CSC415: Introduction to Reinforcement Learning

Lecture 1: Introduction and MDP Structure

Dr. Amey Pore

Winter 2026

January 7, 2026

Today's Plan

- **Overview of Reinforcement Learning (RL)**
  - What is reinforcement learning?
  - Key characteristics: RL vs Supervised Learning
  - Where is reinforcement learning used?
- Course structure
- RL formulation

Reinforcement Learning

Learning through experience/data to make good decisions under uncertainty

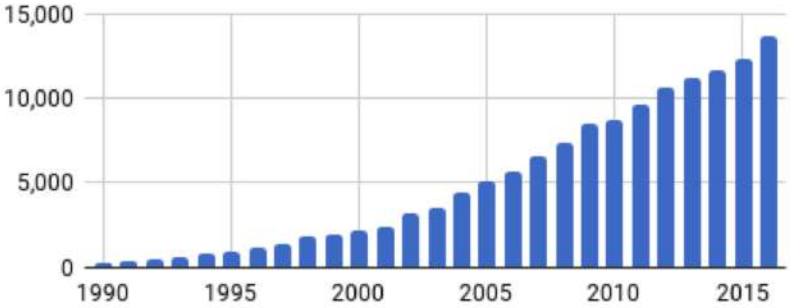
Reinforcement Learning

- Learning through experience/data to make good decisions under uncertainty
- Essential part of intelligence
- Builds strongly from theory and ideas starting in the 1950s with Richard Bellman

Reinforcement Learning

- Learning through experience/data to make good decisions under uncertainty
- Essential part of intelligence
- Builds strongly from theory and ideas starting in the 1950s with Richard Bellman
- A number of impressive successes in the last decade

Huge Increase in Interest



Henderson et al., "Deep Reinforcement Learning that Matters" AAAI 2018.

Characteristics of Reinforcement Learning

- Optimization
- Delayed consequences
- Exploration
- Generalization

Key Characteristic 1: Optimization

- Goal: Find optimal way to make decisions yielding best outcomes
- Explicit notion of decision utility
- Example: Finding minimum distance route between two cities given network of roads

Key Characteristic 2: Delayed Feedback

The Credit Assignment Problem

- Actions have **long-term consequences**
- Rewards may come **much later**
- Which action caused the reward?
- Decisions now can impact things much later...

Examples

- **Chess Game**: Move 1 (pawn) $\rightarrow$ ... Move 50 (checkmate $+1$). Which move(s) led to winning?
- **Saving for retirement**: Decisions now affect financial security decades later
- **Video games**: Finding a key in Montezuma's revenge - early actions enable later rewards

Key Characteristic 3: Exploration

- Learning about the world by making decisions
  - Agent as scientist
  - Learn to ride a bike by trying (and failing)
- 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 Waterloo instead of UofT, we will have different later experiences...

Example: Restaurant Selection

- **Exploitation**: Go to your favourite restaurant (safe, known reward)
- **Exploration**: Try a new restaurant (learn whether it's better)

Key Characteristic 4: Generalization

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

Three Types of Machine Learning

Supervised Learning

- Given: Labeled examples (x, y)
- Goal: Learn mapping $f : X \to Y$
- Example: Image classification (image $\to$ label)

Unsupervised Learning

- Given: Unlabeled data x
- Goal: Find patterns/structure in data
- Example: Clustering, dimensionality reduction

Reinforcement Learning

- Given: Interaction with environment
- Goal: Learn policy $\pi : S \to A$ to maximize reward
- Example: Game playing, robot control

Key Difference 1: No Supervisor

Supervised Learning

- Teacher provides correct answers
- Example: "This image is a cat" (label provided)

Reinforcement Learning

- **No teacher, only a reward signal**
- Example: Playing a game
    - No one tells you the "correct" move
    - You only know: win (+1) or lose (-1)
    - Must figure out which actions lead to rewards

Key Difference 2: Sequential Data & Actions Affect Data

Supervised Learning

- Data: Independent and identically distributed (i.i.d.)
- Order doesn't matter: $(x_1, y_1), (x_2, y_2), \ldots$
- Dataset is fixed before training
- Model is **passive** - doesn't affect data collection

Reinforcement Learning

- Data: **Sequential and temporally correlated**
- Order **matters**: $s_1, a_1, r_1, s_2, a_2, r_2, \ldots$
- Current state depends on previous states and actions
- Agent's actions **actively influence** what data it sees next
- Creates a **feedback loop**: actions → new states → new actions

Examples: Actions Affect Data

Robot Navigation

- Action: Turn left $\rightarrow$ See new part of environment
- Action: Turn right $\rightarrow$ See different part
- Agent controls its own experience!

