Reinforcement Learning - Technical Workshop

Julia El Zini, Ph.D.

April 28, 2025

AI Lead, AEMS.ai

Motivation

Why are you interested in RL?

Why Study Reinforcement Learning?

Why Study Reinforcement Learning?

Why Study Reinforcement Learning?

- Combinatorial complexity of Go: $\approx 10^{170}$ possible board positions

Why Study Reinforcement Learning?

- Combinatorial complexity of Go: $\approx 10^{170}$ possible board positions $>>> 10^{80} \approx$ the number of atoms on Earth

Why Study Reinforcement Learning?

- Combinatorial complexity of Go: $\approx 10^{170}$ possible board positions $>>> 10^{80} \approx$ the number of atoms on Earth
- Unclear optimal strategies and classical search methods fail

Show of hands — how many of you heard about RL only after ChatGPT?

Why RL Now?

- RL is a key enabler of modern LLMs:
  - ChatGPT, Claude, Llama trained with **RLHF**

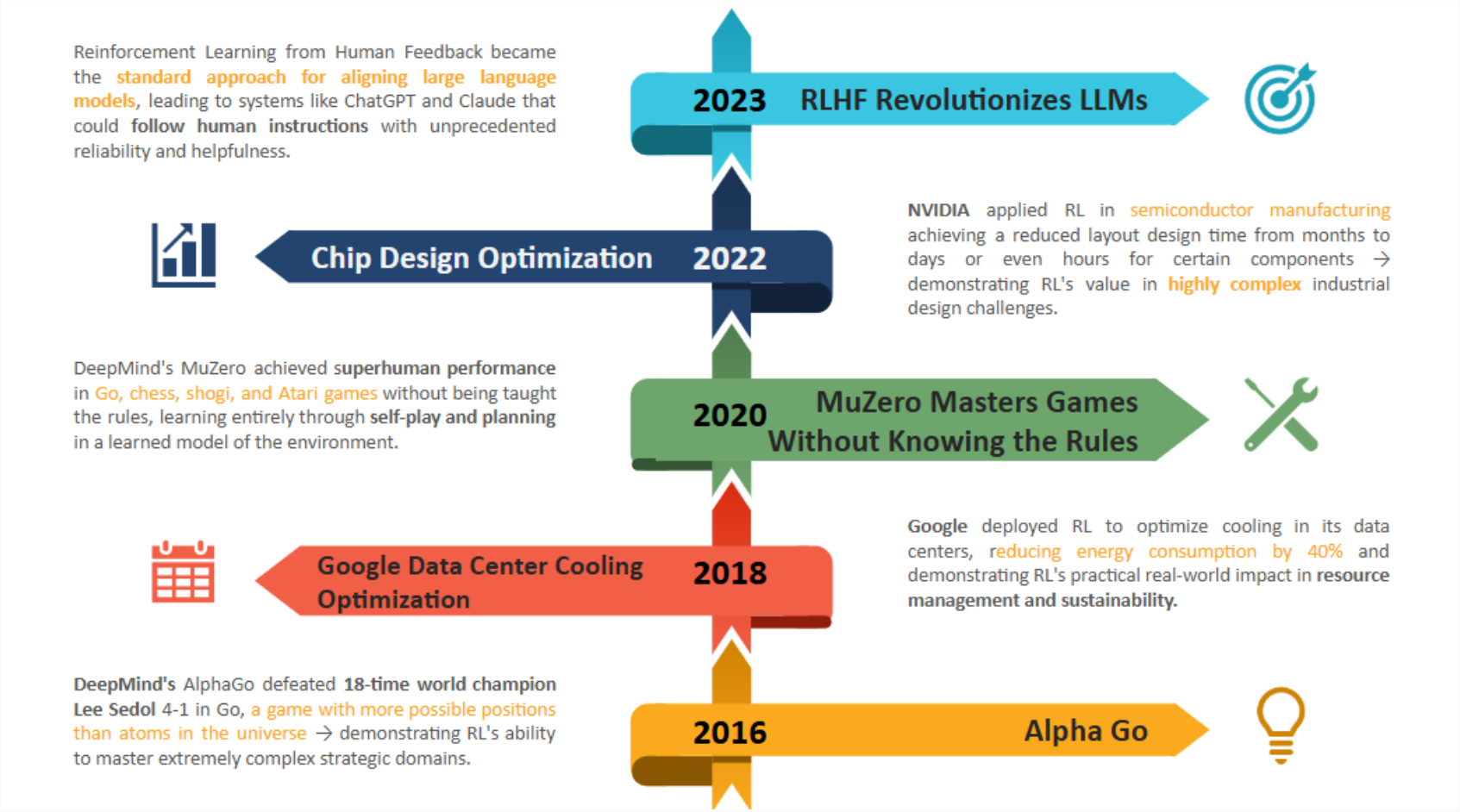
Why RL Now?

- RL is a key enabler of modern LLMs:
  - ChatGPT, Claude, Llama trained with **RLHF**
  - **Aligns** models with human preferences

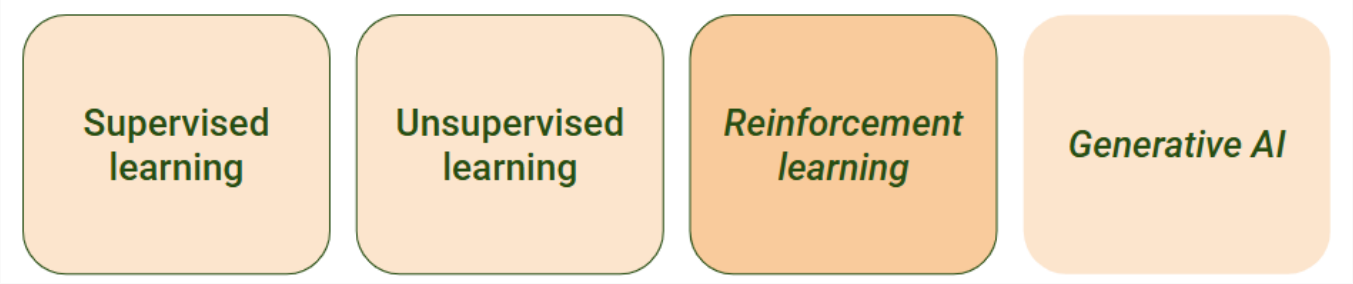
Why RL Now?

- RL is a key enabler of modern LLMs:
  - ChatGPT, Claude, Llama trained with **RLHF**
  - **Aligns** models with human preferences
  - Task adaptation **without explicit programming**

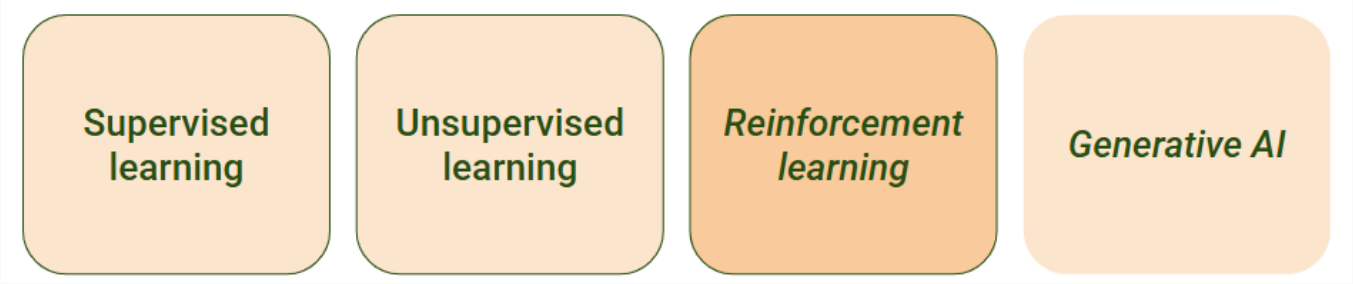
RL Timeline


Reinforcement Learning from Human Feedback became the standard approach for aligning large language models, leading to systems like ChatGPT and Claude that could follow human instructions with unprecedented reliability and helpfulness.
2023 RLHF Revolutionizes LLMs
NVIDIA applied RL in semiconductor manufacturing achieving a reduced layout design time from months to days or even hours for certain components → demonstrating RL's value in highly complex industrial design challenges.
Chip Design Optimization 2022
DeepMind's MuZero achieved superhuman performance in Go, chess, shogi, and Atari games without being taught the rules, learning entirely through self-play and planning in a learned model of the environment.
2020 MuZero Masters Games Without Knowing the Rules
Google deployed RL to optimize cooling in its data centers, reducing energy consumption by 40% and demonstrating RL's practical real-world impact in resource management and sustainability.
Google Data Center Cooling Optimization 2018
DeepMind's AlphaGo defeated 18-time world champion Lee Sedol 4-1 in Go, a game with more possible positions than atoms in the universe → demonstrating RL's ability to master extremely complex strategic domains.
2016 Alpha Go

The Big Picture: Where RL Fits?



The Big Picture: Where RL Fits?



Today's Workshop

We'll explore:

- How to **formulate** problems for RL
- The **mathematical foundations** (MDPs)
- **Solution approaches** at a high-level
- How RL **transformed generative AI**
- The basic features of a Python environment to solve a basic RL problem

RL Definition

Let's Begin with a Simple Question

What would you do if you could only learn from trial and error?

Let's Begin with a Simple Question

What would you do if you could only learn from trial and error?

