

# Lecture 1: Introduction to RL

Professor Emma Brunskill

CS234 RL

Winter 2023

- Today the 3rd part of the lecture includes some slides from David Silver's introduction to RL slides or modifications of those slides

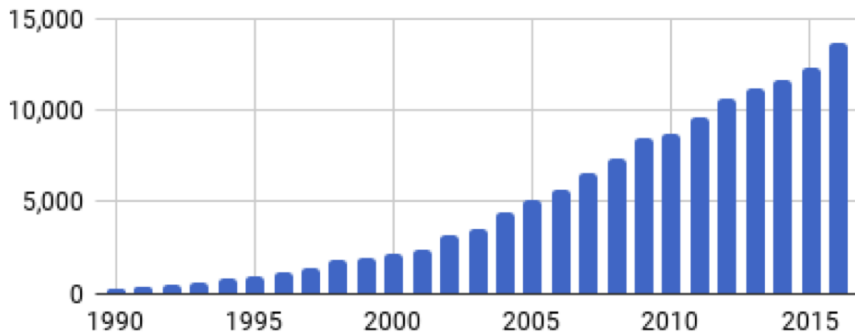
# Today's Plan

- Overview of reinforcement learning
- Course logistics
- Introduction to sequential decision making under uncertainty

Learning to make good decisions under uncertainty

The probability problems involved are formidable. . . [but] the theory of sequential design will be of the greatest importance to mathematical statistics and to science... – Robins 1952

# Huge Increase in Interest<sup>1</sup>



<sup>1</sup>Figure from Henderson et al. 2018 AAAI  
<https://arxiv.org/pdf/1709.06560.pdf>

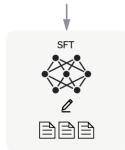
## behavior cloning

Step 1

Collect demonstration data and train a supervised policy.

## imitation learning

A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3.5 with supervised learning.

## model based RL

Step 2

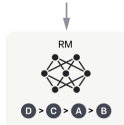
Collect comparison data and train a reward model.

A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



## IL $\rightarrow$ RL

hw 3

Step 3

Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

A new prompt is sampled from the dataset.



The PPO model is initialized from the supervised policy.



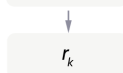
The policy generates an output.



The reward model calculates a reward for the output.



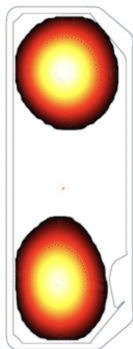
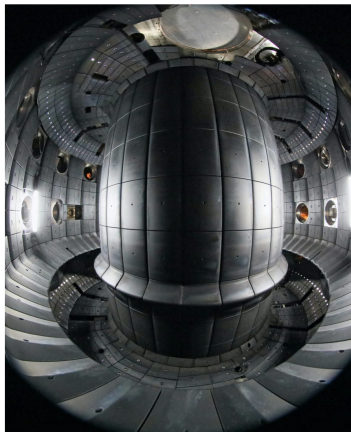
The reward is used to update the policy using PPO.



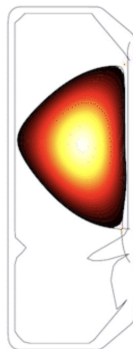
# Learning Plasma Control for Fusion Science<sup>2</sup>

