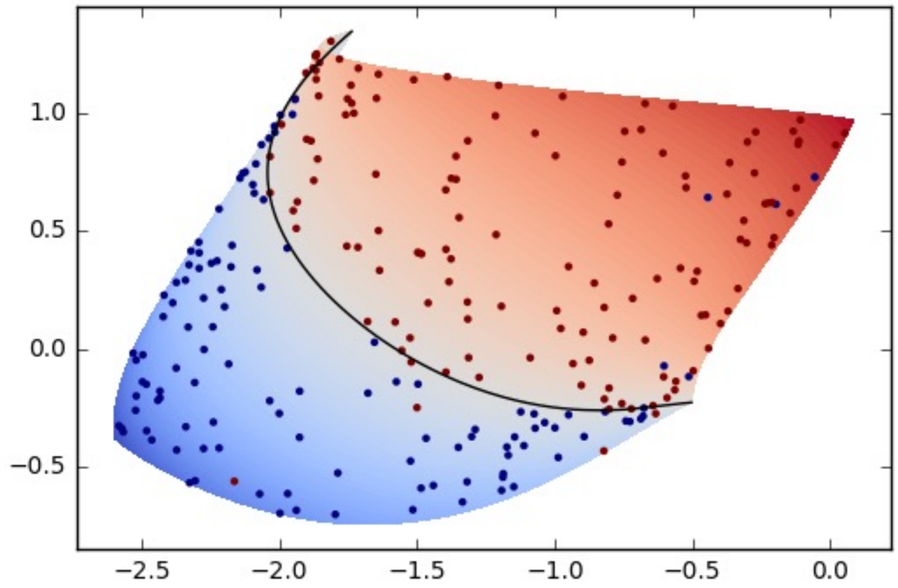
Lecture 24: Reinforcement Learning



Gavin Kerrigan

Spring 2023

Some slides adapted from Padhraic Smyth, Alex Ihler

Announcements

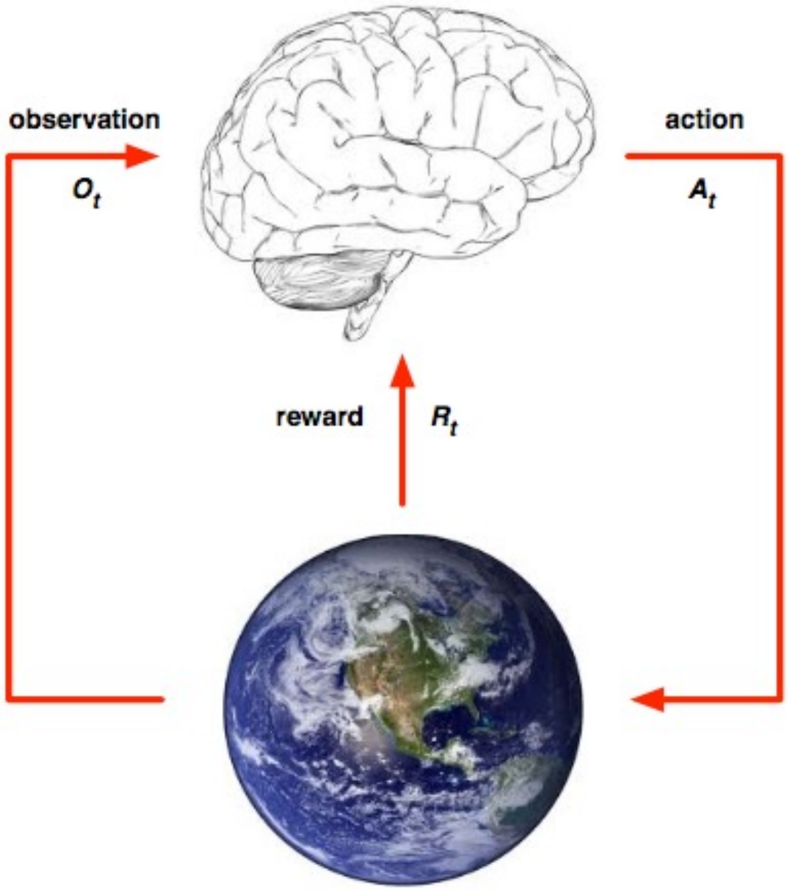
- HW4 due tonight
- HW5 released by Monday
  - Due in 2 weeks (Friday 6/9, last day of class)
  - Short assignment on clustering
- Final exam details discussed next week
  - Similar format to midterm
- No class on Monday (Memorial day)

Intro to Reinforcement Learning

Multi-Armed Bandits

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

Reinforcement Learning is Hard

Agent takes in a history of states, actions, and rewards

Tries to predict the best action to take next

- No direct supervision: only rewards
  - Agent must figure out what the best actions are -- not provided as training data
  - "Exploration"
- Feedback is often delayed
  - Example: only getting a reward if you win a Chess game
- Data has a temporal/sequential aspect
- Environment is changing
  - Agent's actions have an effect