- Imagine learning a new skill where nobody can tell you what's **right or wrong**...
- You only receive a score after completing the entire task...
- And you need to determine which actions contributed to your success or failure...

Let's Begin with a Simple Question

What would you do if you could only learn from trial and error?

- Imagine learning a new skill where nobody can tell you what's **right or wrong**...
- You only receive a score after completing the entire task...
- And you need to determine which actions contributed to your success or failure...

This is the fundamental challenge reinforcement learning solves.

Problems that RL tackles

When to cast a problem into RL?

- When the problem involves **sequential decision making**

Problems that RL tackles

When to cast a problem into RL?

- When the problem involves **sequential decision making**
- When there is a clear **agent-environment interaction**

Problems that RL tackles

When to cast a problem into RL?

- When the problem involves **sequential decision making**
- When there is a clear **agent-environment interaction**
- When immediate **feedback** is not enough - delayed rewards are crucial

Problems that RL tackles

When to cast a problem into RL?

- When the problem involves **sequential decision making**
- When there is a clear **agent-environment interaction**
- When immediate **feedback** is not enough - delayed rewards are crucial
- When the goal is to **optimize long-term performance** rather than immediate outcomes

Problems that RL tackles

When to cast a problem into RL?

- When the problem involves **sequential decision making**
- When there is a clear **agent-environment interaction**
- When immediate **feedback** is not enough - delayed rewards are crucial
- When the goal is to **optimize long-term performance** rather than immediate outcomes
- When the problem has a **temporal component** (actions affect future states)

Problems that RL tackles

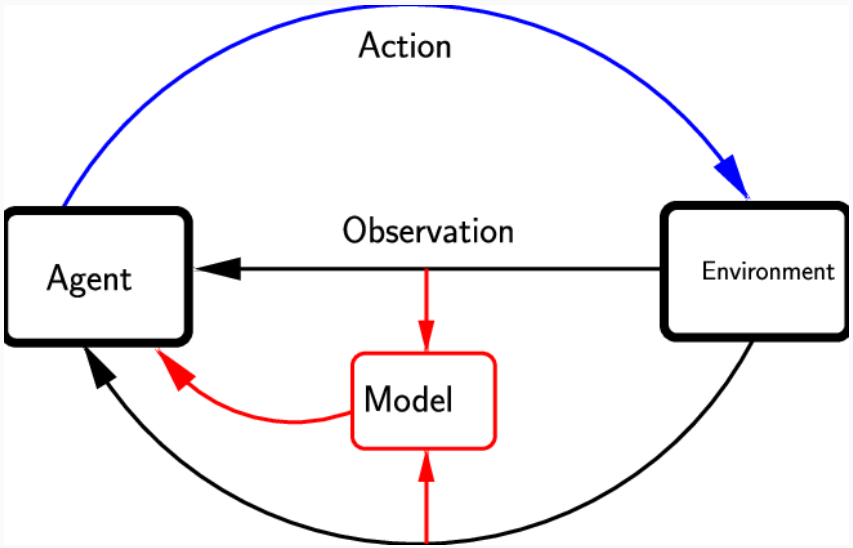
When to cast a problem into RL?

- When the problem involves **sequential decision making**
- When there is a clear **agent-environment interaction**
- When immediate **feedback** is not enough - delayed rewards are crucial
- When the goal is to **optimize long-term performance** rather than immediate outcomes
- When the problem has a **temporal component** (actions affect future states)
- When the environment dynamics are **complex or unknown**

Characteristics of RL Problems

Key Components

- **Agent**: The decision-maker or learner
- **Environment**: Everything that the agent interacts with, the environment models how the agent will move from one state to the other
- **Reward**: Feedback signal indicating success
- **Policy**: Agent's strategy for selecting actions $\rightarrow$ the policy is what the RL model will learn



RL vs. Other Learning Paradigms

Supervised Learning

- Learning from labeled examples
- Immediate feedback
- No sequential decision making
- No concept of delayed rewards
- Example: Classification, regression

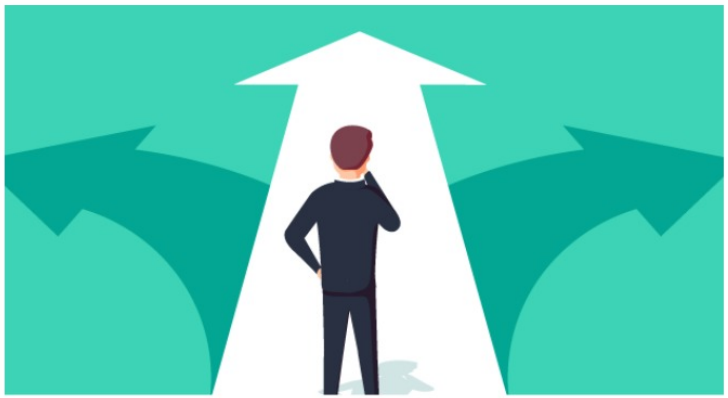
Unsupervised Learning

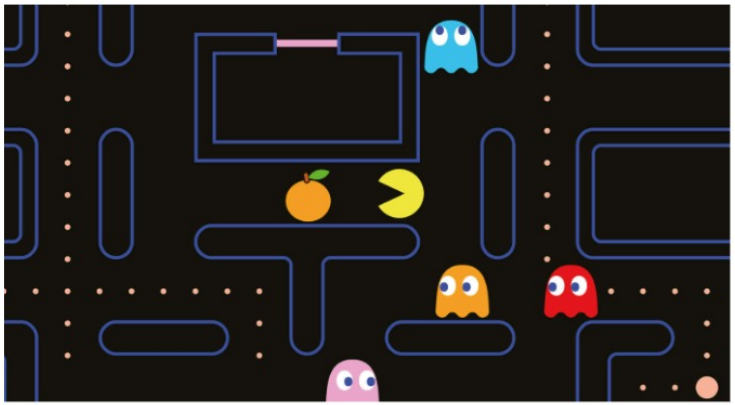
- Learning from unlabeled data
- No explicit feedback
- Finding patterns
- No direct optimization goal
- Example: Clustering, dimensionality reduction

Reinforcement Learning

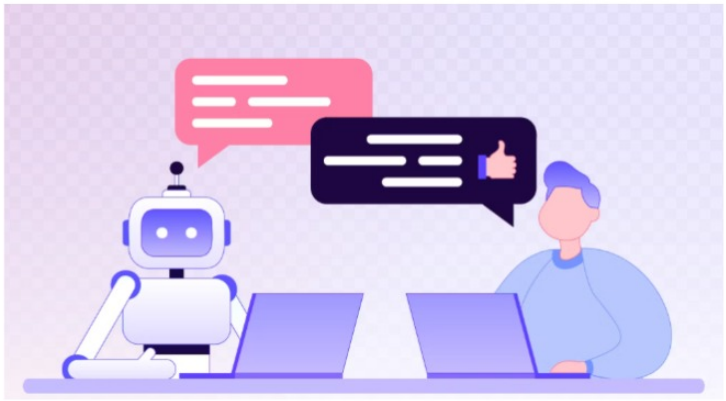
- Learning from interaction
- Sparse, delayed rewards
- Sequential decision making
- Exploration-exploitation dilemma
- Example: Decision making, games

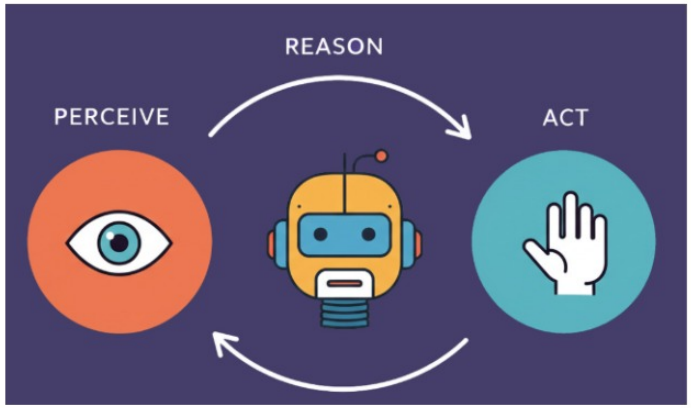
Reinforcement Learning Before The Generative AI Wave





Reinforcement Learning After The Generative AI Wave





Formulating and Solving Real-World Problems as RL

Phase I: Modeling

1. **Identify the agent and environment**
2. **Modeling decisions**:
   - What information does the agent need to make decisions?
   - What actions can the agent take?
   - Is the action space discrete or continuous?
   - What signals success or failure?