*deep RL in 2 actor critic*

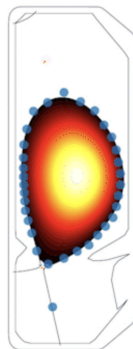
*sim2real*



Droplets



Negative  
Triangularity



ITER-like  
shape

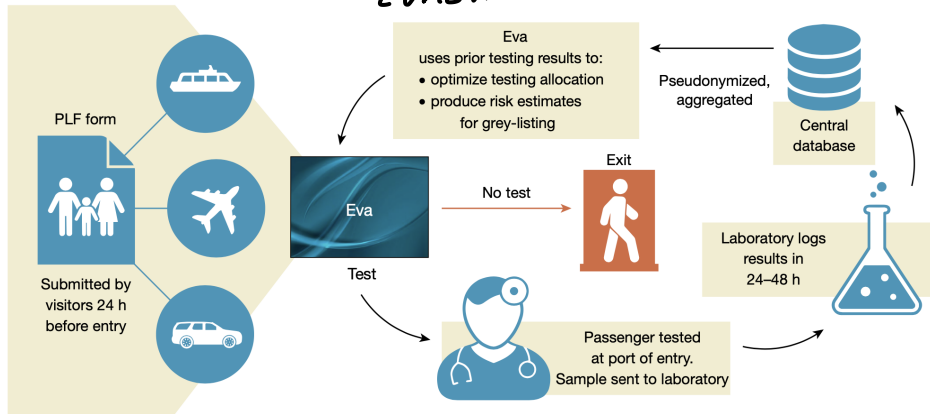
*data constrained*

<sup>2</sup>Image credits: left Alain Herzog / EPFL, right DeepMind & SPC/EPFL. Degrave et al. Nature 2022 <https://www.nature.com/articles/s41586-021-04301-9>

# Efficient and targeted COVID-19 border testing via RL <sup>3</sup>

*contextual multiarmed bandits  
constraints*

*delayed  
outcome*



<sup>3</sup>Bastani et al. Nature 2021



# Reinforcement Learning Involves

- Optimization
- Delayed consequences
- Exploration
- Generalization

- Goal is to find an optimal way to make decisions
  - Yielding best outcomes or at least very good outcomes
- Explicit notion of utility of decisions
- Example: finding minimum distance route between two cities given network of roads

# Delayed Consequences

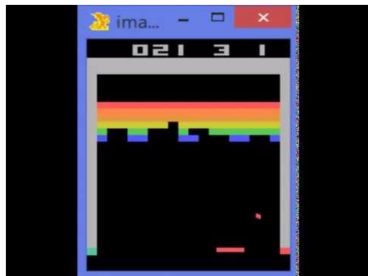
- Decisions now can impact things much later...
  - Saving for retirement
  - Finding a key in video game Montezuma's revenge
- Introduces two challenges
  - When planning: decisions involve reasoning about not just immediate benefit of a decision but also its longer term ramifications
  - When learning: temporal credit assignment is hard (what caused later high or low rewards?)

- Learning about the world by making decisions
  - Agent as scientist
  - Learn to ride a bike by trying (and failing)
  - Finding a key in Montezuma's revenge
- Decisions impact what we learn about
  - Only get a reward for decision made
  - Don't know what would have happened for other decision
  - If we choose to go to Stanford instead of MIT, we will have different later experiences...

# Generalization

*compress*

- Policy is mapping from past experience to action
- Why not just pre-program a policy?



3  
*image*  
 $(256 \times 256)$

Figure: DeepMind Nature, 2015

# RL vs Other AI and Machine Learning

	AI Planning	SL	UL	RL	IL
Optimization					
Learns from experience					
Generalization					
Delayed Consequences					
Exploration					

- SL = Supervised learning; UL = Unsupervised learning; RL = Reinforcement Learning; IL = Imitation Learning

# RL vs Other AI and Machine Learning

	AI Planning	SL	UL	RL	IL
Optimization	X				
Learns from experience					
Generalization	X				
Delayed Consequences	X				
Exploration					

- SL = Supervised learning; UL = Unsupervised learning; RL = Reinforcement Learning; IL = Imitation Learning
- AI planning assumes have a model of how decisions impact environment

# RL vs Other AI and Machine Learning

	AI Planning	SL	UL	RL	IL
Optimization	X				
Learns from experience		X			
Generalization	X	X			
Delayed Consequences	X				
Exploration					

- SL = Supervised learning; UL = Unsupervised learning; RL = Reinforcement Learning; IL = Imitation Learning
- Supervised learning has access to the correct labels



# RL vs Other AI and Machine Learning

	AI Planning	SL	UL	RL	IL
Optimization	X				
Learns from experience		X	X		
Generalization	X	X	X		
Delayed Consequences	X				
Exploration					

