



CS 760: Machine Learning **Reinforcement Learning**

University of Wisconsin-Madison

Outline

- **Intro to Reinforcement Learning**

- Basic concepts, mathematical formulation, MDPs, policies

- **Valuing and Obtaining Policies**

- Value functions, Bellman equation, value iteration, policy iteration

Outline

- **Intro to Reinforcement Learning**

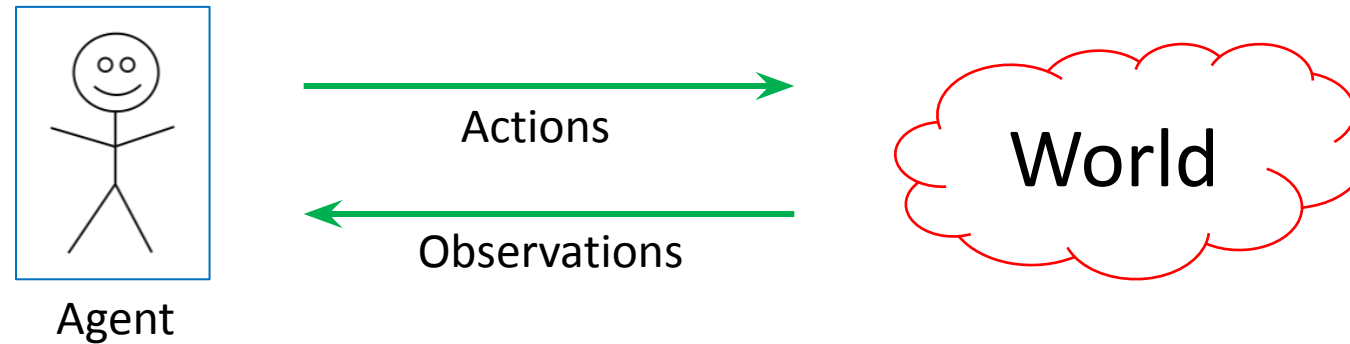
- Basic concepts, mathematical formulation, MDPs, policies

- **Valuing and Obtaining Policies**

- Value functions, Bellman equation, value iteration, policy iteration

A General Model

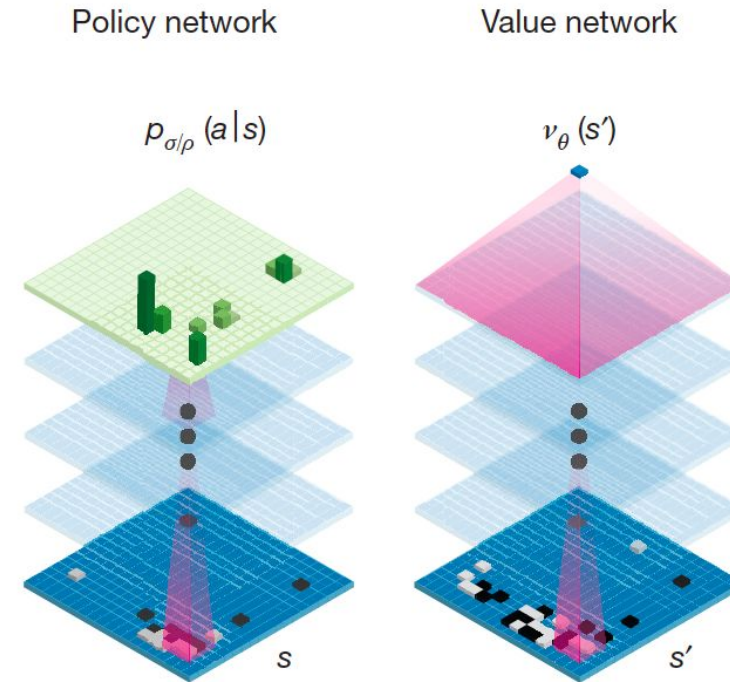
We have an **agent interacting** with the **world**



- 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

Examples: Gameplay Agents

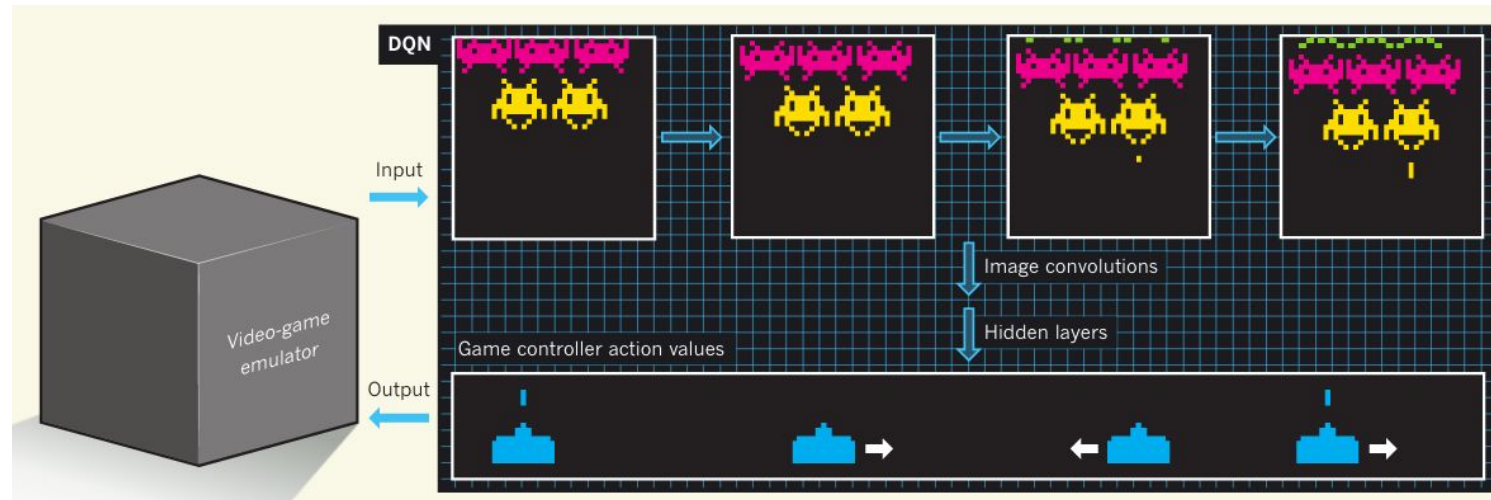
AlphaZero:



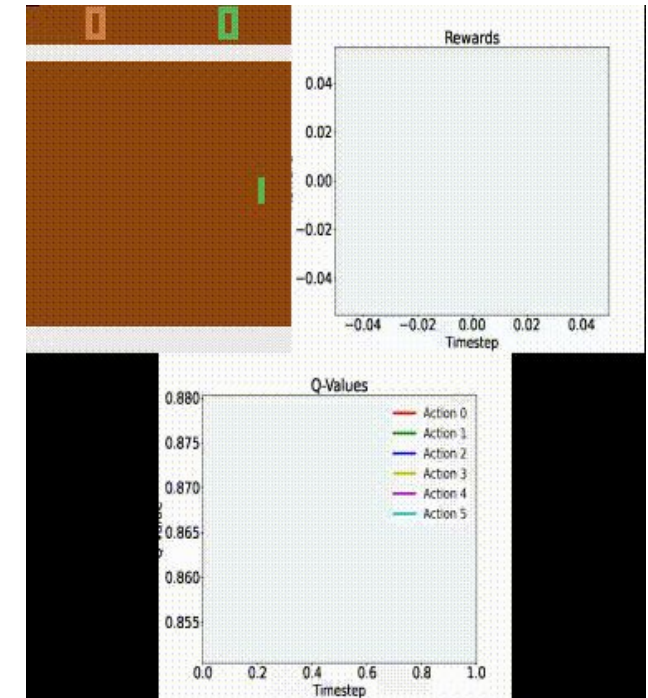
<https://deepmind.com/research/alphago/>

Examples: Video Game Agents

Pong, Atari



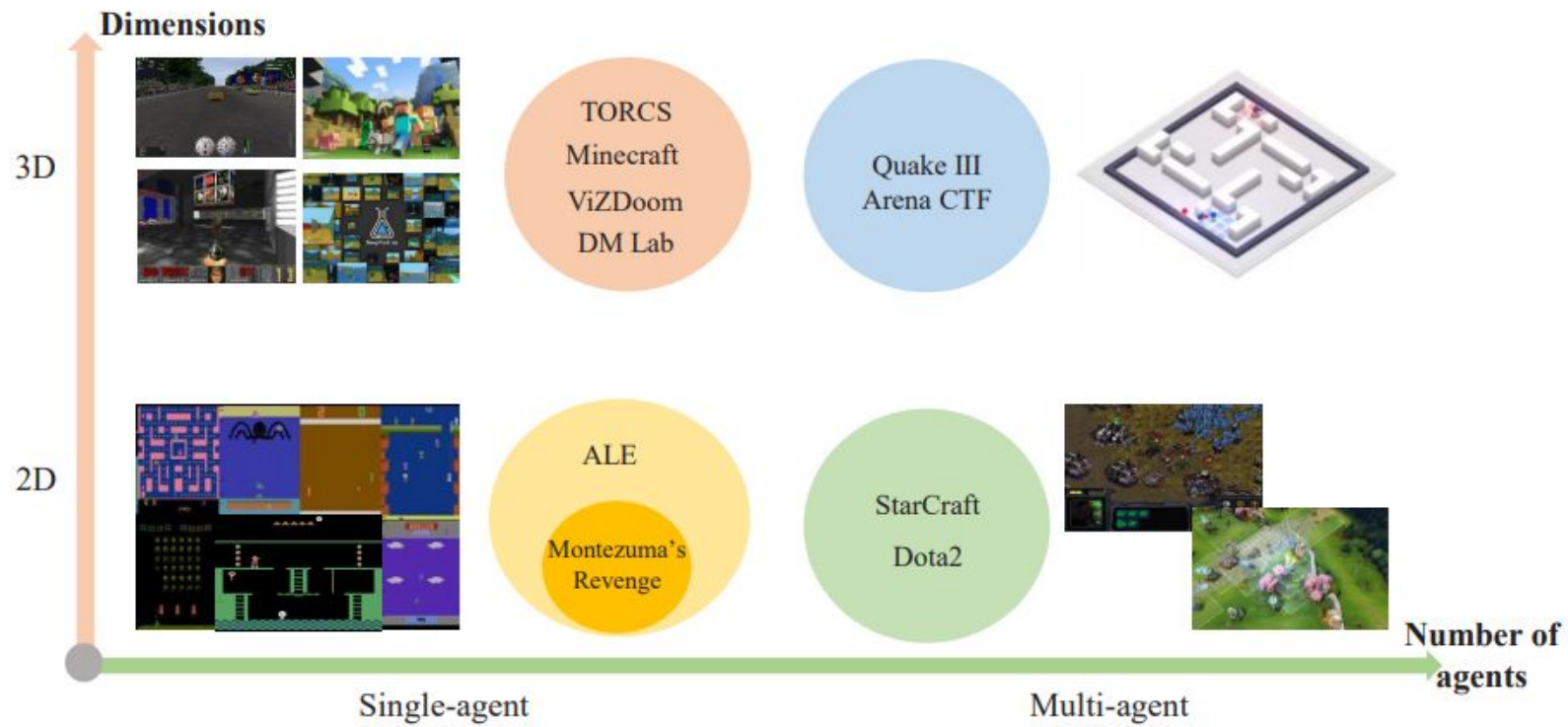
Mnih et al, "Human-level control through deep reinforcement learning"



