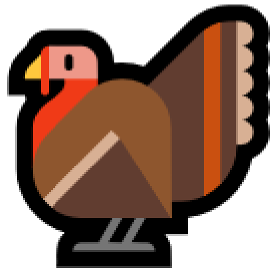
# CS 540 Introduction to Artificial Intelligence
# **Reinforcement Learning I (continued)**

University of Wisconsin-Madison
Fall 2025
November 21, 2025

# Announcements



- Wednesday, November 26
  - **Class is cancelled**
  - Professor Brown’s office hours will be 11:30-12:30

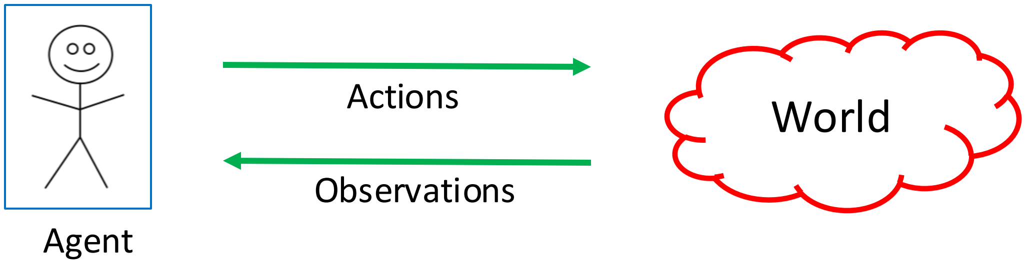
- Homework:
  - HW8 due today at 11:59 PM
  - HW9 released today, due **Tuesday** Dec 2 at 11:59 pm
  - HW10 released Dec 2, due Tuesday Dec 9 at 11:59 pm

# Outline

- Introduction to reinforcement learning
  - Basic concepts, mathematical formulation, MDPs, policies.
- Learning policies
  - Q-learning, action-values, exploration vs exploitation.

# Back to Our General Model

We have an **agent** **interacting** with the **world**




- Agent receives a reward based on state of the world
  - **Goal**: maximize reward / utility **($$$)**
  - Note: **data** consists of actions & observations
    - Compare to unsupervised learning and supervised learning

# Markov Decision Process (MDP)

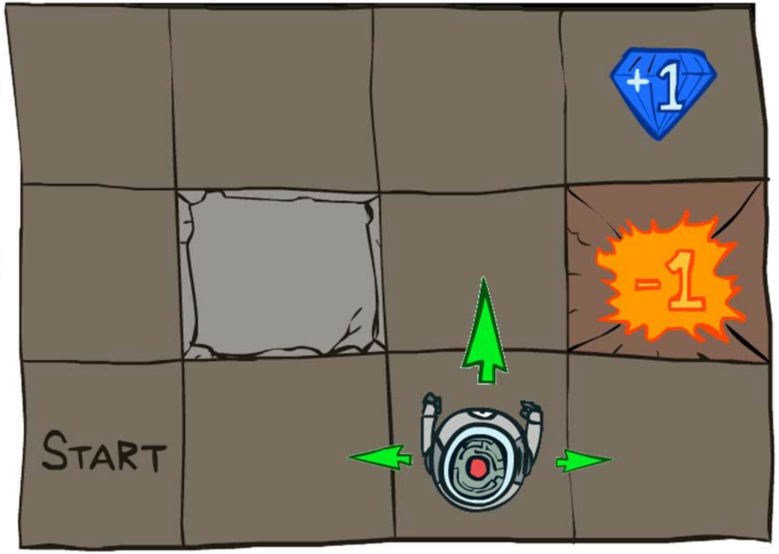
The formal mathematical model:

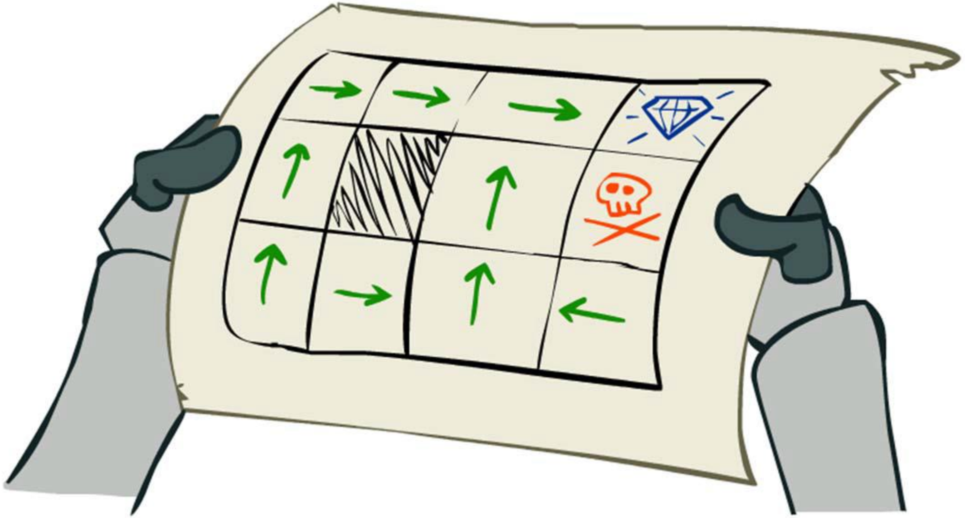
- **State set** $S$. Initial state $s_0$. **Action set** $A$
- **Reward function:** $r(s_t)$
- **State transition model**: $P(s_{t+1}|s_t, a_t)$
  - Markov assumption: transition probability only depends on $s_t$ and $a_t$, and not earlier history (previous actions or states)
- More generally: $r(s_t, a_t)$, potentially random
- **Policy**: $\pi(s) : S \rightarrow A$ action to take at a particular state