- SL = Supervised learning; UL = Unsupervised learning; RL = Reinforcement Learning; IL = Imitation Learning
- Unsupervised learning has access to no labels

# RL vs Other AI and Machine Learning

	AI Planning	SL	UL	RL	IL
Optimization	X			X	
Learns from experience		X	X	X	
Generalization	X	X	X	X	
Delayed Consequences	X			X	
Exploration				X	

- SL = Supervised learning; UL = Unsupervised learning; RL = Reinforcement Learning; IL = Imitation Learning
- Reinforcement learning is given only reward information, and only for states reached and actions taken

RL  $\rightarrow$  SL

	AI Planning	SL	UL	RL	IL
Optimization	X			X	X
Learns from experience		X	X	X	X
Generalization	X	X	X	X	X
Delayed Consequences	X			X	X
Exploration				X	

- SL = Supervised learning; UL = Unsupervised learning; RL = Reinforcement Learning; IL = Imitation Learning
- Imitation learning typically assumes input demonstrations of good policies
- IL reduces RL to SL. IL + RL is promising area

# Today's Plan

- Overview of reinforcement learning
- **Course logistics**
- Introduction to sequential decision making under uncertainty

# Course Outline

- Markov decision processes & planning
- Model-free policy evaluation
- Model-free control
- Reinforcement learning with function approximation & Deep RL
- Policy Search
- Exploration
- Offline RL
- Advanced Topics

# High Level Learning Goals<sup>4</sup>

- Define the key features of RL
- Given an application problem know how (and whether) to use RL for it
- Implement (in code) common RL algorithms
- Understand theoretical and empirical approaches for evaluating the quality of a RL algorithm

---

<sup>4</sup>For more detailed descriptions, see website

# Course Structure Overview

- Live lectures
- Three homeworks
- 1 exam
- 1 multiple choice quiz
- Final project
- Check/Refresh your understanding exercises (Access through your Stanford poll everywhere account)
- Problem sessions

# "Learning is Not a Spectator Sport: Doing is Better than Watching for Learning from a MOOC"<sup>5</sup>

- In a psychology Massive Open Online Class, doing more activities seemed to yield a **6 times larger** learning benefit compared to extra video watching or reading
- "...it appears students actually spend substantially less time per activity (3.4 min) than reading a page (5.0 min)"
- → Engaged practice is likely to be a more efficient and effective way to learn material.
- To achieve the class learning goals, encourage you to do homework, go do problem sessions to get more active practice on conceptual and theoretical parts, do the check your understandings, and try past quiz or exam problems for practice without referring to solutions before you complete them

---

<sup>5</sup>Koedinger et al. L@S 2015. <https://dl.acm.org/doi/pdf/10.1145/2724660.2724681>



# Staff and Support

- Instructor: Emma Brunskill
- CAs: Dilip Arumugam, Anirudhan Badrinath, Skanda Vaidyanath, Max Sobol Mark, Regina Wang and Jian Vora
- Additional information
  - Course webpage: <http://cs234.stanford.edu>
  - Schedule, Ed (fastest way to get help), lecture slides
  - Prerequisites, grading details, late policy, see webpage
- All of you can succeed if you put in the effort
- We, the class staff, and your fellow classmates, are here to help!



*hw 1 out 1/2 for this week*

# Today's Plan

- Overview of reinforcement learning
- Course logistics
- **Introduction to sequential decision making under uncertainty**

# Refresher Exercise: AI Tutor as a Decision Process

- Student initially does not know addition (easier) nor subtraction (harder)
- AI tutor agent can provide practice problems about addition or subtraction
- AI agent gets rewarded +1 if student gets problem right, -1 if get problem wrong
- Model this as a Decision Process. Define state space, action space, and reward model. What does the dynamics model represent? What would a policy to optimize the expected discounted sum of rewards yield?
- Write down your own answers (5 min) and then discuss in small breakout groups..