Sequential Decision Making

Actions have long-term consequences

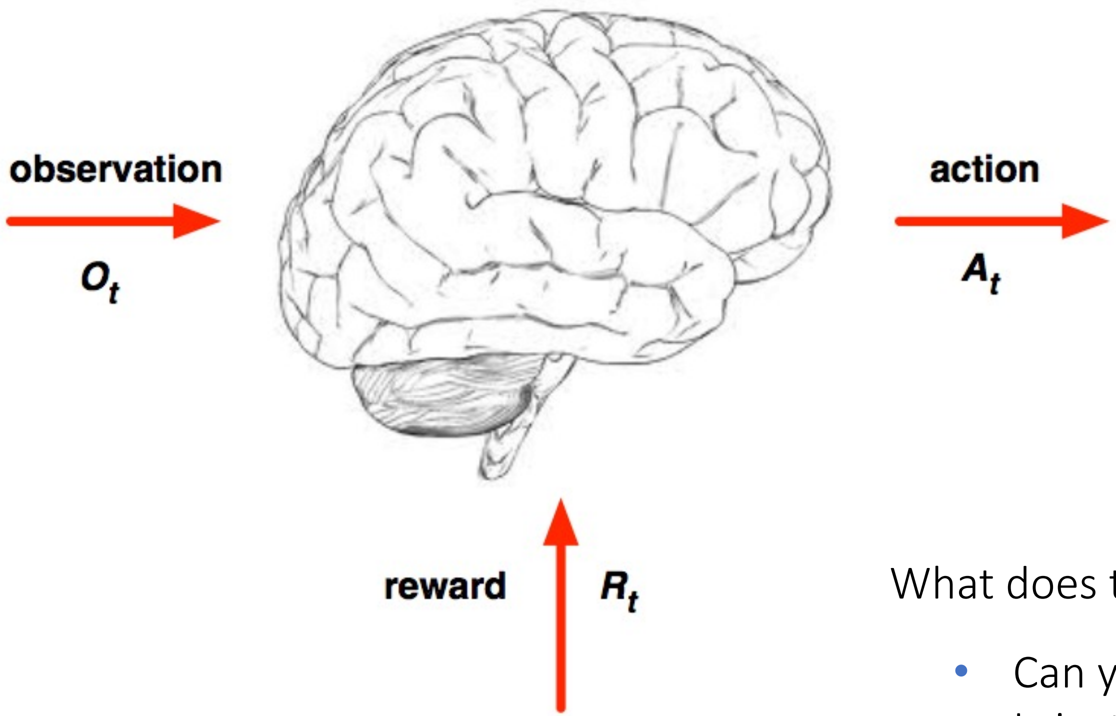
May be better to sacrifice short-term rewards for long-term benefit

Examples:

- Buying Apple stock 20 years ago
- Going into debt to go to college
- Starting on a course project early in the quarter

One aspect of *general intelligence* is the ability to plan far into the future

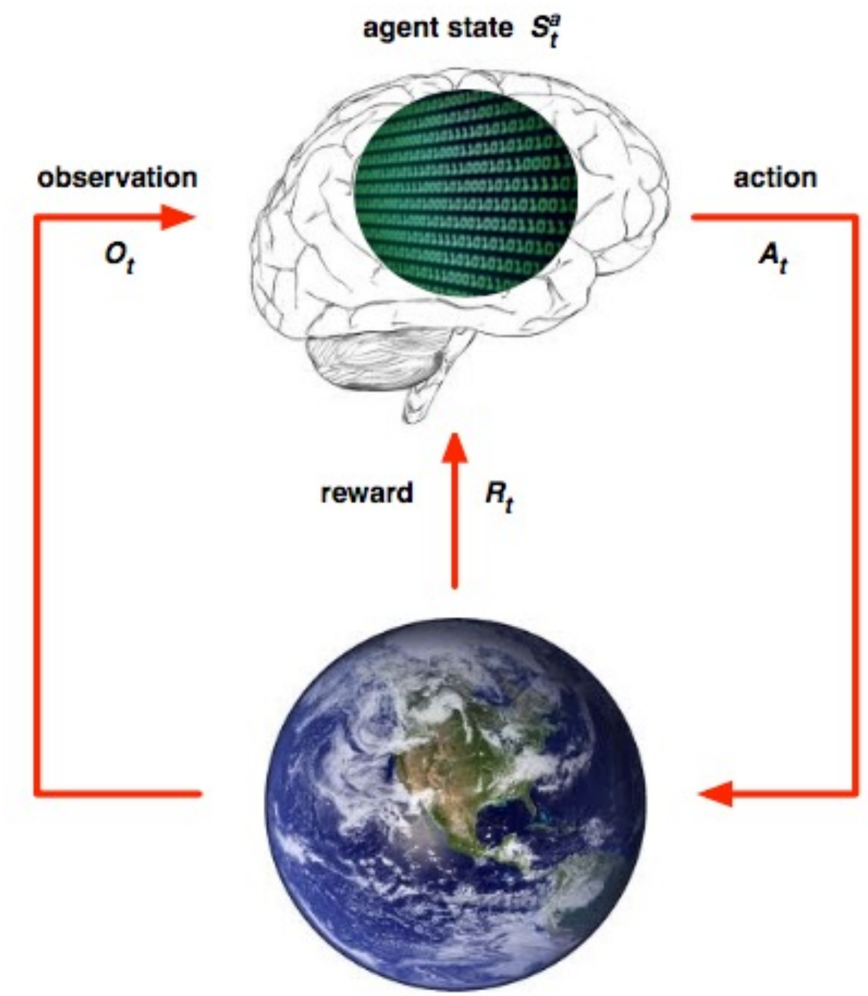
Agents



What does the choice of action depend on?

- Can you ignore O_t completely?
- Is just O_t enough? Or (O_t, A_t)?
- Is it last few observations?
- Is it all observations so far?

