2023

22ND ANNUAL

PARIS C. KANELLAKIS

MEMORIAL LECTURE

"Monitoring Health and Diseases Using Radio Signals and Machine Learning"

Dina Katabi

Thuan and Nicole Pham Professor
of Electrical Engineering and Computer Science, MIT

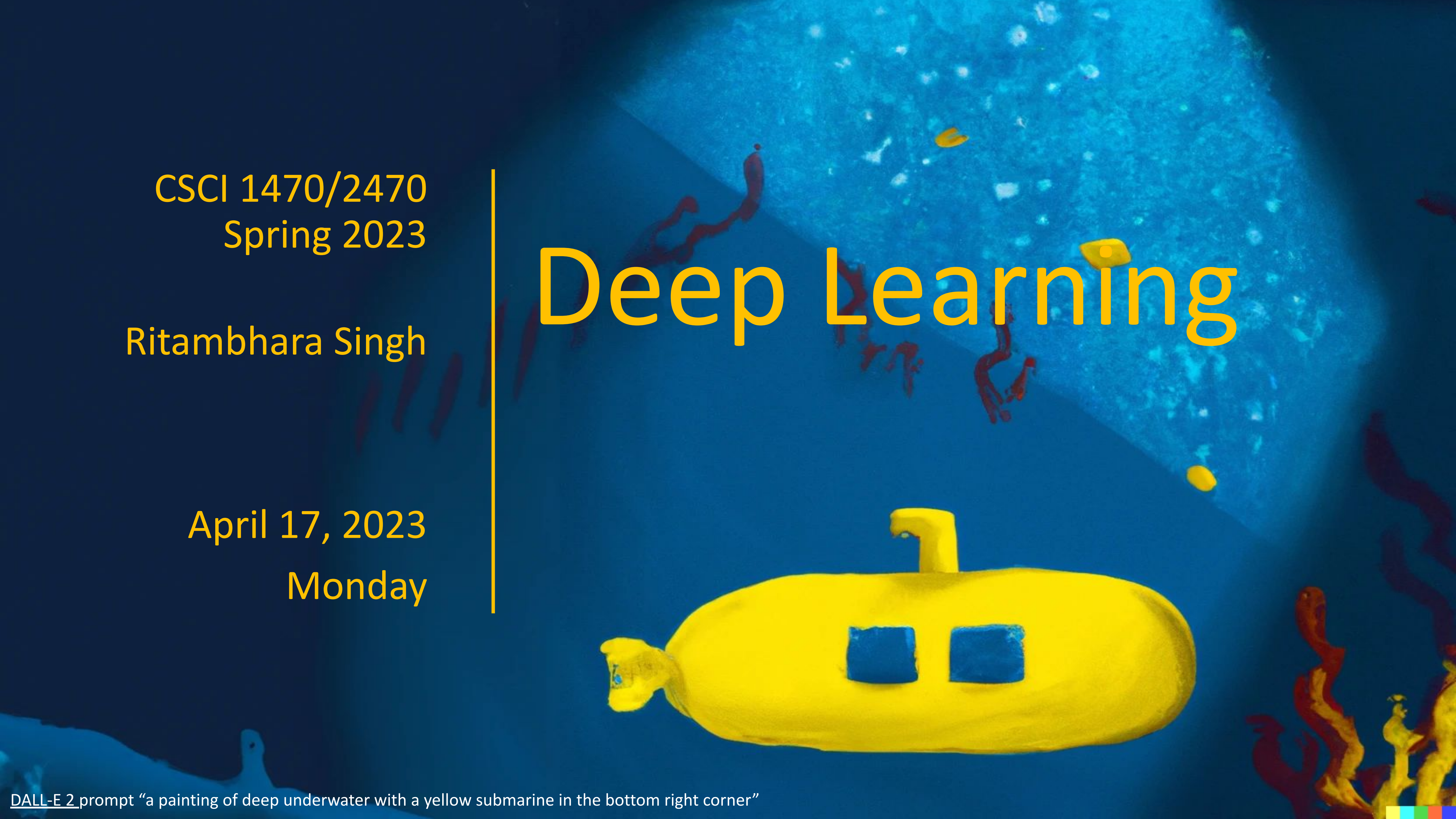
4 PM on April 20 • CIT 368

CSCI 1470/2470
Spring 2023

Ritambhara Singh

April 17, 2023
Monday

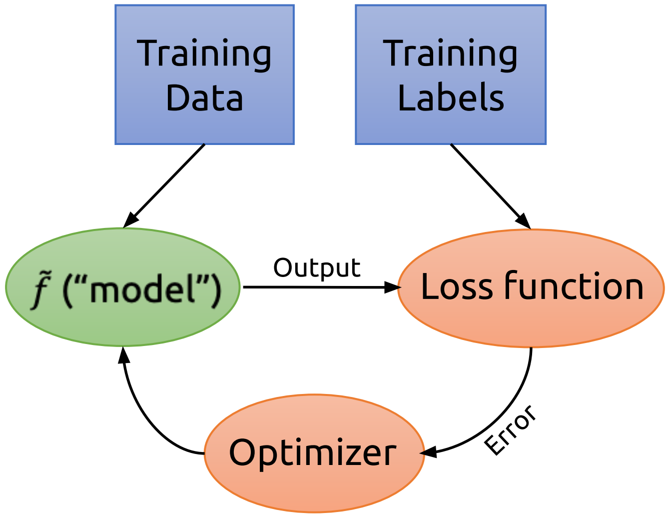
Deep Learning



DALL-E 2 prompt “a painting of deep underwater with a yellow submarine in the bottom right corner”

