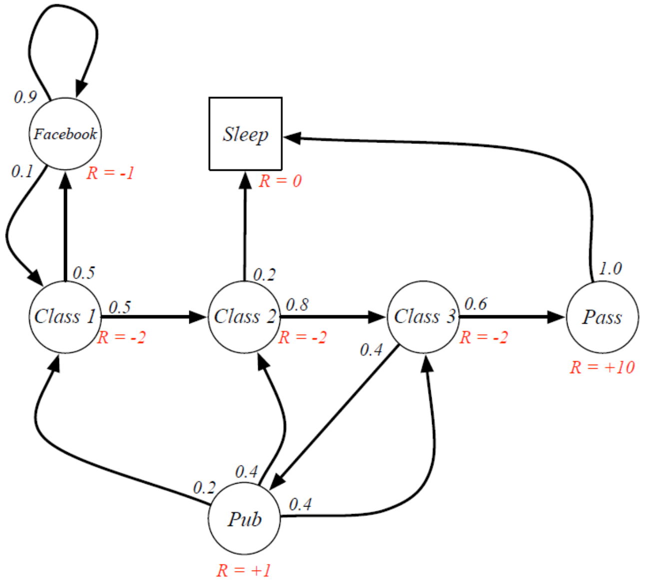
State-Value Function for Student MRP (2)





State-Value Function for Student MRP (3)





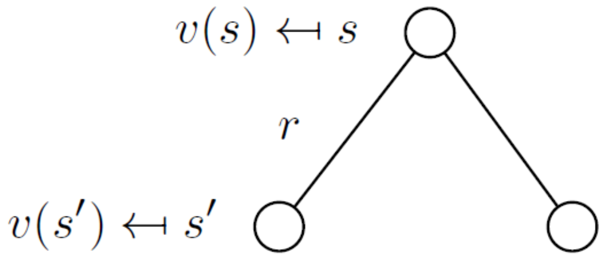
Bellman Equations for MRP (1)

- The value function can be decomposed into two parts:
  - Immediate reward R_{t+1}
  - Discounted value of successor state $\gamma v(S_{t+1})$

$$
\begin{aligned}
v(s) &= \mathbb{E}\left[G_t \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \ldots\right) \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s\right]
\end{aligned}
$$

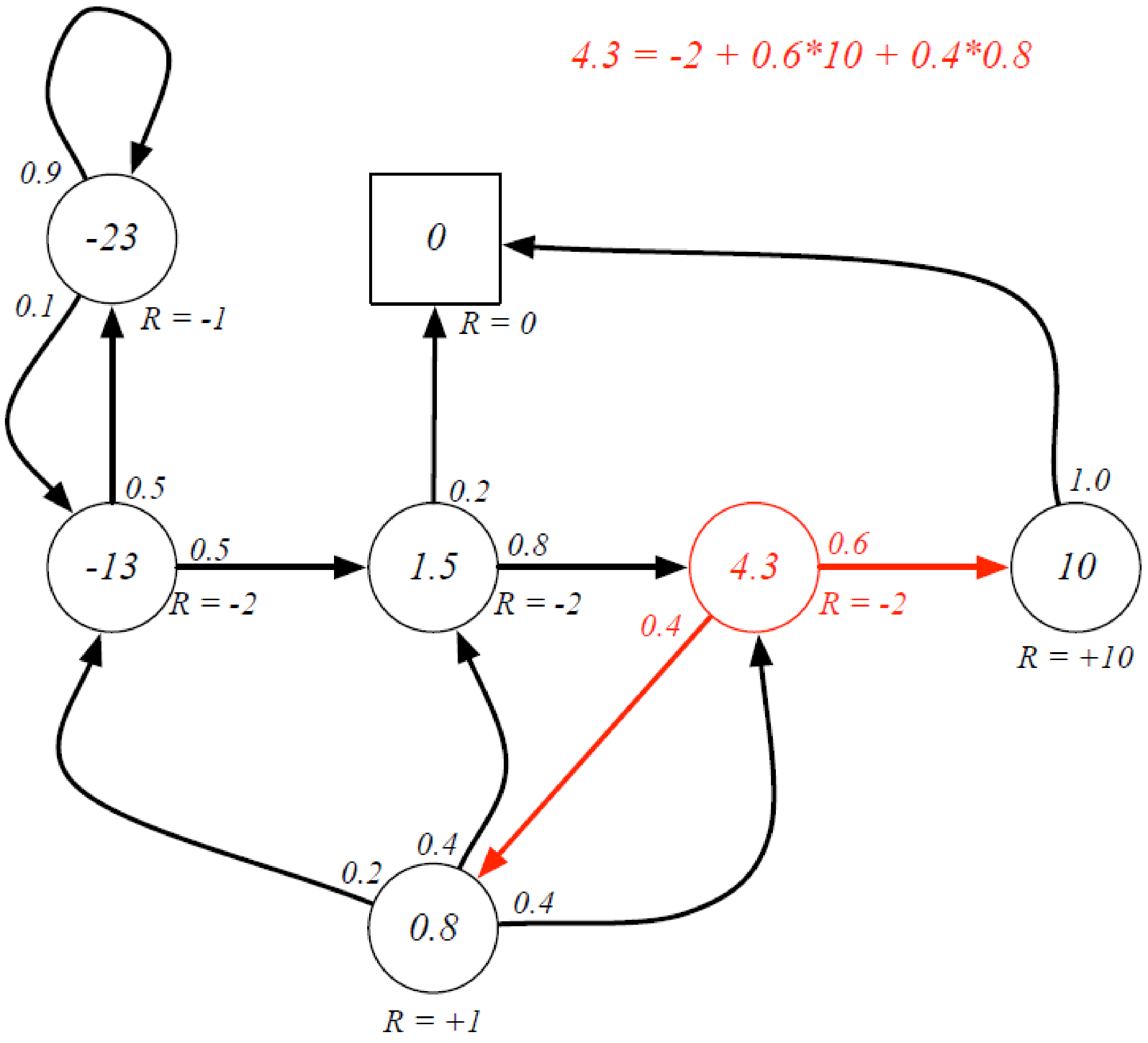
Bellman Equations for MRP (2)