RL vs Other AI and Machine Learning

	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

	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
Planning assumes have a model of how decisions impact environment
Supervised learning is provided correct labels

RL vs Other AI and Machine Learning

	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 is provided no labels

RL vs Other AI and Machine Learning

	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

Sidenote: Imitation Learning

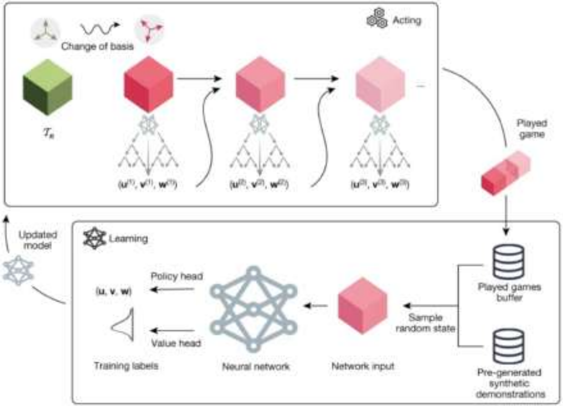
	Planning	SL	UL	RL	IL
Optimization	X			X	
Learns from experience		X	X	X	X
Generalization	X	X	X	X	X
Delayed Consequences	X			X	
Exploration				X	

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

- Imitation learning assumes input demonstrations of good policies
- IL reduces RL to SL. IL + RL is promising area

Where RL is Particularly Powerful

1. **No examples of desired behavior**: e.g. because the goal is to go beyond human performance or there is no existing data for a task.
2. **Enormous search or optimization problem with delayed outcomes**:



Figure, AlphaTensor. Fawzi et al. 2022

Why RL works?

Application 1: Game Playing

Famous Examples

- **AlphaGo** (2016): Defeated world Go champion
- **AlphaZero** (2017): Chess, Go, Shogi from scratch
- **DQN** (2015): Superhuman Atari game performance

Why RL Works Well Here

- Sequential decisions (each move)
- Long-term planning needed
- Clear reward signal (win/lose)
- Can simulate/play many games

Video: DeepMind Atari Game Playing

Application 2: Robotics

Examples

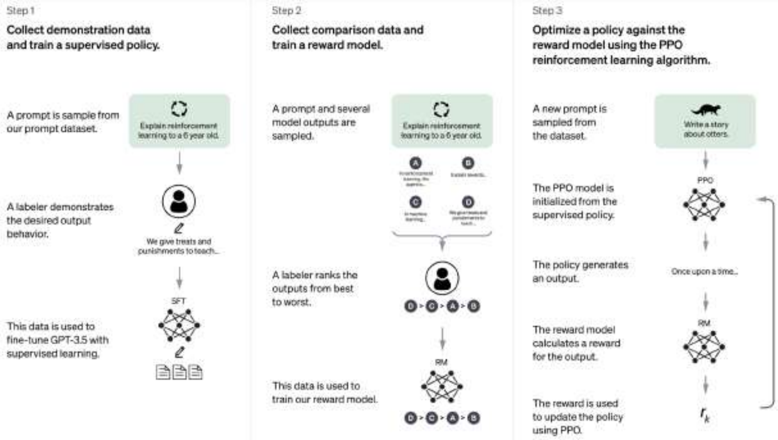
- **Locomotion**: Robots learning to walk, run, jump
- **Manipulation**: Grasping and manipulating objects
- **Helicopter Control**: Acrobatic maneuvers
- **Autonomous Vehicles**: Navigation and decision-making

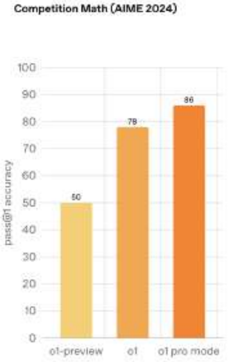
Challenges

- Safety: Real-world failures are costly
- Sample efficiency: Real data is expensive
- Sim-to-real: Transfer from simulation to reality