Phase II: Training

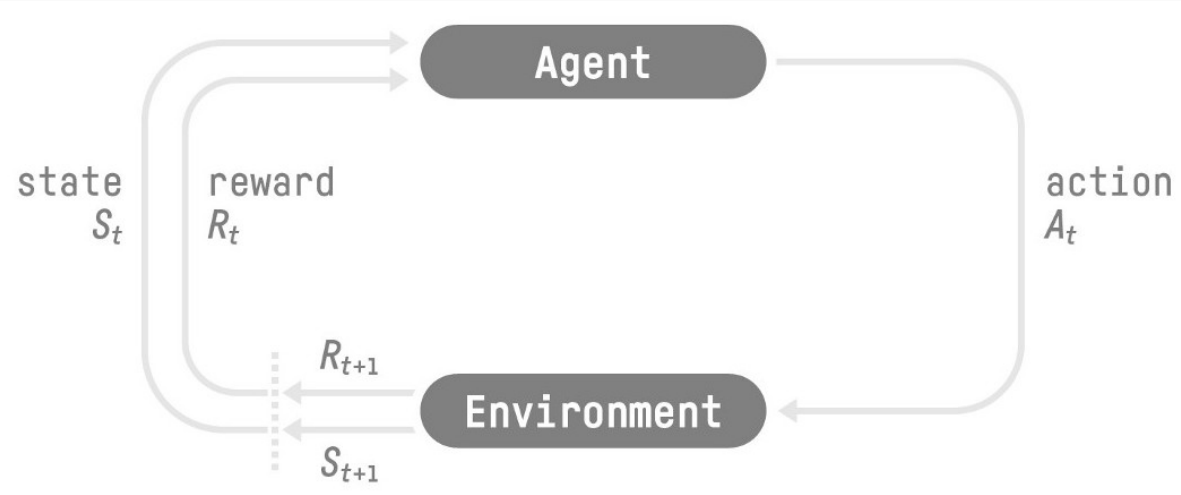
1. **Choose (a) suitable RL algorithm(s)**
2. Train the model and fine-tune its hyperparameters

RL Formulation

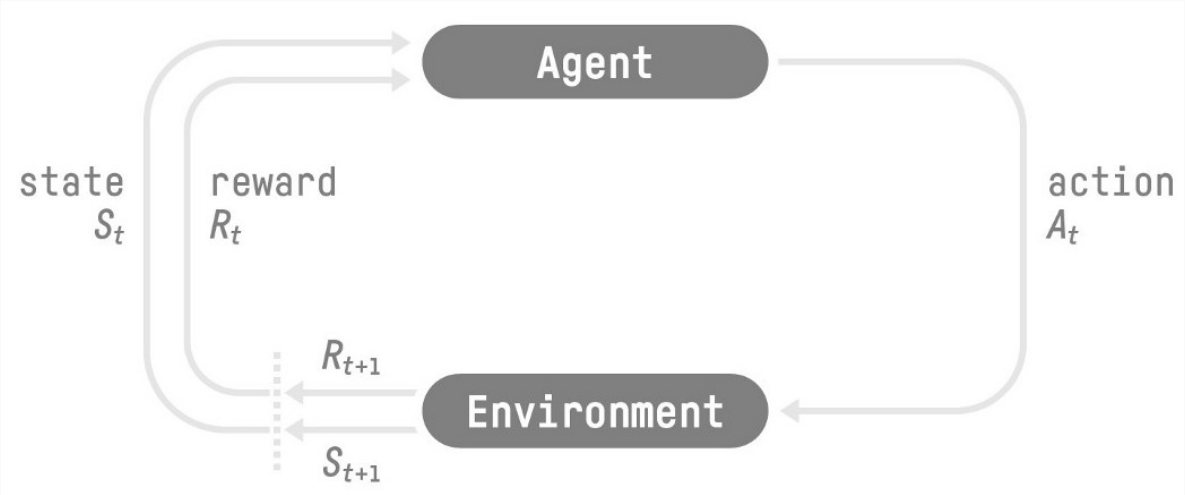
RL Formulation

RL Dynamics

Reinforcement Learning Dynamics



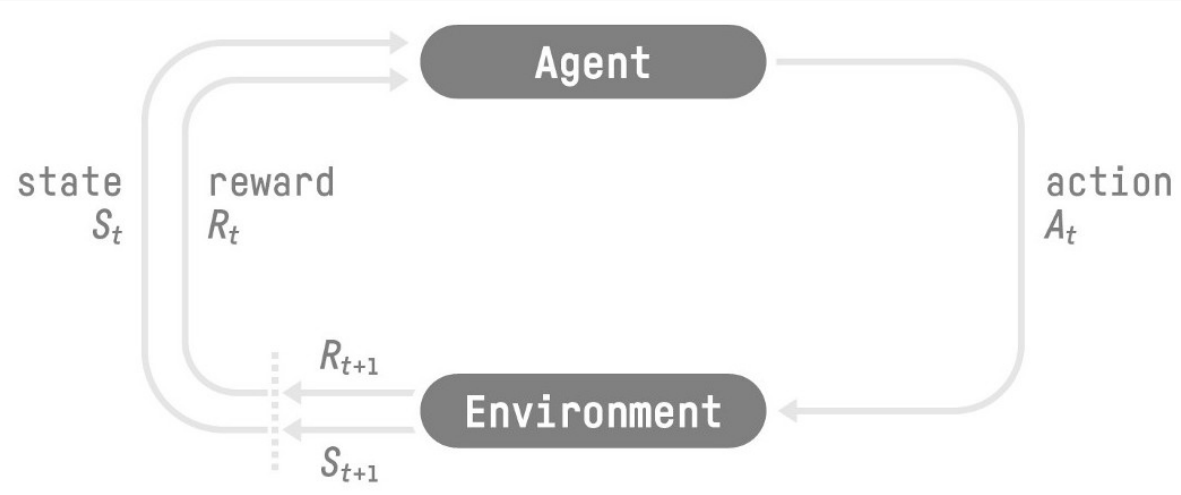
Reinforcement Learning Dynamics



At each time step t,

- the agent is at state s_t

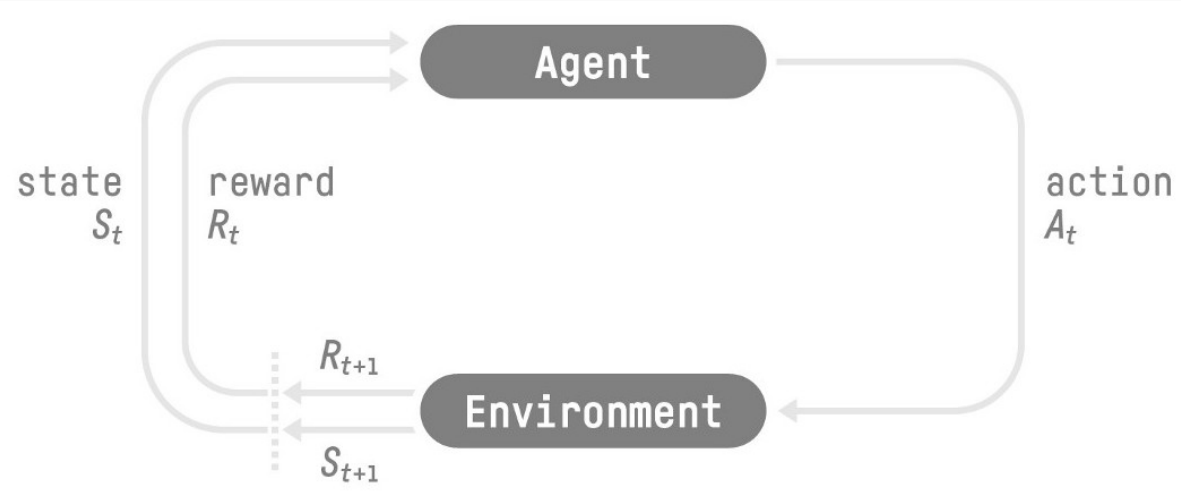
Reinforcement Learning Dynamics



At each time step t,

- the agent is at state s_t
- chooses and action a

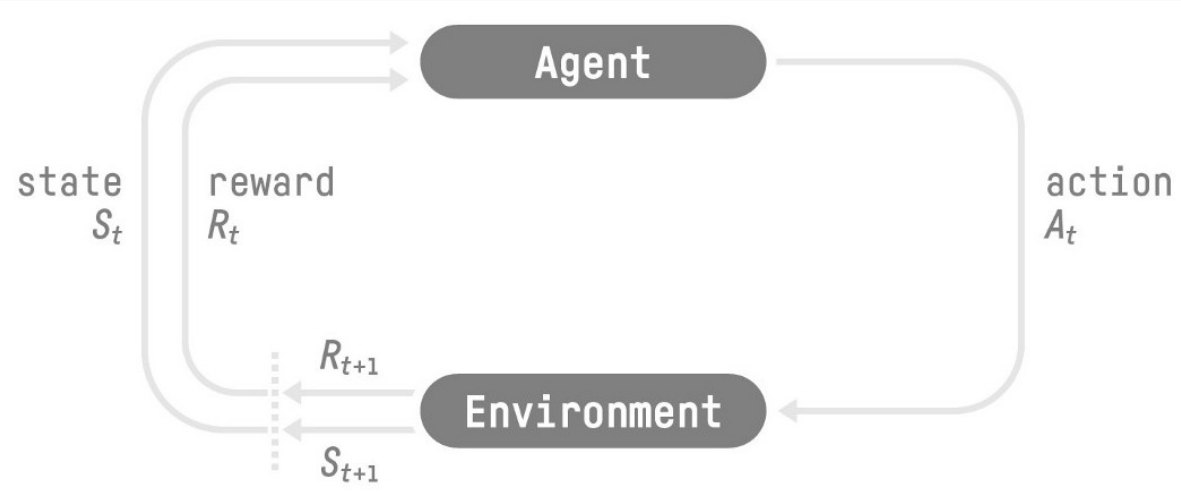
Reinforcement Learning Dynamics



At each time step t,

- the agent is at state s_t
- chooses and action a
- transitions to state s_{t+1}

