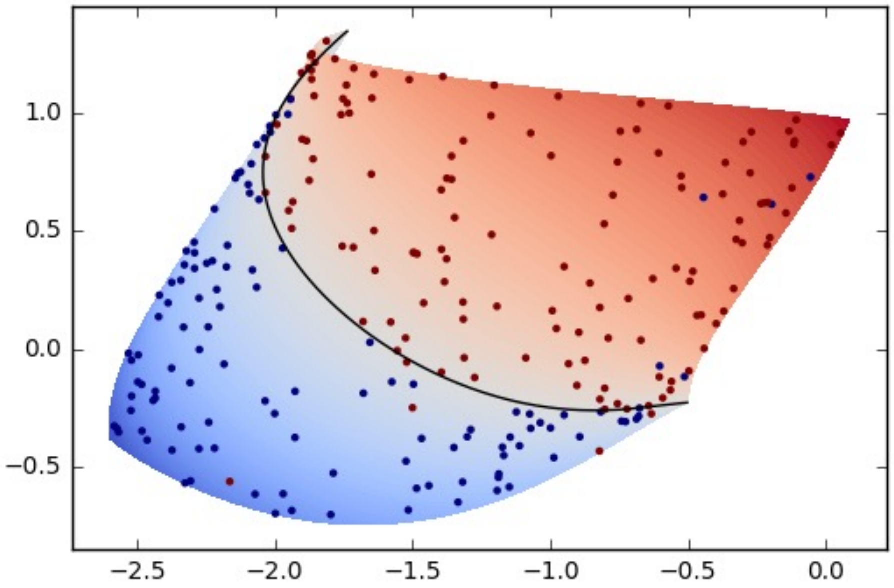
Lecture 25: Reinforcement Learning Part 2



Gavin Kerrigan

Spring 2023

Some slides adapted from Padhraic Smyth, Alex Ihler

Announcements

- HW5 released
  - Implementing & experimenting with kMeans
  - Due in ~2 weeks (6/9)

- Discussion tomorrow
  - Project workshop
  - Come prepared with questions & progress

- Final course eval
  - evaluations.eee.uci.edu
  - Please fill out!
  - Due 6/12

Announcements

Week 9				
Monday 5/29	**No class (Memorial Day)**			
Wednesday 5/31	Lec25	Reinforcement Learning		
Thursday 6/1	Dis09	Project Workshop		
Friday 6/2	Lec26	Reinforcement Learning		
Week 10				
Monday 6/5	Lec27	Advanced Topics		
Wednesday 6/7	Lec28	Advanced Topics		
Thursday 6/8	Dis010	Final Exam Review		
Friday 6/9	Lec29	Final Exam Review	**HW5 Due**	
Finals Week				
Monday 6/12			**Project Due**	
Wednesday 6/14		**Final Exam 1:30-3:30pm**		

Final Exam

- Weds June 14, 1:30-3:30pm
  - In-person, usual lecture hall

- Same format as midterm exam
  - All you need is a pen or pencil
  - Closed book: no notes, books, etc
  - No electronic devices (calculators, phones, etc)

- Assigned Seating
  - Same process as for midterm
  - Seating will be announced via Ed
  - Same list of students requesting left-handed seats
  - Same list of students for DSC accommodation

Final Exam

- Exam is cumulative
  - … but more focused on material after midterm

- What to study?
  - Lecture slides, homeworks
  - Be able to do all algorithms by hand
  - Highly recommend studying midterm exam solutions

- Sample final will be posted soon

Topics not on final exam

- Any slides marked with 
- AdaBoost
- Hierarchical clustering
- Advanced topics lectures (Week 10)
- Python syntax/programming

Questions?