history

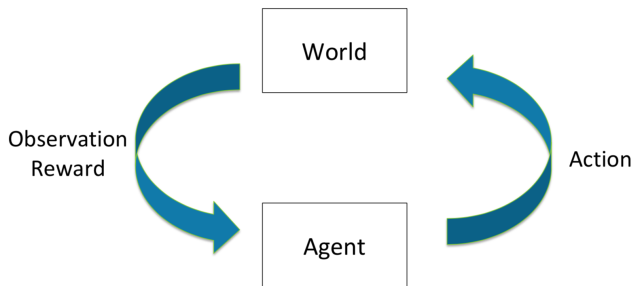
student knew add, knew subtract

- State:
- Actions: add prob, sub prob
- Reward model: 1 if correct else 0
- Meaning of dynamics model:

# Refresher Exercise: AI Tutor as a Decision Process

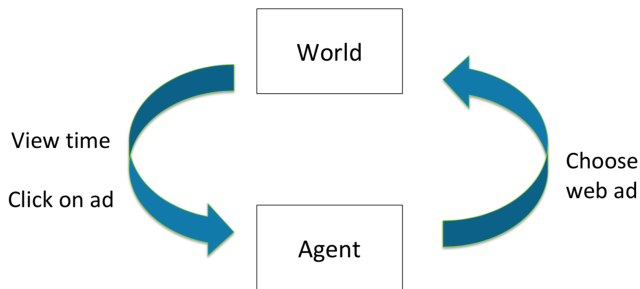
- Student initially does not know addition (easier) nor subtraction (harder)
- Teaching agent can provide activities about addition or subtraction
- Agent gets rewarded for student performance: +1 if student gets problem right, -1 if get problem wrong
- Which items will agent learn to give to max expected reward? Is this the best way to optimize for learning? If not, what other reward might one give to encourage learning?

# Sequential Decision Making



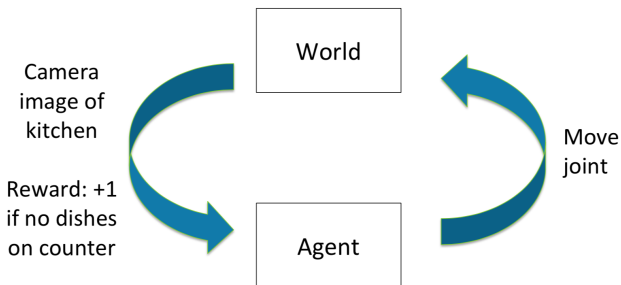
- Goal: Select actions to maximize total expected future reward
- May require balancing immediate & long term rewards

# Example: Web Advertising



- Goal: Select actions to maximize total expected future reward
- May require balancing immediate & long term rewards

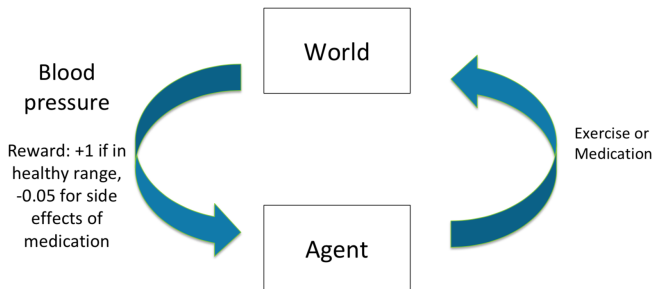
# Example: Robot Unloading Dishwasher



- Goal: Select actions to maximize total expected future reward
- May require balancing immediate & long term rewards