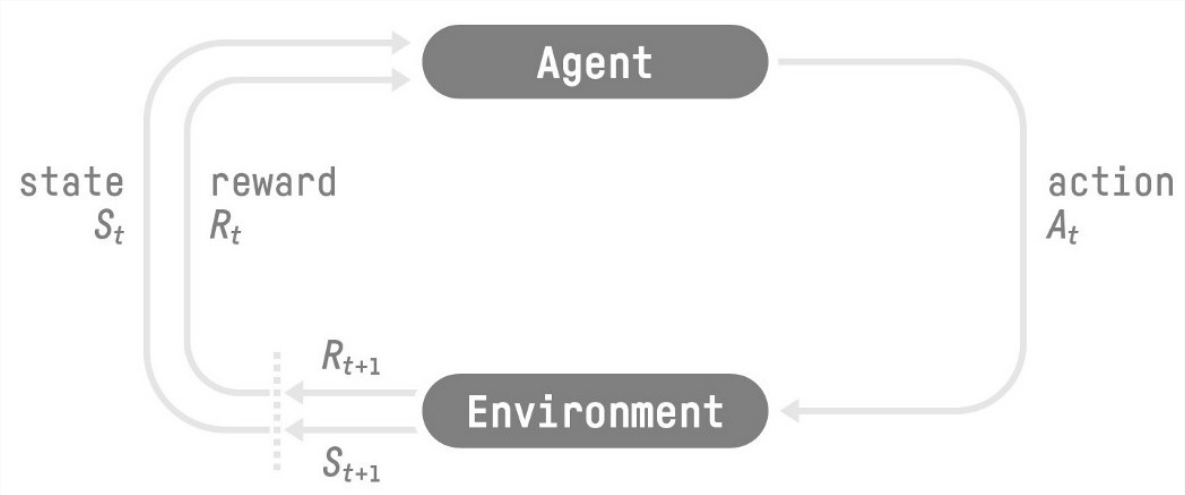
Reinforcement Learning Dynamics



At each time step t,

- the agent is at state s_t
- chooses and action a
- transitions to state s_{t+1}
- and receives a reward r_t

Reinforcement Learning Dynamics



At each time step t,

- the agent is at state s_t
- chooses and action a
- transitions to state s_{t+1}
- and receives a reward r_t

How to formulate such a problem???

RL Formulation

RL as MDP

RL Formulation as MDP

RL as MDP

MDP is defined by:

- a set of states s

RL Formulation as MDP

RL as MDP

MDP is defined by:

- a set of states s
- a set of actions a

RL Formulation as MDP

RL as MDP

MDP is defined by:

- a set of states s
- a set of actions a
- a transition function $\mathcal{P}(s, a, s')$

RL Formulation as MDP

RL as MDP

MDP is defined by:

- a set of states s
- a set of actions a
- a transition function $\mathcal{P}(s, a, s')$
  - probability that action a leads to state s' when executed in state s

RL Formulation as MDP

RL as MDP

MDP is defined by:

- a set of states s
- a set of actions a
- a transition function $\mathcal{P}(s, a, s')$
  - probability that action a leads to state s' when executed in state s
  - also called the model or the dynamics

RL Formulation as MDP

RL as MDP

MDP is defined by:

- a set of states s
- a set of actions a
- a transition function $\mathcal{P}(s, a, s')$
  - probability that action a leads to state s' when executed in state s
  - also called the model or the dynamics
- a reward function $R(s, a, s')$

RL Formulation as MDP

RL as MDP

MDP is defined by:

- a set of states s
- a set of actions a
- a transition function $\mathcal{P}(s, a, s')$
  - probability that action a leads to state s' when executed in state s
  - also called the model or the dynamics
- a reward function $R(s, a, s')$
  - sometimes just $R(s')$

RL Formulation as MDP

RL as MDP

MDP is defined by:

- a set of states s
- a set of actions a
- a transition function $\mathcal{P}(s, a, s')$
  - probability that action a leads to state s' when executed in state s
  - also called the model or the dynamics
- a reward function $R(s, a, s')$
  - sometimes just $R(s')$
- a start state

RL Formulation as MDP

RL as MDP

MDP is defined by:

- a set of states s
- a set of actions a
- a transition function $\mathcal{P}(s, a, s')$
  - probability that action a leads to state s' when executed in state s
  - also called the model or the dynamics
- a reward function $R(s, a, s')$
  - sometimes just $R(s')$
- a start state
- maybe a goal or an end state

RL Formulation as MDP

RL as MDP

MDP is defined by:

- a set of states s
- a set of actions a
- a transition function $\mathcal{P}(s, a, s')$
  - probability that action a leads to state s' when executed in state s
  - also called the model or the dynamics
- a reward function $R(s, a, s')$
  - sometimes just $R(s')$
- a start state
- maybe a goal or an end state
- a discount factor γ

RL Formulation as MDP: Reward discount

Why reward discount

- Prefer rewards now to rewards later
- Rewards are discounted over time by a factor of γ^t for example.