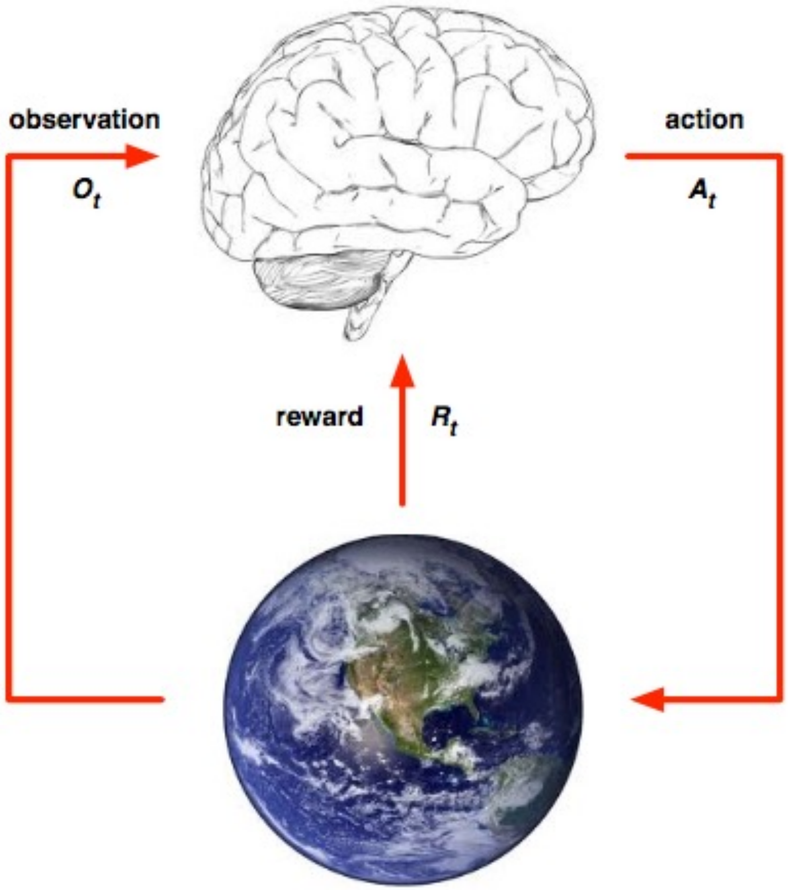
Reinforcement Learning Recap

Markov Processes

Markov Reward Processes

Markov Decision Processes

Agent-Environment Interface



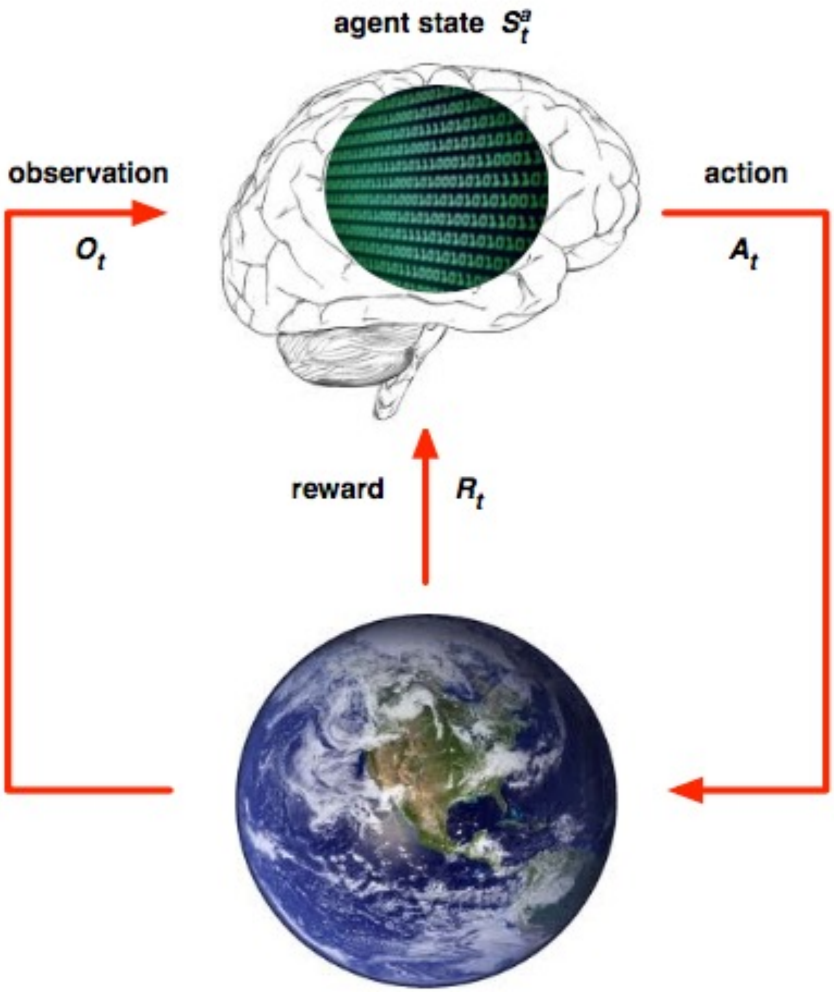
Agent

- Decides on an action
- Observes the state of the environment
- Receives a reward
- Goal: take actions that result in the highest total reward

Environment

- Executes the action
- Computes the next observation
- Computes the next reward

Agent State, S_t



History: everything that happened so far

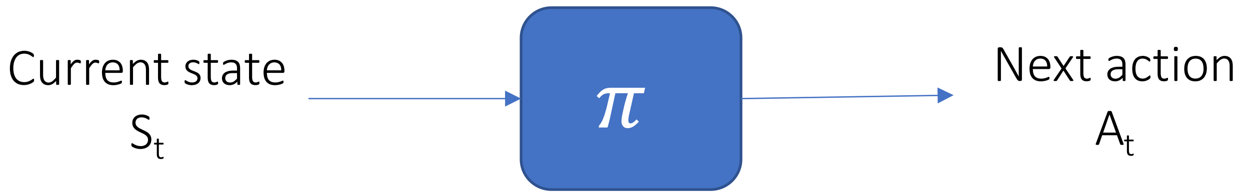
$$H_t = O_1R_1A_1O_2R_2A_2O_3R_3,\ldots,A_{t-1}O_tR_t$$

State, S_t could be... O_t
O_tR_t
$A_{t-1}O_tR_t$
$O_{t-3}O_{t-2}O_{t-1}O_t$

In general, $S_t = f(H_t)$

You, as AI designer, specify this function

Agent Policy, π



Deterministic Policy: $A_t = \pi(S_t)$

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

Good policy: Leads to larger cumulative reward

Bad policy: Leads to worse cumulative reward

Multi-Armed Bandits

A simple RL problem we will explore in-depth

Basic problem:

- Have K different slot machines ("Bandits")
- At each time step, agent can choose one machine and receives a random reward
- Each has some unknown average reward θ_i (e.g. a number between 0 and 1)

Multi-Armed Bandits

A simple RL problem we will explore in-depth

Agent must balance between…

- Playing machines where little is known about the reward (“Exploration”)
- Playing machines where the reward is believed to be high (“Exploitation”)

Various strategies for doing this:

- Explore-then-Exploit
- ϵ-Greedy
- Decreasing ϵ

Questions?