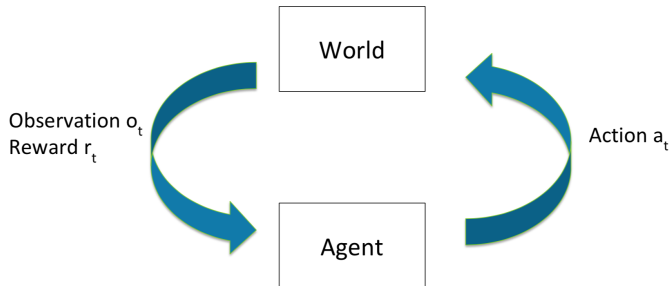


# Example: Blood Pressure Control



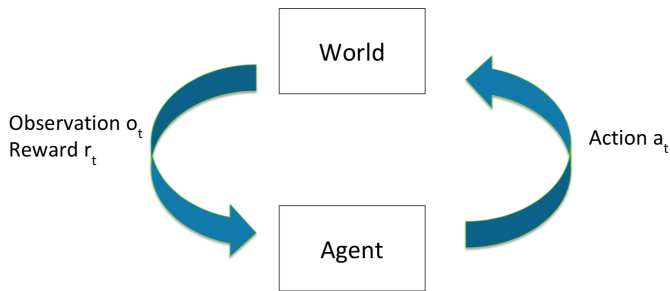
- Goal: Select actions to maximize total expected future reward
- May require balancing immediate & long term rewards

# Sequential Decision Process: Agent & the World (Discrete Time)



- Each time step  $t$ :
  - Agent takes an action  $a_t$
  - World updates given action  $a_t$ , emits observation  $o_t$  and reward  $r_t$
  - Agent receives observation  $o_t$  and reward  $r_t$

# History: Sequence of Past Observations, Actions & Rewards



- History  $h_t = (a_1, o_1, r_1, \dots, a_t, o_t, r_t)$
- Agent chooses action based on history
- State is information assumed to determine what happens next
  - Function of history:  $s_t = (h_t)$

# Markov Assumption

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

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

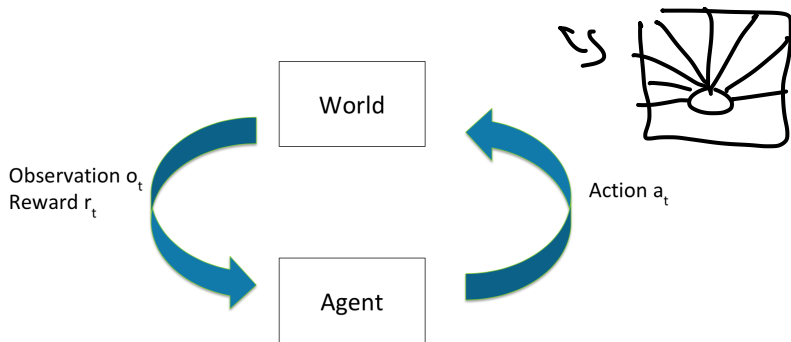
- Future is independent of past given present

# Why is Markov Assumption Popular?

- Simple and often can be satisfied if include some history as part of the state
- In practice often assume most recent observation is sufficient statistic of history:  $s_t = o_t$
- State representation has big implications for:
  - Computational complexity
  - Data required
  - Resulting performance



# Types of Sequential Decision Processes



- Is state Markov? Is world partially observable? (POMDP)
- Are dynamics deterministic or stochastic?
- Do actions influence only immediate reward (bandits) or reward and next state ?

# Example: Mars Rover as a Markov Decision Process


$s_1$	$s_2$	$s_3$	$s_4$	$s_5$	$s_6$	$s_7$
						

Figure: Mars rover image: NASA/JPL-Caltech

- States: Location of rover ( $s_1, \dots, s_7$ )
- Actions: TryLeft or TryRight
- Rewards:
  - +1 in state  $s_1$
  - +10 in state  $s_7$
  - 0 in all other states

# RL Algorithm Components

- Often includes one or more of: Model, Policy, Value Function



- Agent's representation of how world changes given agent's action
- Transition / dynamics model predicts next agent state

$$p(s_{t+1} = s' | s_t = s, a_t = a)$$

- Reward model predicts immediate reward

$$r(s_t = s, a_t = a) = \mathbb{E}[r_t | s_t = s, a_t = a]$$

# Example: Mars Rover Stochastic Markov Model

$+1$   $f(\circ)$

$s_1$	$s_2$	$s_3$	$s_4$	$s_5$	$s_6$	$s_7$
$\hat{r} = 0$	$\hat{r} = 0$	$\hat{r} = 0$	$\hat{r} = 0$	$\hat{r} = 0$	$\hat{r} = 0$	$\hat{r} = 0$