1

Worth Now

γ

Worth Next Step

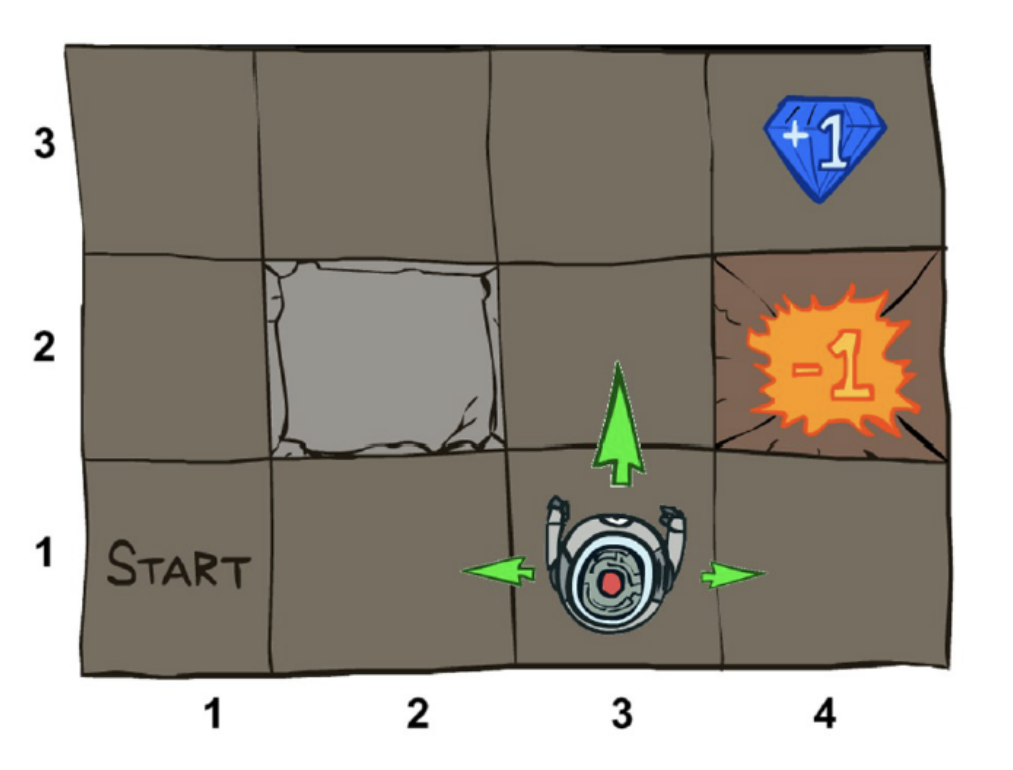
γ^2

Worth In Two Steps

RL Formulation

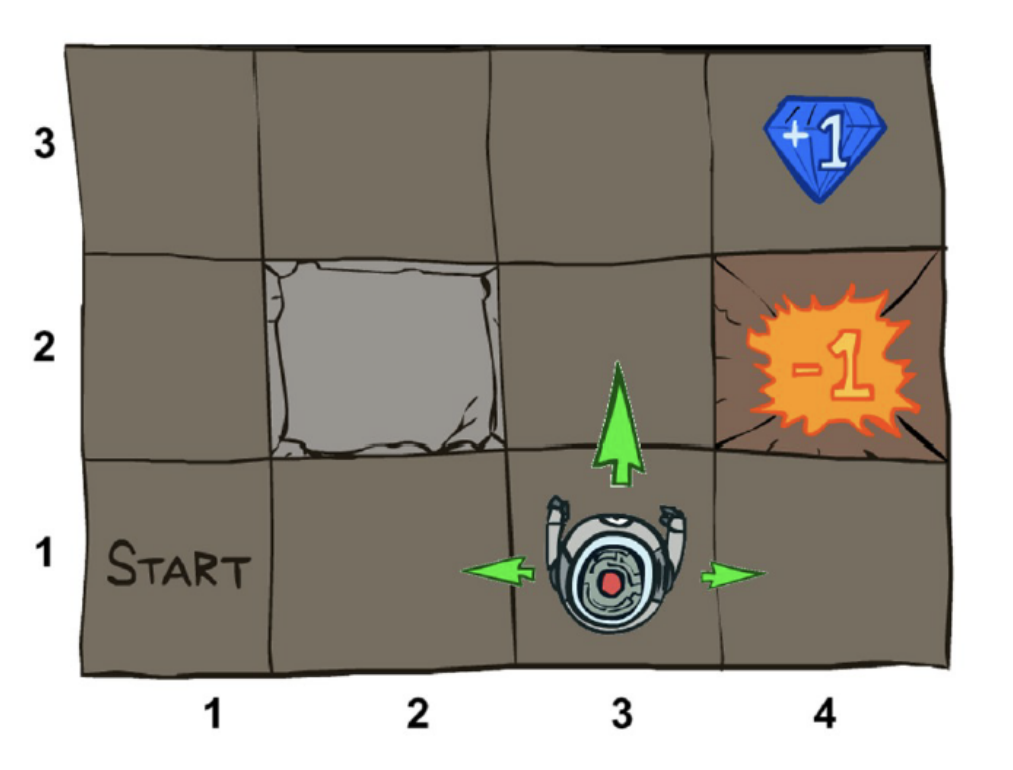
MDP Examples

RL Formulation as MDP: Example



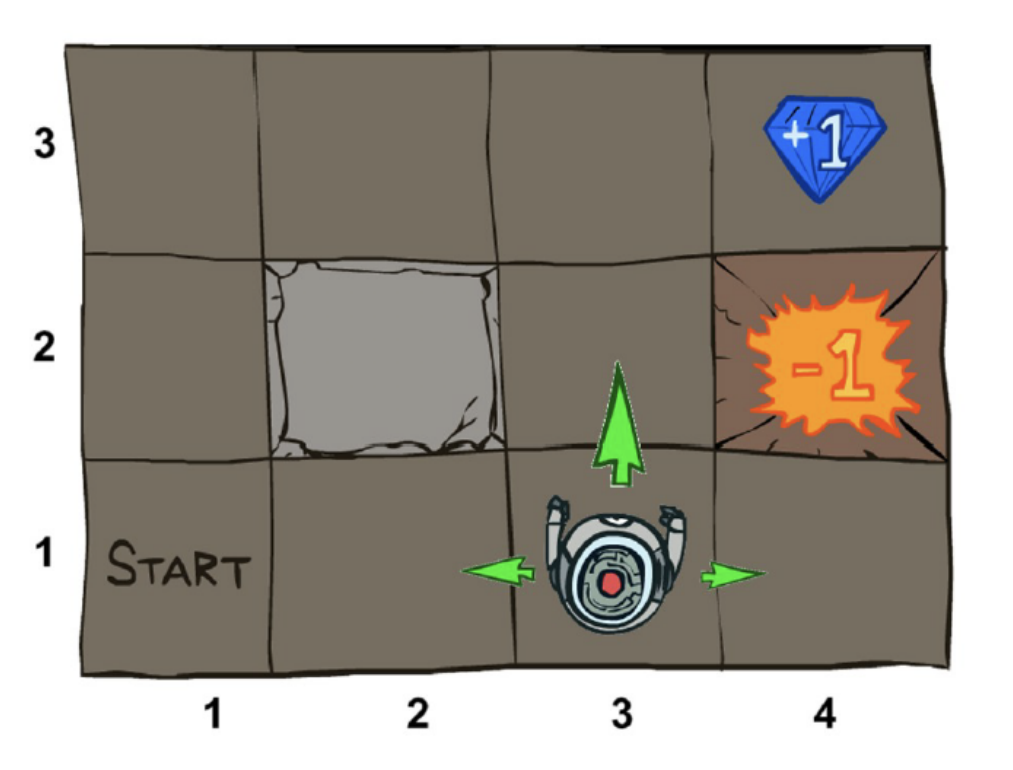
- 3×4 grid

RL Formulation as MDP: Example



- 3×4 grid
- Goal is to reach the diamond ($+1$) while staying away from fire (-1).

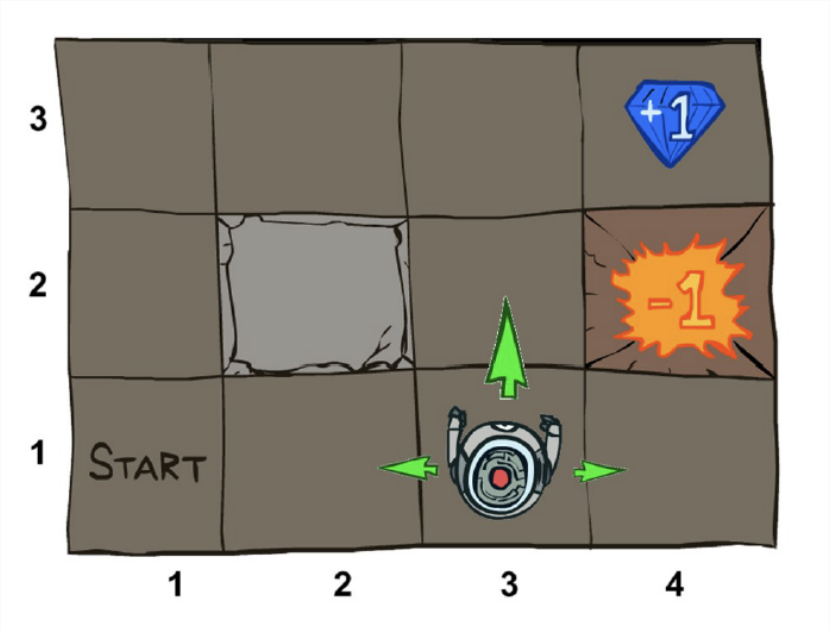
RL Formulation as MDP: Example



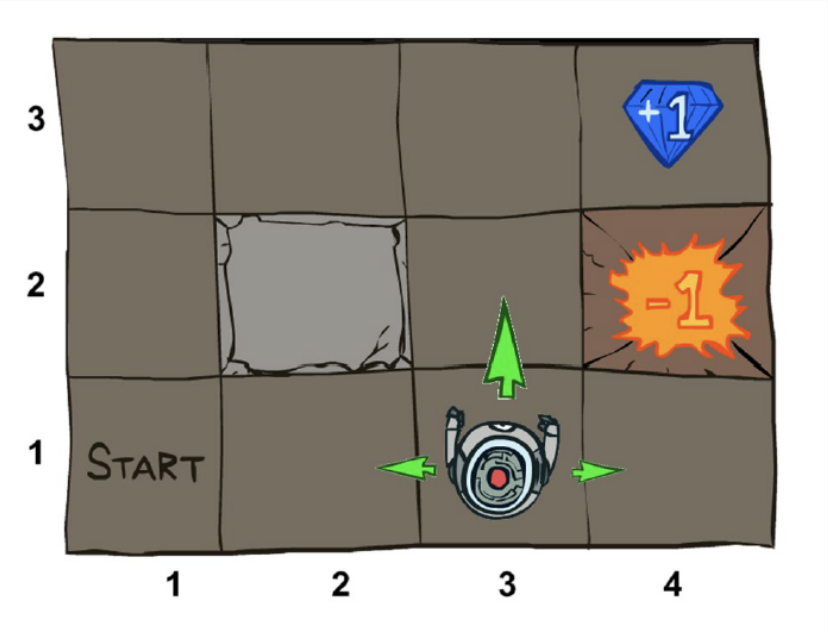
- 3×4 grid
- Goal is to reach the diamond ($+1$) while staying away from fire (-1).
- What do you think a formulation would include?

RL Formulation as MDP: Example Cont'd

- States: $(1,1), (1,2), \ldots, (1,4)$

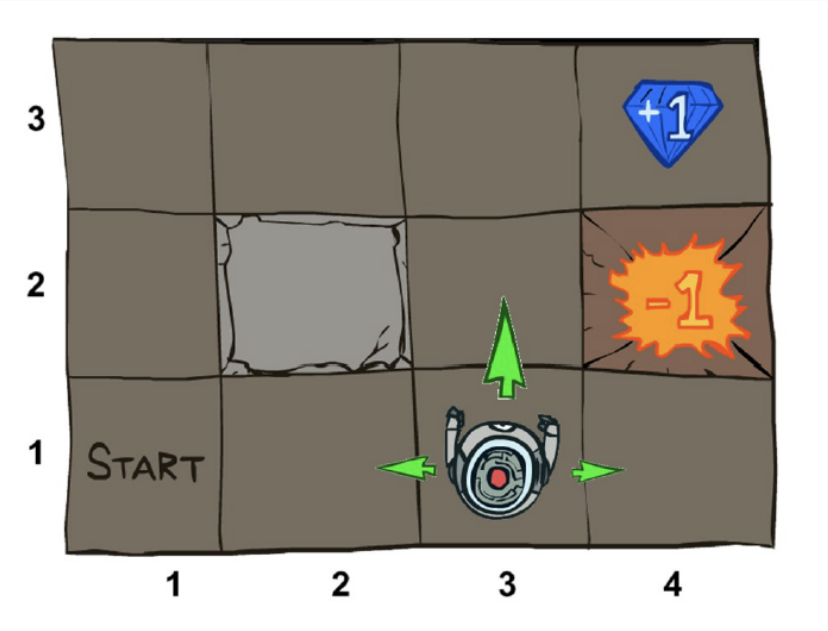


RL Formulation as MDP: Example Cont'd



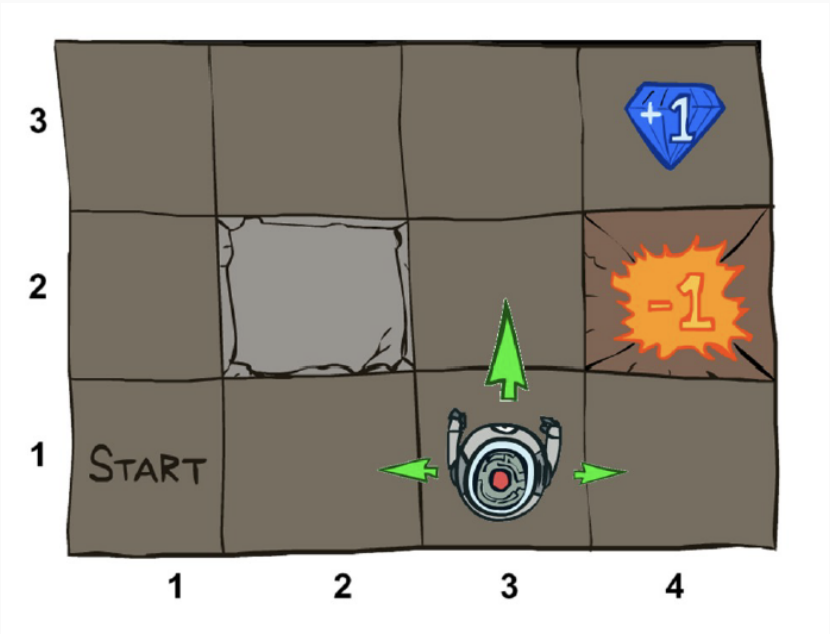
- States: $(1,1),(1,2),\ldots,(1,4)$
- Actions: left, right, up, down