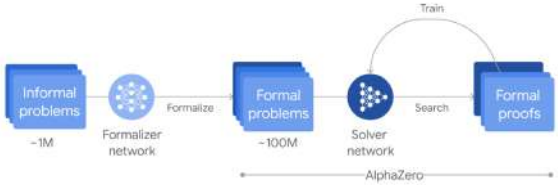
Video: RL in Robotics

Application 3: ChatGPT





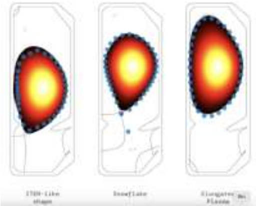
AI achieves gold medal in IMO



Google Deepmind, "Towards Robust Mathematical Reasoning" Nov 2025

Application 3: Plasma Control

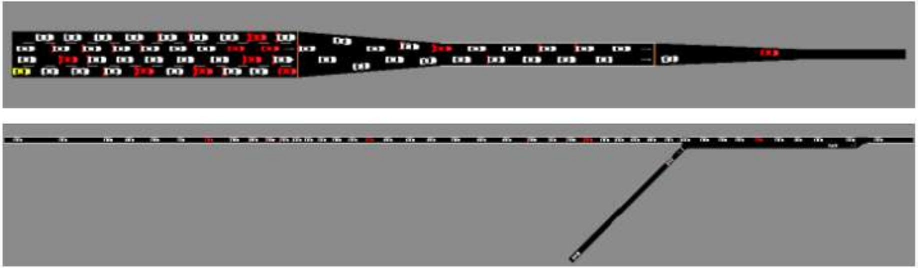
- Controlling plasma in fusion reactors is extremely complex
- RL learns optimal control strategies from simulations
- Achieves stable plasma configurations for longer durations



Application 4: Traffic Management

Smart Traffic Control

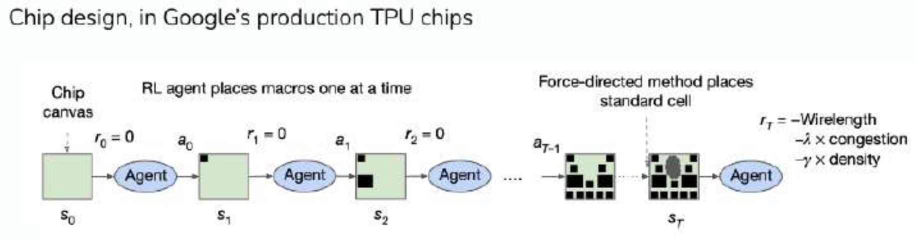
- **Traffic Light Optimization**: RL learns optimal timing patterns to reduce congestion
- **Route Planning**: Dynamic routing based on real-time traffic conditions
- **Autonomous Vehicle Coordination**: Multi-agent RL for traffic flow optimization



Application 5: Chip Design

Key Applications

- **Placement**: Optimal positioning of circuit components
- **Routing**: Efficient wire routing between components
- **Power Optimization**: Minimizing power consumption while meeting performance targets



https://research.google/blog/chip-design-with-deep-reinforcement-learning/

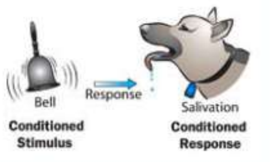
Applications: Other Domains

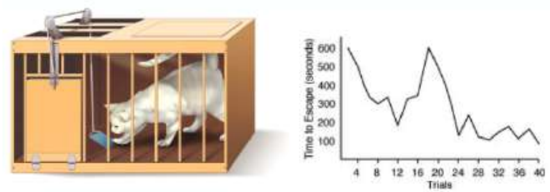
- **Finance**: Algorithmic trading, portfolio management
- **Recommendation Systems**: Personalized content delivery
- **Healthcare**: Treatment optimization, drug discovery

Fundamental Aspect of Intelligence: Biological Motivation for RL

Historical Foundations

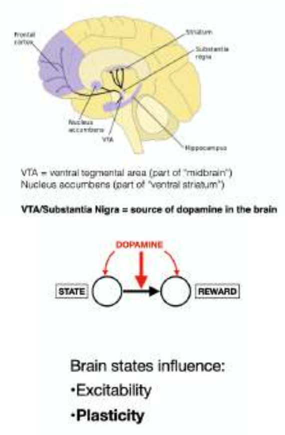
- **Pavlovian Conditioning**: Learning associations between stimuli and rewards
- **Operant Conditioning**: Behaviors that lead to rewards are repeated

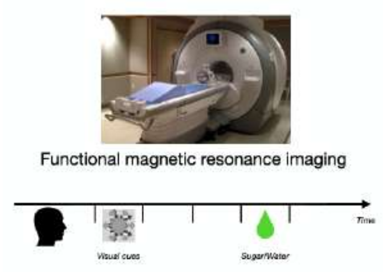


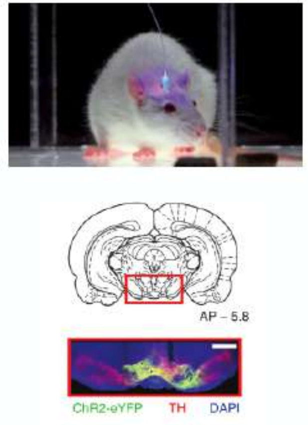


Thorndike (1898)