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



$$v(s) = \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} v(s')$$

Bellman Equation for Student MRP



Bellman Equation in Matrix Form

- The Bellman equation can be expressed concisely using matrices,

$$v = \mathcal{R} + \gamma \mathcal{P} v$$

where v is a column vector with one entry per state

$$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} R_1 \\ \vdots \\ R_n \end{bmatrix} + \gamma \begin{bmatrix} p_{11} & \cdots & p_{1n} \\ & \vdots & \\ p_{n1} & \cdots & p_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix}$$

Solving the Bellman Equation

- The Bellman equation is a linear equation
- It can be solved directly:

$$
\begin{aligned}
v &= \mathcal{R} + \gamma \mathcal{P} v \\
(I - \gamma \mathcal{P})\, v &= \mathcal{R} \\
v &= (I - \gamma \mathcal{P})^{-1} \mathcal{R}
\end{aligned}
$$

- Direct solution only possible for small MRP
- There are many *iterative* methods for large MRP
  - Dynamic programming
  - Monte-Carlo simulation
  - Temporal-difference learning

Markov Decision Process

Markov Decision Process

- So far, we analyzed the passive behavior of a Markov chain with rewards

- A Markov decision process (MDP) is a Markov reward process with decisions (or actions).

Definition

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

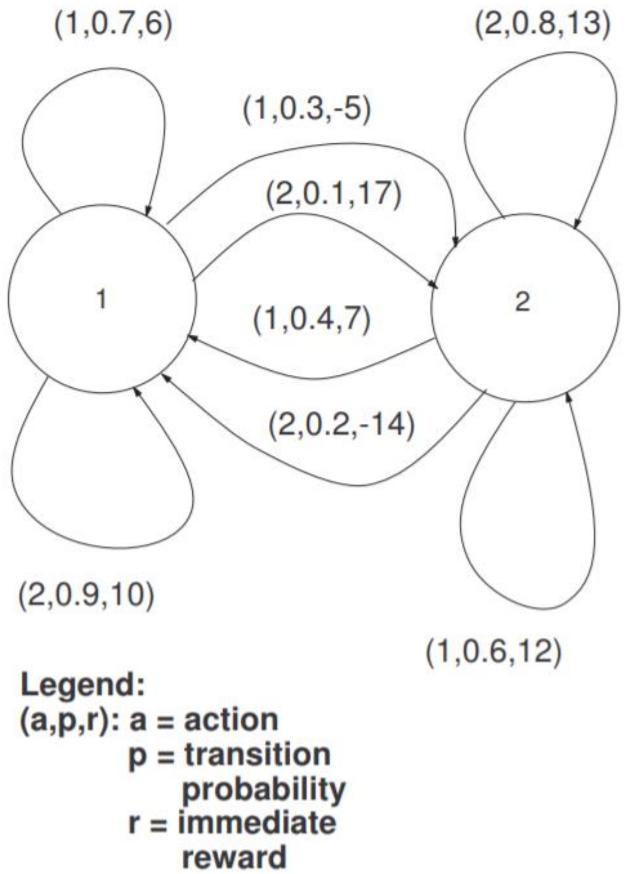
- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}^a_s = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$
- γ is a discount factor $\gamma \in [0, 1]$.