RL Formulation as MDP: Example Cont'd



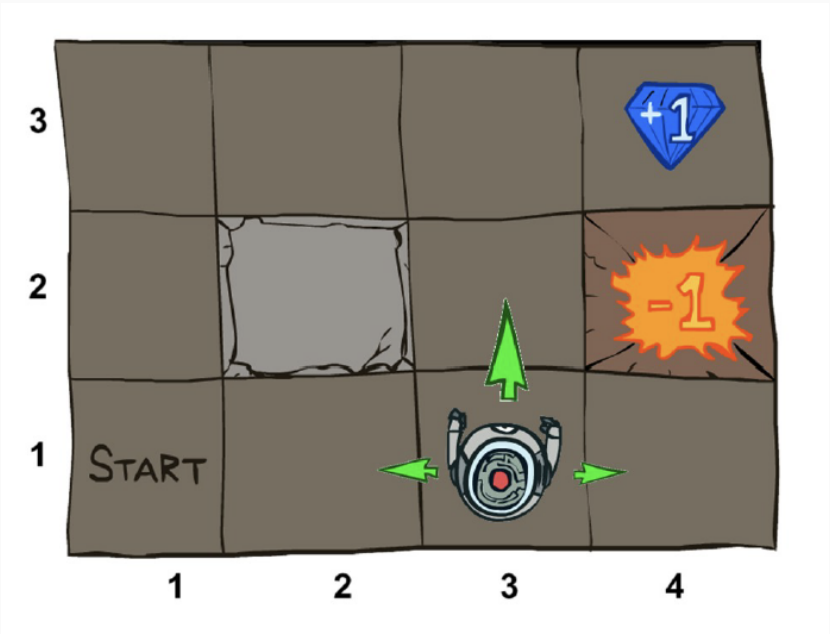
- States: $(1,1),(1,2),\ldots,(1,4)$
- Actions: left, right, up, down
- Deterministic transition. Ex:
  $\mathcal{P}((1,1),\text{right},(1,2)) = 1$,
  $\mathcal{P}((1,2),\text{up},(1,2)) = 1$

RL Formulation as MDP: Example Cont'd



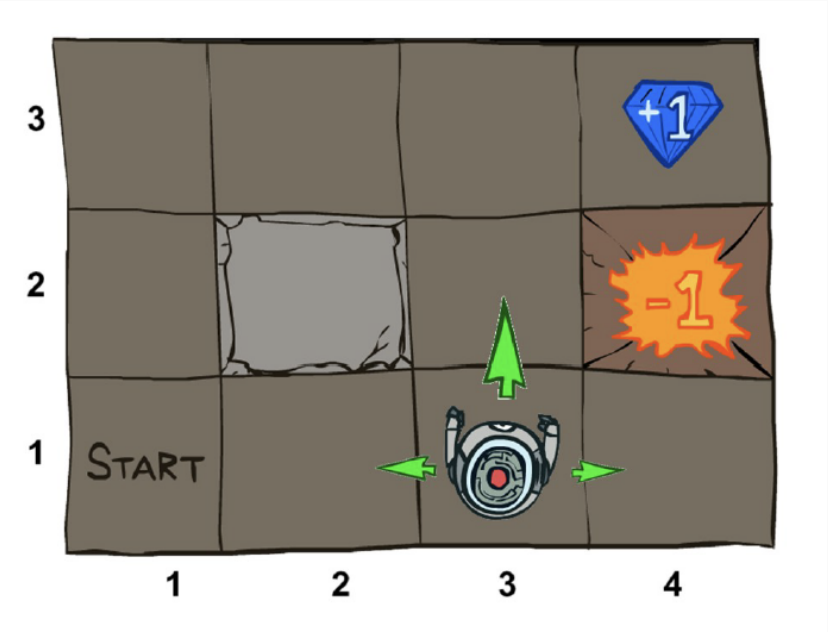
- States: $(1,1),(1,2),\ldots,(1,4)$
- Actions: left, right, up, down
- Deterministic transition. Ex:
  $\mathcal{P}((1,1),\text{right},(1,2)) = 1$,
  $\mathcal{P}((1,2),\text{up},(1,2)) = 1$
- $R((2,4)) = -1$, $R((3,4)) = 1$ and the rest of the states s'' are such that $R(s'') = -0.03$

RL Formulation as MDP: Example Cont'd



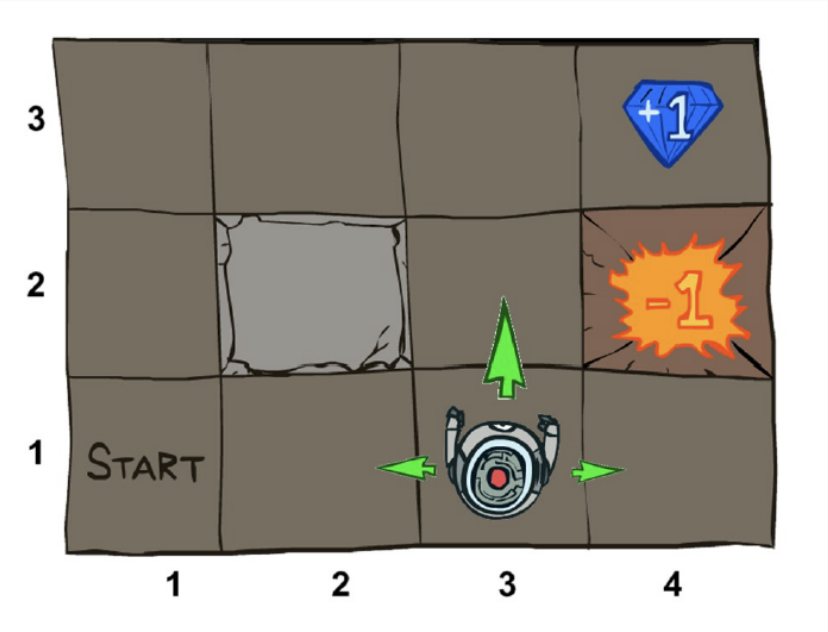
- States: $(1,1),(1,2),\ldots,(1,4)$
- Actions: left, right, up, down
- Deterministic transition. Ex:
  $\mathcal{P}((1,1),\text{right},(1,2)) = 1$,
  $\mathcal{P}((1,2),\text{up},(1,2)) = 1$
- $R((2,4)) = -1$, $R((3,4)) = 1$ and the rest of the states s'' are such that $R(s'') = -0.03$
- Start $= (1,1)$

RL Formulation as MDP: Example Cont'd



- States: $(1,1),(1,2),\ldots,(1,4)$
- Actions: left, right, up, down
- Deterministic transition. Ex:
  $\mathcal{P}((1,1),\text{right},(1,2)) = 1$,
  $\mathcal{P}((1,2),\text{up},(1,2)) = 1$
- $R((2,4)) = -1$, $R((3,4)) = 1$ and the rest of the states s'' are such that $R(s'') = -0.03$
- Start $= (1,1)$
- Goal $= (3,4)$

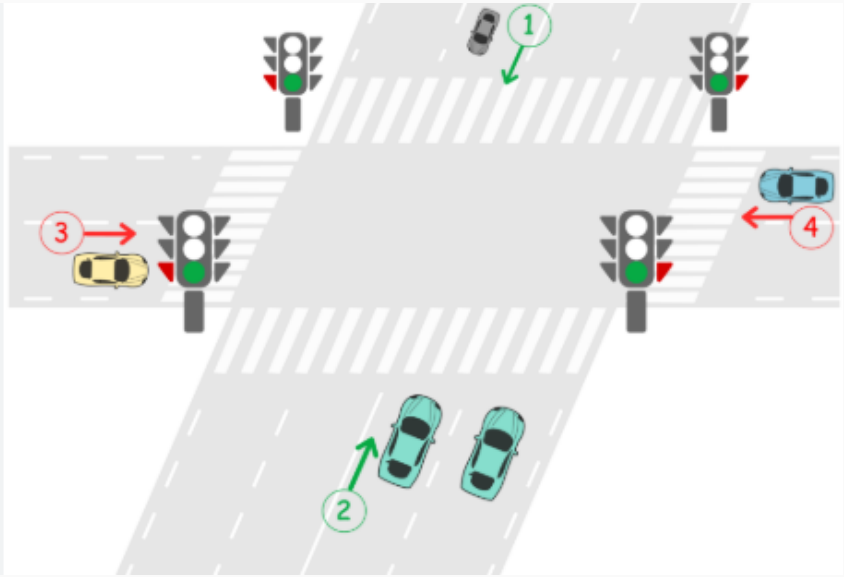
RL Formulation as MDP: Example Cont'd



- States: $(1,1),(1,2),\ldots,(1,4)$
- Actions: left, right, up, down
- Deterministic transition. Ex: $\mathcal{P}((1,1),\text{right},(1,2)) = 1$, $\mathcal{P}((1,2),\text{up},(1,2)) = 1$
- $R((2,4)) = -1$, $R((3,4)) = 1$ and the rest of the states s'' are such that $R(s'') = -0.03$
- Start $= (1,1)$
- Goal $= (3,4)$
- Discount factor could be $0 < \gamma \leq 1$ (1 means no discount).