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \ldots$$

# Example of MDP: Grid World

Robot on a grid; goal: find the best policy



Source: P. Abbeel and D. Klein

# Markov Decision Process (MDP)

The formal mathematical model:

- **State set** S. Initial state $s_0$. **Action set** A
- **Reward function:** $r(s_t)$
- **State transition model**: $P(s_{t+1}|s_t, a_t)$
- **Policy**: $\pi(s) : S \rightarrow A$ action to take at a particular state

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \ldots$$

# Reinforcement Learning Challenges

**Credit-assignment:**

- May take many actions before reward is received. Which ones were most important?
- <u>Example</u>: You study 15 minutes a day all semester. The morning of the final exam, you eat a bowl of yogurt. You receive an A on the final. Was it the studying or the yogurt that led to the A?

**Exploration vs. Exploitation:**

- Transition probabilities and reward may be unknown to the learner.
- Should you keep trying actions that led to reward in the past or try new actions that might lead to even more reward?

# Defining the Optimal Policy

For policy π, **expected utility** over all possible state sequences from $s_0$ produced by following that policy:

Utility of sequence

$$V^{\pi}(s_0) = \sum_{\substack{\text{sequences} \\ \text{starting from } s_0}} P(\text{sequence})U(\text{sequence})$$

Probability of sequence
when following $\pi$

Called the **value function** (for π, $s_0$)

# Discounting Rewards

Utility can add up the rewards over a sequence of states, but how should we treat the future?

- Solution: **discount** future rewards.

$$U(s_0, s_1 \ldots) = r(s_0) + \gamma r(s_1) + \gamma^2 r(s_2) + \ldots = \sum_{t \geq 0} \gamma^t r(s_t)$$

- Discount factor $\gamma$ between 0 and 1
  - Set according to how important **present** is versus **future**
  - Note: has to be less than 1 for convergence

# From Value to Policy

Now that $V^{\pi}(s_0)$ is defined, what $a$ should we take?

- First, let $\pi^*$ be the **optimal** policy for $V^{\pi}(s_0)$, and $V^*(s_0)$ its expected utility.
- What's the expected utility following an action?
  - Specifically, action $a$ in state $s$?

$$\sum_{s'} P(s'|s,a)V^*(s')$$

All the states we could go to

Transition probability

Expected rewards

# Bellman Equation

Let’s walk over one step for the value function:

$$V^*(s) = r(s) + \underbrace{\gamma \max_a \sum_{s'} P(s'|s,a) V^*(s')}_{\text{Discounted expected future } \textbf{rewards}}$$

Current state reward

# Value Iteration

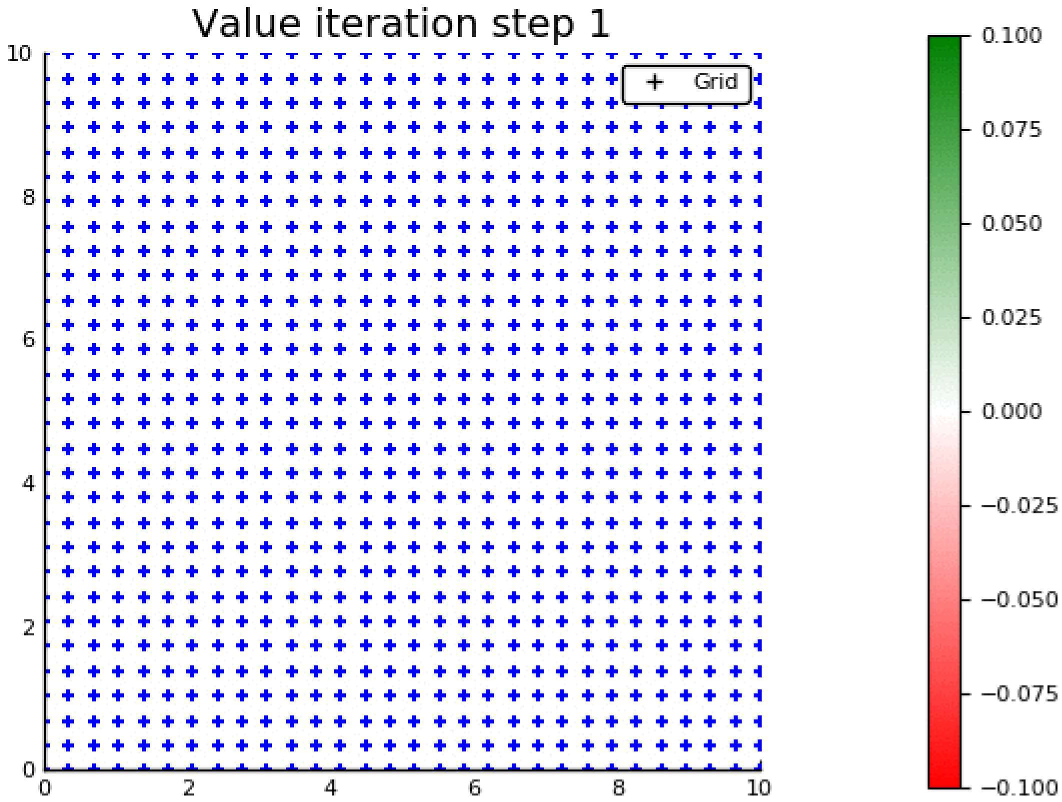
**Q**: how do we find $V^*(s)$?