Agent State, S_t



History: everything that happened so far

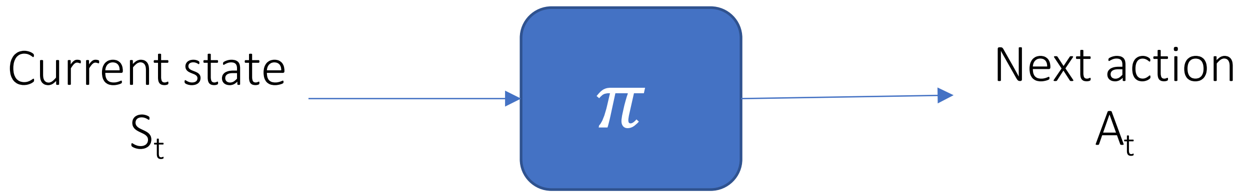
$H_t = O_1R_1A_1O_2R_2A_2O_3R_3,\ldots,A_{t-1}O_tR_t$

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: $A_t = \pi(S_t)$

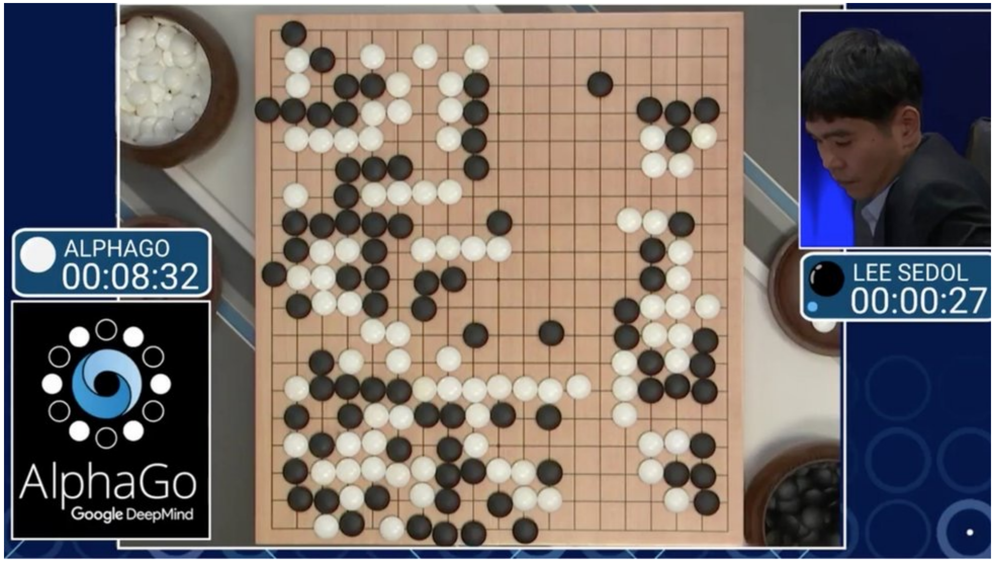
Stochastic Policy: $\pi(a|s) = P(A_t = a|S_t = s)$

Good policy: Leads to larger cumulative reward

Bad policy: Leads to worse cumulative reward

Examples of RL Problems

Learning to Play Go



https://www.bbc.com/news/technology-35785875

State:

- Location of all pieces on the board
- History of last few board positions

Actions:

- Where to place a piece

Rewards:

- Positive if you win; negative if you lose

Examples of RL Problems

Learning to Get Rich

https://www.cnet.com/personal-finance/investing/its-been-a-wild-ride-stock-market-predictions-for-the-next-year/

State:

- Prices of many stocks
- How much money you have in the bank
- Additional market data

Actions:

- What stocks to buy/sell

Rewards:

- Proportional to how much profit you make

Examples of RL Problems

Learning to Fly Autonomous Drones

https://www.ansys.com/blog/challenges-developing-fully-autonomous-drone-technology

State:

- Location and orientation of drone
- Velocity of drone
- Current weather

Actions:

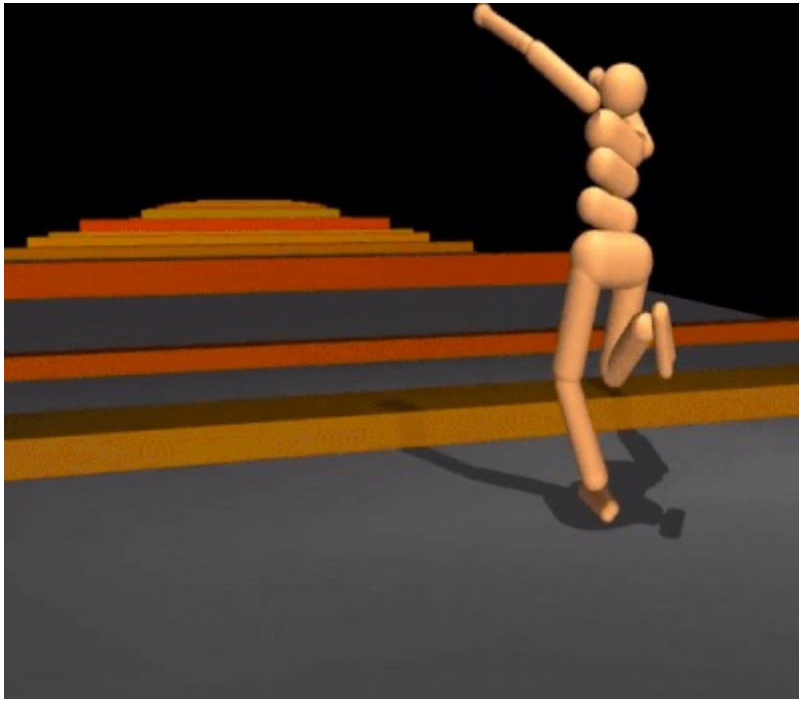
- Moving propellers in a particular fashion