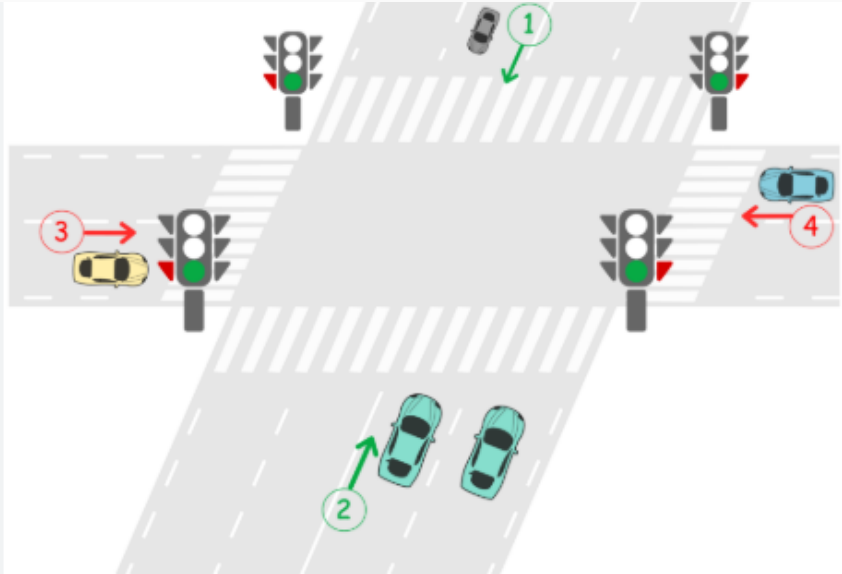
Example 2: Traffic Light Control as RL

Goal: Minimize average waiting time across all vehicles



Example 2: Traffic Light Control as RL

Goal: Minimize average waiting time across all vehicles

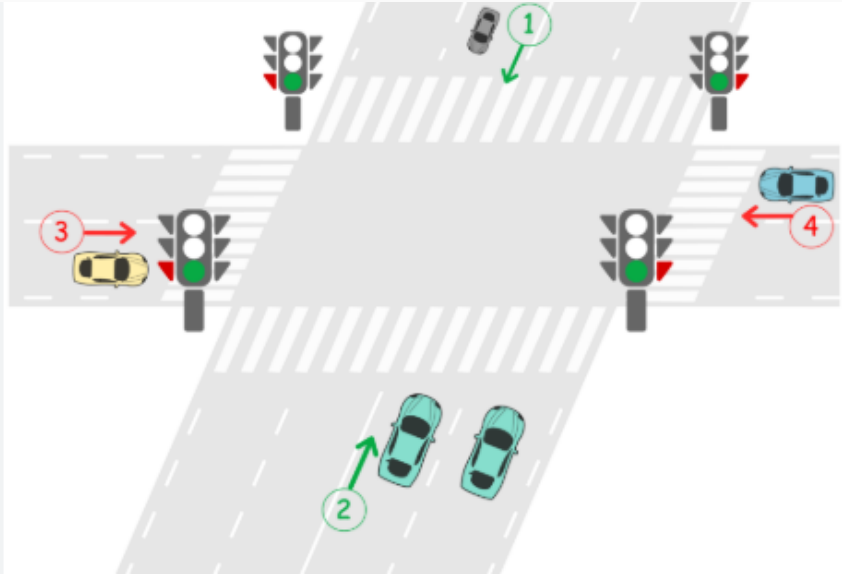


Problem Formulation

- **Agent**: Traffic light controller

Example 2: Traffic Light Control as RL

Goal: Minimize average waiting time across all vehicles

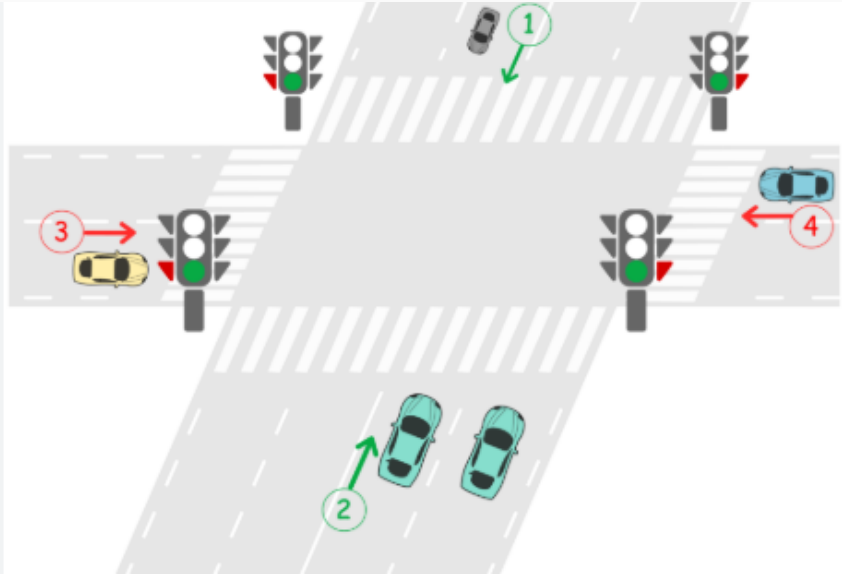


Problem Formulation

- **Agent**: Traffic light controller
- **State**: Queue lengths, waiting times, current signal phase

Example 2: Traffic Light Control as RL

Goal: Minimize average waiting time across all vehicles

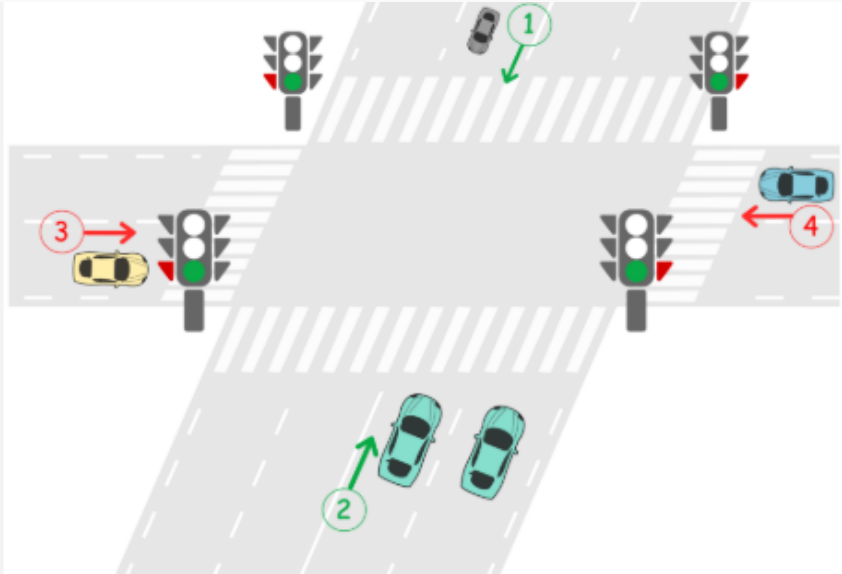


Problem Formulation

- **Agent**: Traffic light controller
- **State**: Queue lengths, waiting times, current signal phase
- **Actions**: Change light phases (green, yellow, red) of each traffic light

Example 2: Traffic Light Control as RL

Goal: Minimize average waiting time across all vehicles

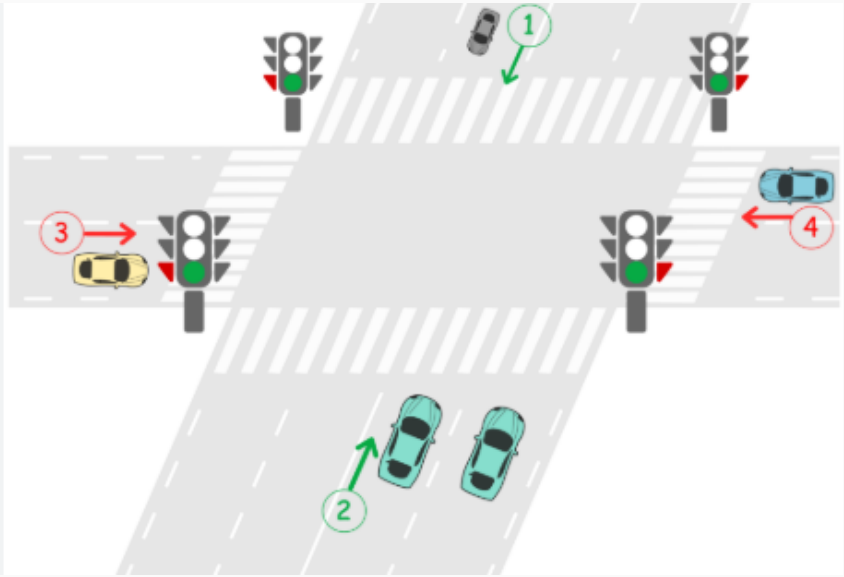


Problem Formulation

- **Agent**: Traffic light controller
- **State**: Queue lengths, waiting times, current signal phase
- **Actions**: Change light phases (green, yellow, red) of each traffic light
- **Reward**: Negative of total waiting time or queue length