Example

- P_a: transition probability matrix for action a
- R_a: transition reward matrix for action a

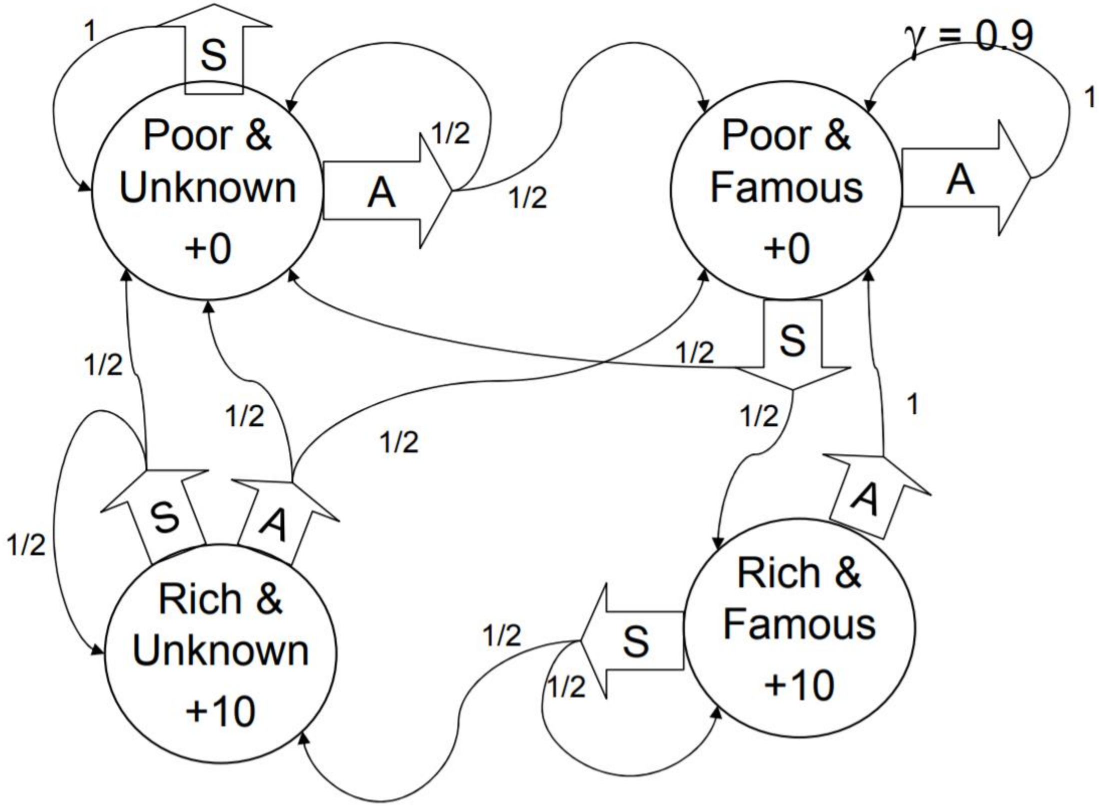
$$\mathbf{P}_1 = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}; \mathbf{P}_2 = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix};$$

$$\mathbf{R}_1 = \begin{bmatrix} 6 & -5 \\ 7 & 12 \end{bmatrix}; \mathbf{R}_2 = \begin{bmatrix} 10 & 17 \\ -14 & 13 \end{bmatrix}.$$



Example

- You run a startup company.
  - In every state, you must choose between Saving money or Advertising

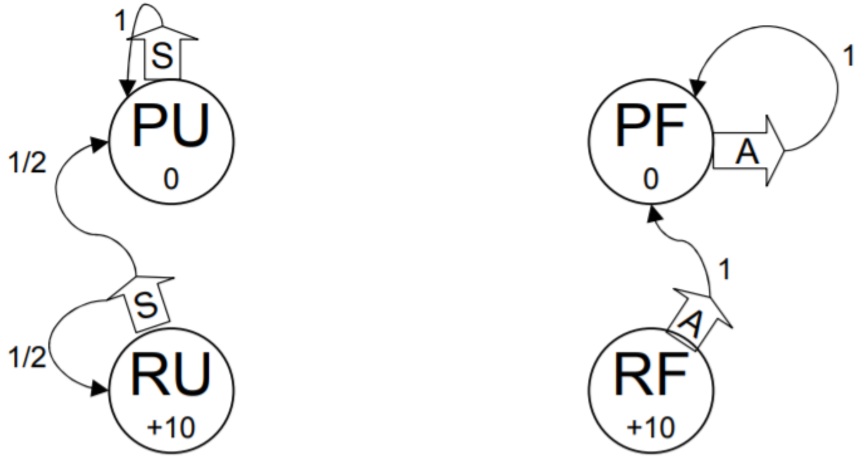


Policy

- A policy is a mapping from states to actions, $\pi: S \to A$
- Example: two policies

