AI in Mathematics
Lecture 10
Reinforcement Learning

Bar-Ilan University
Nebius Academy | Stevens Institute of Technology

May 27, 2025

About This Course

~~1 week: Intro~~

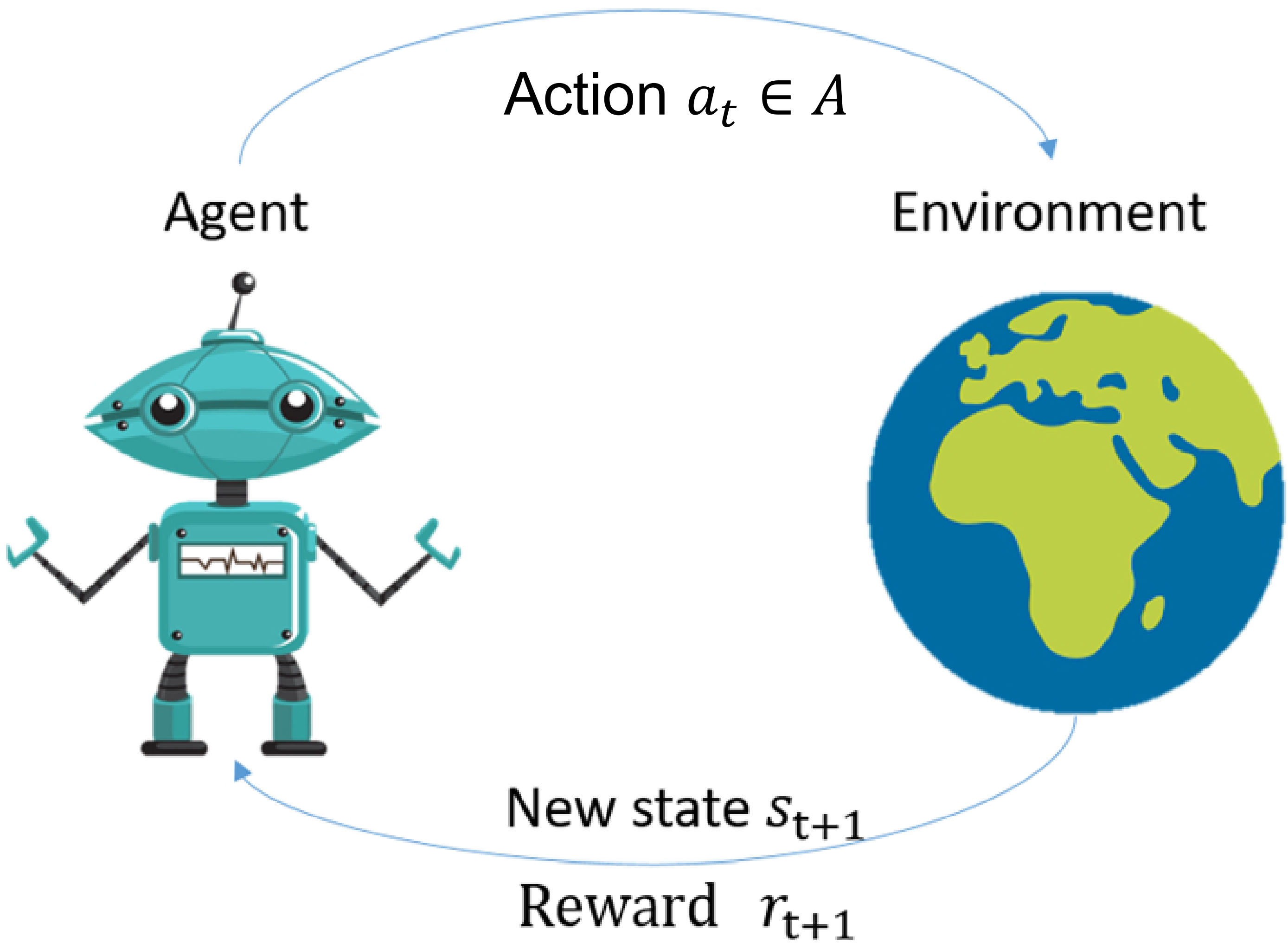
~~2 weeks: Classic ML~~

~~2 weeks: Deep Learning in Mathematics~~

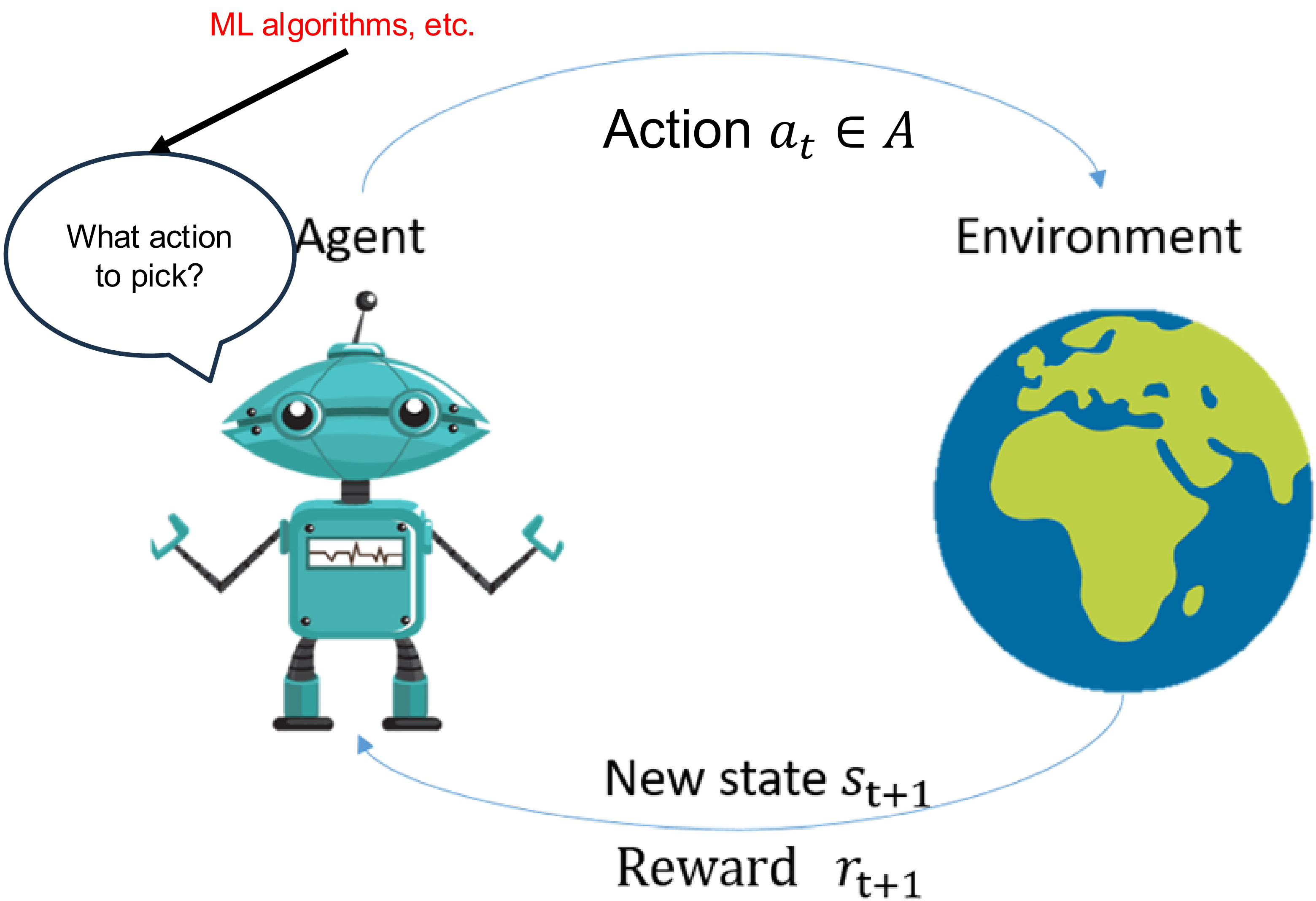
~~4 weeks: Math as an NLP problem (LLMs etc.)~~