- Numbers above show RL agent's reward model
- Part of agent's transition model:
  - $0.5 = P(s_1|s_1, \text{TryRight}) = P(s_2|s_1, \text{TryRight})$
  - $0.5 = P(s_2|s_2, \text{TryRight}) = P(s_3|s_2, \text{TryRight}) \dots$
- Model may be wrong


- Policy  $\pi$  determines how the agent chooses actions
- $\pi : S \rightarrow A$ , mapping from states to actions
- Deterministic policy:

$$\pi(s) = a$$

- Stochastic policy:

$$\pi(a|s) = Pr(a_t = a | s_t = s)$$

# Example: Mars Rover Policy

$s_1$	$s_2$	$s_3$	$s_4$	$s_5$	$s_6$	$s_7$
						

- $\pi(s_1) = \pi(s_2) = \dots = \pi(s_7) = \text{TryRight}$
- Quick check: is this a deterministic policy or a stochastic policy?

# Value Function

- Value function  $V^\pi$ : expected discounted sum of future rewards under a particular policy  $\pi$

$$V^\pi(s_t = s) = \mathbb{E}_\pi[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \dots | s_t = s]$$

- Discount factor  $\gamma$  weighs immediate vs future rewards
- Can be used to quantify goodness/badness of states and actions
- And decide how to act by comparing policies

# Example: Mars Rover Value Function

$r(s)$   
 $\rightarrow r(s,a)$   
 $r(s,a,s')$

$s_1$	$s_2$	$s_3$	$s_4$	$s_5$	$s_6$	$s_7$
$V^\pi(s_1) = +1$	$V^\pi(s_2) = 0$	$V^\pi(s_3) = 0$	$V^\pi(s_4) = 0$	$V^\pi(s_5) = 0$	$V^\pi(s_6) = 0$	$V^\pi(s_7) = +10$

- Discount factor,  $\gamma = 0$
- $\pi(s_1) = \pi(s_2) = \dots = \pi(s_7) = \text{TryRight}$
- Numbers show value  $V^\pi(s)$  for this policy and this discount factor

$$V^\pi(s) = r(s)$$

# Types of RL Agents

$$S \rightarrow a \quad V(s)$$

- Model-based
    - Explicit: Model
    - May or may not have policy and/or value function
  - Model-free
    - Explicit: Value function and/or policy function
    - No model
- dynamics & reward*

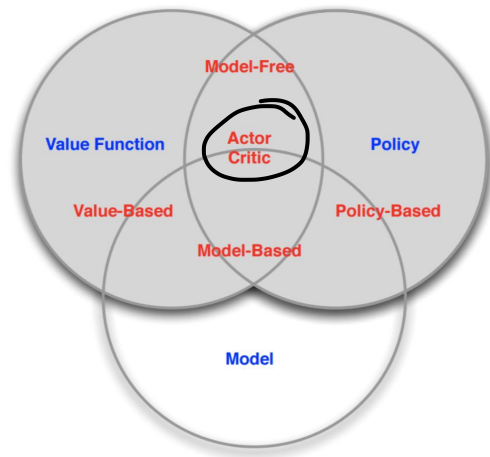


Figure: Figure from David Silver's RL course

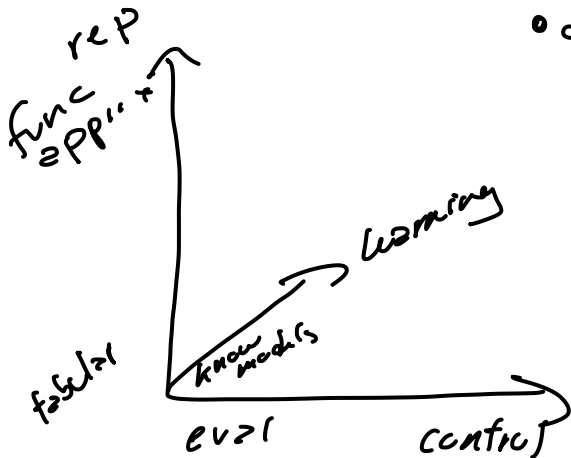


# Evaluation and Control

- Evaluation
  - Estimate/predict the expected rewards from following a given policy
- Control
  - Optimization: find the best policy