Different Learning Paradigms

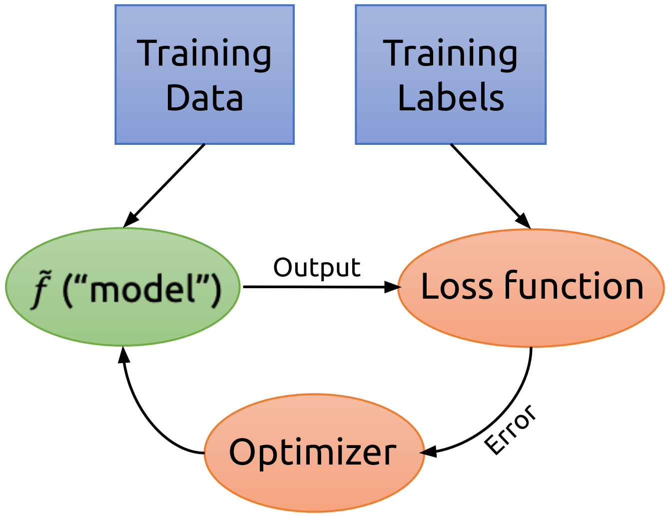
Supervised Learning



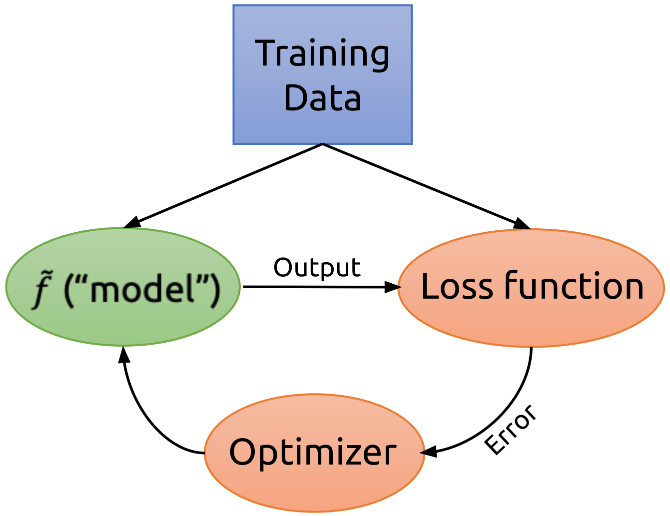
We’ve focused on this thus far…

Different Learning Paradigms

Supervised Learning



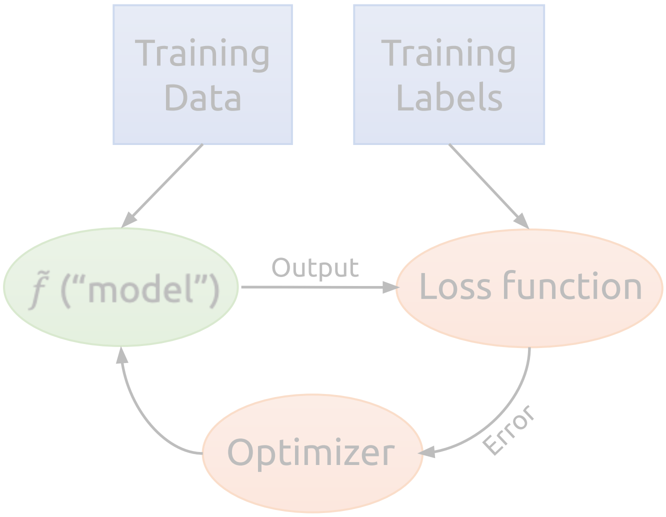
Unsupervised Learning



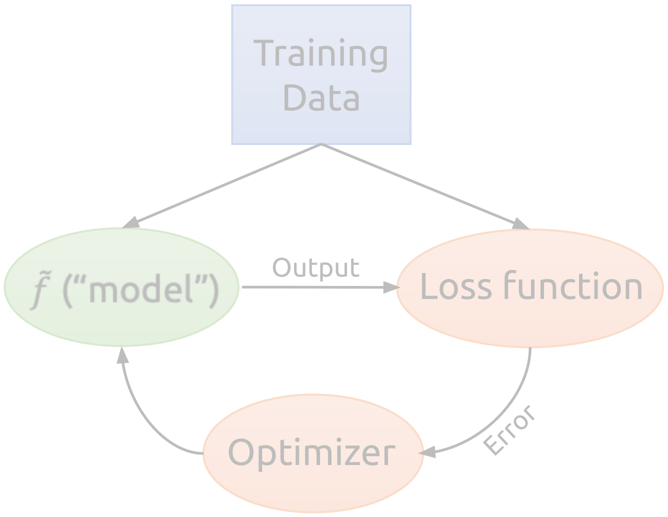
We've focused on this thus far…

Different Learning Paradigms

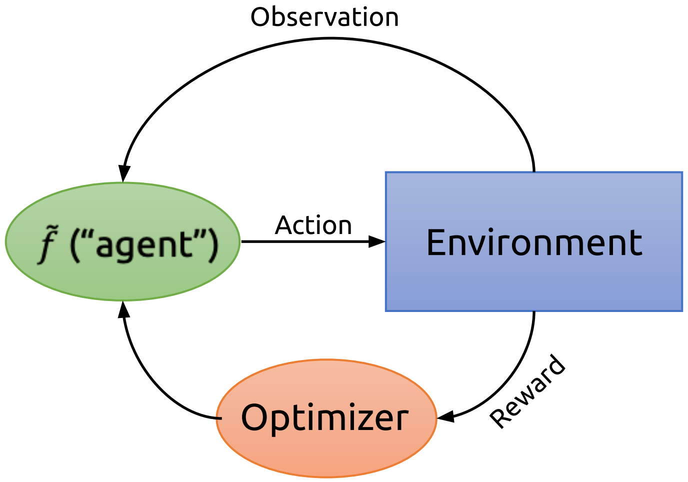
Supervised Learning



Unsupervised Learning



Reinforcement Learning



Now it's time to look at this

Why Reinforcement Learning?

Why Reinforcement Learning?

- Requires less data (e.g. no need for explicit training labels)
- Humans learn from experience, not just labels (e.g. touching hot tea hurts!)

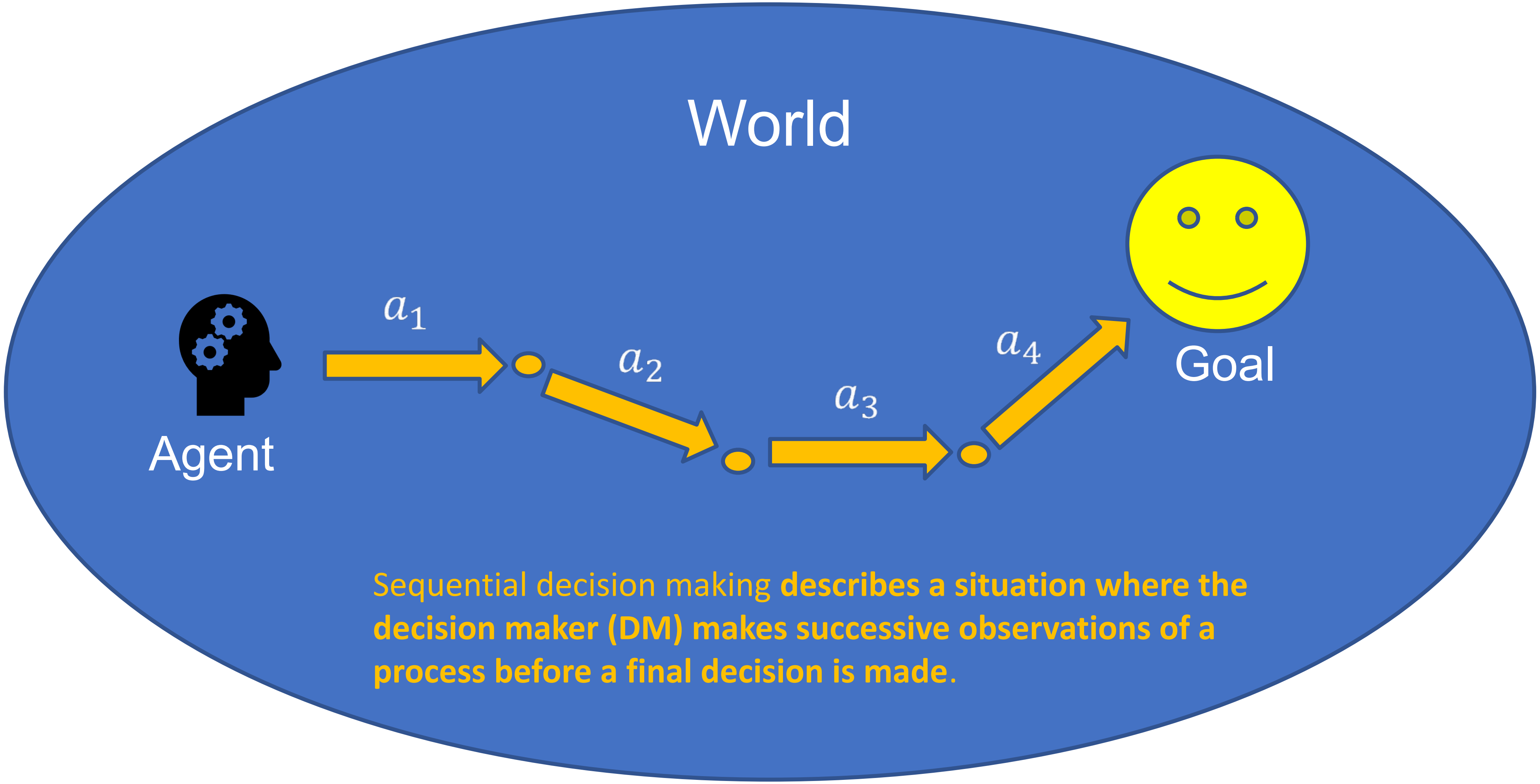
Today’s goal – learn about Reinforcement Learning (RL)