A. Nielsen

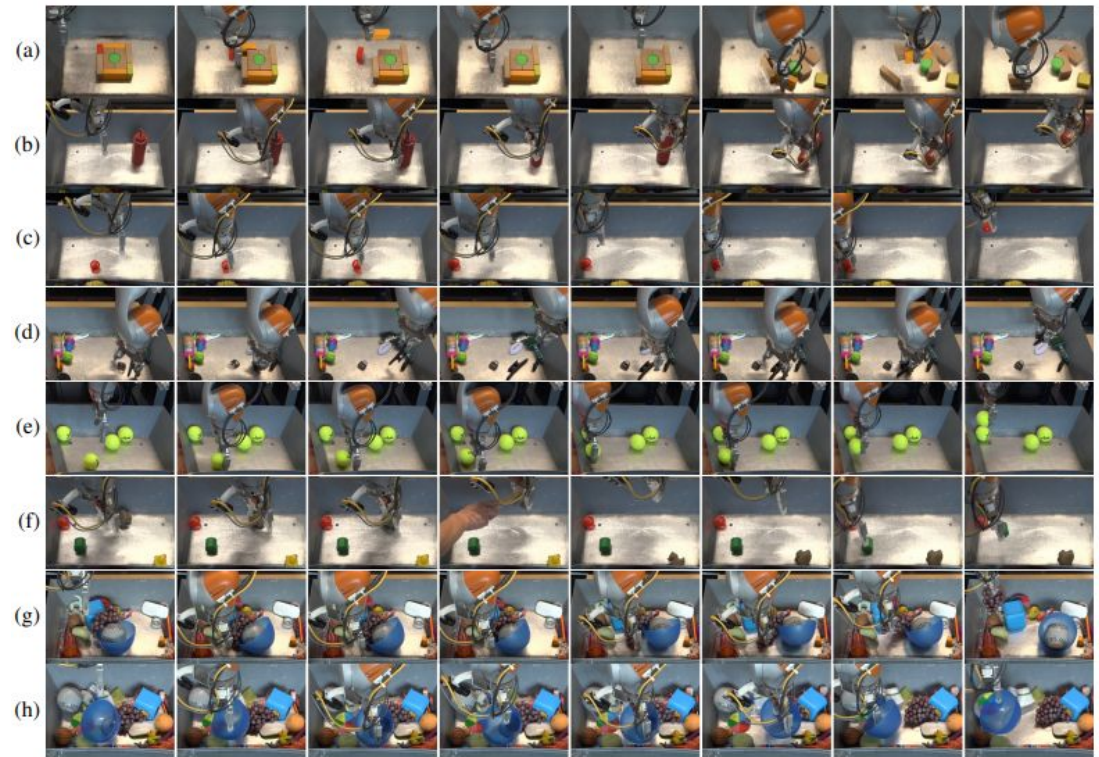
Examples: Video Game Agents

Minecraft, Quake, StarCraft, and more!



Examples: Robotics

Training robots to perform tasks (e.g. grasp!)