4 weeks: Reinforcement Learning (RL) in Math

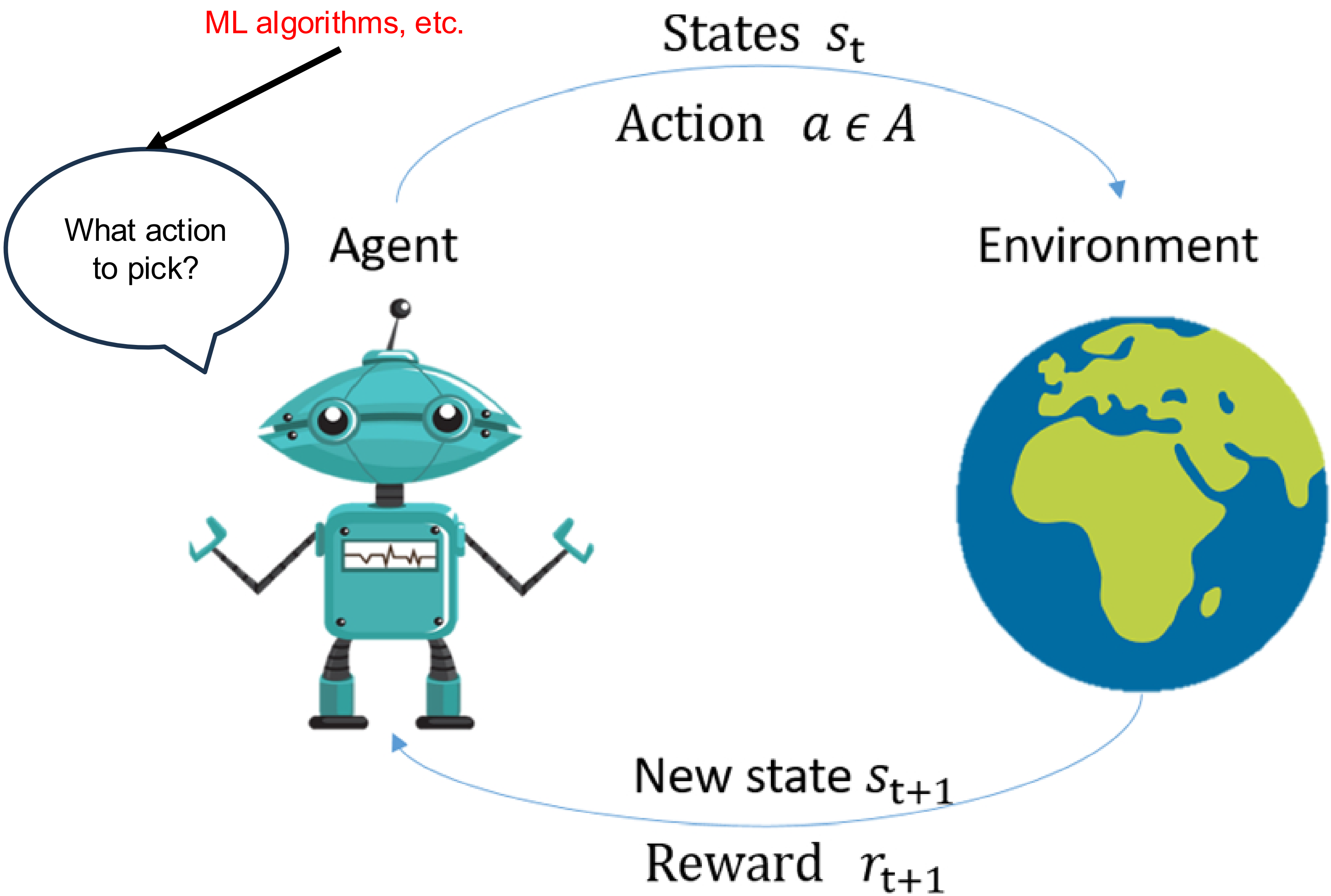
Reinforcement Learning



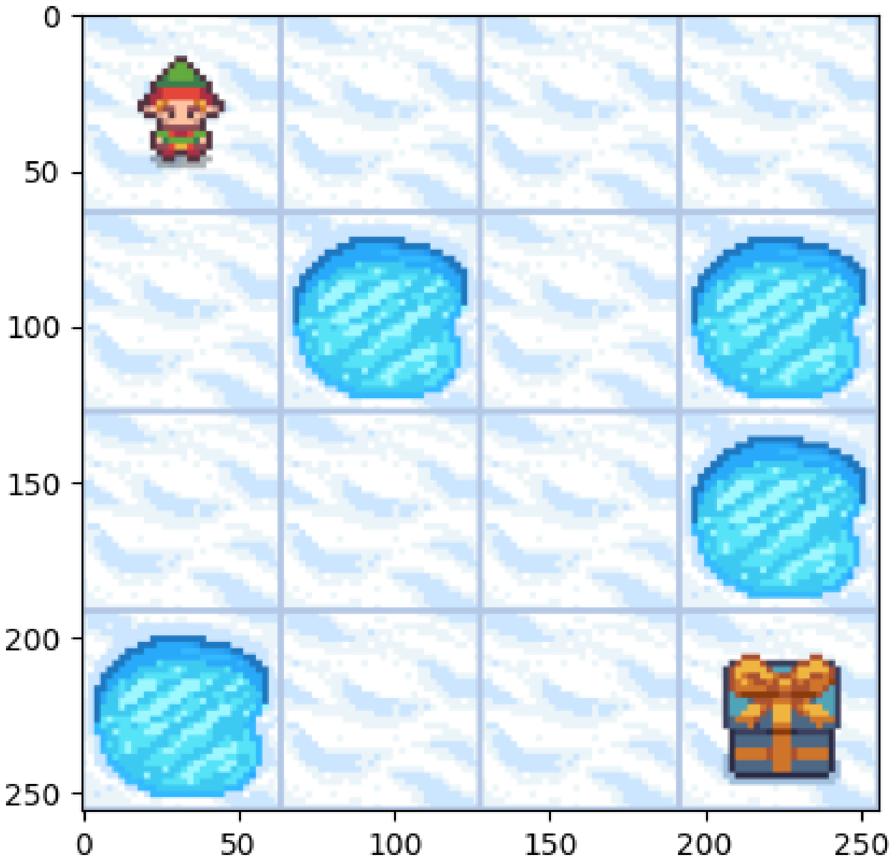
Reinforcement Learning



Reinforcement Learning



Example



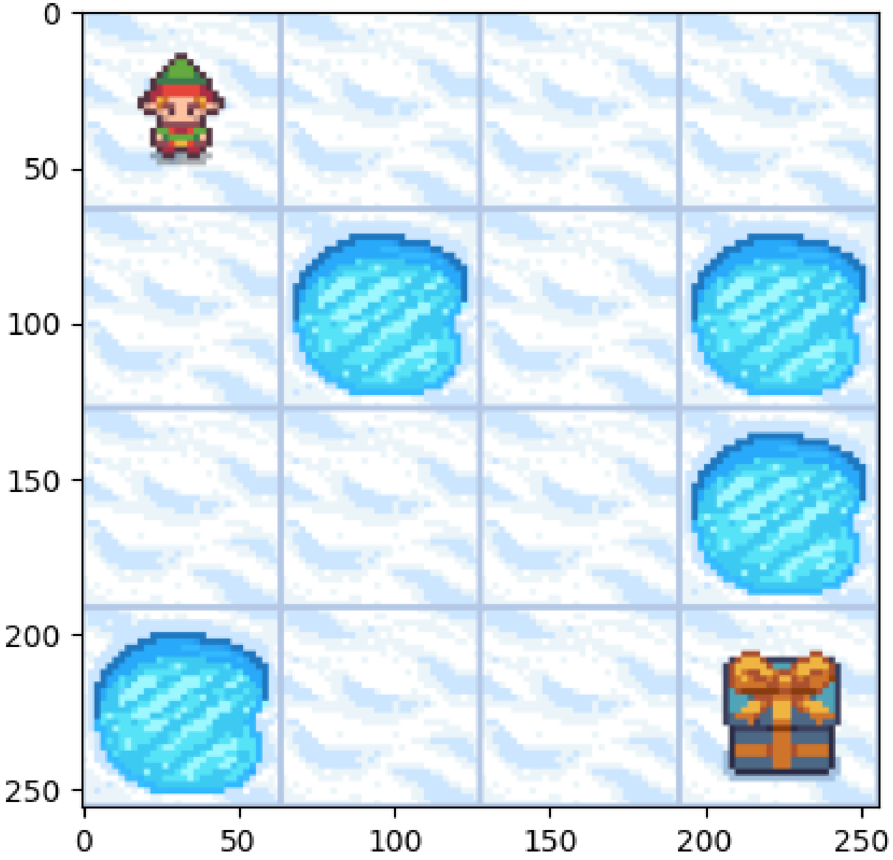
What are the states, actions and rewards in this case?