# Build Up in Complexity



• deep RL

# Making Sequences of Good Decisions Given a Model of the World

- Assume finite set of states and actions
- Given models of the world (dynamics and reward)
- Evaluate the performance of a particular decision policy
- Compute the best policy
- This can be viewed as an AI planning problem

# Making Sequences of Good Decisions Given a Model of the World

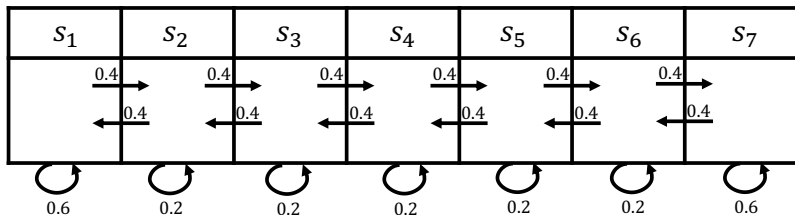
- Markov Processes
- Markov Reward Processes (MRPs)
- Markov Decision Processes (MDPs)
- Evaluation and Control in MDPs

# 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  $p(s_{t+1} = s' | s_t = s)$
- Note: no rewards, no actions
- If finite number ( $N$ ) of states, can express  $P$  as a matrix

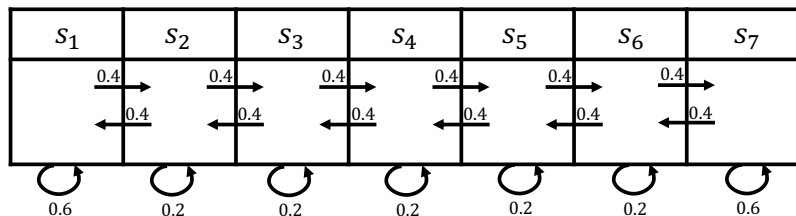
$$P = \begin{pmatrix} P(s_1|s_1) & P(s_2|s_1) & \cdots & P(s_N|s_1) \\ P(s_1|s_2) & P(s_2|s_2) & \cdots & P(s_N|s_2) \\ \vdots & \vdots & \ddots & \vdots \\ P(s_1|s_N) & P(s_2|s_N) & \cdots & P(s_N|s_N) \end{pmatrix}$$

# 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  $S_4$

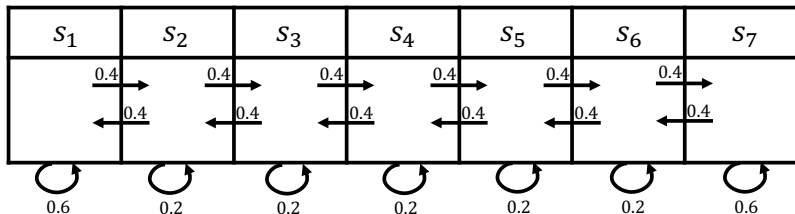
- $S_4, S_5, S_6, S_7, S_7, S_7, \dots$
- $S_4, S_4, S_5, S_4, S_5, S_6, \dots$
- $S_4, S_3, S_2, S_1, \dots$

# 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  $P(s_{t+1} = s' | s_t = s)$
  - $R$  is a reward function  $R(s_t = s) = \mathbb{E}[r_t | s_t = s]$
  - Discount factor  $\gamma \in [0, 1]$
- Note: no actions
- If finite number ( $N$ ) of states, can express  $R$  as a vector