Dopamine and Reward Learning







Schultz, Dayan, Montague (1997); O'Doherty et al. (2003); Steinberg et al. (2013)

Today's Plan

- Overview of reinforcement learning
  - What is reinforcement learning?
  - Key characteristics: RL vs Supervised Learning
  - Where is reinforcement learning used?
- **Course structure**
- RL formulation

Information & Resources

Course Website `https://ameyapores.github.io/csc415/`

We have put a lot of info here. Please read it. :)

Instructor

- Dr. Amey Pore
- **Office Hours**: Wednesday, 6:00 PM - 7:00 PM (MN3110)

Teaching Assistants

- **Deniz Jafari** - Office Hours: Tuesday 4pm-5pm Online zoom
- **Quentin Clark** - Office Hours: Tuesday 4pm-5pm Online zoom

Schedule

- **Lecture**: Wednesday, 11:00 AM - 1:00 PM (DH 2070)
- **Practical**: Wednesday, 6:00 PM - 7:00 PM (DH 2026)

Course Resources

Required Textbook Reinforcement Learning: An Introduction (2nd Edition)
Richard S. Sutton and Andrew G. Barto
`http://incompleteideas.net/book/`

Additional Resources

- **UCL Course on RL** by David Silver (DeepMind)
- **CSC234 Introduction to Reinforcement learning, Stanford** (Emma Brunskill)
- **CS224: Deep Reinforcement Learning, Stanford** (Chelsea Finn)

Coursework and Grading

Assessment Breakdown

- **Laboratory Exercises** (25%): 6 lab exercises (top 5 count)
- **Midterm Exam** (15%): Jan 29, 2026
- **Assignment 1** (10%): Literature review + implementation
- **Project Proposal** (5%): Feb 24, 2026
- **Final Project Paper** (25%): Mar 24, 2026
- **Assignment 2 (Peer Review)** (10%): Mar 31, 2026
- **Final Project Presentation** (10%): Apr 2, 2026

Coursework

Laboratory Exercises Hands-on programming assignments in Python using Gymnasium and PyTorch:

- Lab 1: Tabular value-iteration agent on Gridworld
- Lab 2: Compare MC and TD methods; Q-Learning
- Lab 3: Implement DQN in Gymnasium
- Lab 4: Train PPO agent on Pendulum-v1
- Lab 5: Implement RND agent in MiniGrid
- Lab 6: Train CNN encoder on Atari frames
- Lab 7: RL for LLM alignment

Final Project Conference-level research paper applying RL concepts to domains such as robotics.

A Bit of Advice

Important Notes

- **RL methods take time to learn behavior!**
- We try to make labs fast to train (using simple environments)
- But, they will still take some time
- You may choose to be more ambitious in your project

Recommendation

- **Don't start labs/project deliverables the night before the deadline**. :)
- Doing is better than watching for learning. [4]

Prerequisites

- **Recommended**: CSC413
- Some familiarity with PyTorch and deep learning concepts

[4] Koedinger et al. 2015. https://dl.acm.org/doi/pdf/10.1145/2724660.2724681

Course Policies

Late Submission Policy

- **Laboratory Exercises**: Late submissions **prohibited**
- **Assignments/Project**: Maximum 3 days late, 15% penalty per day

Academic Integrity

- All work submitted must be your own
- Collaboration allowed but must be acknowledged
- Please read course website for honor code and AI tools policy

Generative AI Policy

- AI tools permitted as learning aids (with citation)
- Include "AI Statement" detailing tool usage
- Midterm Exam: Closed environment - AI tools prohibited

Break!

CSC415: Introduction to Reinforcement Learning

Lecture 1: Introduction and MDP Structure

Dr. Amey Pore

Winter 2026

January 7, 2026

Structure and content adapted from David Silver's and Emma Brunskill's course on Introduction to RL.