Actions – possible movements: right, left, up, down.

States – possible positions of an agent.

Rewards – For example: -10 for falling into the lake and $+100$ for finishing in left bottom corner.

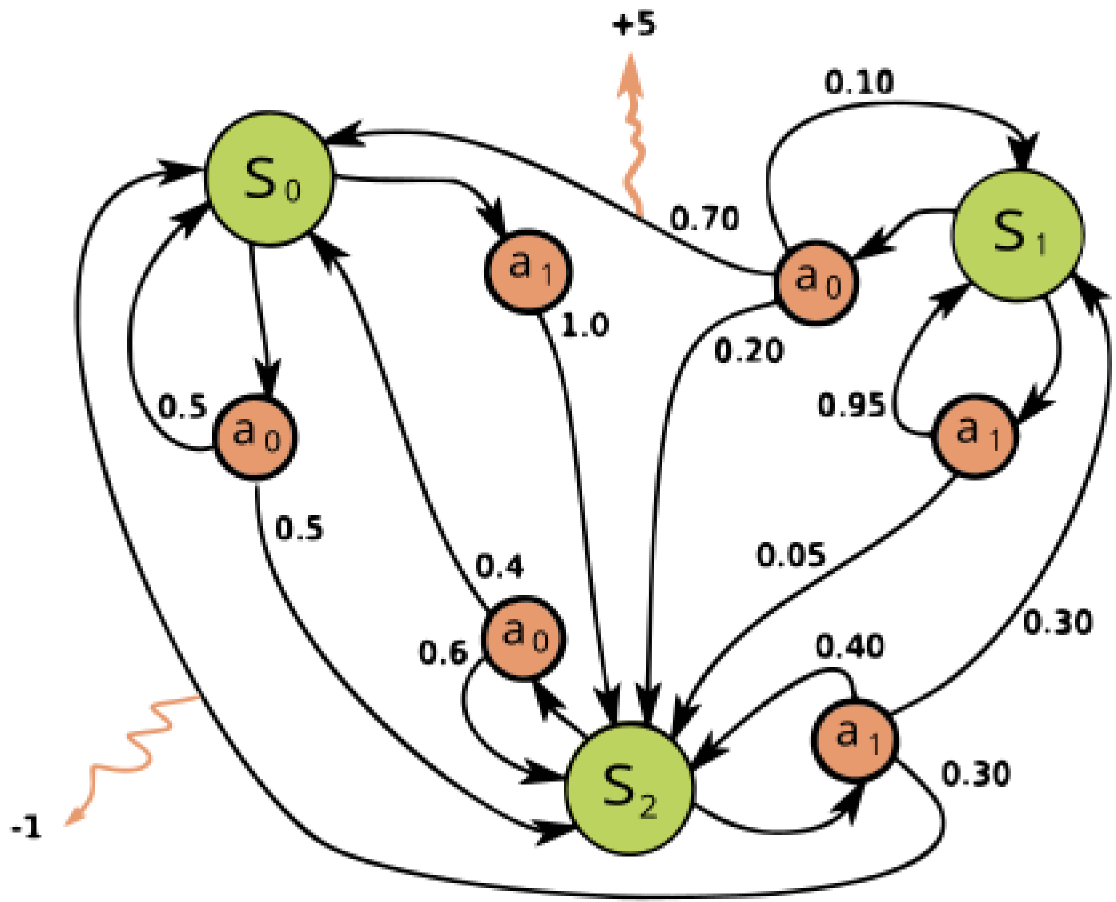
Frozen Lake



❓ Does the agent see the layout?