Reinforcement Learning Recap

Markov Processes

Markov Reward Processes

Markov Decision Processes

Where We're Headed

Markov Property

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

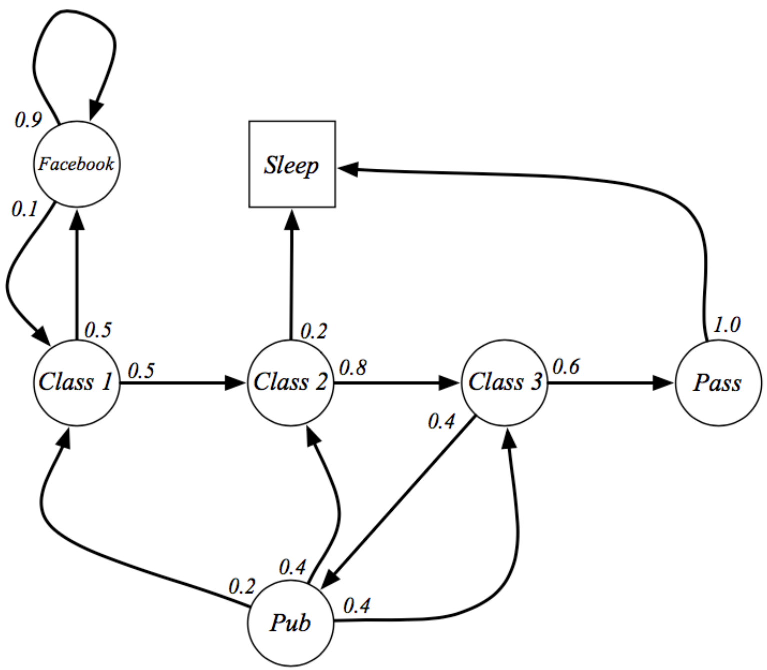
Definition

A state S_t is *Markov* if and only if

$$\mathbb{P}\left[S_{t+1} \mid S_t\right] = \mathbb{P}\left[S_{t+1} \mid S_1, ..., S_t\right]$$

- 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

Student Markov Chain



$$\mathcal{P} = \begin{array}{rc} & \begin{array}{ccccccc} C1 & C2 & C3 & Pass & Pub & FB & Sleep \end{array} \\ \begin{array}{r} C1 \\ C2 \\ C3 \\ Pass \\ Pub \\ FB \\ Sleep \end{array} & \left[\begin{array}{ccccccc} & 0.5 & & & & 0.5 & \\ & & 0.8 & & & & 0.2 \\ & & & 0.6 & 0.4 & & \\ & & & & & & 1.0 \\ 0.2 & 0.4 & 0.4 & & & & \\ 0.1 & & & & & 0.9 & \\ & & & & & & 1 \end{array} \right] \end{array}$$

Reinforcement Learning Recap

Markov Processes

Markov Reward Processes

Markov Decision Processes

Expected Values

Expected values are ways of formalizing averages

For a discrete random variable X, its expected value is:

$$\mathbb{E}[X] = \sum x \, P(x)$$

Expected Values

Expected values are ways of formalizing averages

For a discrete random variable X, its expected value is:

$$\mathbb{E}[X] = \sum x \, P(x)$$

Example:

$$P(X = 1) = 1/4 \quad P(X = 2) = 1/2 \quad P(X = 3) = 1/4$$

$$\mathbb{E}[X] = \frac{1}{4}(1) + \frac{1}{2}(2) + \frac{1}{4}(3) = 2$$

Markov Reward Process

A Markov reward process is a Markov chain with values.

Markov Reward Process

A Markov reward process is a Markov chain with values.

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,
  $\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s]$

Markov Reward Process

A Markov reward process is a Markov chain with values.

Definition

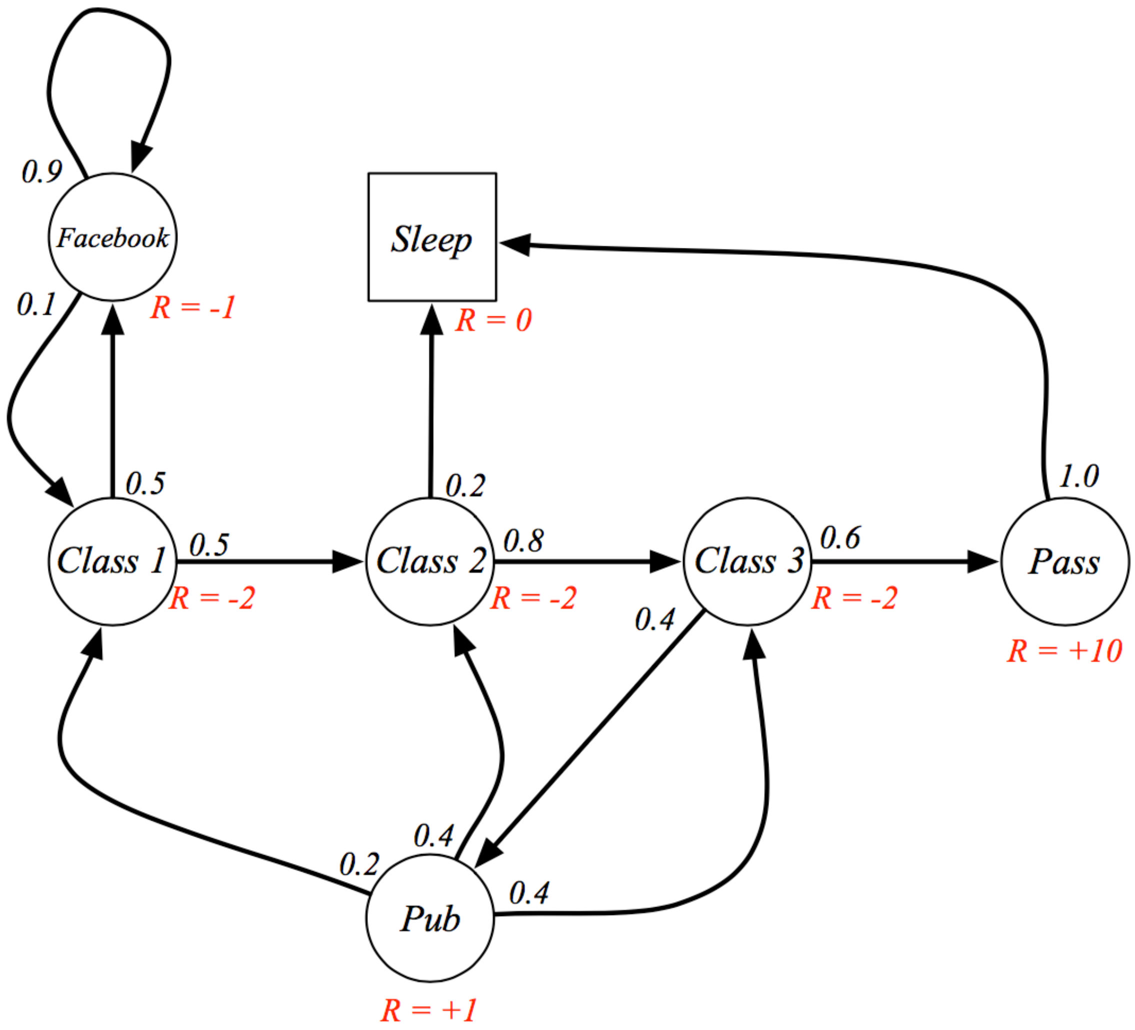
A *Markov Reward Process* is a tuple $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ 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, $\gamma \in [0, 1]$