Today's Plan

- Overview of reinforcement learning
- Course structure
- **RL formulation**
    - The RL Problem
    - key components: Agent, Environment, Reward
    - Understand State and Observations
    - Inside an RL Agent: Policy, Value Function, Model

Rewards

- A reward R_t is a scalar feedback signal
- Indicates how well agent is doing at step t
- The agent's job is to maximise cumulative reward

Reinforcement learning is based on the **reward hypothesis**

Definition (Reward Hypothesis)

All goals can be described by the maximisation of expected cumulative reward

Examples of Rewards

- **Defeat the world champion at Backgammon**
  - +/−ve reward for winning/losing a game
- **Manage an investment portfolio**
  - +ve reward for each $ in bank
- **Control a power station**
  - +ve reward for producing power
  - −ve reward for exceeding safety thresholds
- **Make a humanoid robot walk**
  - +ve reward for forward motion
  - −ve reward for falling over
- **Play many different Atari games better than humans**
  - +/−ve reward for increasing/decreasing score

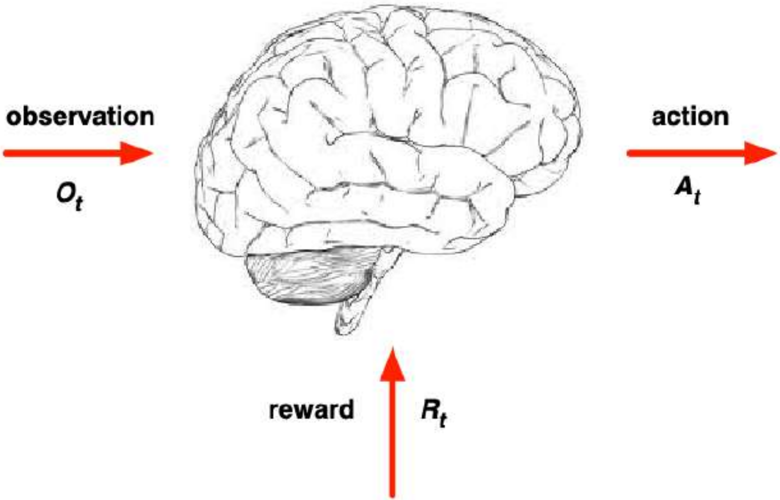
Sequential Decision Making

- **Goal**: select actions to maximise total future reward
- Actions may have long term consequences
- Reward may be delayed
- It may be better to sacrifice immediate reward to gain more long-term reward

Examples:

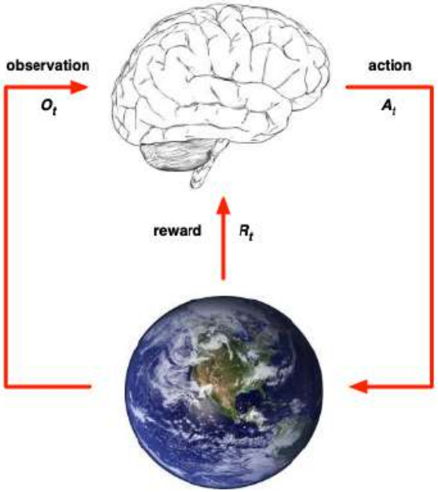
- A financial investment (may take months to mature)
- Refuelling a helicopter (might prevent a crash in several hours)
- Blocking opponent moves (might help winning chances many moves from now)

Agent and Environment



Agent and 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

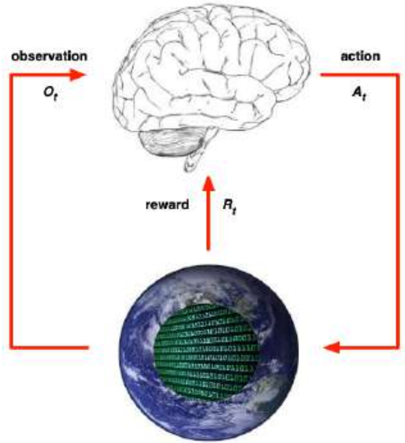
- The history is the sequence of observations, actions, rewards

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

- 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
- State is the information used to determine what happens next
- Formally, state is a function of the history:

$$S_t = f(H_t)$$

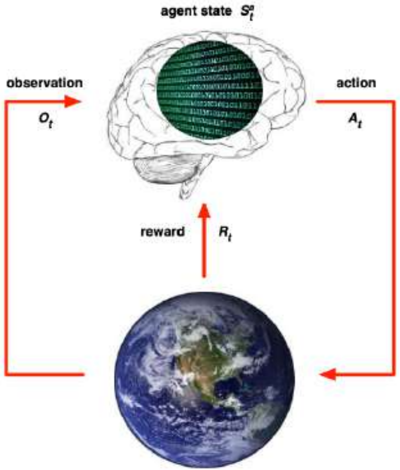
Environment State



The **environment state** S_t^e is the environment's private representation

- i.e. whatever data the environment uses to pick the next observation/reward
- The environment state is not usually visible to the agent
- Even if S_t^e is 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:

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

Information State

An information state (a.k.a. Markov state) contains all useful information from the history.