- Why do we want it? Can use it to get the best policy
- Know: reward $r(s)$, transition probability $P(s'|s,a)$
  - Knowing r and P is the “planning” problem. In reality r and P must be estimated from interactions : “reinforcement learning”
- Also know $V^*(s)$ satisfies Bellman equation (recursion above)

**A**: Use the property. Start with $V_0(s)$=0. Then, update

$$V_{i+1}(s) = r(s) + \gamma \max_a \sum_{s'} P(s'|s,a) V_i(s')$$

# Value Iteration: Demo




Source: POMDPBGallery Julia Package

# Q-Learning

- Our **next** reinforcement learning algorithm.
- Does not require knowing *r* or *P*. Learn from data of the form:$\{(s_t, a_t, r_t, s_{t+1})\}$.
- Learns an action-value function *Q**(*s*,*a*) that tells us the expected value of taking *a* in state *s*.
  - Note: $V^*(s) = \max_a Q^*(s, a)$.
- Optimal policy is formed as $\pi^*(s) = \mathrm{arg}\max_a Q^*(s, a)$

# The Q*(s,a) function

- Starting from state s, perform (perhaps suboptimal) action *a*. THEN follow the optimal policy

$$Q^*(s,a) = r(s) + \gamma \sum_{s'} P(s'|s,a) V^*(s')$$

- Equivalent to

$$Q^*(s,a) = r(s) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q^*(s',a')$$

# Q-Learning Iteration

How do we get Q(*s*,*a*)?

- Iterative procedure

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r(s_t) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

Learning rate

**Idea**: combine old value and new estimate of future value.

Note: We are using a policy to take actions; based on the estimated Q!

# Q-Learning

Estimate Q*(*s*,*a*) from data $\{(s_t, a_t, r_t, s_{t+1})\}$:

Learning rate

# Q-Learning

Estimate Q*(*s*,*a*) from data $\{(s_t, a_t, r_t, s_{t+1})\}$:

1. Initialize Q(.,.) arbitrarily (eg all zeros)
   1. Except terminal states $Q(s_{terminal},.)=0$
2. Iterate over data until Q(.,.) converges:

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max_b Q(s_{t+1}, b))$$

**Idea**: update is an empirical version of our Q table recursion:

$$Q^*(s,a) = r(s) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q^*(s',a')$$

# Exploration Vs. Exploitation

General question!

- **Exploration:** take an action with unknown consequences
  - Pros:
    - Get a more accurate model of the environment
    - Discover higher-reward states than the ones found so far
  - Cons:
    - When exploring, not maximizing your utility
    - Something bad might happen
- **Exploitation:** go with the best strategy found so far
  - Pros:
    - Maximize reward as reflected in the current utility estimates
    - Avoid bad stuff
  - Cons:
    - Might prevent you from discovering the true optimal strategy

# Q-Learning: ε-Greedy Behavior Policy

Getting data with both **exploration and exploitation**

- With probability ε, take a random action; else the action with the highest (current) Q(*s*,*a*) value.

$$a = \begin{cases} \operatorname{argmax}_{a \in A} Q(s, a) & \text{uniform}(0,1) > \epsilon \\ \text{random } a \in A & \text{otherwise} \end{cases}$$

# Q-learning Algorithm

Input: step size $\alpha$, exploration probability $\epsilon$

1. set Q(*s*,*a*) = 0 for all *s*, *a*.
2. For each episode:
3. Get initial state *s*.
4. While (*s* not a terminal state):
5. Perform a = $\epsilon$-greedy(Q, *s*), receive *r*, *s*'
6. $Q(s,a) = (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a'))$
7. $s \leftarrow s'$
8. End While
9. End For

Explore: take action to see what happens.

Update action-value based on result.

# Summary

- Reinforcement learning setup
- Mathematical formulation: MDP
- Bellman Equation
- Value Iteration Algorithm
- The Q-learning Algorithm