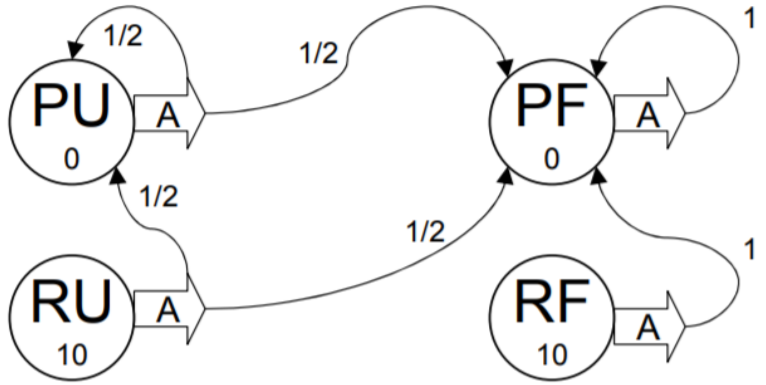
Policy Number 1:

STATE →	ACTION
PU	S
PF	A
RU	S
RF	A



Policy Number 2:

STATE →	ACTION
PU	A
PF	A
RU	A
RF	A



Policies

- A policy is a mapping from states to actions, $\pi: S \rightarrow A$
- A policy fully defines the behavior of an agent
- Let P^{π} be a matrix containing probabilities for each transition under policy π
- Given an MDP $\mathcal{M} = \langle S, A, P, R, \gamma \rangle$ and a policy π
  - The state sequence $s_1, s_2, \cdots$ is a Markov process $\langle S, P^{\pi} \rangle$
  - The state and reward sequence is a Markov reward process $\langle S, P^{\pi}, R^{\pi}, \gamma \rangle$

Questions on MDP Policy

- How many possible policies in our example?
- Which of the above two policies is best?
- How do you compute the *optimal* policy?

State-Value Function

Definition

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

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

- Given the policy π, the state-value function can again be decomposed into immediate reward plus discounted value of successor state (recursively)

$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s \right]$$

Bellman Expectation Equation

$$v_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s\right]$$

$$\Downarrow$$

$$\upsilon_\pi(s) = R(s) + \gamma \sum_{s' \in S} P^\pi_{ss'}\, \upsilon_\pi\left(s'\right)$$

- The Bellman expectation equation can be expressed concisely in a matrix form,

$$\upsilon_\pi = R + \gamma P^\pi \upsilon_\pi$$

with direct solution

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

Optimal Policy and Optimal Value Function

- The optimal policy is the policy that achieves the highest value for every state

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

  and its optimal value function is written $v^*(s)$

- We can directly define the *optimal value function* using Bellman optimality equation

$$v^*(s) = R(s) + \gamma \max_{a} \sum_{s' \in S} P_{ss'}^{a} \, v^*\left(s'\right)$$

  and *optimal policy* is simply the action that attains this max

$$\pi^*(s) = \arg\max_{a} \sum_{s' \in S} P_{ss'}^{a} \, v_\pi(s')$$

Computing the Optimal Policy

- Value iteration
  - According to Bellman optimality equation

1) initialize an estimate for the value function arbitrarily

$$v(s) \leftarrow 0 \quad \forall s \in S$$

2) Repeat, update

$$v(s) \leftarrow R(s) + \gamma \max_a \sum_{s' \in S} P_{ss'}^a \, v\left(s'\right), \quad \forall s \in S$$

Solving the Bellman Optimality Equation

- Bellman Optimality Equation is non-linear
- No closed form solution (in general)

- (Will learn later) many iterative solution methods
  - Value Iteration
  - Policy Iteration
  - Q-learning
  - SARSA

➢ You will get into details in the course of reinforcement learning

Example: Gridworld Domain

- Simple grid world with a goal state with reward and a “bad state” with reward -100
- Actions move in the desired direction with probably 0.8, in one of the perpendicular directions with
- Taking an action that would bump into a wall leaves agent where it is