Definition

A state S_t is Markov if and only if

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

- "The future is independent of the past given the present"
$$H_{1:t} \rightarrow S_t \rightarrow H_{t+1:\infty}$$
- Once the state is known, the history may be thrown away
- i.e. The state is a sufficient statistic of the future
- The environment state S_t^e is Markov
- The history H_t is Markov

Examples

state $\mathbf{s}$ - RGB images, joint positions, joint velocities

action $\mathbf{a}$ - commanded next joint position

trajectory $\boldsymbol{\tau}$ - 10-sec sequence of camera, joint readings, controls at 20 Hz

$$(\mathbf{s}_1, \mathbf{a}_1, \mathbf{s}_2, \mathbf{a}_2, \ldots, \mathbf{s}_T, \mathbf{a}_T),\ T = 200$$

reward $r(\mathbf{s}, \mathbf{a}) =$ 1 if the towel is on the hook in state $\mathbf{s}$

0 otherwise

observation $\mathbf{o}$ - the user's most recent message

action $\mathbf{a}$ - chatbot's next message

trajectory $\boldsymbol{\tau}$ - variable length conversation trace

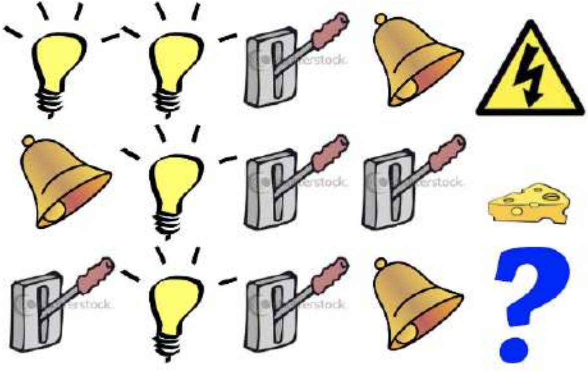
$$(\mathbf{o}_1, \mathbf{a}_1, \mathbf{o}_2, \mathbf{a}_2, \ldots, \mathbf{o}_T, \mathbf{a}_T)$$

reward $r(\mathbf{s}, \mathbf{a}) =$ 1 if the user gives upvote

-10 if the user downvotes

0 if no user feedback

Rat Example



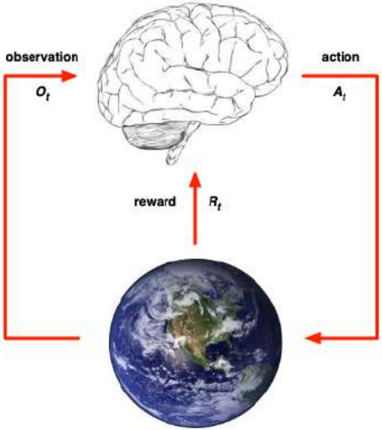
- What if agent state = last 3 items in sequence?
- What if agent state = counts for lights, bells and levers?
- What if agent state = complete sequence?

Think Pair wise

Define

- state $\mathbf{s}$ or observation $\mathbf{o}$
- action $\mathbf{a}$
- trajectory $\boldsymbol{\tau}$
- reward $r(\mathbf{s}, \mathbf{a})$

Fully Observable Environments



Full observability: agent directly observes environment state

$$O_t = S_t^a = S_t^e$$

- Agent state = environment state = information state
- Formally, this is a **Markov decision process (MDP)**

Partially Observable Environments

- **Partial observability**: agent **indirectly** observes environment:
  - A robot with camera vision isn't told its absolute location
  - A trading agent only observes current prices
  - A poker playing agent only observes public cards
- Now agent state $\neq$ environment state
- Formally this is a **partially observable Markov decision process (POMDP)**
- Agent must construct its own state representation S_t^a, e.g.
  - Complete history: $S_t^a = H_t$
  - **Beliefs** of environment state: $S_t^a = (\mathbb{P}[S_t^e = s^1], \ldots, \mathbb{P}[S_t^e = s^n])$
  - Recurrent neural network: $S_t^a = \sigma(S_{t-1}^a W_s + O_t W_o)$

Major Components of an RL Agent

An RL agent may include one or more of these components:

- **Policy**: agent's behaviour function
- **Model**: agent's representation of the environment
- **Value function**: how good is each state and/or action

Example: Mars Rover as a Markov Decision Process

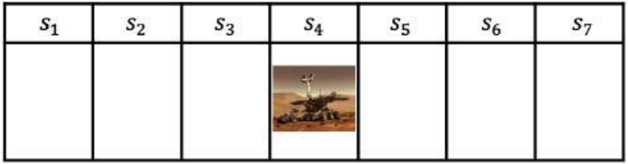


Figure: Mars rover image: NASA/JPL-Caltech

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

Policy

Policy π determines how the agent chooses actions
$\pi : S \to A$, mapping from states to actions