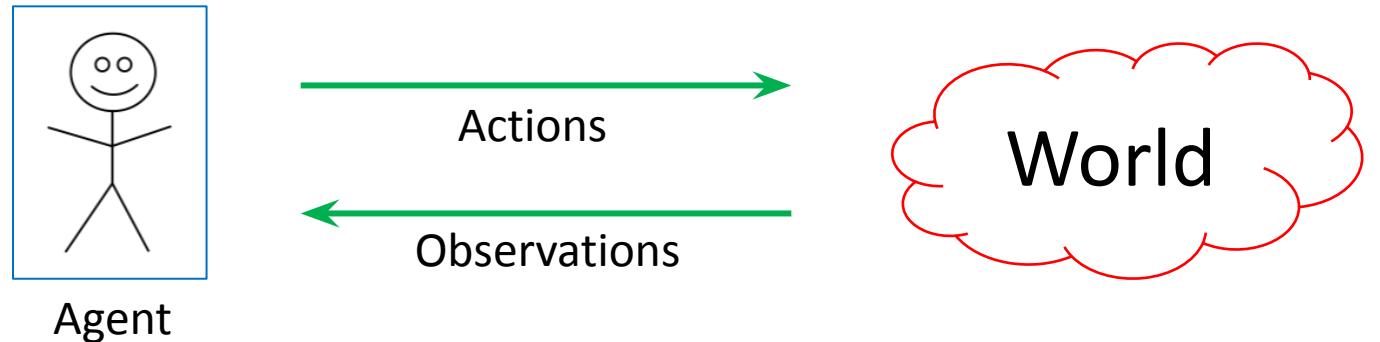


Ibarz et al, " How to Train Your Robot with Deep Reinforcement Learning – Lessons We've Learned "

Building The Theoretical Model

Basic setup:

- Set of states S
- Set of actions A
- Information: at time t , observe state $s_t \in S$. Get reward r_t
- Agent makes choice $a_t \in A$. State changes to s_{t+1} , continue



Goal: find a map from **states to actions** maximize rewards.

↑
a “policy”

Markov Decision Process (MDP)

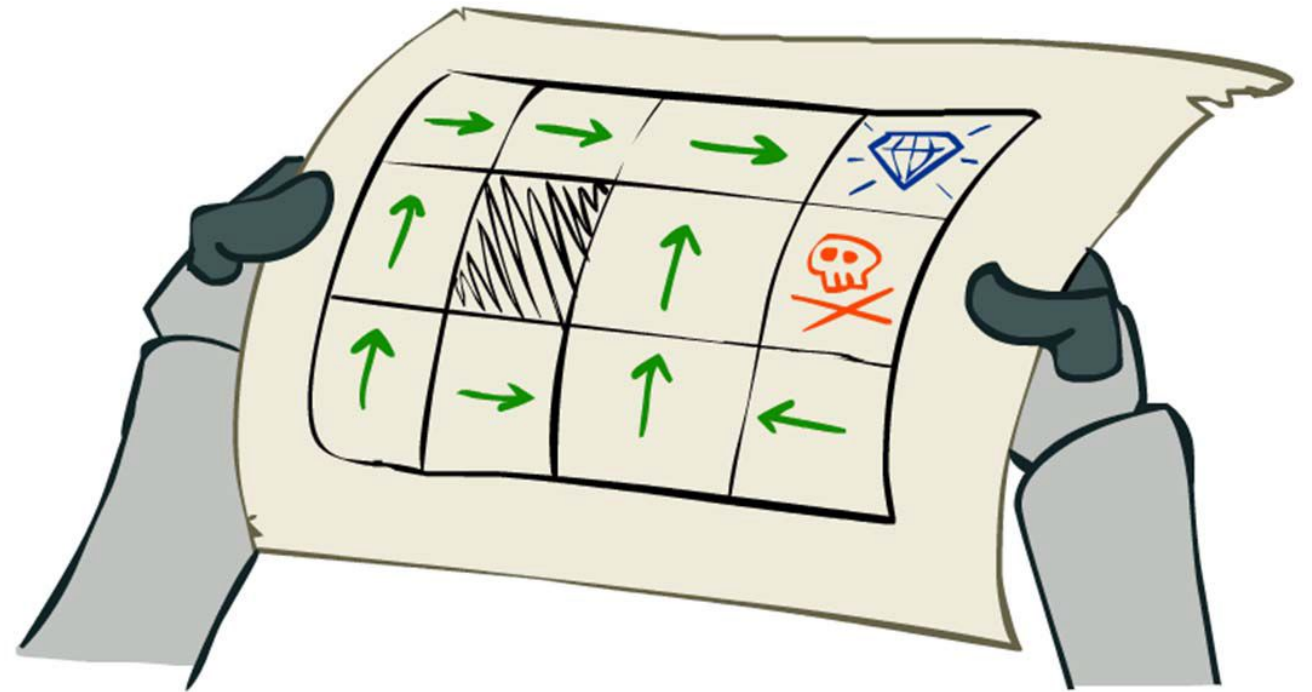
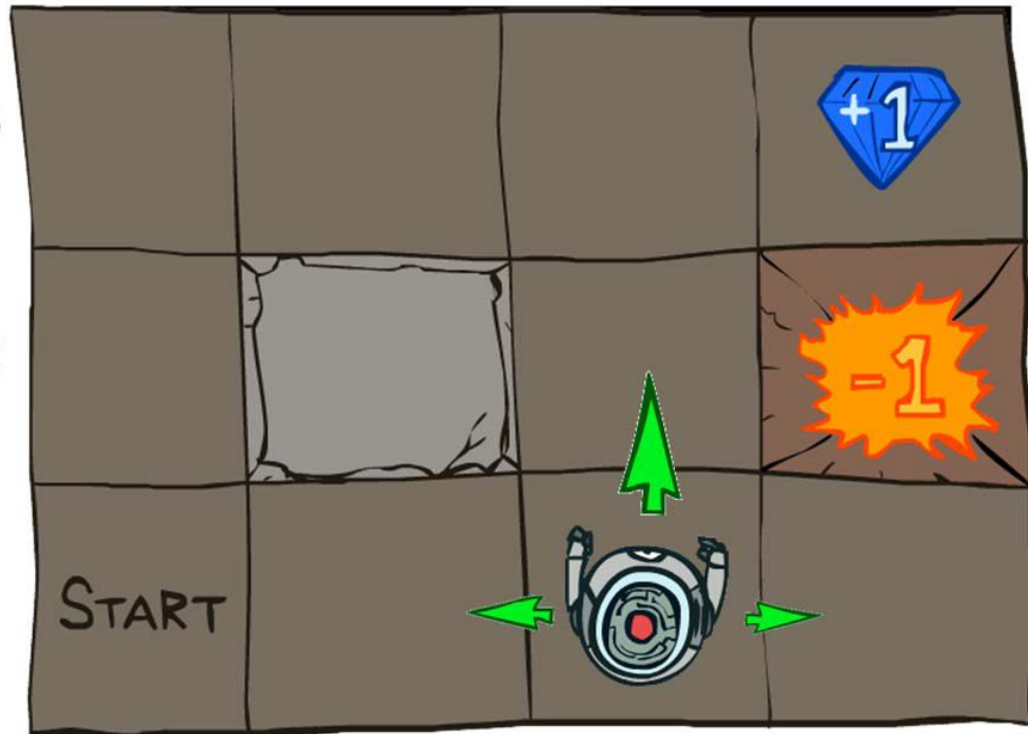
The formal mathematical model:

- **State set** S . Initial state s_0 . **Action set** A
- **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 previous actions or states.
- **Reward function:** $r(s_t)$
- **Policy:** $\pi(s) : S \rightarrow A$ action to take at a particular state.

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

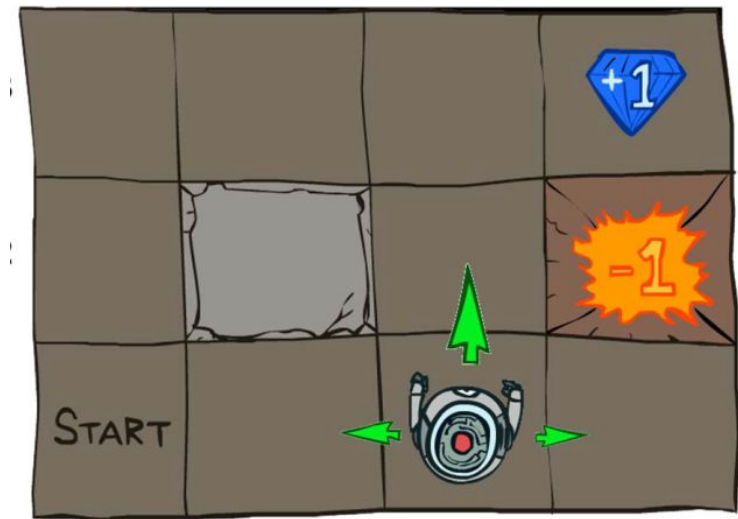
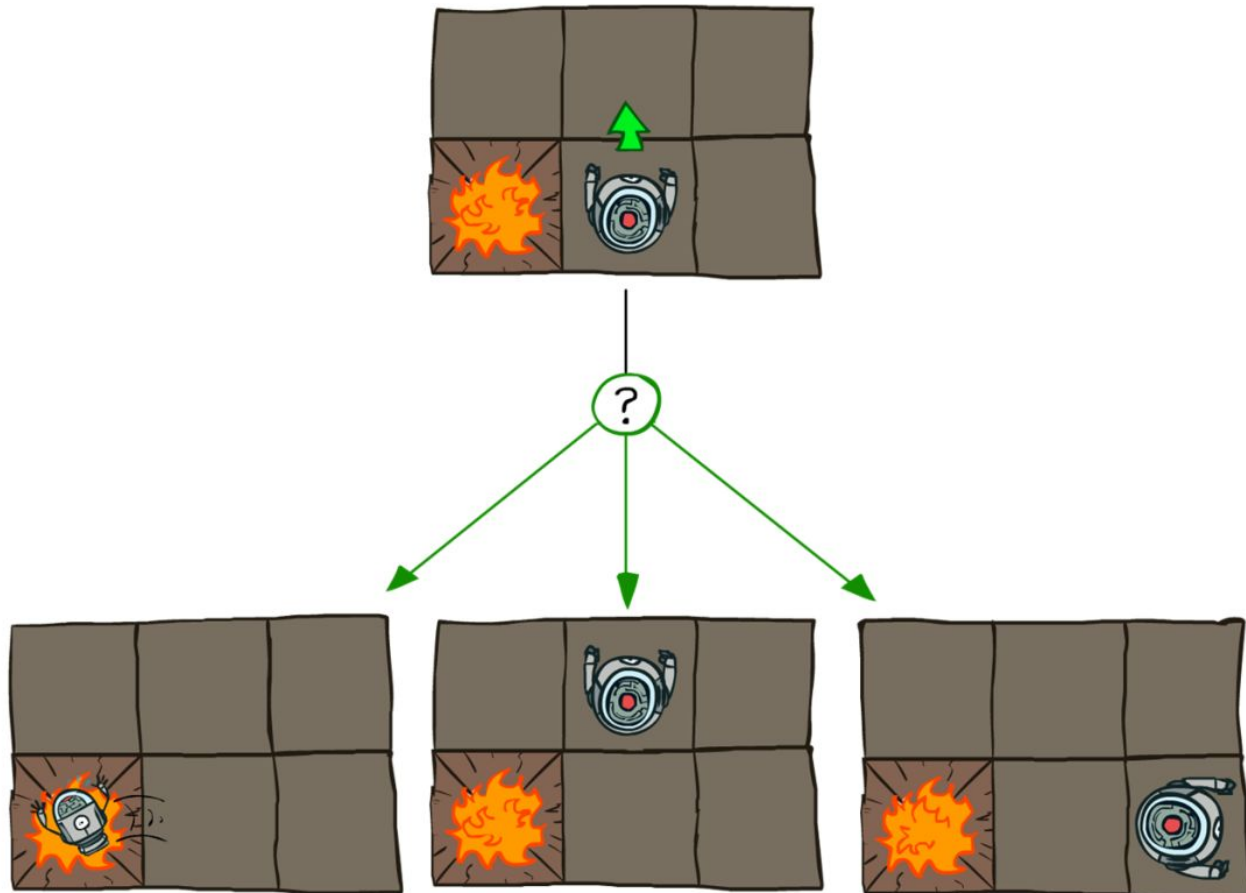
Example of MDP: Grid World