(1) Sequential Decision making

(2) Formalizing RL – Markov Decision Processes

(3) Policies – defining agent behavior

RL: Sequential Decision Making



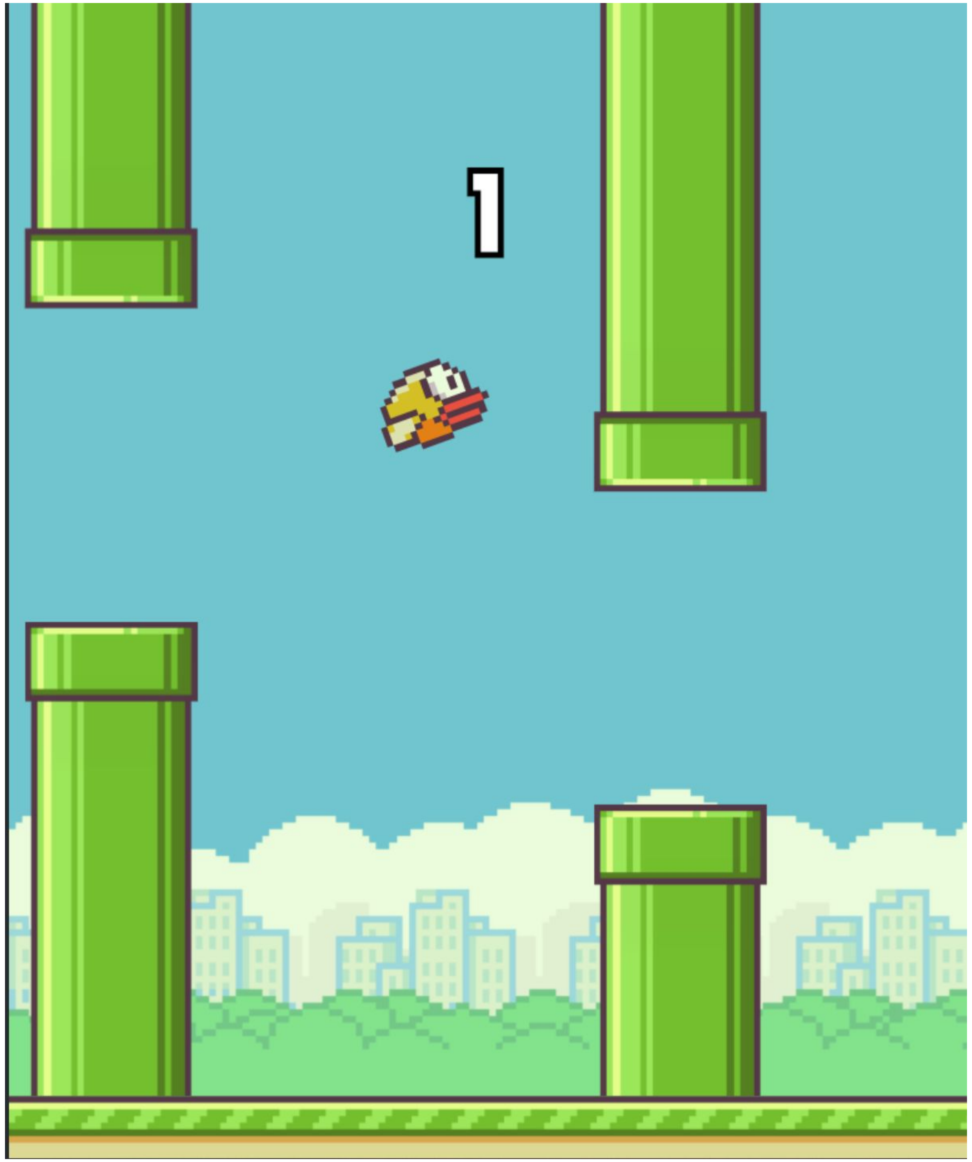
Sequential decision making **describes a situation where the decision maker (DM) makes successive observations of a process before a final decision is made.**

What's a common example of a sequential decision making process?

- Playing games!
- Let’s look at a specific example…

Flappy Bird

https://flappybird.io/



Flappy Bird

What is the goal here?

What are the possible actions?

- Goal: maximize “score” – number of green pipes the bird passes without hitting one
- Actions: **flap** or **don’t flap**

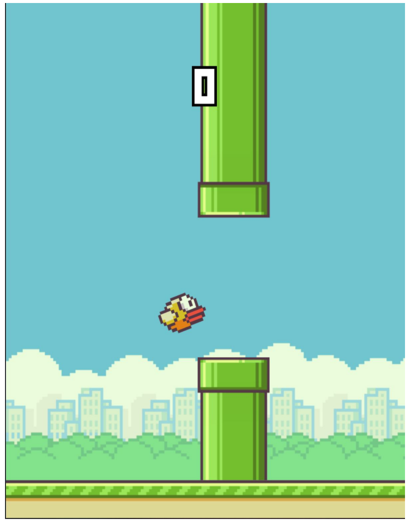
Supervised Learning for Flappy Bird

How can you learn to play using supervised learning?

- Input
  - Training data: image frame
  - Labels: best action to take in that frame

- Goal: learn which action to take for a given frame

Data:

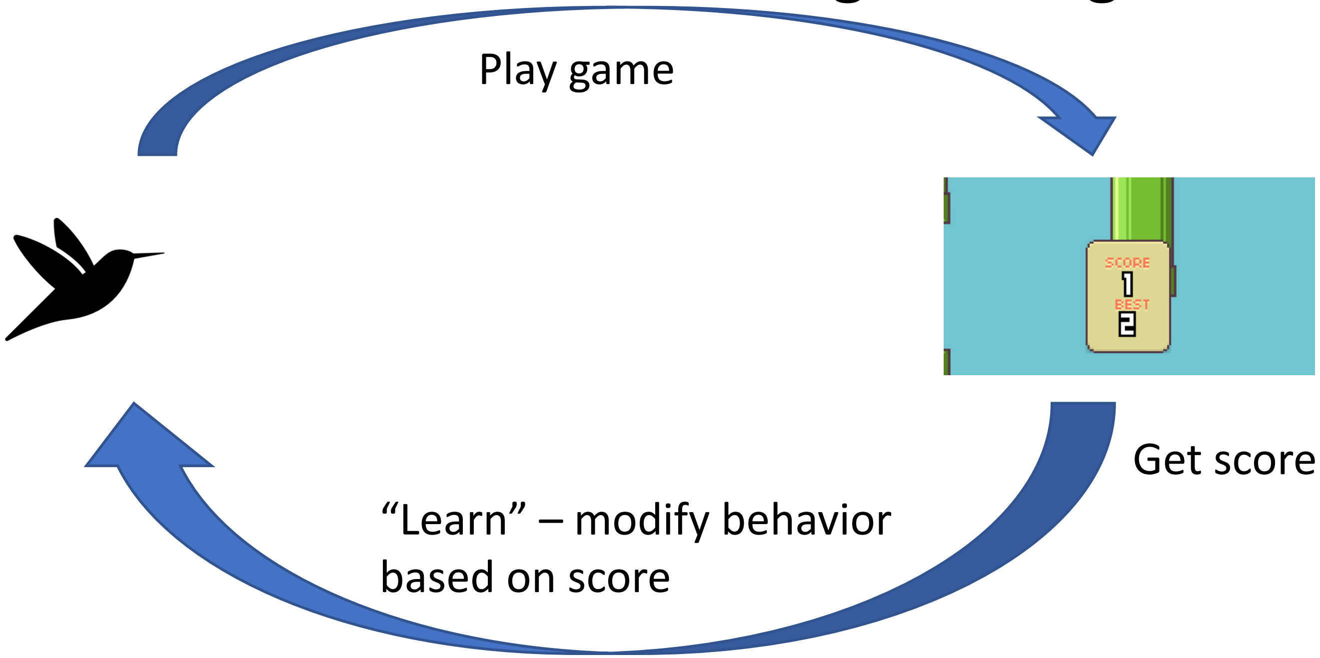


Label:

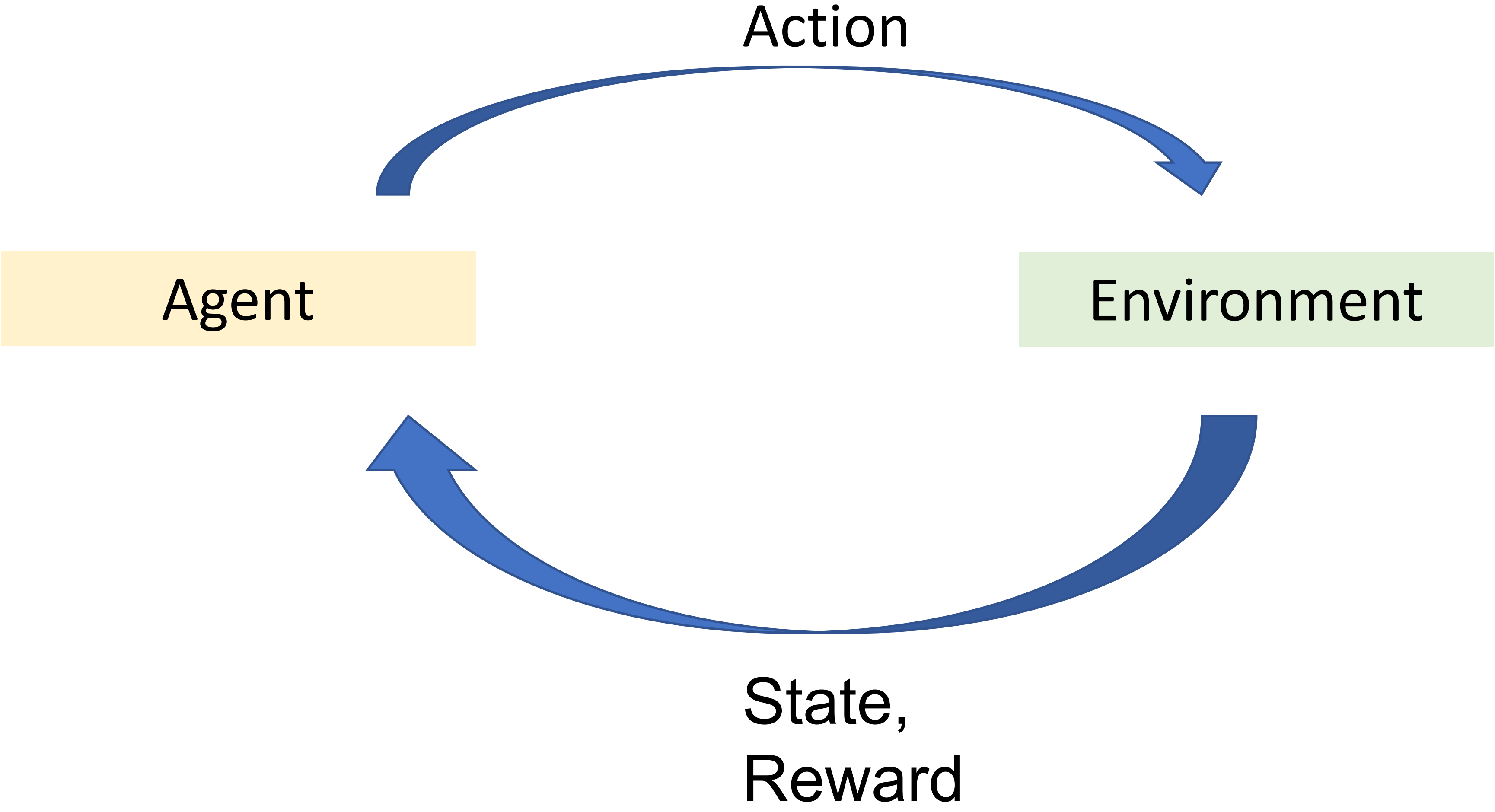
Flap

How does RL learn Flappy Bird?

- Requires no pre-collected data!
- Learn from experience by playing game over and over and over again
- Agent wants to choose actions that will give it higher scores

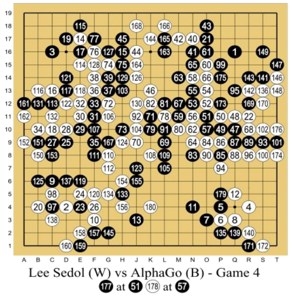


General RL framework



Recent RL Successes

- AlphaGo (DeepMind, 2016)
    - Defeated best Go player in the world
    - 2-hour documentary: https://www.alphagomovie.com/
- AlphaZero (DeepMind, 2017)
    - Defeated best chess, shogi, and Go computers in the world by learning *only from self-play*
- Dota 2 (OpenAI Five, 2019)
    - First AI to beat world champions in esports game
- AlphaStar (DeepMind, 2019)
    - First AI to beat professional StarCraft players
- Autonomous Helicopter Flight (2017)
- Atari games (2015)



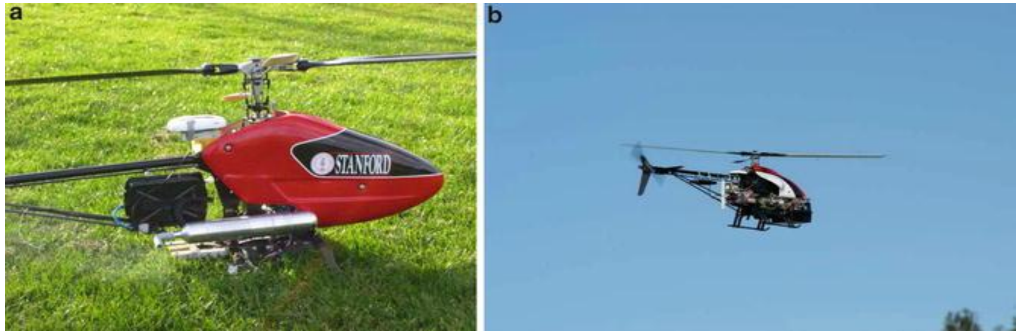
[commons.Wikimedia.org]


[pexels.com]


[Kansky 2017, "Scheme Networks"]


[flickr]

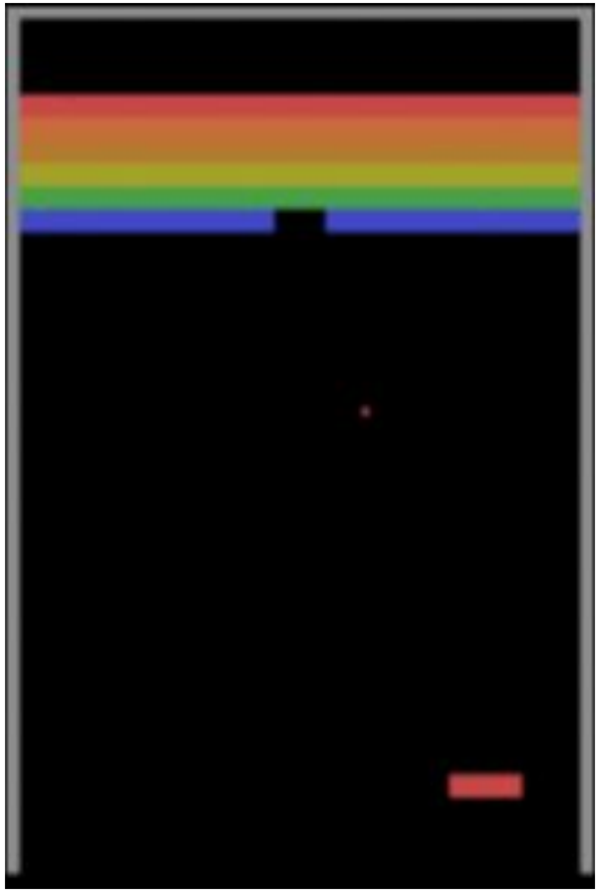
[pngimg.com]



[https://link.springer.com/referenceworkentry/10.1007%2F978-1-4899-7687-1_16]

RL Reservations

- Inefficient with data
  - AlphaGo learned from playing ~100 million games
  - Human Go champion only has played ~50,000 games total
- Sensitive to small perturbations in the environment
  - E.g. Agent successfully trained on Atari Breakout will completely fail if the paddle is shifted a few pixels upward.