Reward can be stochastic or deterministic (here, we often consider deterministic)

R_s is the average reward we receive from being in state s

Student Markov Chain with Rewards



Returns

Definition

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

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

Returns

Definition

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

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

- 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 $\gamma^k R$.

Returns

Definition

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

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

- 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 "myopic" evaluation
  - γ close to 1 leads to "far-sighted" evaluation

Why discount?

Most Markov reward and decision processes are discounted. Why?

Why discount?

Most Markov reward and decision processes are discounted. Why?

- Mathematically convenient to discount rewards
- Avoids infinite returns in cyclic Markov processes

Why discount?

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 interest than delayed rewards

Why discount?

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 interest 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.

Questions?

Value Functions

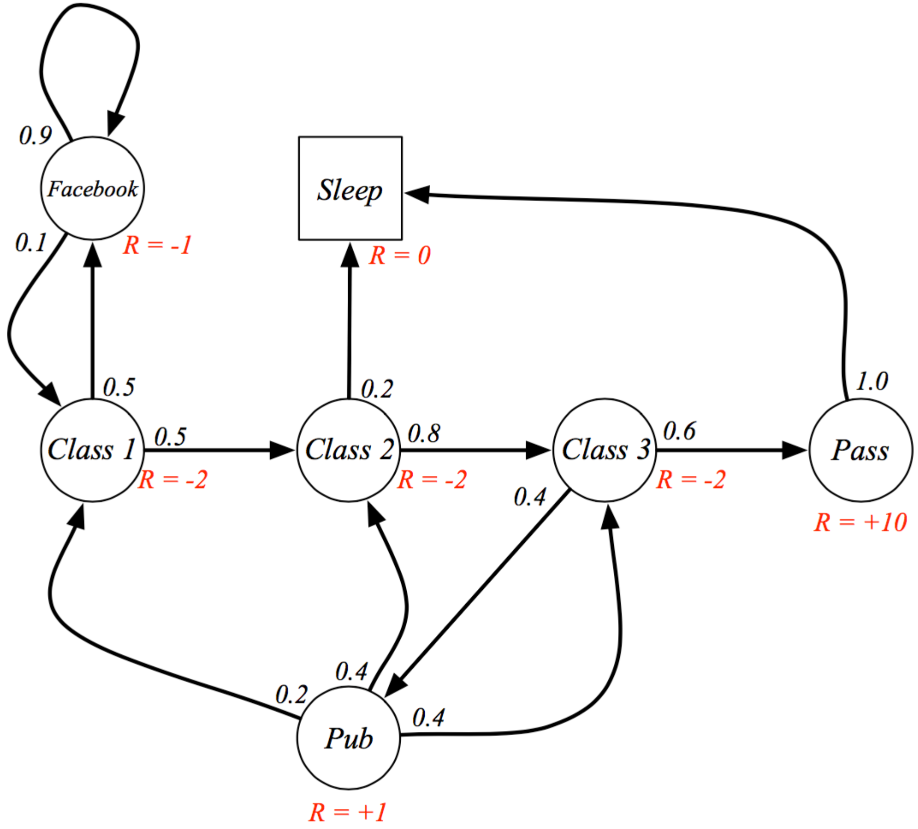
The value function $v(s)$ gives the long-term value of state s