Rewards:

- Positive reward if drone successfully flies to destination
- Negative reward if the drone crashes

Examples of RL Problems

Learning to Walk



https://www.deepmind.com/blog/producing-flexible-behaviours-in-simulated-environments

State:

- Velocity, orientation
- Position of all joints

Actions:

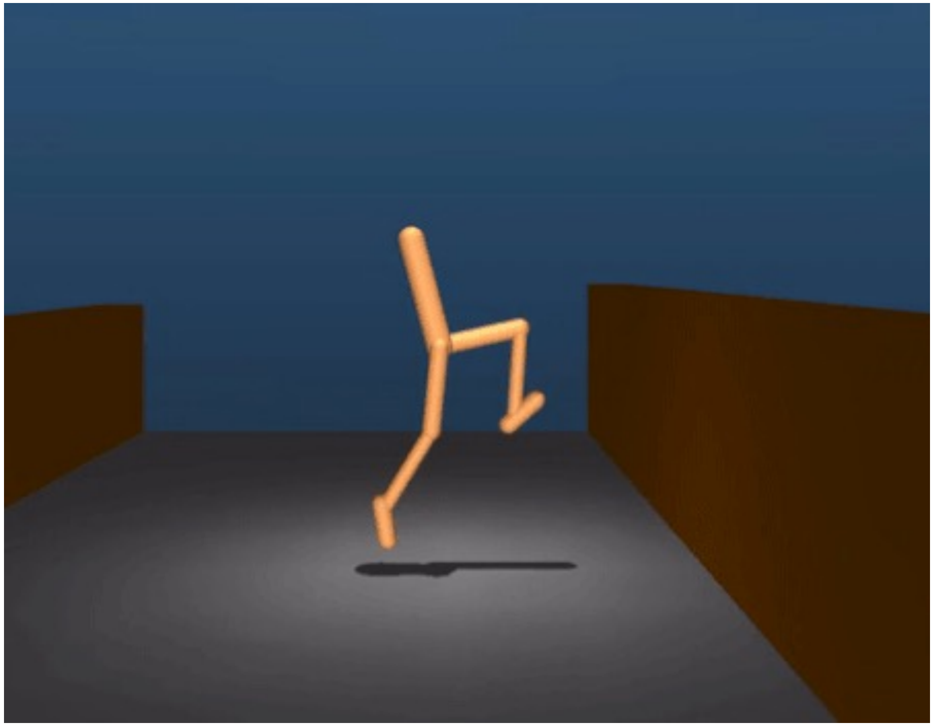
- How to move each individual joint

Rewards:

- Positive reward if you stay upright, make it to goal
- Negative reward if you fall

Examples of RL Problems

Learning to Walk



https://www.deepmind.com/blog/producing-flexible-behaviours-in-simulated-environments

State:

- Velocity, orientation
- Position of all joints

Actions:

- How to move each individual joint

Rewards:

- Positive reward if you stay upright, make it to goal
- Negative reward if you fall

Examples of RL Problems

Playing Atari Games

State:

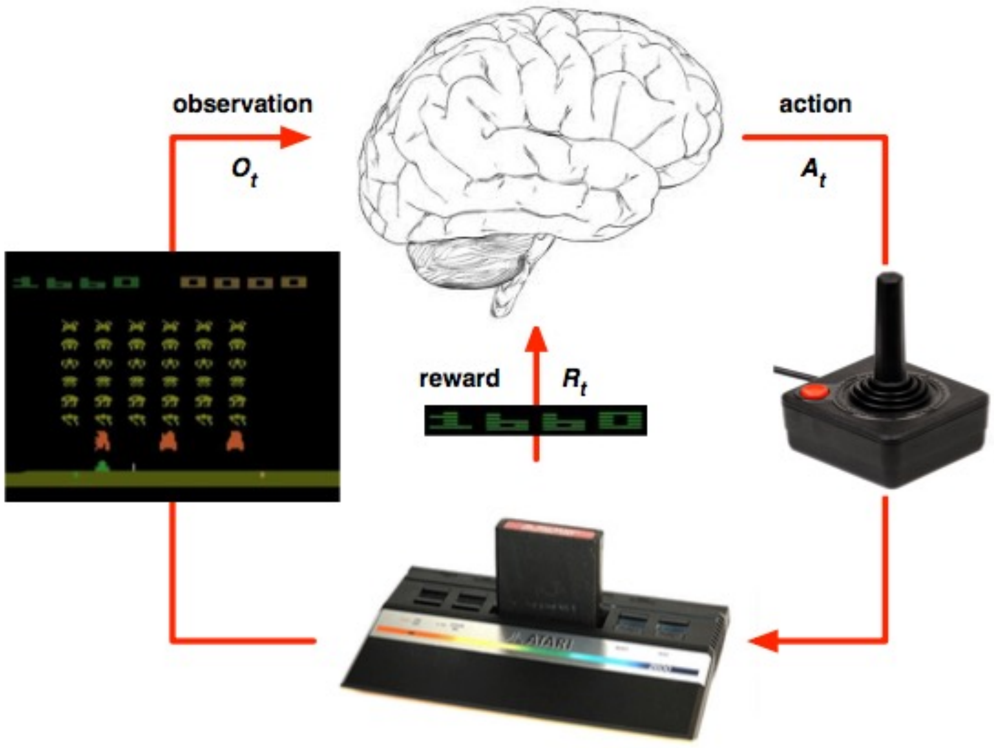
- Pixels on screen

Actions:

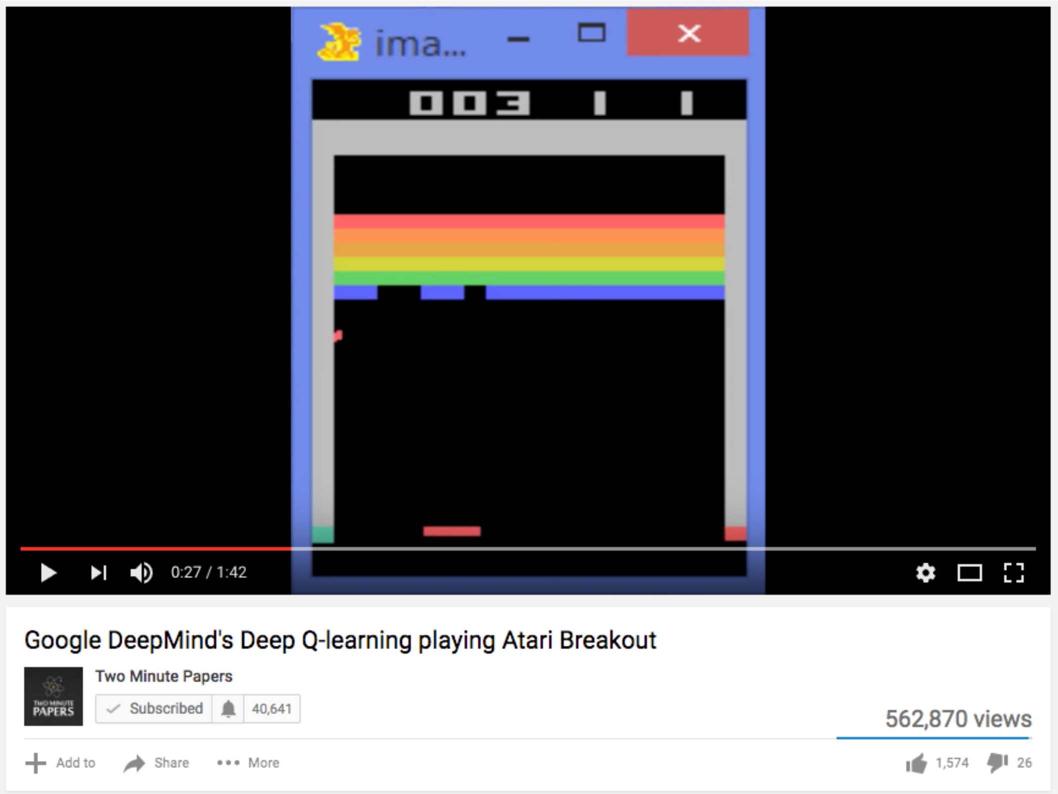
- Moving joystick
- Pressing buttons

Rewards:

- Computed based on current score, number of lives left, etc.



Examples of RL Problems



https://www.youtube.com/watch?v=V1eYniJ0Rnk

Questions?

Intro to Reinforcement Learning

Multi-Armed Bandits

Markov Decision Processes

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”)

Examples of MAB Problems

Online advertising: which ad will result in the most clicks?

Show Ad 1

Show Ad 2

. . . . . .

Show Ad K

Reward:

- +1 if user clicks on the ad
- 0 if user does not click on ad

Average reward for each ad is what proportion of users click on the ad

Examples of MAB Problems

Clinical trials: which medicine will result in the best health outcomes?

Medication 1

Medication 2

. . . . .

Medication K

Reward:

- +1 if patient is cured
- 0 if patient's health does not change
- -1 if patient's health gets worse

Average reward over many trials summarizes how often drug is helpful/harmful

MAB as an RL problem

Actions:

- Which arm to pull at each trial

State:

- Agent's current estimate of average reward per arm
- $S_t = [\hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_K]$ is a K-dimensional vector
- $\hat{\theta}_j =$ (total reward received from arm j) / (total number of times arm j was pulled)

Goal of our agent is to maximize the total reward across T trials

Strategies for the MAB Problem

Note: in each strategy, agent updates S_t after every trial

Strategy 1: Random Guessing

- At each trial, pull a random arm
- Not very smart: we don't adapt to choose arms which give high rewards
- All explore, no exploit

Strategy 2: Explore-then-Exploit

- Pull random arms for the first $S < T$ trials
- Pull the best arm for the remaining $T – S$ trials
- What is wrong with this?
    - Usually want S to be small
    - Can get unlucky and choose a suboptimal arm

Strategies for the MAB Problem

Note: in each strategy, agent updates S_t after every trial

Strategy 3: ϵ-Greedy

- Fix a number $0 < \epsilon < 1$
- At each trial:
    - pull the best arm (highest $\hat{\theta}_j$) with probability 1 - ϵ
    - pull a random arm with probability ϵ
- ϵ explicitly trades-off between exploration and exploitation
    - Needs to be chosen by us, usually small (0.05-0.1)

Strategies for the MAB Problem

Note: in each strategy, agent updates S_t after every trial

Strategy 4: ϵ-Decreasing

- Fix numbers $0 < \epsilon < 1$ and $0 < \alpha < 1$
  - α is a "decay" factor
- At trial t:
  - pull the best arm (highest $\hat{\theta}_j$) with probability 1 - $\epsilon \cdot \alpha^t$
  - pull a random arm with probability $\epsilon \cdot \alpha^t$
- Idea:
  - When t is small, want to explore more (high ϵ)
  - When t is large, want to exploit more (low ϵ)