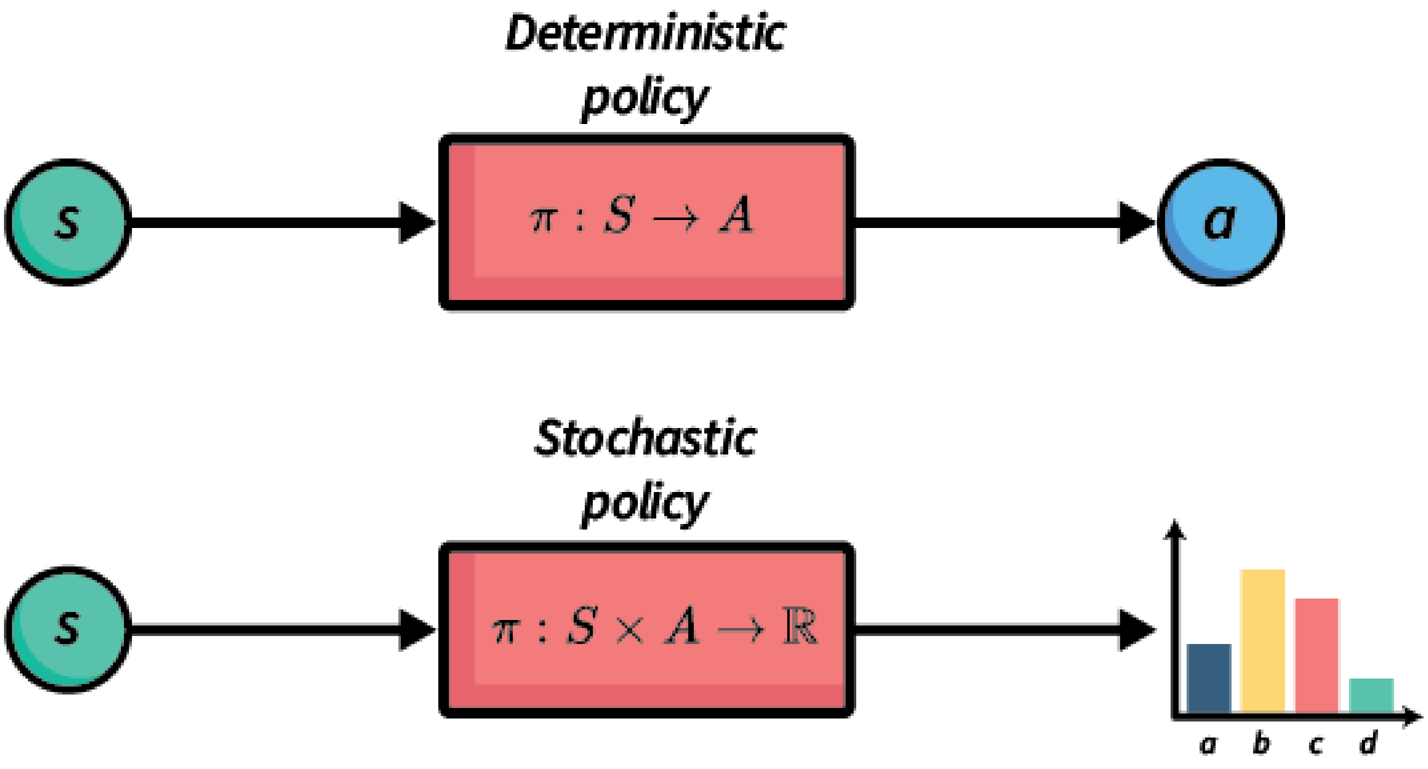
❓ How can it avoid holes if it doesn't see them?

Markov Decision Process



Policy

A **policy** is the agent's strategy for choosing actions.



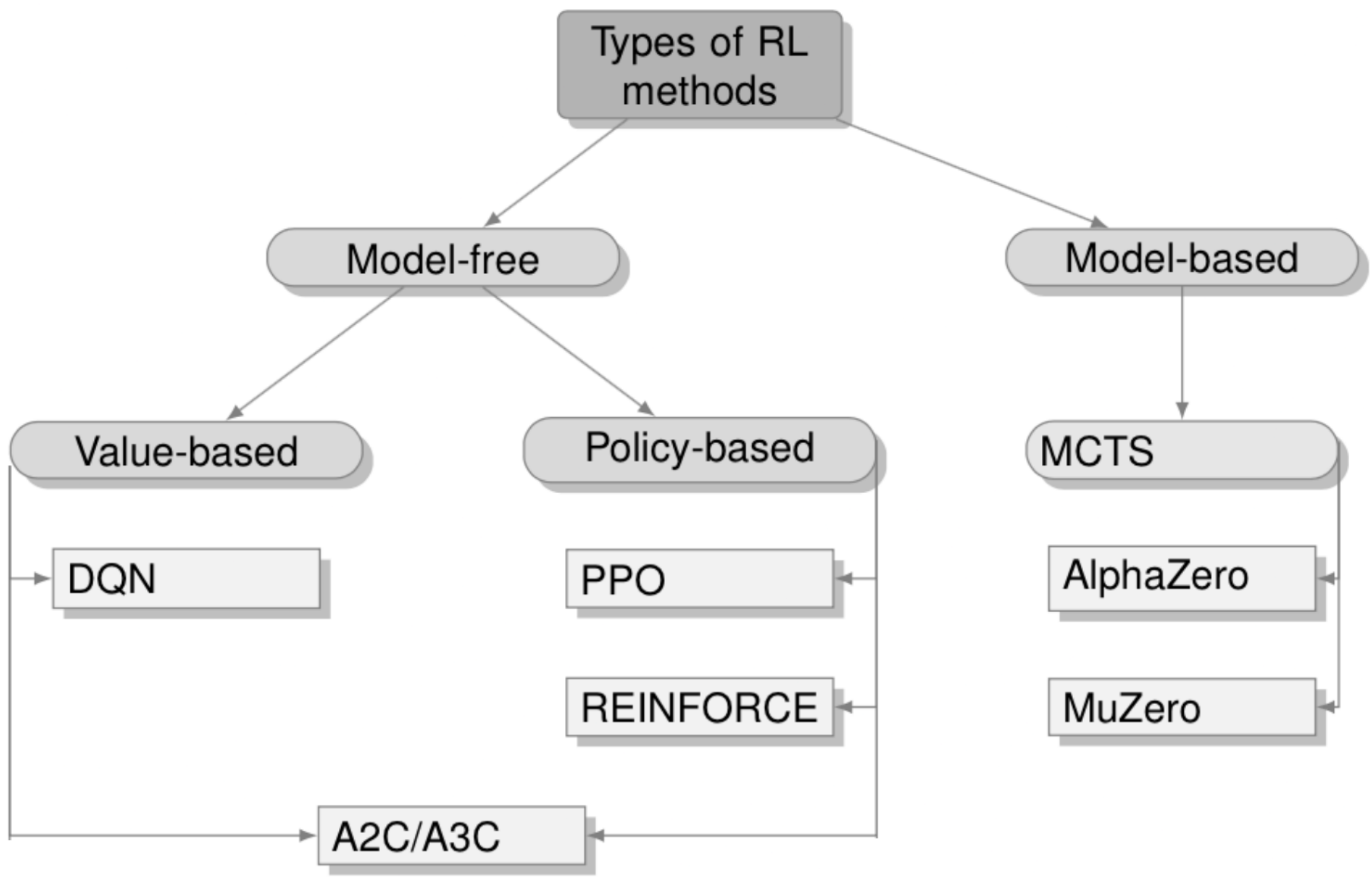
Policy

A **policy** is the agent's strategy for choosing actions.