Definition

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

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

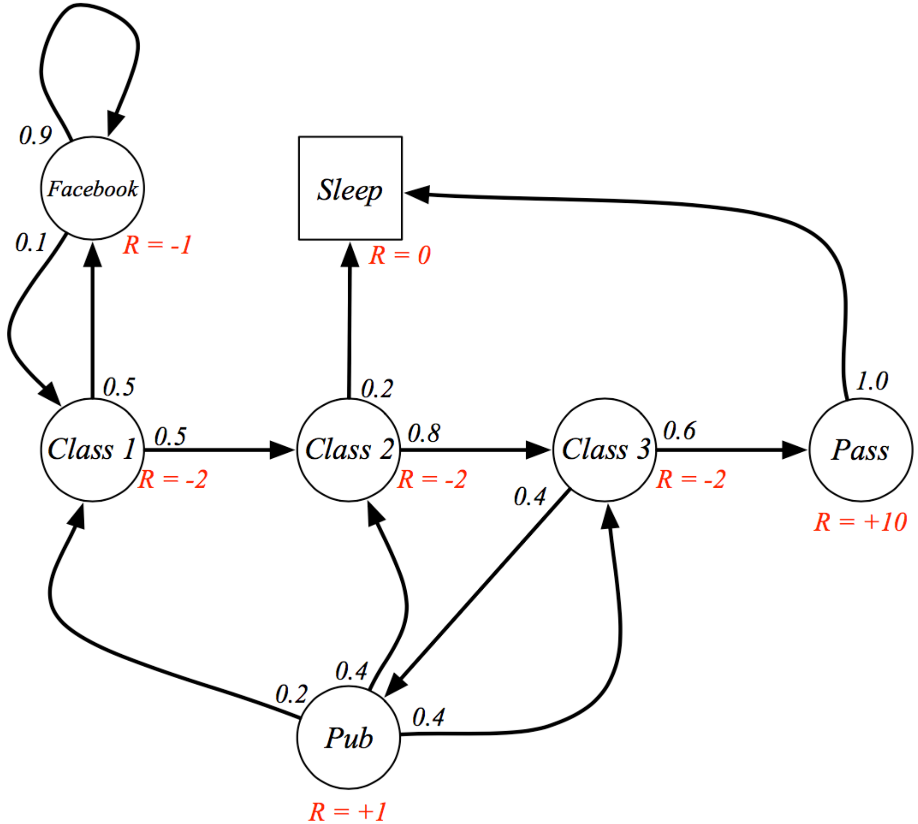
Estimating v(s)



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

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

Estimating v(s)



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

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

C1 C2 C3 Pass Sleep

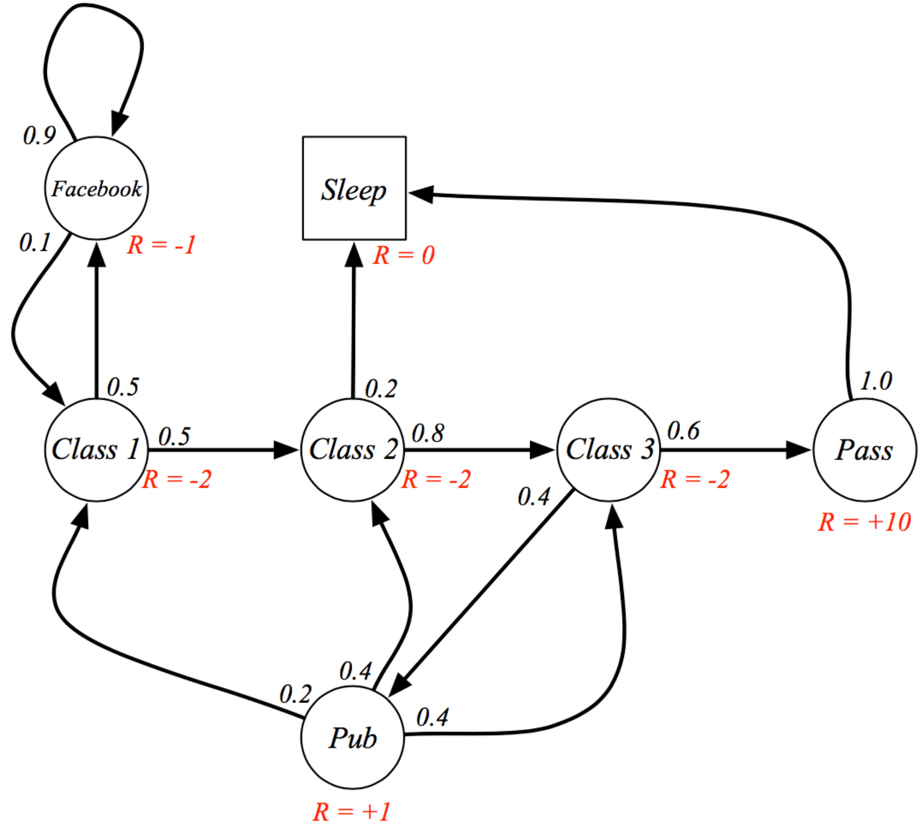
C1 FB FB C1 C2 Sleep

C1 C2 C3 Pub C2 C3 Pass Sleep

C1 FB FB C1 C2 C3 Pub C1 ...

FB FB FB C1 C2 C3 Pub C2 Sleep

Estimating v(s)