Many more strategies exist: we won't cover them all here

MAB Example

$\theta_1 = 0.1$

$\theta_2 = 0.3$

$\theta_3 = 0.7$

$\theta_4 = 0.2$

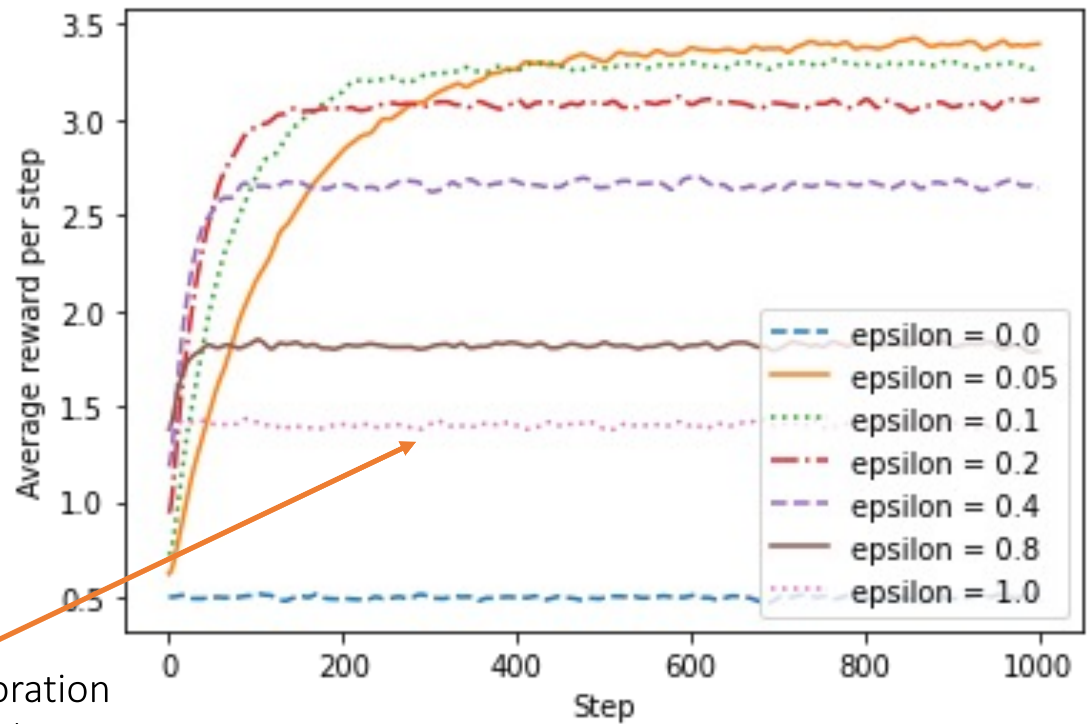
$\theta_5 = 0.1$

Arm k gives a random reward:

- +1 with probability θ_k
- 0 with probability 1 - θ_k

MAB Example

ϵ-Greedy Strategy



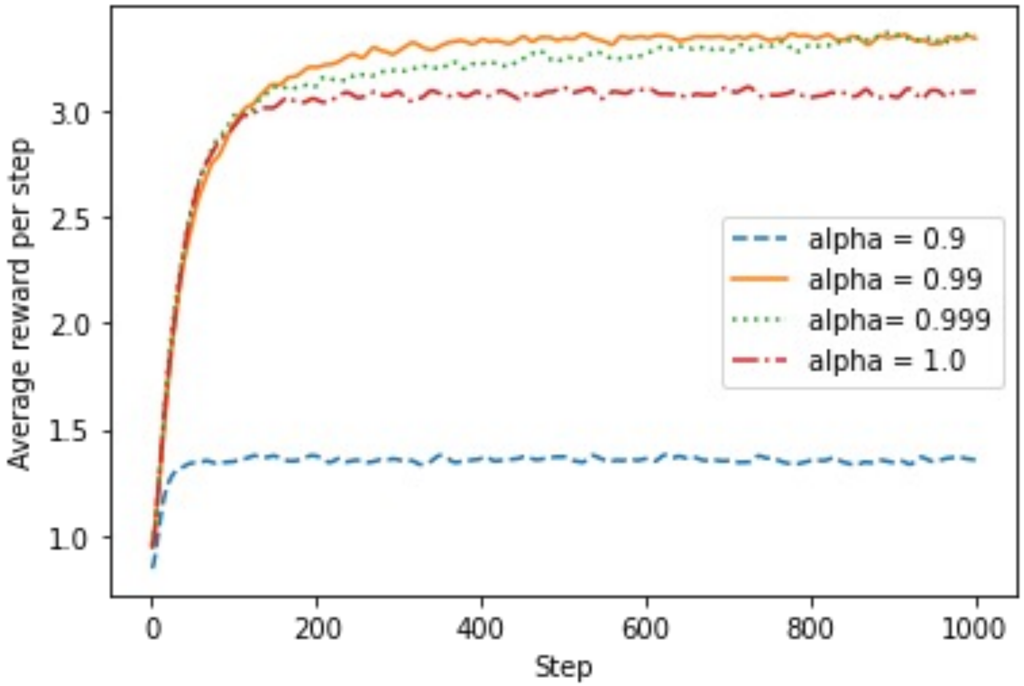
Pure exploration
(Strategy 1)

https://gibberblot.github.io/rl-notes/single-agent/multi-armed-bandits.html

MAB Example

Decreasing ϵ Strategy

Initial ϵ is 0.2



https://gibberblot.github.io/rl-notes/single-agent/multi-armed-bandits.html

Summary of MABs

- Simple RL problem that can model many real-world problems
  - Which ads to show, where to place them
  - Medical trials
  - General alternative to A/B testing

- Need to balance exploration against exploitation
  - Many strategies for explicitly doing so

- Fairly deep topic
  - Can prove bounds on the “regret”
  - comparisons between your strategy and the optimal strategy
  - See e.g. “Regret Analysis…” (Bubeck and Cesa-Bianchi, 2012)

Questions?