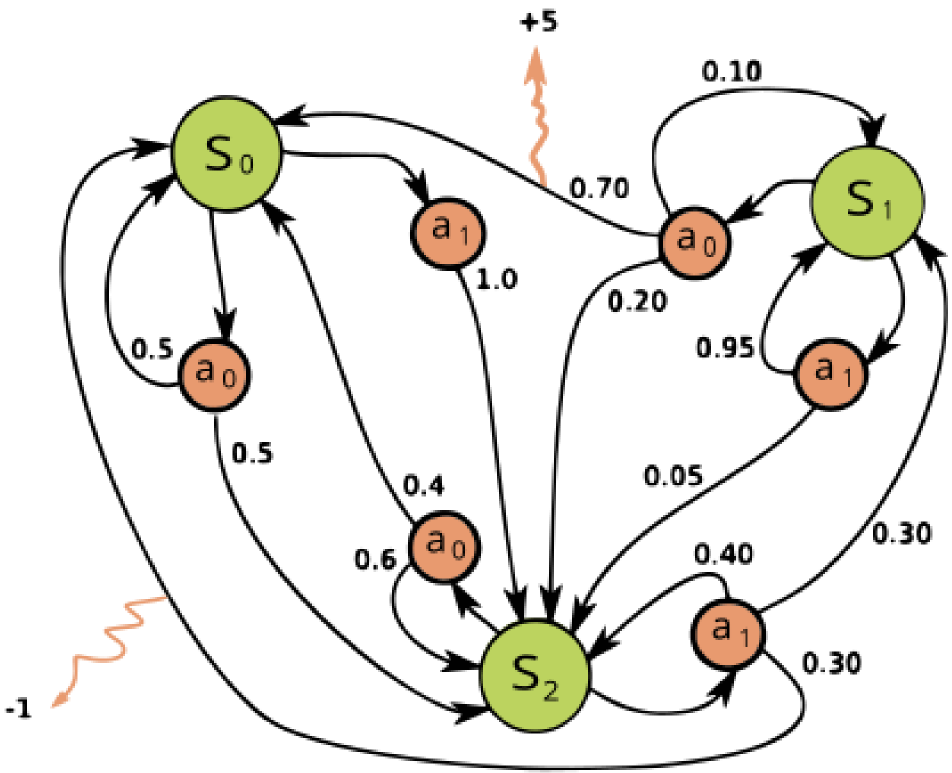
Deterministic policy:
$\pi(s) = a$ – always choose action a in state s.

Stochastic policy:
$\pi(a \mid s) = P(a_t = a \mid s_t = s)$ – probability of taking action a in state s.

RL Algorithms



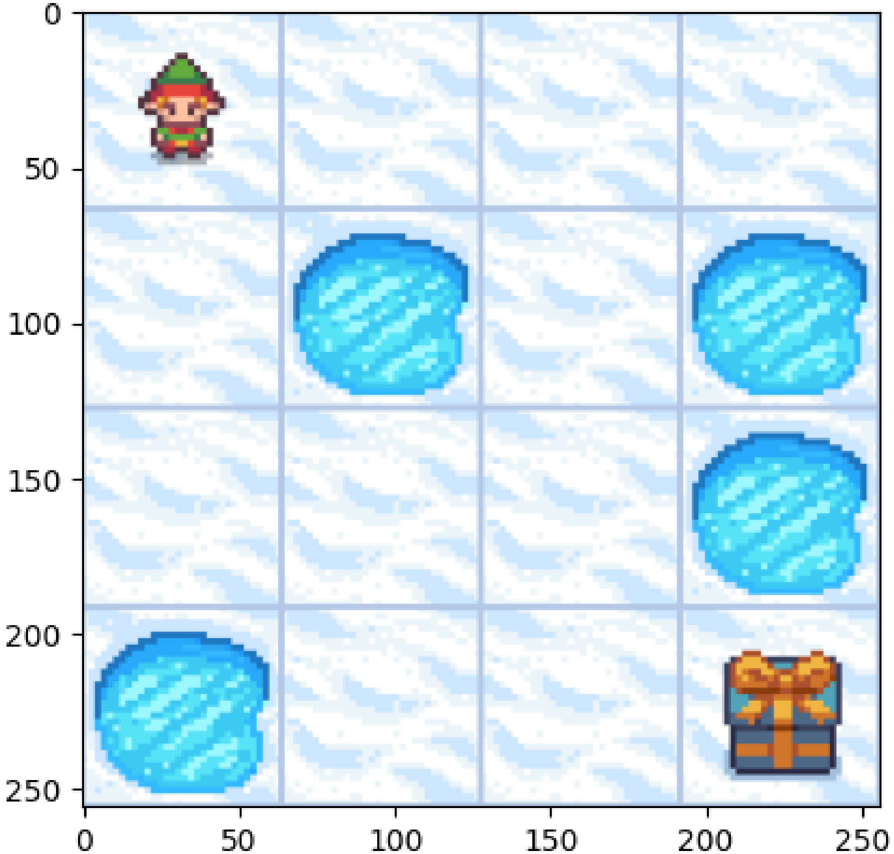
Model based methods



The agent explicitly learns or uses a model of the environment's dynamics, meaning we have intentionally designed it to do so.

If the agent discovers a way to represent probabilities on its own without being guided to build or use a model, it is still considered model-free.