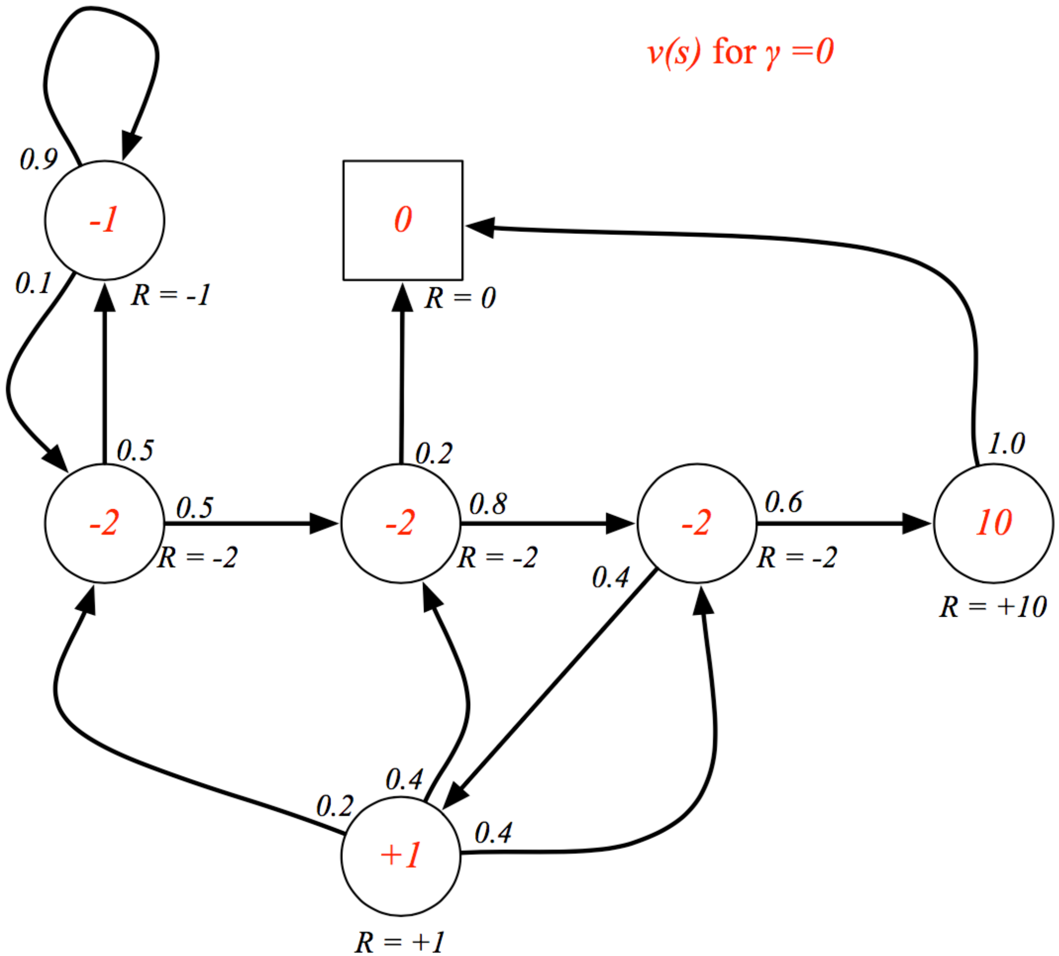


Sample returns for Student MRP:
Starting from $S_1 = \text{C1}$ with $\gamma = \frac{1}{2}$

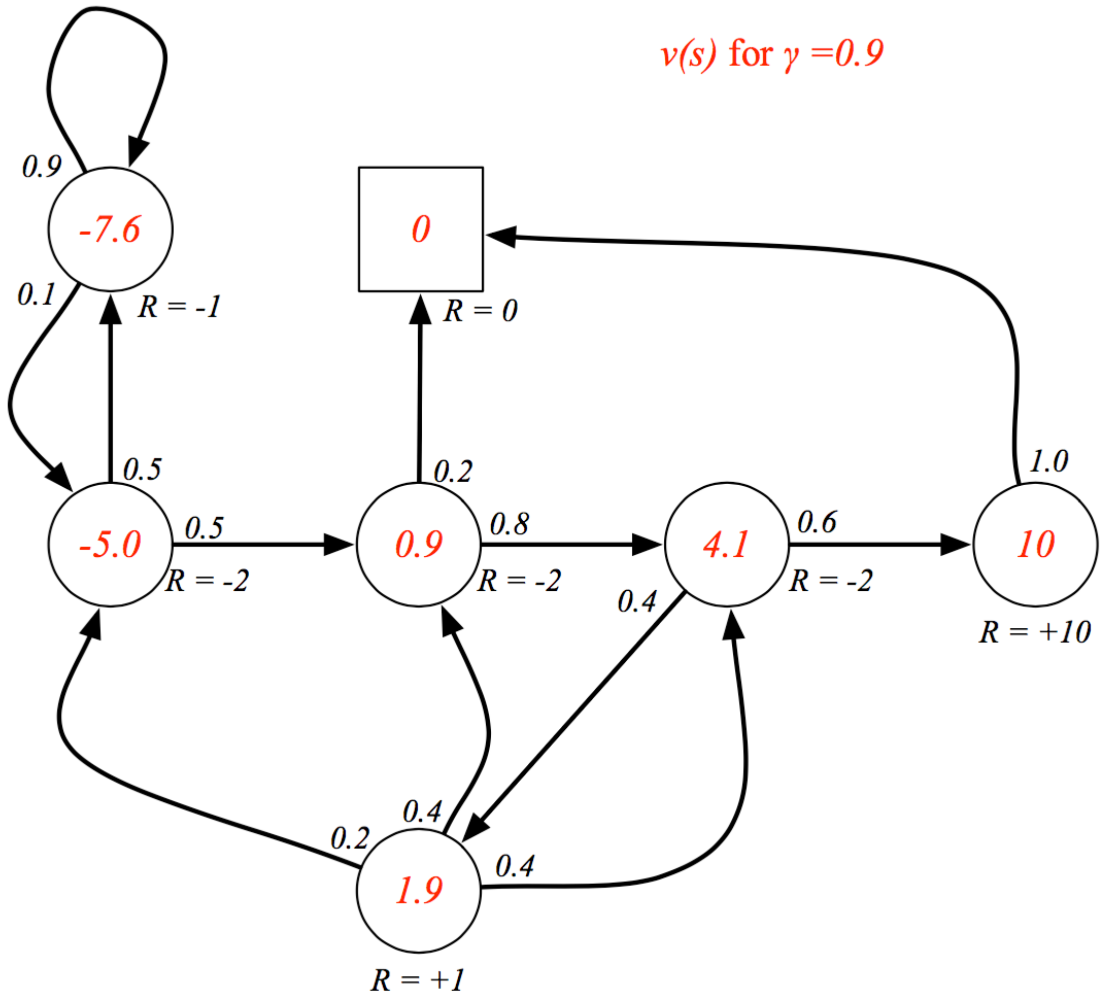
$$G_1 = R_2 + \gamma R_3 + \ldots + \gamma^{T-2} R_T$$