[https://www.vicarious.com/2017/08/07/general-game-playing-with-schema-networks/]

Formalizing RL:
Markov Decision Processes

Markov Decision Process (MDP)

Markov Decision Process (MDP)

- States – set of possible situations in a world, denoted S

*Sometimes researchers like to make the distinction -

A **state** s is a complete description of the state of the world. There is no information about the world which is hidden from the state.
An **observation** o is a partial description of a state, which may omit information.

Markov Decision Process (MDP)

- States – set of possible situations in a world, denoted S
- Actions – set of different actions an agent can take, denoted A

Markov Decision Process (MDP)

- States – set of possible situations in a world, denoted S
- Actions – set of different actions an agent can take, denoted A
- Transition function – returns the probability of transitioning to state s' after taking action a in state s, denoted $T(s, a, s')$

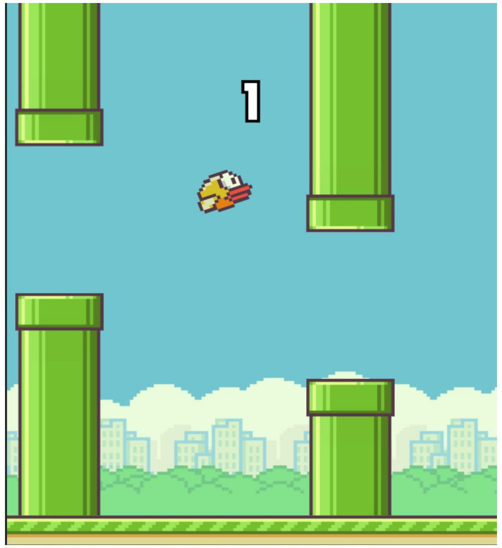
Markov Decision Process (MDP)

- States – set of possible situations in a world, denoted S
- Actions – set of different actions an agent can take, denoted A
- Transition function – returns the probability of transitioning to state s' after taking action a in state s, denoted $T(s, a, s')$
- Reward function – returns the reward received by the agent for transitioning to state s' after taking action a in state s, denoted $R(s, a, s')$

State

- Representation of the "situation" at a point in time

Flappy Bird – full image frame containing bird's location, pipe locations, etc.



You – how you're feeling on a scale of 1-10? – say you feel tired.

Actions

- Set of different actions an agent can do

Flappy Bird – {flap, don’t flap}

You – {take a nap, go out partying}

Transition Function

Transition Function

- Given a current state s, an action a, and another state s', returns the probability of transitioning into s' after taking action a in state s

Transition Function

- Given a current state s, an action a, and another state s', returns the probability of transitioning into s' after taking action a in state s

- In other words: how does choosing an action in one state transition you to the next state?

Transition Function

- Given a current state s, an action a, and another state s', returns the probability of transitioning into s' after taking action a in state s

- In other words: how does choosing an action in one state transition you to the next state?

- $T: S \times A \times S \rightarrow \mathbb{R}$

Transition Function

- Given a current state s, an action a, and another state s', returns the probability of transitioning into s' after taking action a in state s

- In other words: how does choosing an action in one state transition you to the next state?

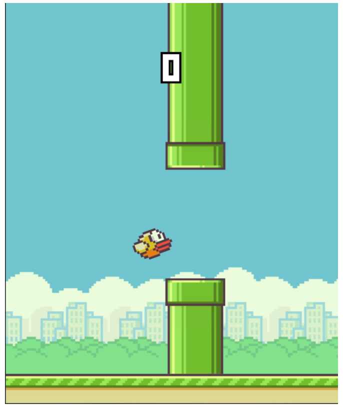
- $T: S \times A \times S \rightarrow \mathbb{R}$

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

Transition Function Examples

- Flappy Bird

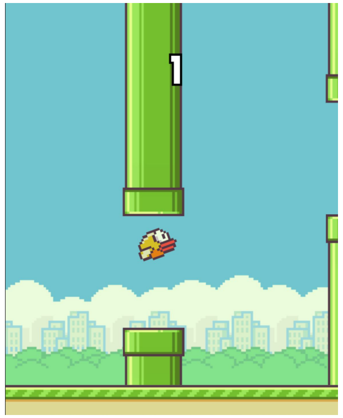
Current state s:



Action a:

Flap!

Next state s':



- Flappy Bird is deterministic, so probability is 1

- You
  - Current state: feeling tired
  - Action: go out partying
  - What's probability your next state is too tired to attend your friend's birthday brunch?

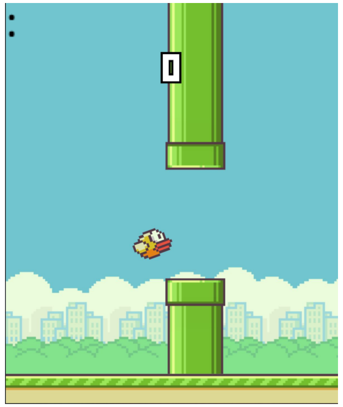
Any questions?

Reward Function

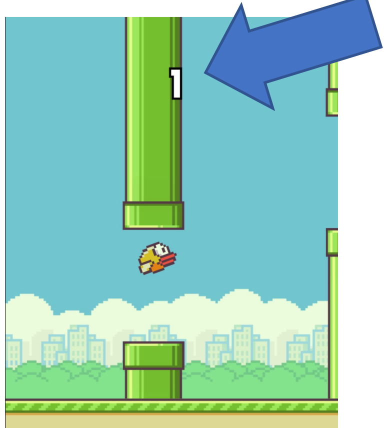
- Given s, a, s', return the reward from taking a in s and transitioning to s'

- Flappy Bird
    - Reward is 1 when passing tubes, 0 otherwise

Current state s: | Action a: | Next state s': | $R(s, a, s') = 1$



Flap!



- You
    - Current state: feeling tired
    - Action: go out partying
    - Next state: too tired to attend your friend's birthday brunch

    Reward:

[pixabay.com[

Why is it called “Markov Decision Process”?

Why is it called “Markov Decision Process”?

- “Decision” – agent “decides” on which action to take

Why is it called “Markov Decision Process”?

- “Decision” – agent “decides” on which action to take
- “Process” – stuff happens over time (e.g. states change)

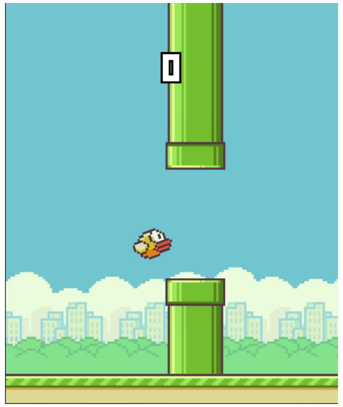
Why is it called “Markov Decision Process”?

- “Decision” – agent “decides” on which action to take
- “Process” – stuff happens over time (e.g. states change)
- “Markov” – “history doesn’t matter”; next state depends only on current state and action, not a history of the previous states

Why is it called "Markov Decision Process"?

- "Decision" – agent "decides" on which action to take
- "Process" – stuff happens over time (e.g. states change)
- "Markov" – "history doesn't matter"; next state depends only on current state and action, not a history of the previous states
- Formally: $P(s_{t+1} \mid s_t) = P(s_{t+1} \mid s_t, s_{t-1}, \ldots, s_1)$

Current state s:

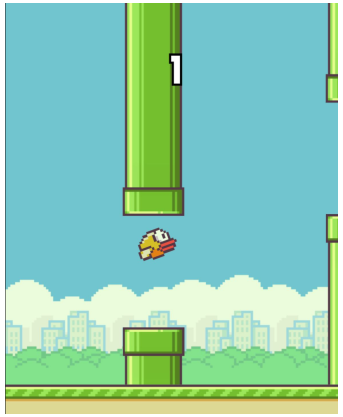


Action a:

Flap!