Robot on a grid; goal: find the best policy



Example of MDP: Grid World

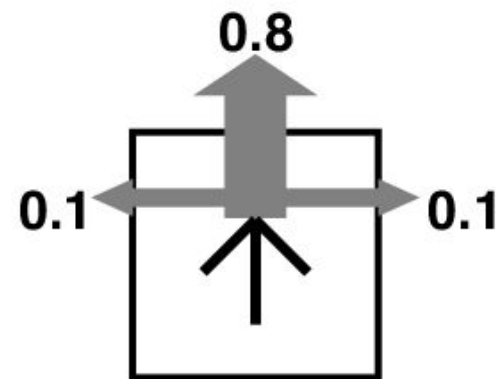
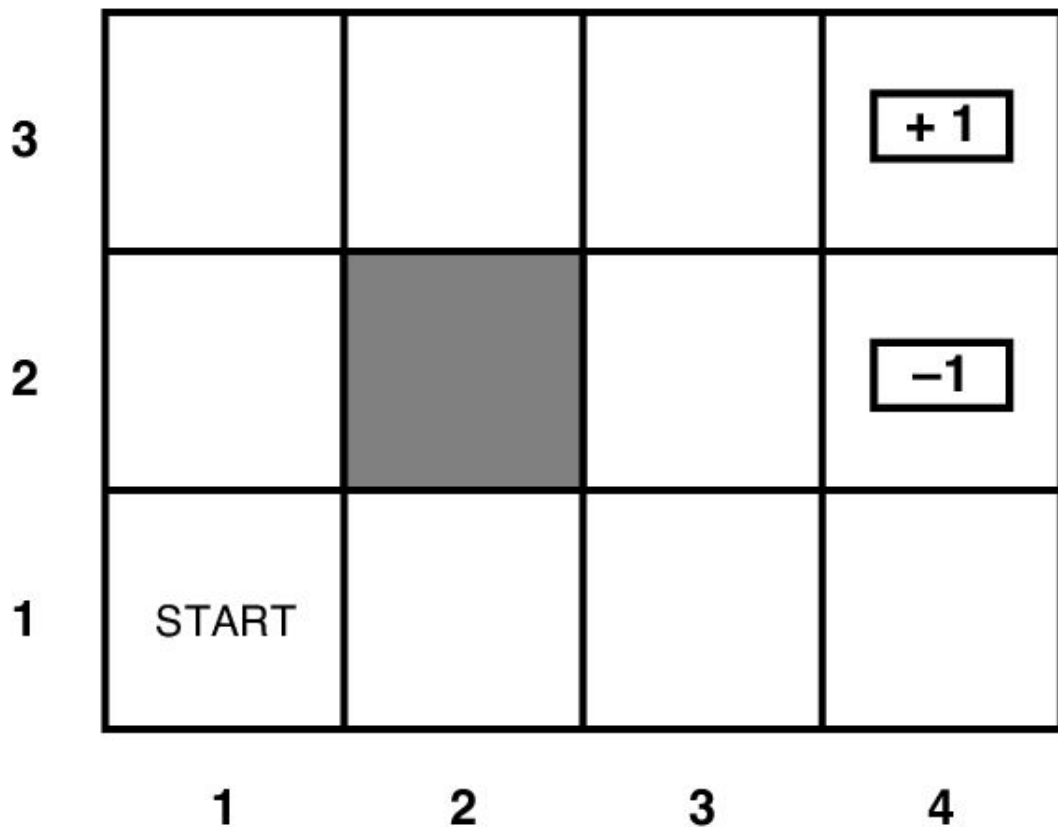
Note: (i) Robot is unreliable (ii) Reach target fast