# Example: Mars Rover Markov Reward Process



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

# Return & Value Function

- Definition of Horizon ( $H$ )
  - Number of time steps in each episode
  - Can be infinite
  - Otherwise called **finite** Markov reward process
- Definition of Return,  $G_t$  (for a Markov Reward Process)
  - Discounted sum of rewards from time step  $t$  to horizon  $H$

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots + \gamma^{H-1} r_{t+H-1}$$

- Definition of State Value Function,  $V(s)$  (for a Markov Reward Process)
  - Expected return from starting in state  $s$

$$V(s) = \mathbb{E}[G_t | s_t = s] = \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots + \gamma^{H-1} r_{t+H-1} | s_t = s]$$

# Discount Factor

- Mathematically convenient (avoid infinite returns and values)
- Humans often act as if there's a discount factor  $< 1$
- If episode lengths are always finite ( $H < \infty$ ), can use  $\gamma = 1$

# 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 ( $H < \infty$ ), can use  $\gamma = 1$

# Computing the Value of a Markov Reward Process

- Markov property provides structure
- MRP value function satisfies

$$V(s) = \underbrace{R(s)}_{\text{Immediate reward}} + \underbrace{\gamma \sum_{s' \in \mathcal{S}} P(s'|s)V(s')}_{\text{Discounted sum of future rewards}}$$

# Matrix Form of Bellman Equation for MRP

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

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

$$V = R + \gamma PV$$

# Analytic Solution for Value of MRP

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

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

$$V = R + \gamma PV$$

$$V - \gamma PV = R$$

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

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

- Solving directly requires taking a matrix inverse  $\sim O(N^3)$
- Note that  $(I - \gamma P)$  is invertible

# Iterative Algorithm for Computing Value of a MRP

- Dynamic programming
- Initialize  $V_0(s) = 0$  for all  $s$
- For  $k = 1$  until convergence
  - For all  $s$  in  $S$

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

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



# Summary of Today

- Reinforcement learning involves learning, optimization, generalization and exploration
- Goal is to learn to make good decisions under uncertainty

# Course Outline

- Markov decision processes & planning
- Model-free policy evaluation
- Model-free control
- Reinforcement learning with function approximation & Deep RL
- Policy Search
- Exploration
- Advanced Topics

# Tasks

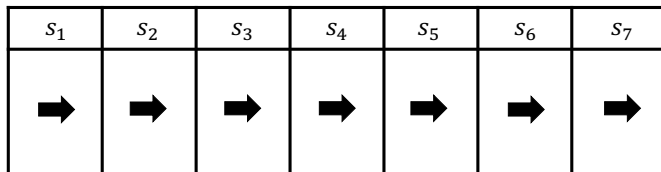
- Homework 1 will be released this week.
- Check your understanding exercises will be announced in lectures and on Ed. These will be for participation points: to receive credit, you need to log in to poll everywhere using your stanford sunid account.
- See website for more details

# Course Outline

- Markov decision processes & planning )
- Model-free policy evaluation
- Model-free control
- Reinforcement learning with function approximation & Deep RL
- Policy Search
- Exploration
- Advanced Topics

See website for more details

# Example: Mars Rover Policy Evaluation



- $\pi(s_1) = \pi(s_2) = \dots = \pi(s_7) = \text{TryRight}$
- Discount factor,  $\gamma = 0$
- What is the value of this policy?

$$V^\pi(s_t = s) = \mathbb{E}_\pi[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots | s_t = s]$$

-

# Example: Mars Rover Policy Evaluation

$s_1$	$s_2$	$s_3$	$s_4$	$s_5$	$s_6$	$s_7$
→	→	→	→	→	→	→

- $\pi(s_1) = \pi(s_2) = \dots = \pi(s_7) = \text{TryRight}$
- Discount factor,  $\gamma = 0$
- What is the value of this policy?

$$V^\pi(s_t = s) = \mathbb{E}_\pi[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots | s_t = s]$$

- Answer:

$$V^\pi(s_t = s) = r(s)$$