ONLY NEED s, a

Next state s':



Solving an MDP

What is the goal here?

Goal : To maximize the cumulative reward (sum over future rewards)

$$G_t = R_{t+1} + R_{t+2} + \dots$$

$$G_t = \sum_{k=0}^{T} R_{t+k+1}$$

In reality, we can't just add the rewards like that…

https://medium.com/free-code-camp/an-introduction-to-reinforcement-learning-4339519de419

Solving an MDP

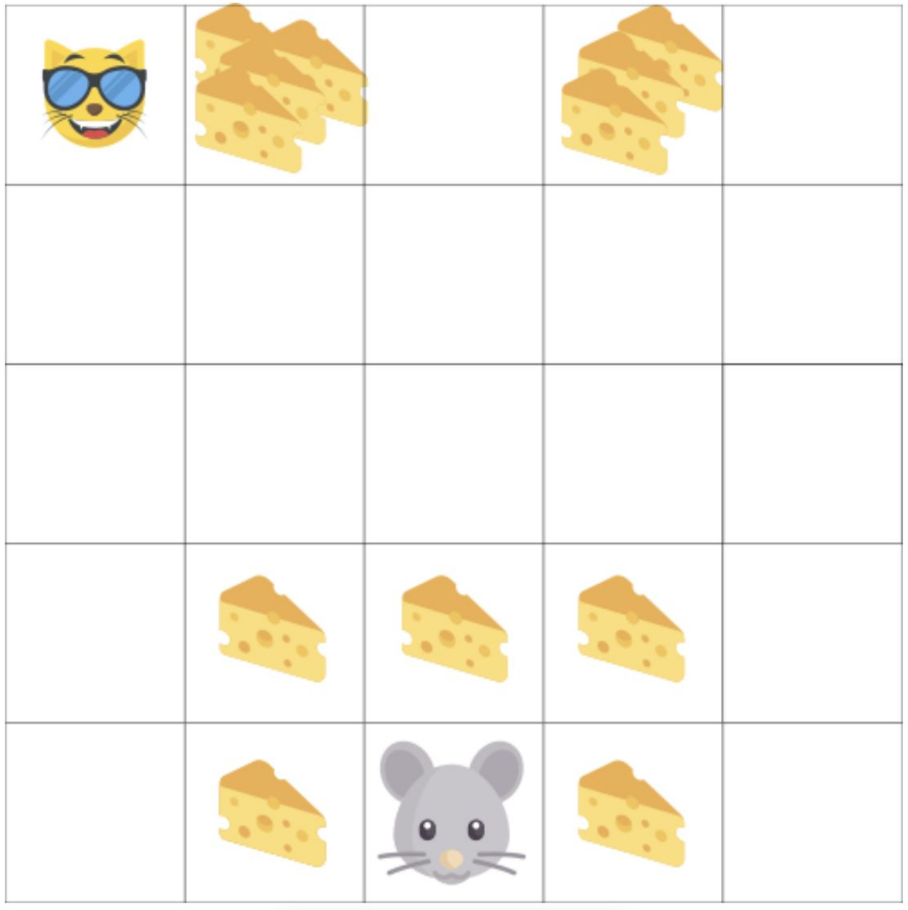
Agent: mouse

Goal: eat the maximum amount of cheese before being eaten by the cat.

It is more probable to eat the cheese near us than the cheese close to the cat (the closer we are to the cat, the more dangerous it is).

As a consequence, the reward near the cat, even if it is bigger (more cheese), will be discounted.

We're not really sure we'll be able to eat it.



https://medium.com/free-code-camp/an-introduction-to-reinforcement-learning-4339519de419

Solving an MDP

- Goal: maximize sum of discounted future rewards, G_t, aka “return”

Solving an MDP

- Goal: maximize sum of discounted future rewards, G_t, aka "return"
- $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

Solving an MDP

- Goal: maximize sum of discounted future rewards, G_t, aka “return”
- $G_t \ = R_{t+1} + \ \gamma R_{t+2} + \gamma^2 R_{t+3} + \ \dots = \ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$
- γ is “discount factor” - value rewards sooner rather than later (prefer \$100 now over \$100 in ten years)

Solving an MDP

- Goal: maximize sum of discounted future rewards, G_t, aka "return"
- $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$
- γ is "discount factor" - value rewards sooner rather than later (prefer $100 now over $100 in ten years)
- Rewards exponentially decay the later in time they are received from the present!

Sum of Infinite Geometric series

$$\gamma \in [0, 1)$$

Solving an MDP

- Goal: maximize sum of discounted future rewards, G_t, aka "return"
- $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$
- γ is "discount factor" - value rewards sooner rather than later (prefer $100 now over $100 in ten years)
- Rewards exponentially decay the later in time they are received from the present!
  - E.g. if reward = 1 at every time step and discount factor is 0.9, then

Let's calculate G_t

Solving an MDP

Any questions?

- Goal: maximize sum of discounted future rewards, G_t, aka "return"
- $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$
- γ is "discount factor" - value rewards sooner rather than later (prefer \$100 now over \$100 in ten years)
- Rewards exponentially decay the later in time they are received from the present!
  - E.g. if reward = 1 at every time step and discount factor is 0.9, then

$$G_t = 1 + 0.9 + 0.9^2 + \cdots = \sum_{k=0}^{\infty} 0.9^k = 10$$

Today’s goal – learn about Reinforcement Learning (RL)

(1) Sequential Decision making

(2) Formalizing RL – Markov Decision Processes

(3) Policies – defining agent behavior

Policies: defining agent behavior

Policy Function

Policy Function

- What action should the agent take in a given state?

Policy Function

- What action should the agent take in a given state?
- Concretely:

Policy Function

- What action should the agent take in a given state?
- Concretely:
- $\pi: S \rightarrow A$

Policy Function

- What action should the agent take in a given state?
- Concretely:
- $\pi: S \rightarrow A$
- Input: state $s \in S$

Policy Function

- What action should the agent take in a given state?
- Concretely:
- $\pi: S \rightarrow A$
- Input: state $s \in S$
- Output: action to be chosen in that state

Policy Function

- What action should the agent take in a given state?
- Concretely:
- $\pi: S \to A$
- Input: state $s \in S$
- Output: action to be chosen in that state
- $\pi(s) = a$ means in state s, take action a

Policy Function Examples

- Flappy Bird:
  - Good policy might be "**flap**" in any state in which the bird is just below the upcoming opening (will lead to passing through opening)
  - Bad policies might be "**never flap**" (will lead to falling into ground and losing)
  - Infinite number of policies!
- You:
  - Policy #1: Never go out partying (bad!)
    - Independent of inputted state
  - Policy #2: Go out partying only if you are not tired

Goal of RL

Goal of RL

- Learn optimal policy $\pi*$ that maximizes the expected future cumulative reward