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			

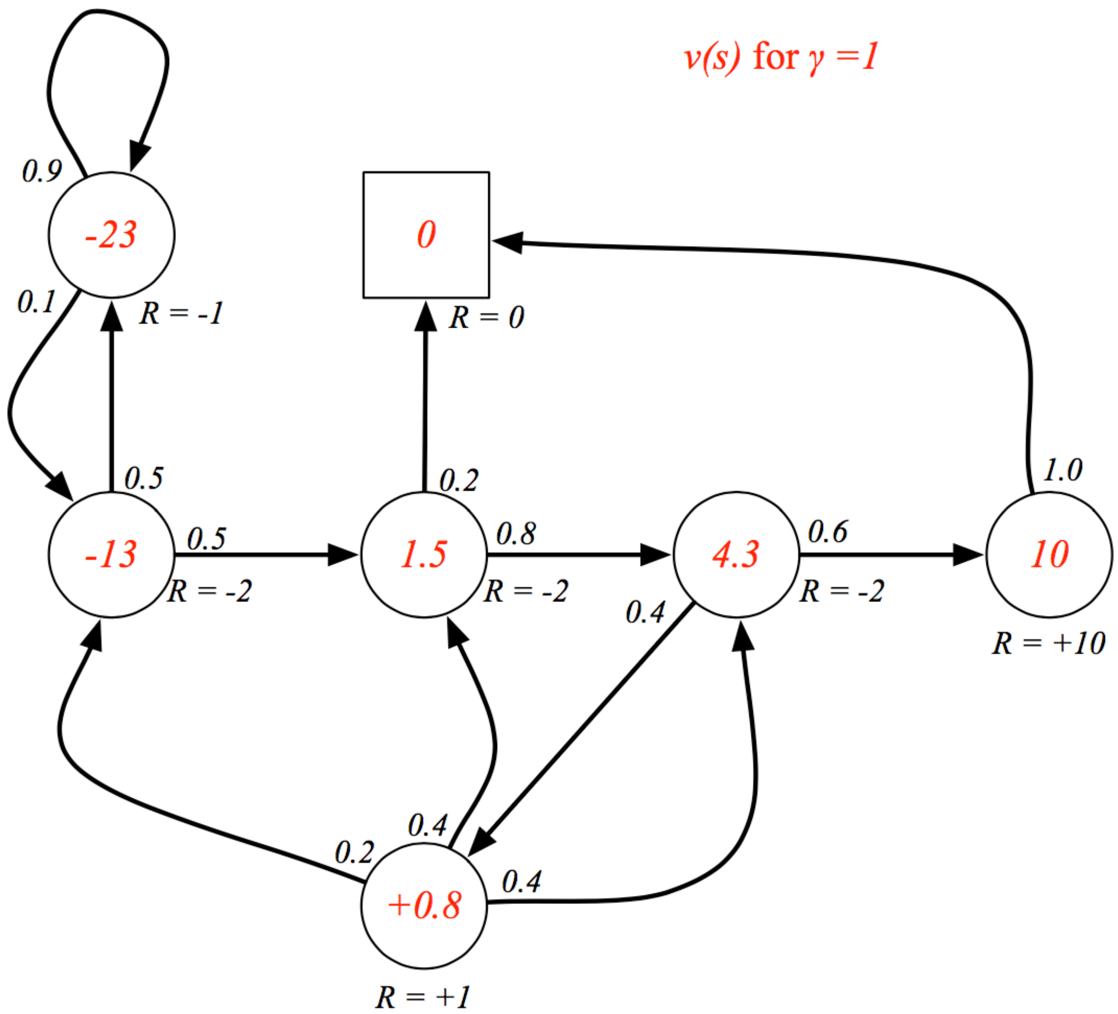
Value Function for Student MRP



Value Function for Student MRP



Value Function for Student MRP



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

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

Definition

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

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

The value function can be decomposed into two parts:

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

Bellman Equations for MRP

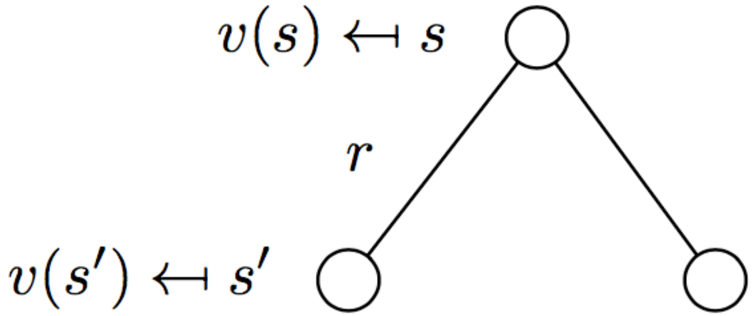
The value function can be decomposed into two parts:

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

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

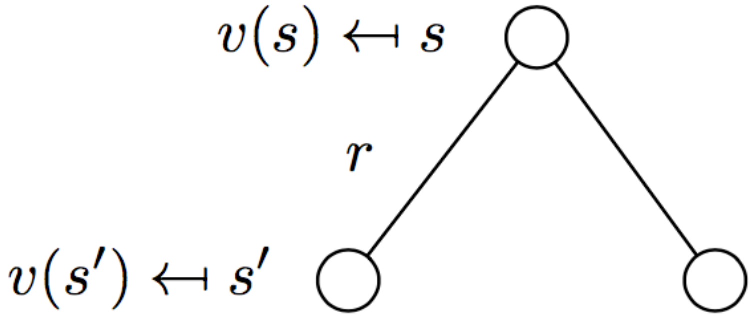
Backup Diagrams

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



Backup Diagrams

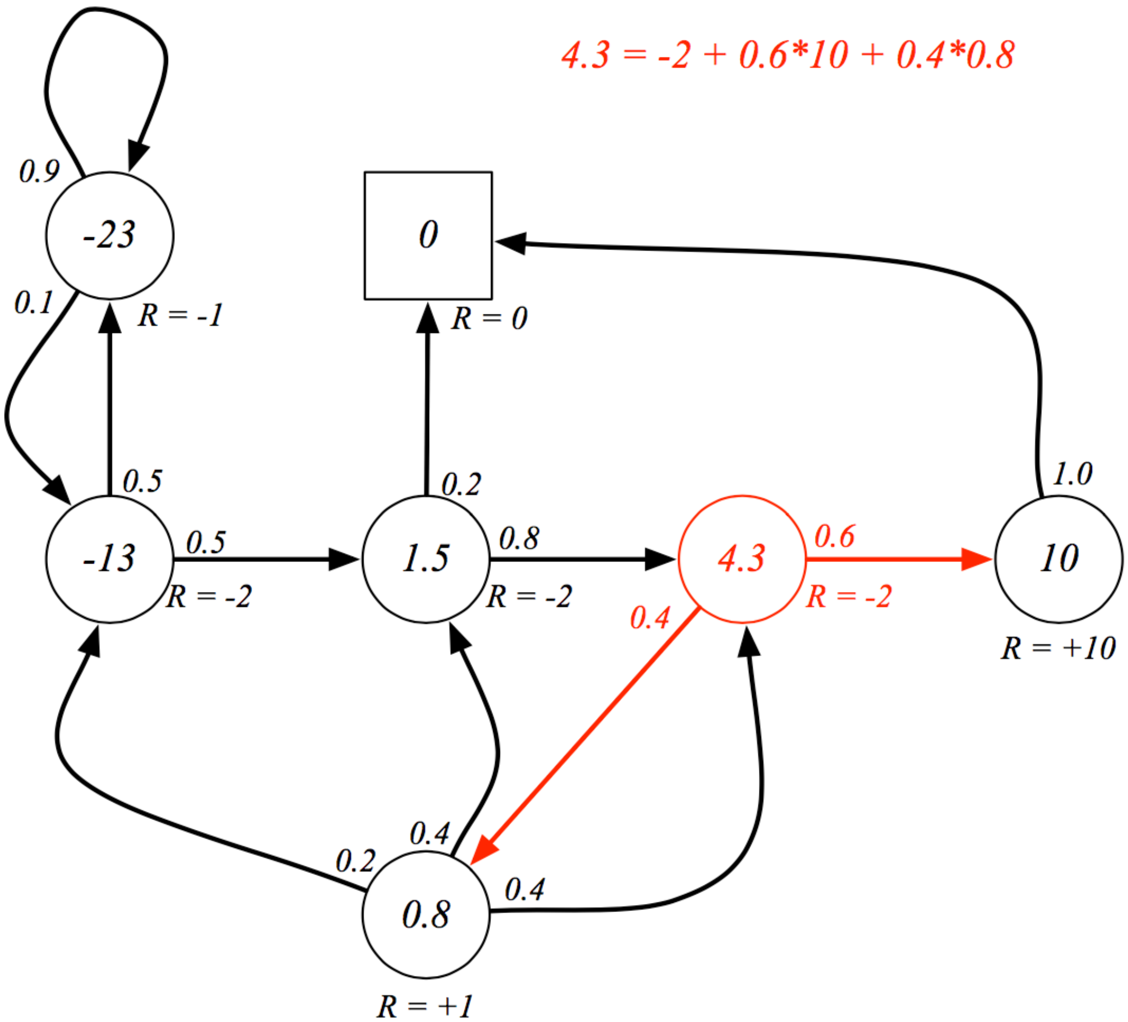
$$v(s) = \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s\right]$$

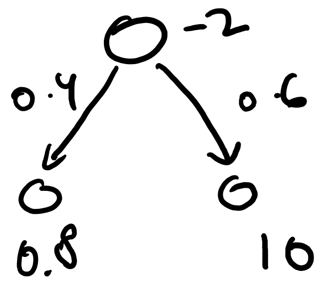


$$v(s) = \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} v(s')$$

Student MRP: Bellman Equations

$4.3 = -2 + 0.6*10 + 0.4*0.8$





Matrix Form of Bellman Equation

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

$$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} \mathcal{R}_1 \\ \vdots \\ \mathcal{R}_n \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{11} & \dots & \mathcal{P}_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix}$$

Matrix Form of Bellman Equation

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

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

Matrix Form of Bellman Equation

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

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

- Computational complexity is $O(n^3)$ for n states
- Direct solution only possible for small MRPs
- There are many iterative methods for large MRPs, e.g.
  - Dynamic programming
  - Monte-Carlo evaluation
  - Temporal-Difference learning

Questions?

Where We're Headed

Wrapup

Markov Processes

- Describe the evolution of states over time
- Characterized by transition matrix

Markov Reward Processes

- Each state has an associated reward (possibly zero)
- Value function of a state
  - long-term average future reward from being in said state
- Value functions can be computed for small MRPs via Bellman equations