Intro to Reinforcement Learning

Multi-Armed Bandits

Markov Decision Processes

Where We're Headed

Markov Property

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

Markov Property

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

Definition

A state S_t is *Markov* if and only if

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

Markov Property

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

Definition

A state S_t is *Markov* if and only if

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

- 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

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

State Transition Matrix

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

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

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

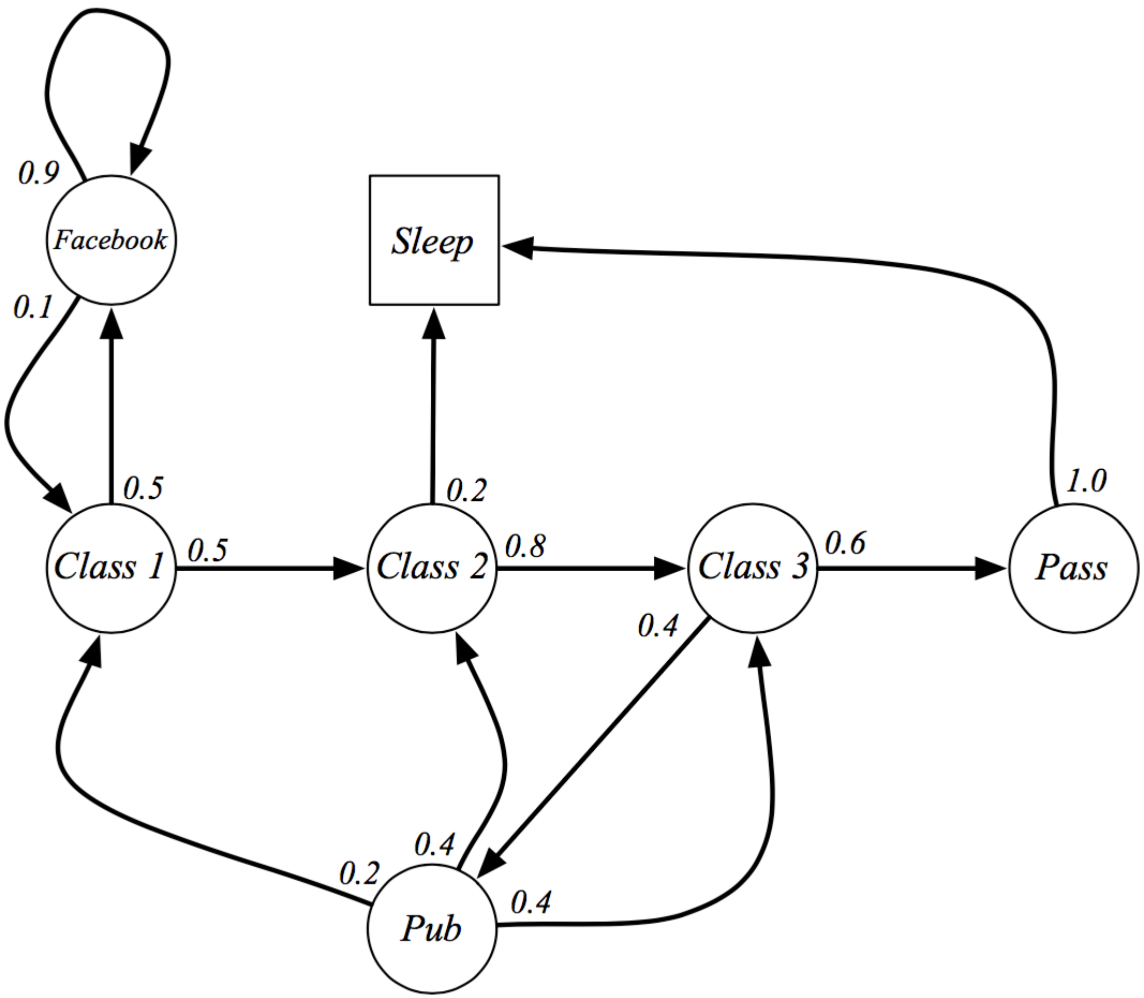
$$\mathcal{P} \;=\; \begin{array}{cc} & \textit{to} \\ \textit{from} & \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{n1} & \dots & \mathcal{P}_{nn} \end{bmatrix} \end{array}$$

where each row of the matrix sums to 1.