Goal of RL

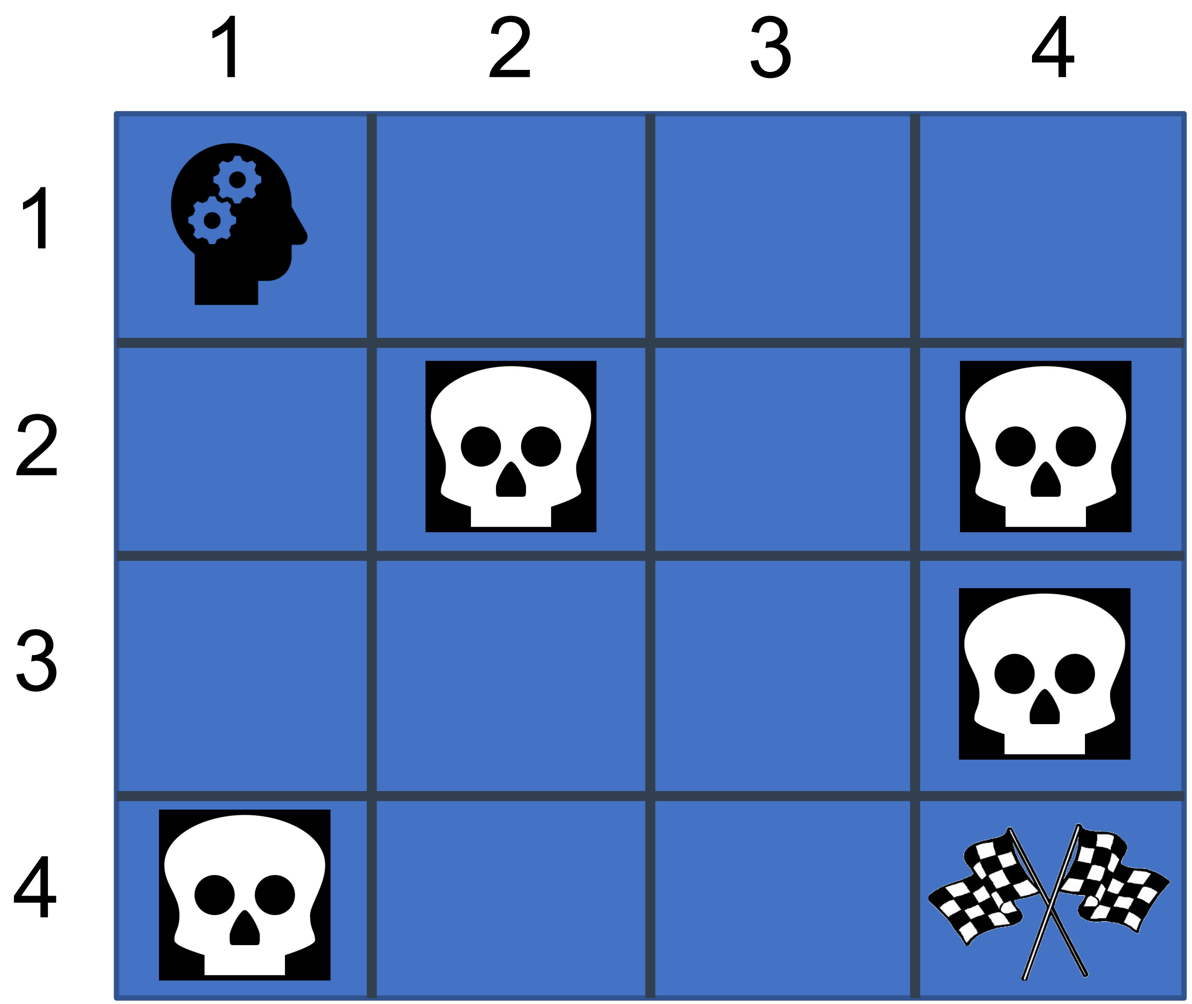
- Learn optimal policy π^* that maximizes the expected future cumulative reward
  - "Expected" because transitions can be non-deterministic

Any questions?

Goal of RL

- Learn optimal policy π^* that maximizes the expected future cumulative reward
  - "Expected" because transitions can be non-deterministic
- Solving MDPs ←→ find this optimal policy!

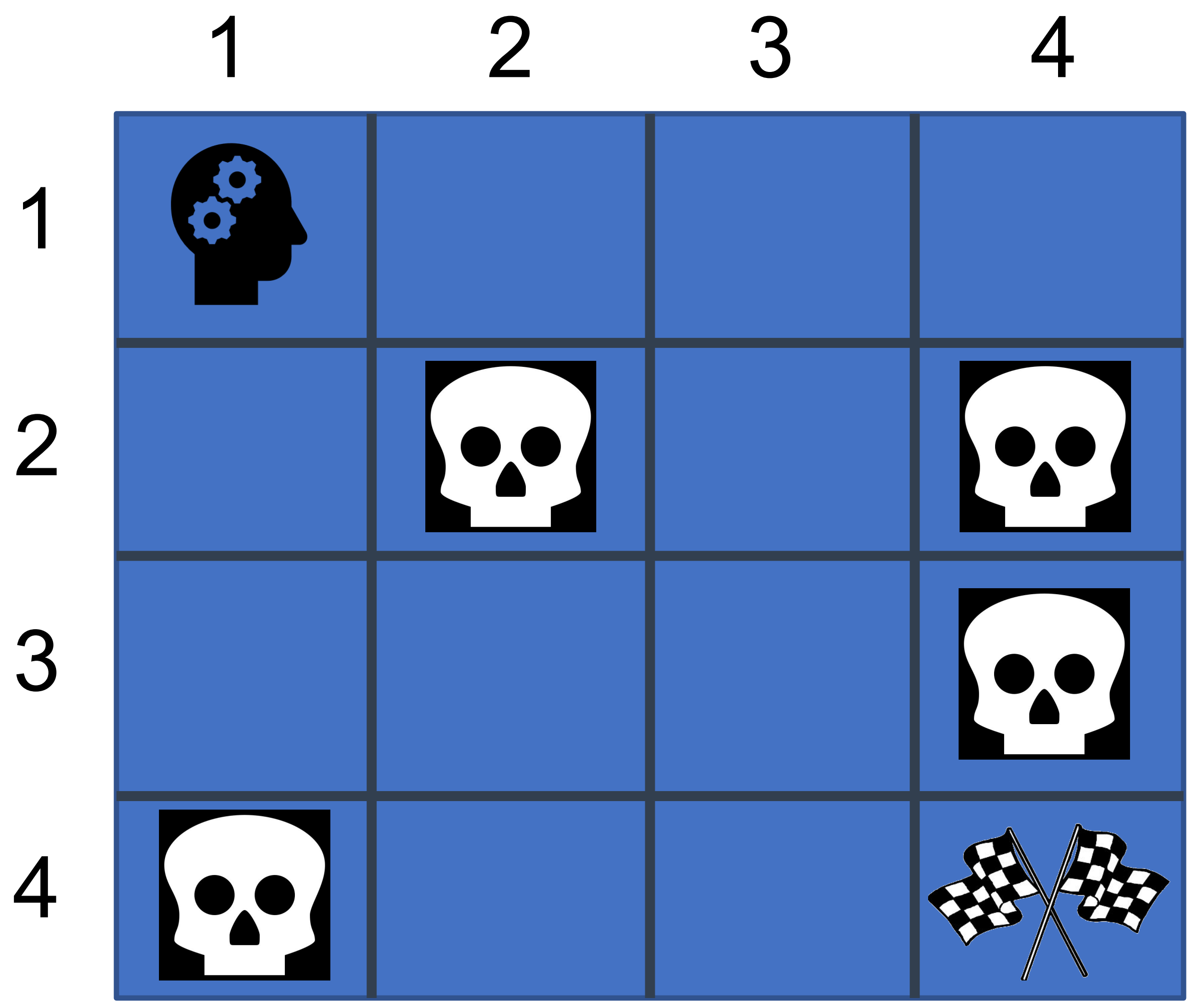
Putting it all together



- Agent starts in top left corner
- Goal: Reach the bottom right without dying (skulls)
- Game terminates when agent dies or reaches goal