Example

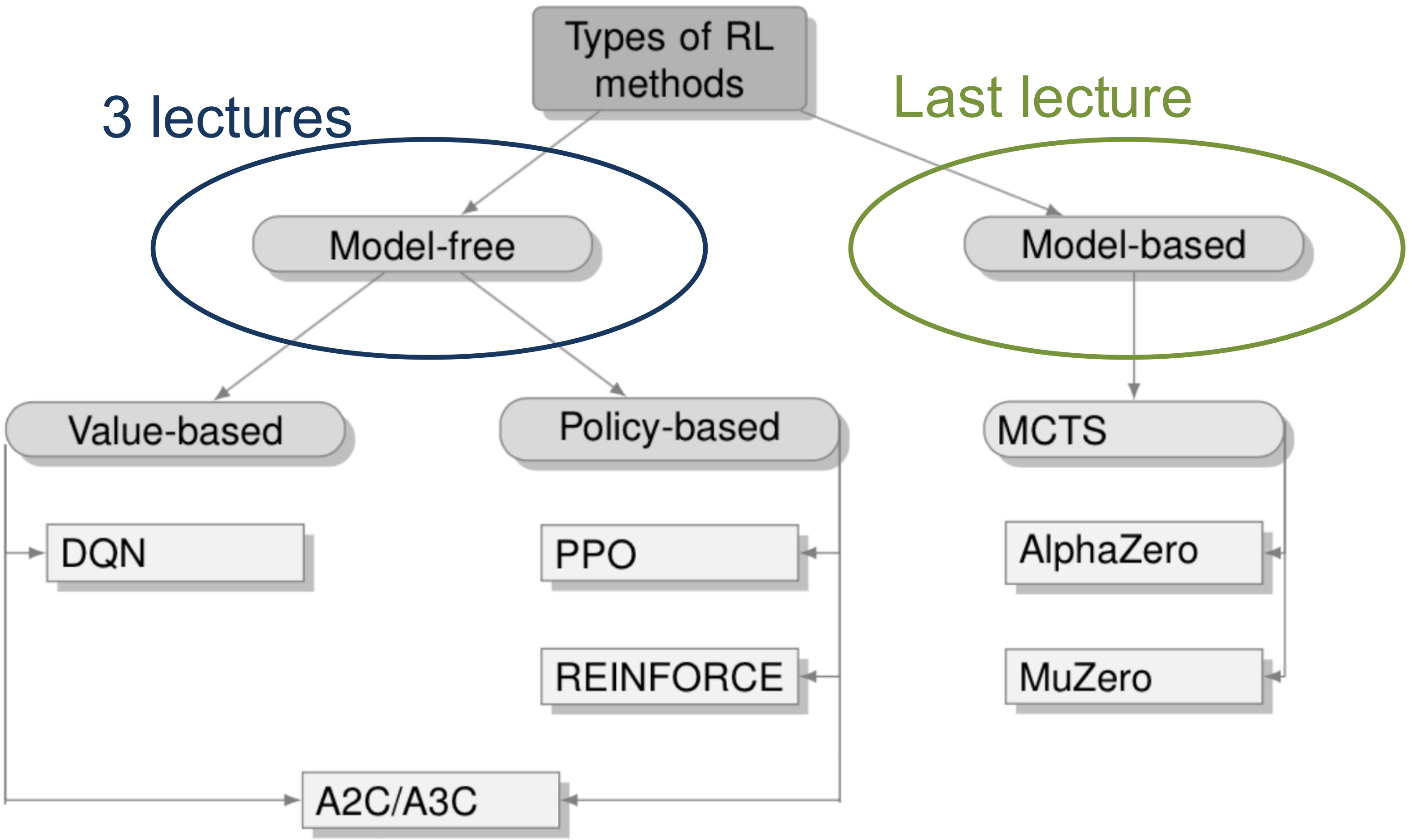


The difference between model free and model based algorithms in the Frozen Lake problem is as follows:

Model free:
The agent only observes its current position (as a state index) and the reward after each move.

Model based:
Either know or learn how likely the agent is to slip when moving in a given direction. They also have access to or reconstruct the entire state space and update a transition model P(s'|s,a).

RL Algorithms



CEM (Cross Entropy Method)

The **Cross Entropy Method (CEM)** is an optimization algorithm that iteratively refines a probability distribution over candidate solutions. At each step, it samples solutions from the current distribution, evaluates their performance, and then updates the distribution to focus more on the best-performing samples.

Over time, this process increases the likelihood of generating high-quality solutions, effectively guiding the search toward optimal or near-optimal outcomes.

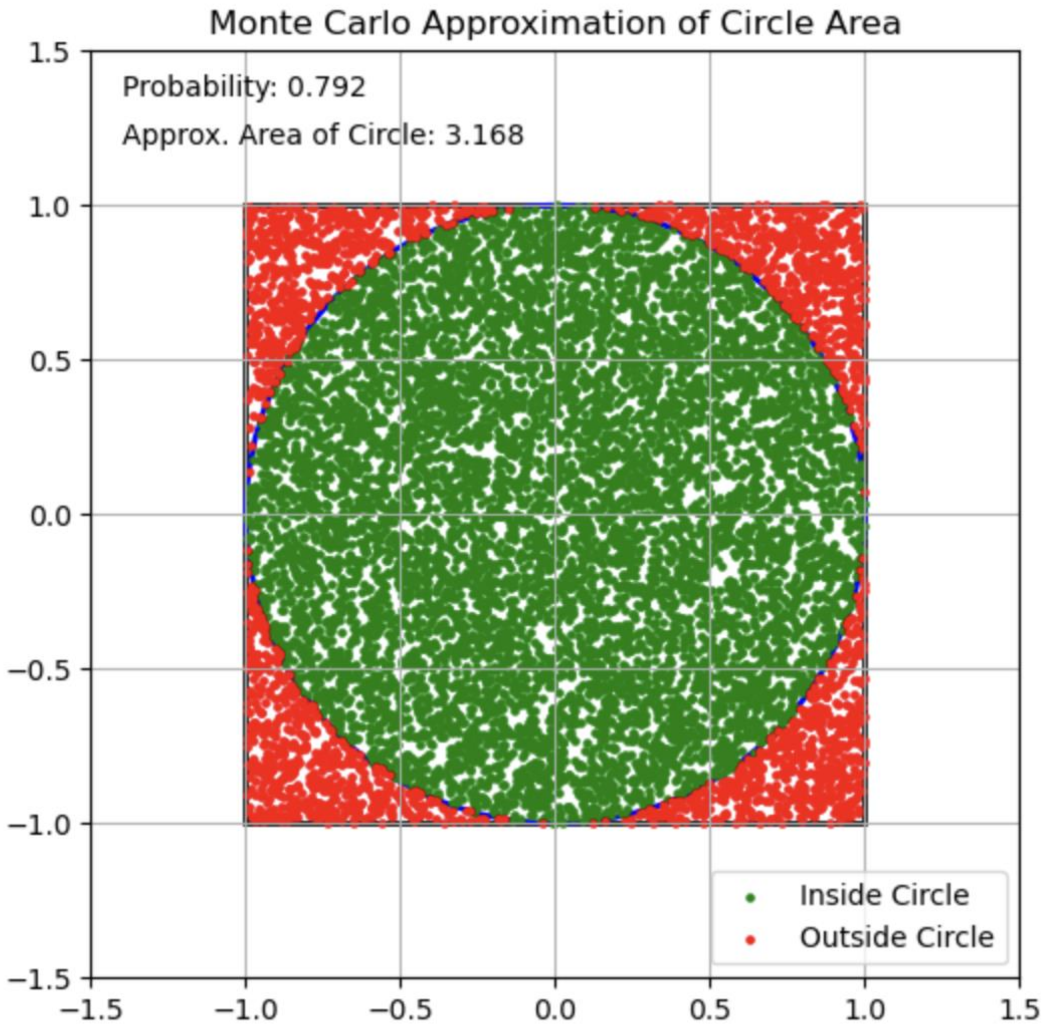
Example

Suppose you want to estimate the area of a circle with radius $r = 1$, but without using the formula $S = \pi\, r$^2. Instead, you can approach the problem as a **stochastic optimization or estimation task**, suitable for methods like the **Monte Carlo simulation**.