$r(s) = -0.04$ for every non-terminal state

Grid World Abstraction

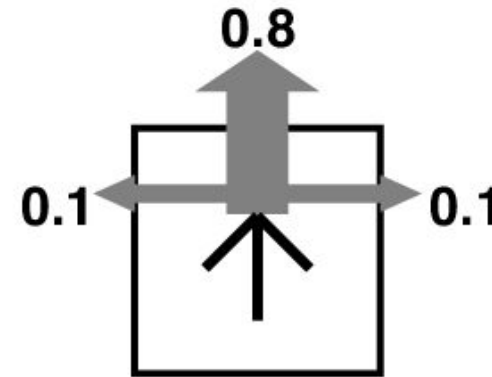
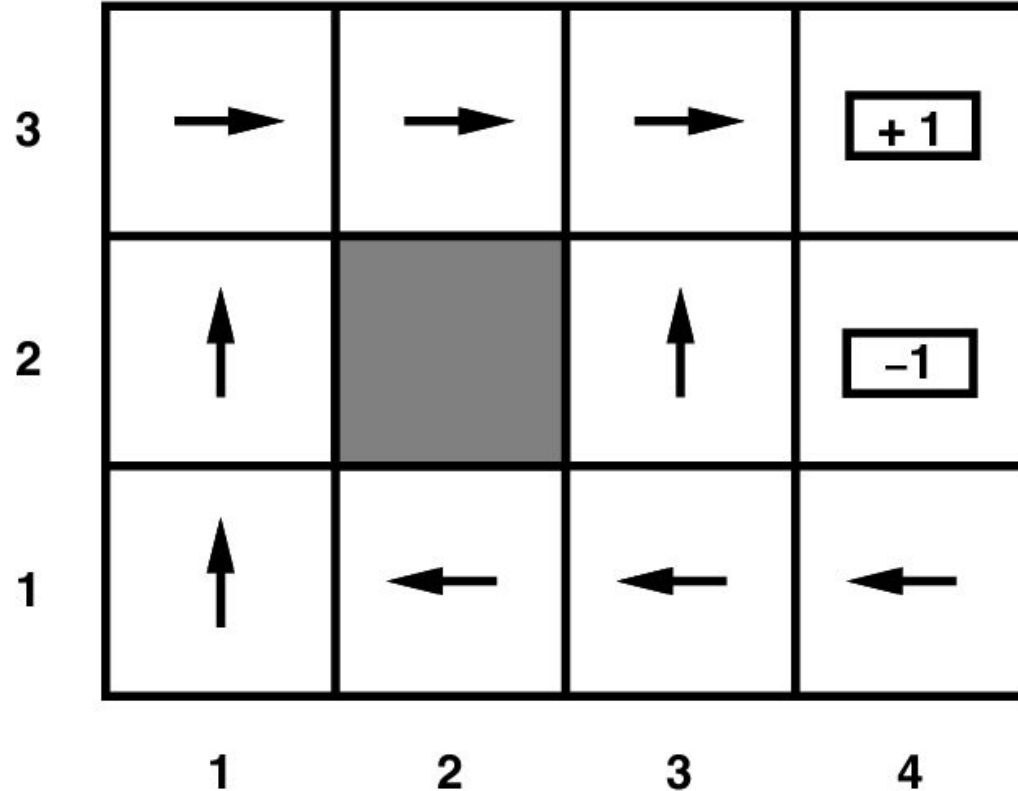
Note: (i) Robot is unreliable (ii) Reach target fast



$r(s) = -0.04$ for every non-terminal state

Grid World Optimal Policy

Note: (i) Robot is unreliable (ii) Reach target fast



$r(s) = -0.04$ for every non-terminal state

Back to MDP Setup

The formal mathematical model:

- **State set** S . Initial state s_0 . **Action set** A
- **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 previous actions or states.
- **Reward function:** $r(s_t)$
- **Policy:** $\pi(s) : S \rightarrow A$ action to take at a particular state.


How do we find
the best policy?

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

Outline

- Intro to Reinforcement Learning

 - Basic concepts, mathematical formulation, MDPs, policies

- Valuing and Obtaining Policies**

 - Value functions, Bellman equation, value iteration, policy iteration

Defining the Optimal Policy

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

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

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

Discounting Rewards

One issue: these are infinite series. **Convergence?**

- Solution

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

- Discount factor γ between 0 and 1


- Set according to how important **present** is VS **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, set $V^*(s)$ to be expected utility for **optimal** policy from s
- What's the expected utility of 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