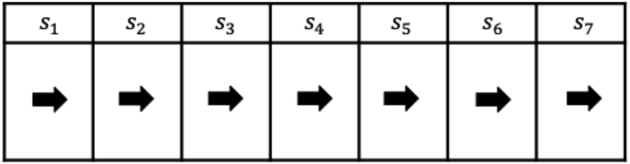
Deterministic policy:

$$\pi(s) = a$$

Stochastic policy:

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

Example: Mars Rover Policy



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

Model

- A **model** predicts what the environment will do next
- **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

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

- Agent may have an internal model of the environment
- Dynamics: how actions change the state
- Rewards: how much reward from each state
- The model may be imperfect
- Numbers above show immediate reward from each state
- Part of agent's transition model:
  - $0.5 = P(s_1|s_1, \text{TryRight}) = P(s_2|s_1, \text{TryRight})$
  - $0.5 = P(s_2|s_2, \text{TryRight}) = P(s_3|s_2, \text{TryRight}) \cdots$

Value Function

- Value function V^π: is a prediction of future reward
- Can be used to quantify goodness/badness of states and actions
- And therefore to select between actions, e.g.

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

Example: Mars Rover Value Function

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

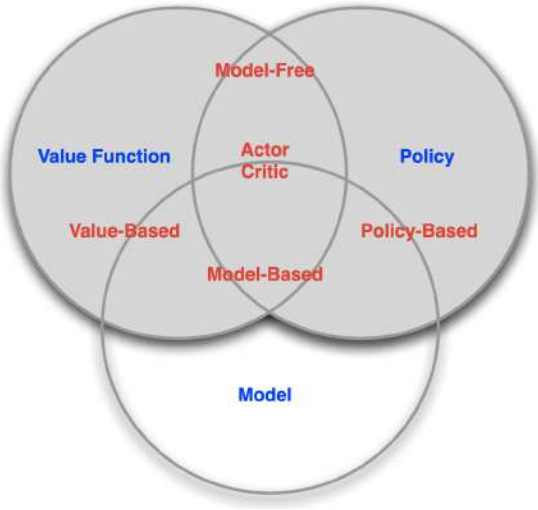
- Discount factor, $\gamma = 0$
- $\pi(s_1) = \pi(s_2) = \cdots = \pi(s_7) = \text{TryRight}$
- Numbers show value $V^\pi(s)$ for this policy.

Types of RL Agents

- **Value Based**
  - No Policy (Implicit)
  - Value Function
- **Policy Based**
  - Policy
  - No Value Function
- **Actor Critic**
  - Policy
  - Value Function

- **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

RL Taxonomy

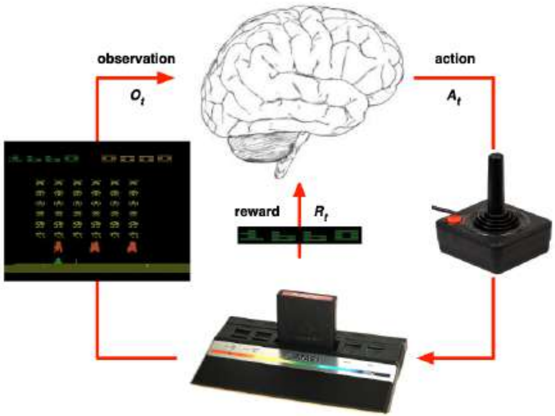

Model-Free
Value Function
Actor
Critic
Policy
Value-Based
Policy-Based
Model-Based
Model

Learning and Planning

Two fundamental problems in sequential decision making

- **Reinforcement Learning:**
  - The environment is initially unknown
  - The agent interacts with the environment
  - The agent improves its policy
- **Planning:**
  - A model of the environment is known
  - The agent performs computations with its model (without any external interaction)
  - The agent improves its policy
  - a.k.a. deliberation, reasoning, introspection, pondering, thought, search

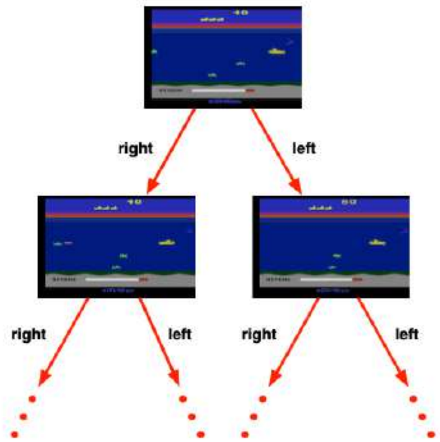
Atari Example: Reinforcement Learning



- Rules of the game are unknown
- Learn directly from interactive game-play
- Pick actions on joystick, see pixels and scores