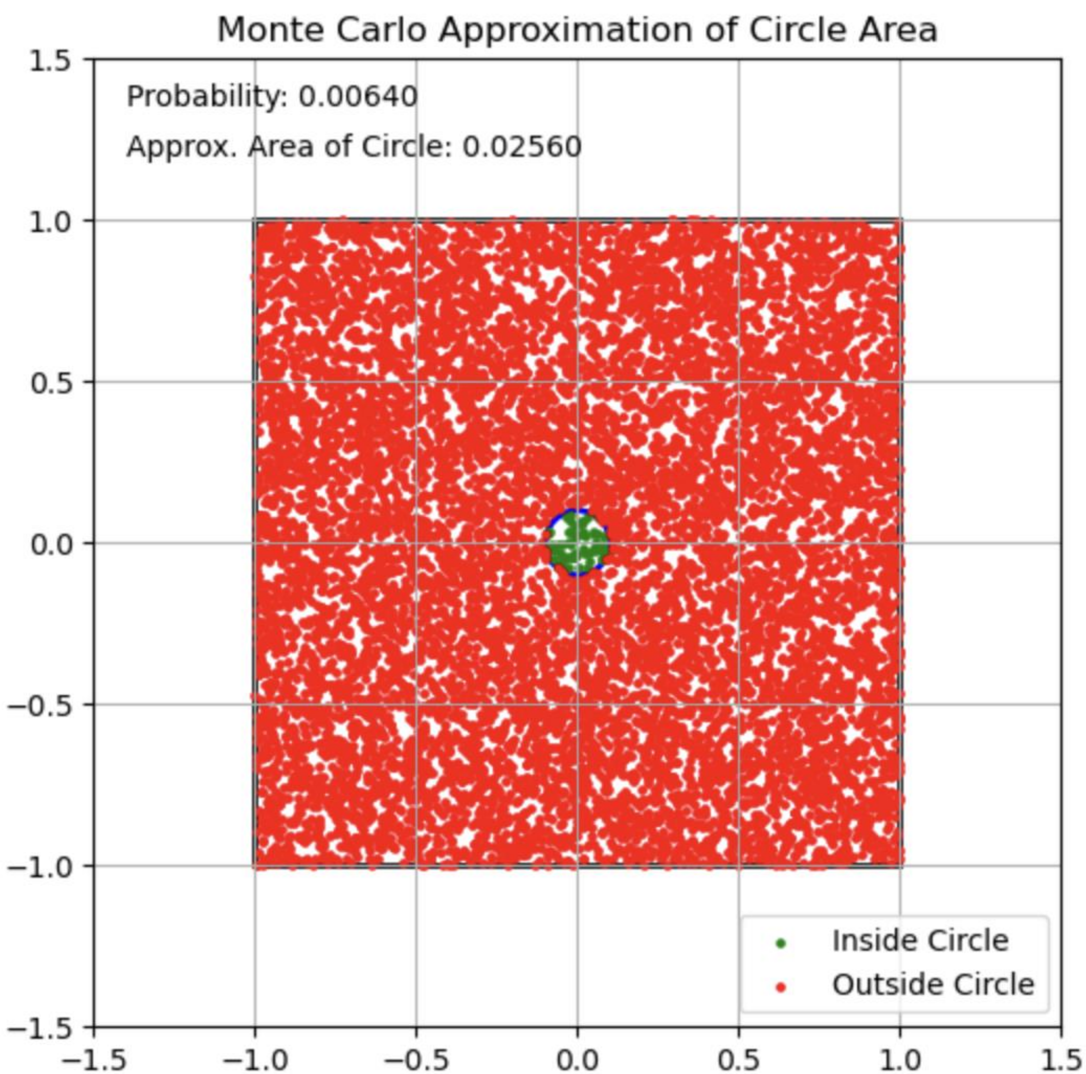
Setup:

- The unit circle (radius 1) is centered at the origin (0,0).
- Enclose the circle within a square of side length 2, spanning from (-1, -1) to (1, 1).
- The area of the square is known: 4.
- Estimate the area of the circle by randomly sampling points in the square and measuring how many fall inside the circle.