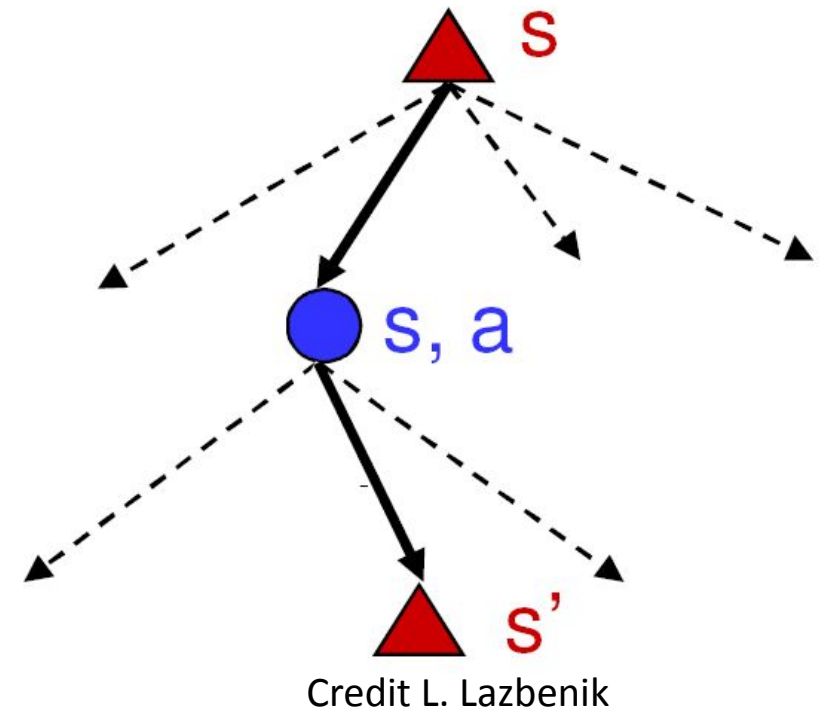
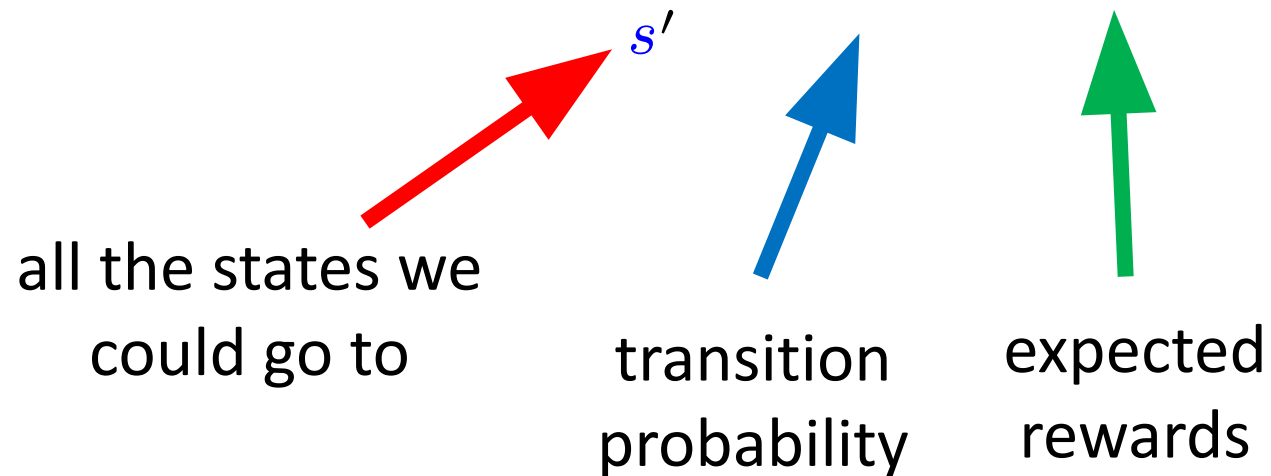
expected rewards

Obtaining the Optimal Policy

We know the expected utility of an action.

- So, to get the optimal policy, compute

$$\pi^*(s) = \operatorname{argmax}_a \sum_{s'} P(s'|s, a) V^*(s')$$



Slight Problem...

Now we can get the optimal policy by doing

$$\pi^*(s) = \operatorname{argmax}_a \sum_{s'} P(s'|s, a) V^*(s')$$

- So we need to know $V^*(s)$.
 - But it was defined in terms of the optimal policy!
 - So we need some other approach to get $V^*(s)$.
 - Need some other **property** of the value function!

Bellman Equation

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

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



current state
reward



discounted expected
future **rewards**

- Bellman: inventor of dynamic programming



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 $P(s' | s, a)$
- Also know $V^*(s)$ satisfies Bellman equation (recursion above)

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

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

Policy Iteration

With value iteration, we estimate V^*

- Then get policy (i.e., indirect estimate of policy)
- Could also try to get policies directly

- This is **policy iteration**. Basic idea:
 - Start with random policy π
 - Use it to compute value function V^π (for that policy)
 - Improve the policy: obtain π'

Policy Iteration: Algorithm

What if don't know the transition probability?
(next time)

Policy iteration. Algorithm

- Start with random policy π
- Use it to compute value function V^π : a set of linear equations

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

- Improve the policy: obtain π'

$$\pi'(\mathbf{s}) = \arg \max_{\mathbf{a}} r(\mathbf{s}) + \gamma \sum_{\mathbf{s}'} P(\mathbf{s}' | \mathbf{s}, \mathbf{a}) V^\pi(\mathbf{s}')$$

- Repeat



Thanks Everyone!

Some of the slides in these lectures have been adapted/borrowed from materials developed by Misha Khodak, Mark Craven, David Page, Jude Shavlik, Tom Mitchell, Nina Balcan, Elad Hazan, Tom Dietterich, Pedro Domingos, Jerry Zhu, Yingyu Liang, Volodymyr Kuleshov, Fred Sala