Atari Example: Planning

- Rules of the game are known
- Can query emulator
    - perfect model inside agent's brain
- If I take action a from state s:
    - what would the next state be?
    - what would the score be?
- Plan ahead to find optimal policy
    - e.g. tree search



Evaluation and Control

Evaluation

Estimate/predict the expected rewards from following a given policy

Control

Optimization: find the best policy

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

Markov models

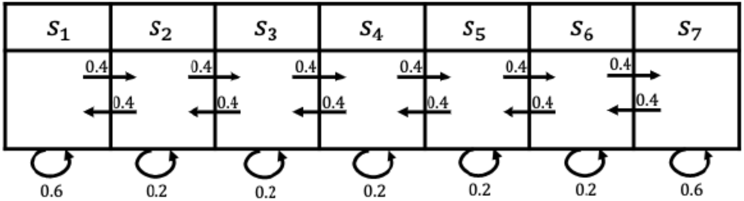
- Markov Processes
- Markov Reward Processes (MRPs)
- Markov Decision Processes (MDPs)
- Evaluation and Control in MDPs

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

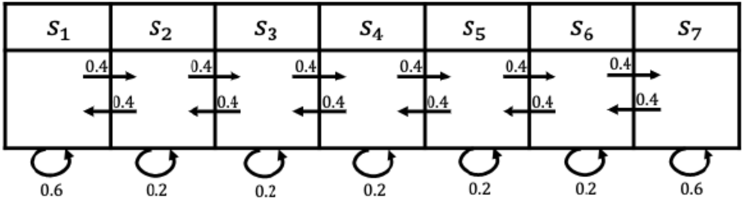
$$P = \begin{pmatrix} P(s_1 \mid s_1) & P(s_2 \mid s_1) & \cdots & P(s_N \mid s_1) \\ P(s_1 \mid s_2) & P(s_2 \mid s_2) & \cdots & P(s_N \mid s_2) \\ \vdots & \vdots & \ddots & \vdots \\ P(s_1 \mid s_N) & P(s_2 \mid s_N) & \cdots & P(s_N \mid s_N) \end{pmatrix}$$

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



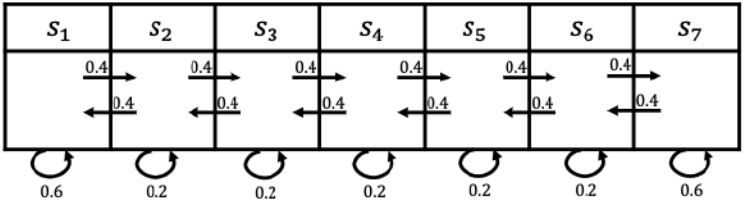
Example: Sample episodes starting from s_4

- $s_4, s_5, s_6, s_7, s_7, \ldots$
- $s_4, s_4, s_5, s_4, s_5, s_6, \ldots$
- $s_4, s_3, s_2, s_1, \ldots$

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 $P(s_{t+1} = s' | s_t = s)$
  - 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



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

Return

Definition

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

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

- 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

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 $\gamma = 1$

Value Function

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

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

Bellman Equation

Bellman Equation for MRPs

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

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

Computing the Value of a Markov Reward Process

- Markov property provides structure
- MRP value function satisfies

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

Matrix Form of Bellman Equation for MRP

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

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

$$V = R + \gamma P V$$

Analytic Solution for Value of MRP

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

$$V = R + \gamma P V$$

$$V - \gamma P V = R$$

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

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

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

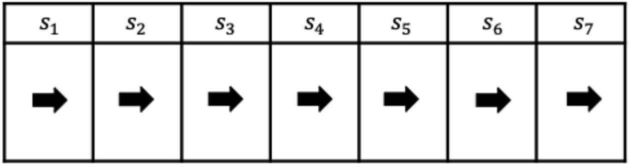
Iterative Algorithm for Computing Value of a MRP

- 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 ($|S| = N$)

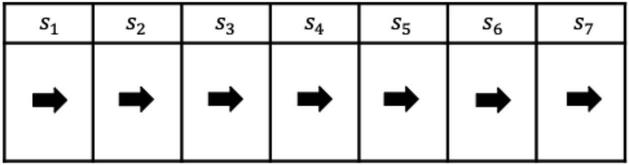
Example: Mars Rover Policy Evaluation



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

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

Example: Mars Rover Policy Evaluation



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

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

Answer:

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