Example

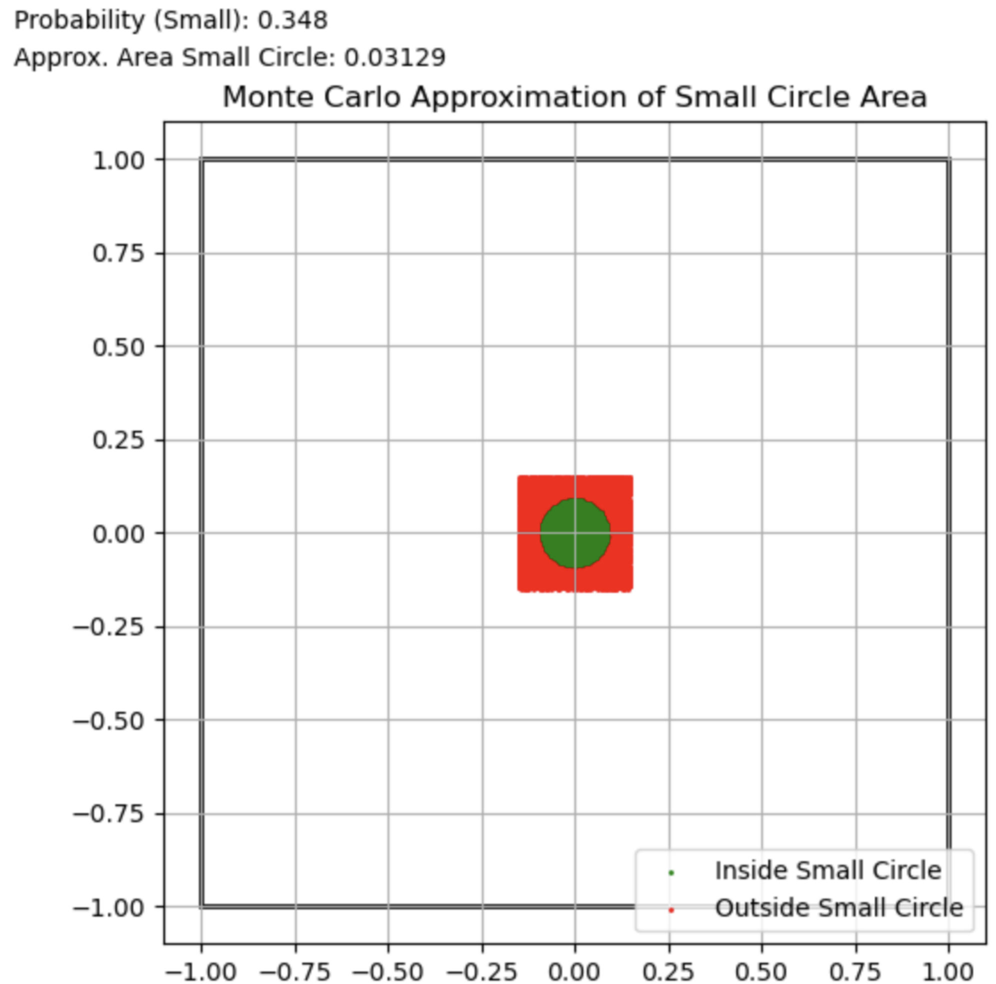

Monte Carlo Approximation of Circle Area
Probability: 0.792
Approx. Area of Circle: 3.168
Inside Circle
Outside Circle
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

Circle with radius 0.1



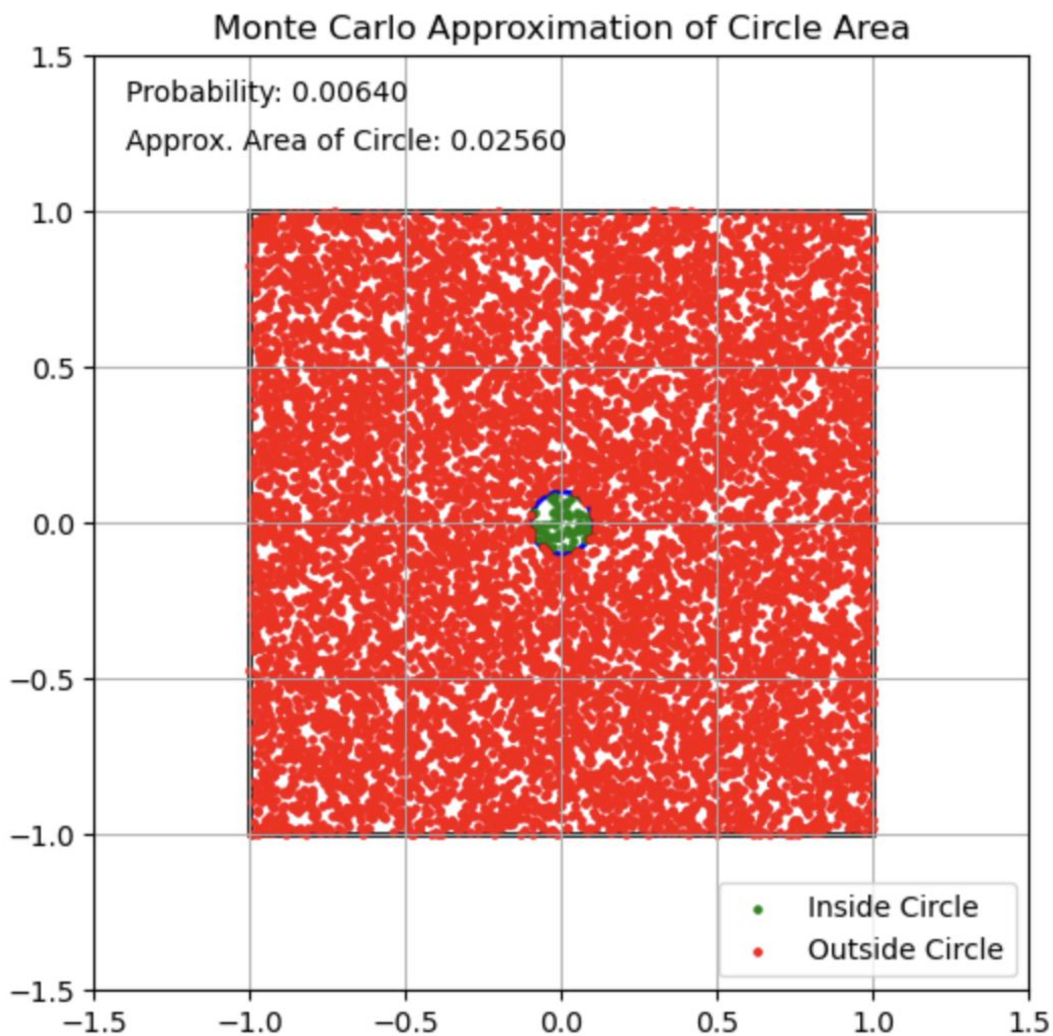
Example

Seems like for smaller circle we need to select randomly from a smaller square uniformly, right?



Idea

Let's update probability distribution after first step, such that samples Inside will have a larger probability.



Formally CEM

Suppose we have a probability distribution $p(x, \theta_t)$.

Iteratively:

- Sample $x_1, x_2 \ldots x_N$ from $p(x, \theta_t)$.
- Evaluate samples using the target function.
- Select the elite set: $S_t \subset \{x_1, \ldots, x_N\}$ consisting of the top $\rho \cdot N$ samples (e. g. $\rho = 0.1$).
- **Update parameters**:

$$\theta_{t+1} = argmax_\theta \sum_{x \in S_t} \log p(x, \theta).$$

How to apply in RL?

We have a policy $\pi(s, \theta_0)$.

Iteratevly:

- Generate N episodes by acting in the environment using policy $\pi(s, \theta_0)$.
- Evaluate each episode by computing the total return (sum of rewards).
- Select the elite set: $S_t \subset \{x_1, \ldots, x_N\}$ consisting of the top $\rho \cdot N$ episodes (e. g. $\rho = 0.1$).
- Update policy parameters using only the elite episodes.

Example

Suppose we are interested in such an unusual graph property:
What is the largest possible ratio between the smallest and largest eigenvalues of a graph's adjacency matrix?

This is a challenging combinatorial question, and we can approach it using **reinforcement learning techniques**.

Example

Let’s define a **score function** for a graph with adjacency matrix Adj:

$$L(Adj) = \frac{\lambda_{min}}{\lambda_{max}}$$

We’ll apply the **Cross-Entropy Method (CEM)** to optimize this score in the following setup:

- We **generate the adjacency matrix element by element**, making predictions based on the current partial state of the matrix.
- Once a full matrix is constructed, we **evaluate it using the score function**.
- The agent is then **trained to imitate the decisions** that led to the highest-scoring matrices.

- We’ll explore the results of this method during the practical session!