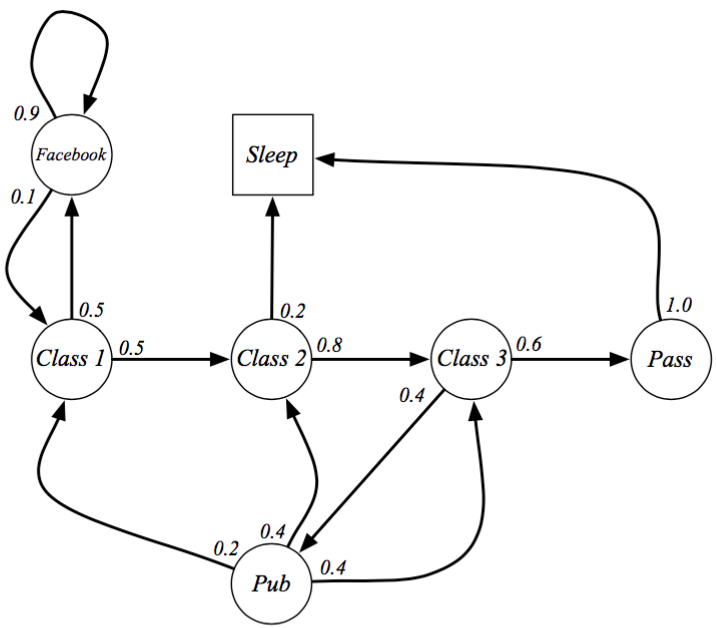
State Transition Matrix

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

Student Markov Chain



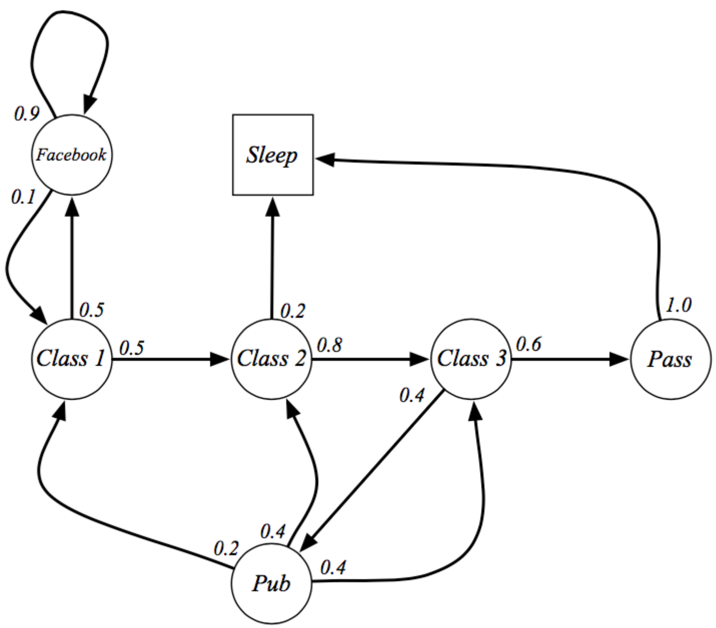
Student Markov Chain



Sample episodes for Student Markov Chain starting from $S_1 = \text{C1}$

$$S_1, S_2, \ldots, S_T$$

Student Markov Chain

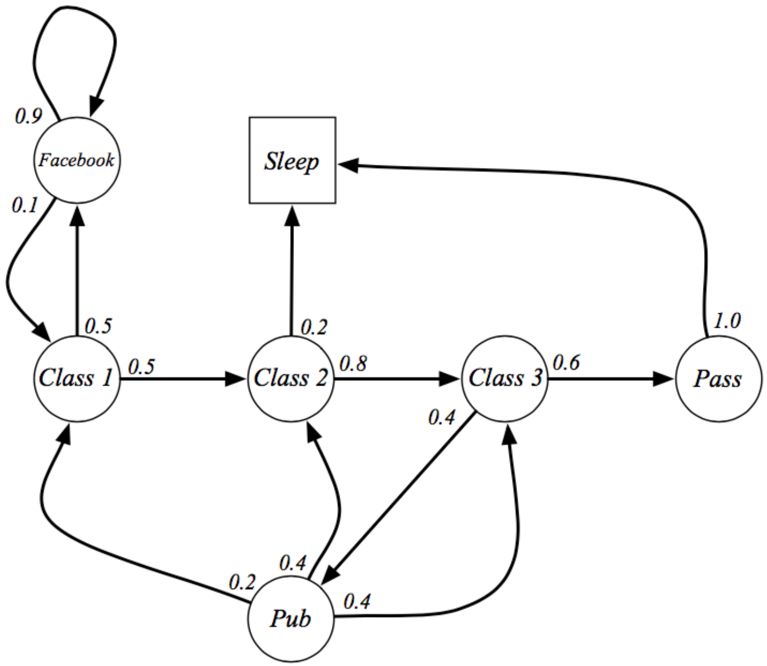


Sample episodes for Student Markov Chain starting from $S_1 = \text{C1}$

$$S_1, S_2, \ldots, S_T$$

- 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

Student Markov Chain



$$\mathcal{P} = \begin{array}{c|ccccccc|} & C1 & C2 & C3 & Pass & Pub & FB & Sleep \\ \hline C1 & & 0.5 & & & & 0.5 & \\ C2 & & & 0.8 & & & & 0.2 \\ C3 & & & & 0.6 & 0.4 & & \\ Pass & & & & & & & 1.0 \\ Pub & 0.2 & 0.4 & 0.4 & & & & \\ FB & 0.1 & & & & & 0.9 & \\ Sleep & & & & & & & 1 \\ \hline \end{array}$$

Markov Chain Demo

http://setosa.io/ev/markov-chains/

Where We're Headed

Wrapup

- Reinforcement Learning
  - An agent interacts with an environment over time
  - Must explore this environment in order to maximize reward over time
- Multi-Armed Bandits
  - Simple RL problem with many real-world applications
- Markov Processes
  - Next state depends only on current state
  - Characterized by transition matrix