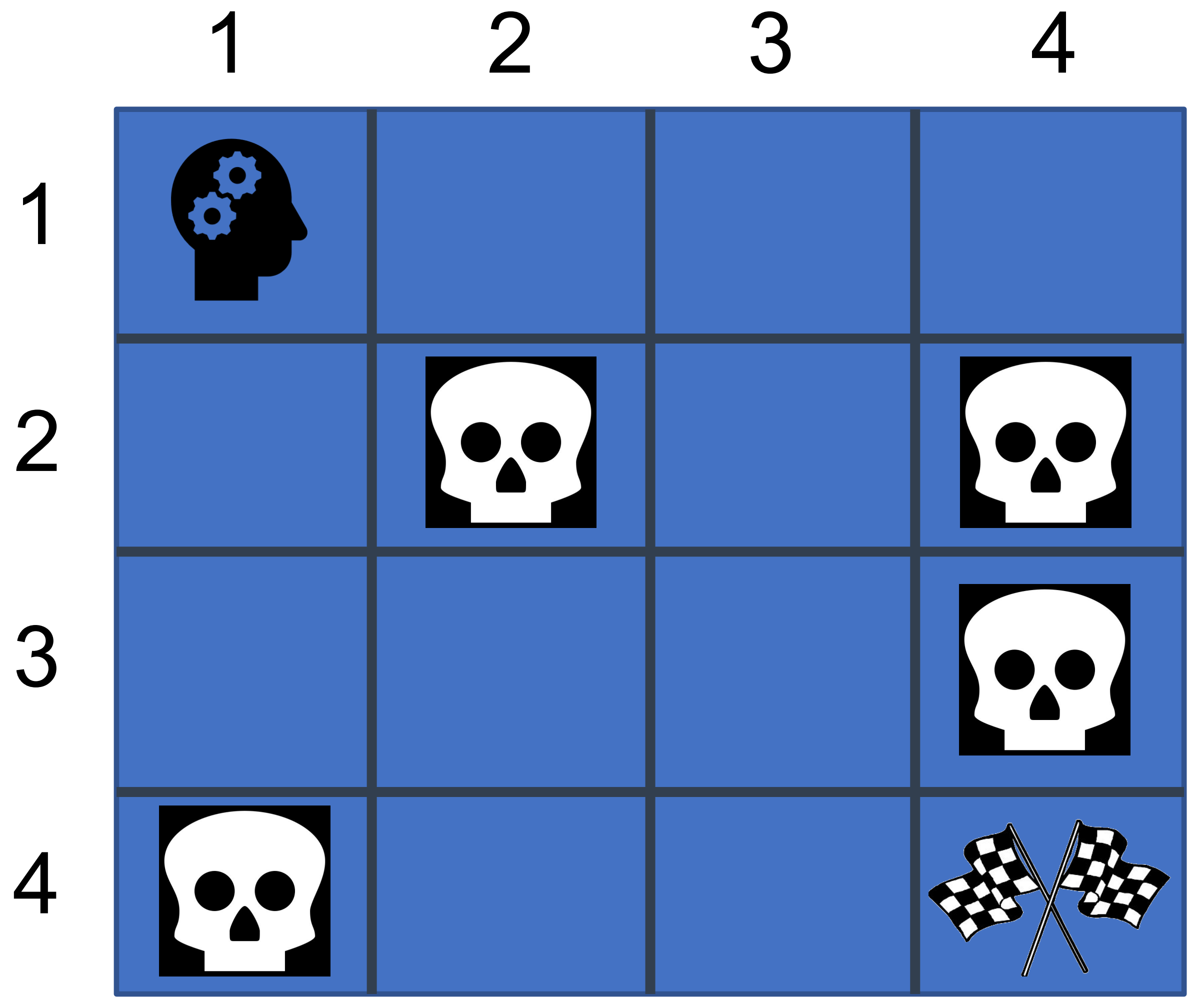
Putting it all together

Can you think of some examples for :



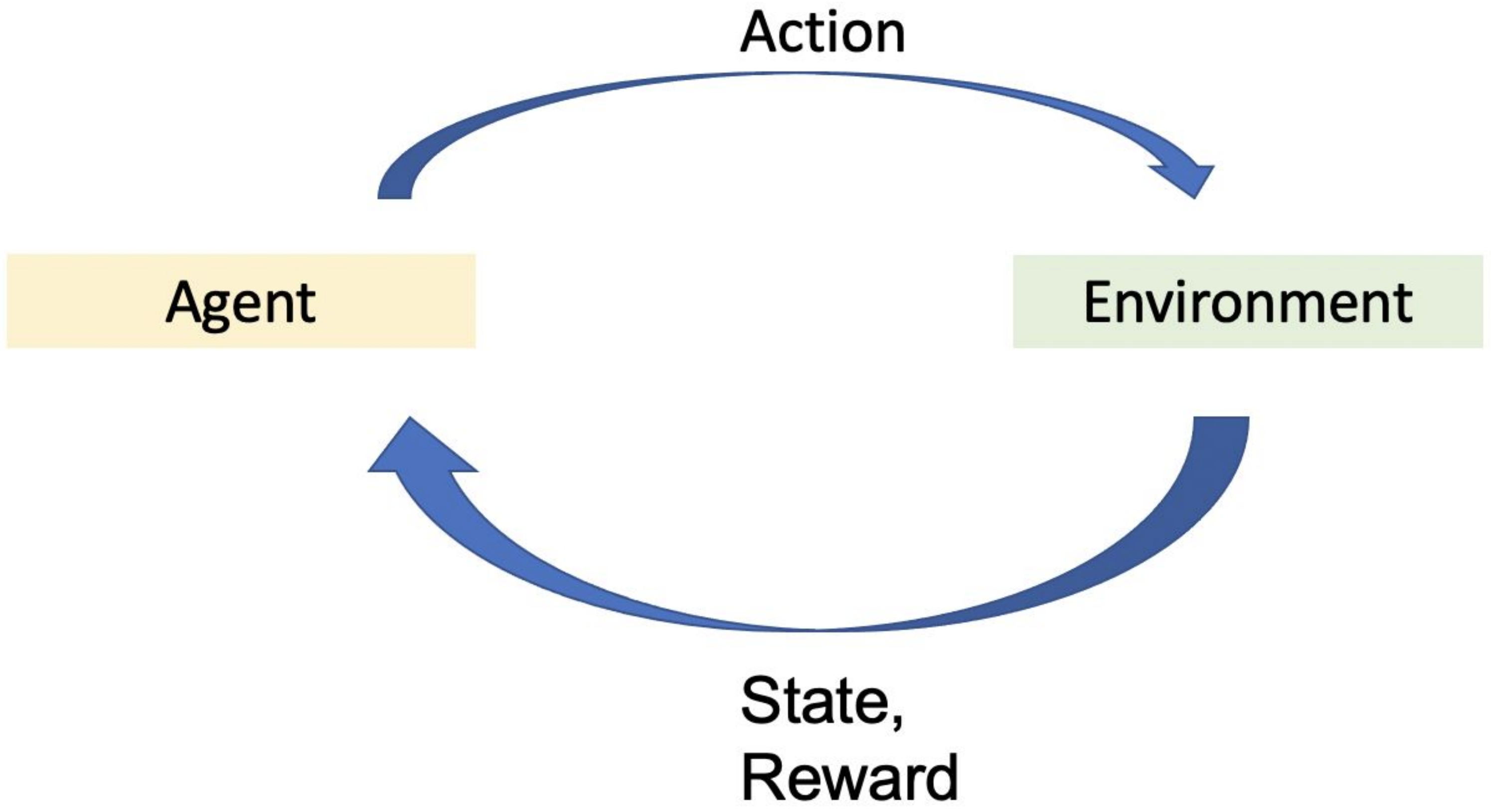
- States:
- Actions:
- Reward:
- Transition functions:
- Policy:

Putting it all together



- States: each square - (1, 1), (1, 2), … , (4, 4)
- Actions: left, right, up, down
- Reward: +1 when you reach the goal, 0 elsewhere
- Transition function: deterministic (for now) Probability =1 for moving a direction given the chosen action, e.g. if agent is in (1, 3):
  - If action is down: move to to (2, 3)
  - If action is left: move to (1, 2)
  - If action is right: move to (1,4)
  - If action is up: stay in (1,3)
- Policy: At (1, 3) move down, at (2,3) move down …

Putting it all together



Taxonomy of RL problems/algorithms

Organizing RL problems/algorithms

	Know T and R	Don't know T and R
Simple/discrete	Value iteration	Q-Learning
Complex/continuous		Deep Q-Networks REINFORCE Actor-Critic

For a more complete taxonomy of RL algorithms, see https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html#citations-below

RL in complex and continuous spaces

Recap

Intro to Reinforcement Learning

- Why RL?
- Sequential Decision Making
- RL Framework

RL components

- Markov Decision Process
- Policies
- Goal of RL