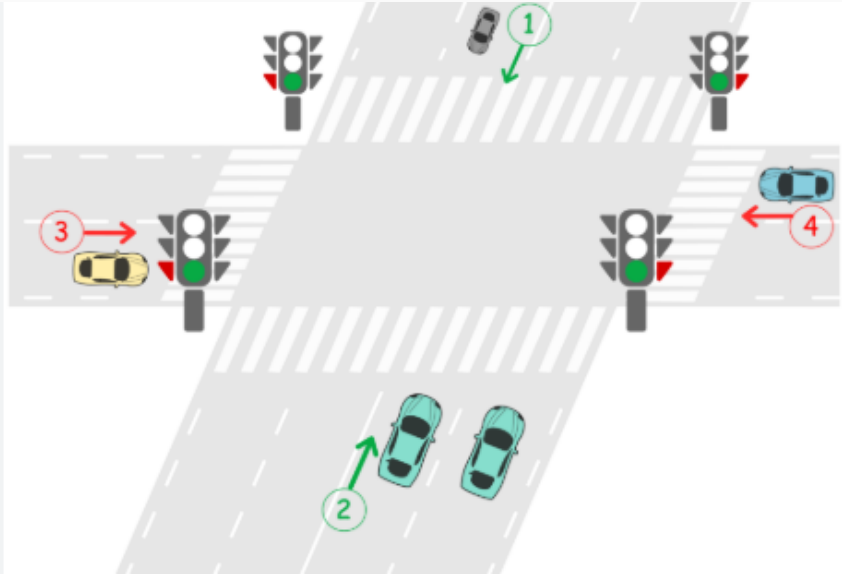
Example 2: Traffic Light Control as RL (Cont'd)

Goal: Minimize average waiting time across all vehicles



Example 2: Traffic Light Control as RL (Cont'd)

Goal: Minimize average waiting time across all vehicles



Reward Formulation

- Negative of total waiting time or queue length
- Big penalty if two opposing lights were green simultaneously
- Penalty for having all lights as red??

RL Solution

RL Solution

Goal of the Agent

Goal of RL Agent in MDP: Optimal Policy

Goal of Agent

- At each time step t, given state s_t and possible actions $a_1, a_2, \ldots$

Goal of RL Agent in MDP: Optimal Policy

Goal of Agent

- At each time step t, given state s_t and possible actions $a_1, a_2, \ldots$
- Decide on an action a_i that will most likely move the agent to state s_{t+1} given $\mathcal{P}$ leading to the maximum reward according to $\mathcal{R}$.

Goal of RL Agent in MDP: Optimal Policy

Goal of Agent

- At each time step t, given state s_t and possible actions $a_1, a_2, \ldots$
- Decide on an action a_i that will most likely move the agent to state s_{t+1} given $\mathcal{P}$ leading to the maximum reward according to $\mathcal{R}$.
- How to formalize this decision process???

Goal of RL Agent in MDP: Optimal Policy

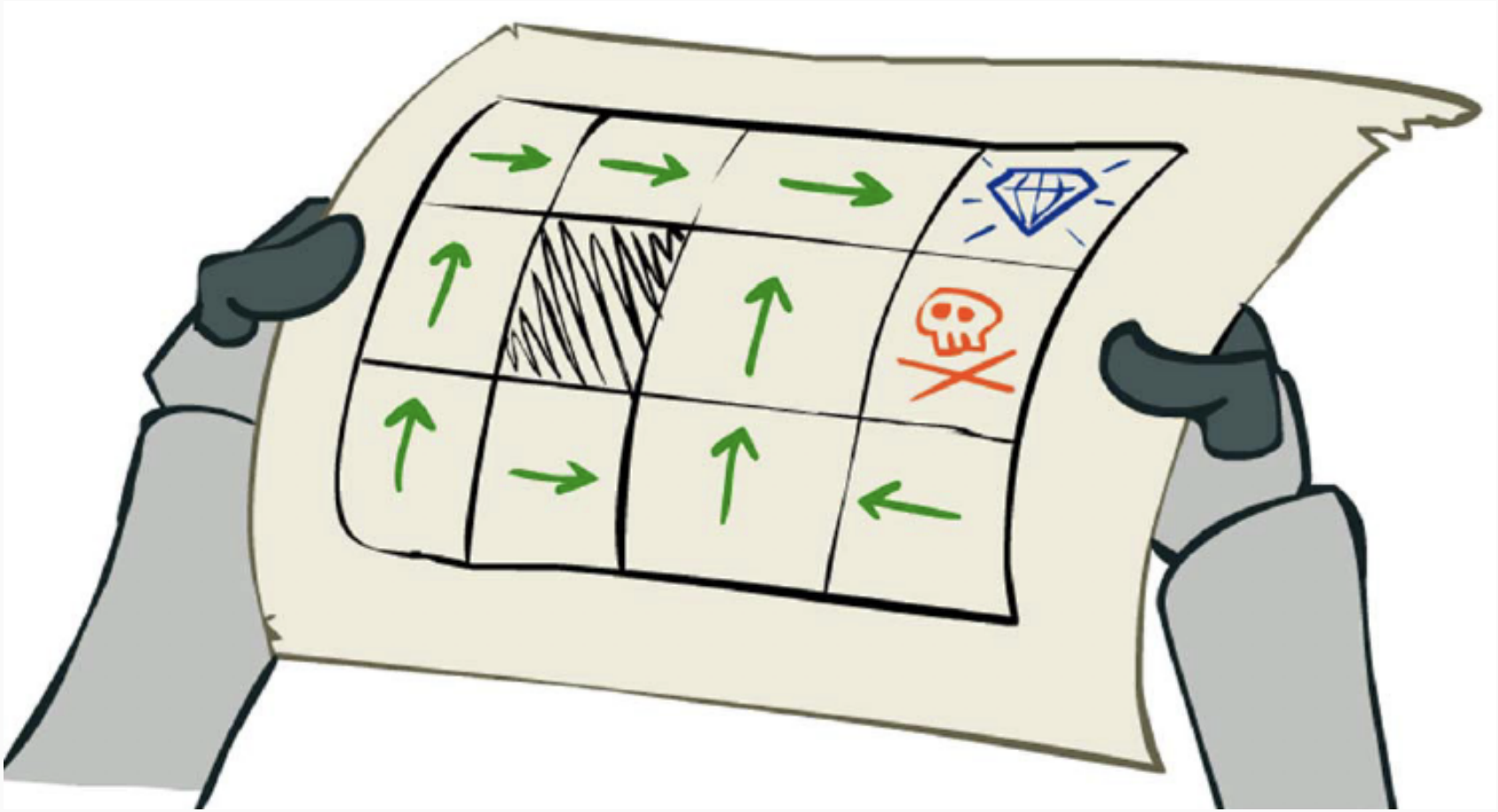
Goal of Agent

- At each time step t, given state s_t and possible actions $a_1, a_2, \ldots$
- Decide on an action a_i that will most likely move the agent to state s_{t+1} given $\mathcal{P}$ leading to the maximum reward according to $\mathcal{R}$.
- How to formalize this decision process???

The decision is dictated by the policy π defined as a mapping from the state space to the action space.

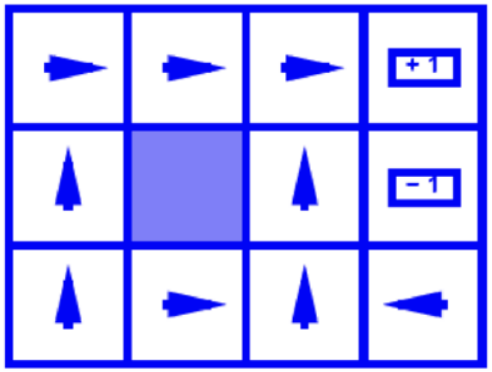
Goal of RL Agent in MDP: Optimal Policy Example

The optimal policy of our previous example would be:

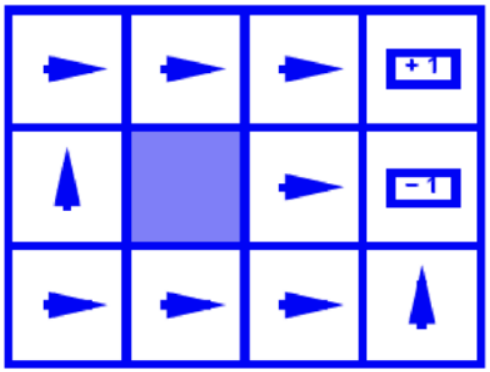


Goal of RL Agent in MDP: Optimal Policy Cont'd

Reward function changed $\mapsto$ New Policy!!

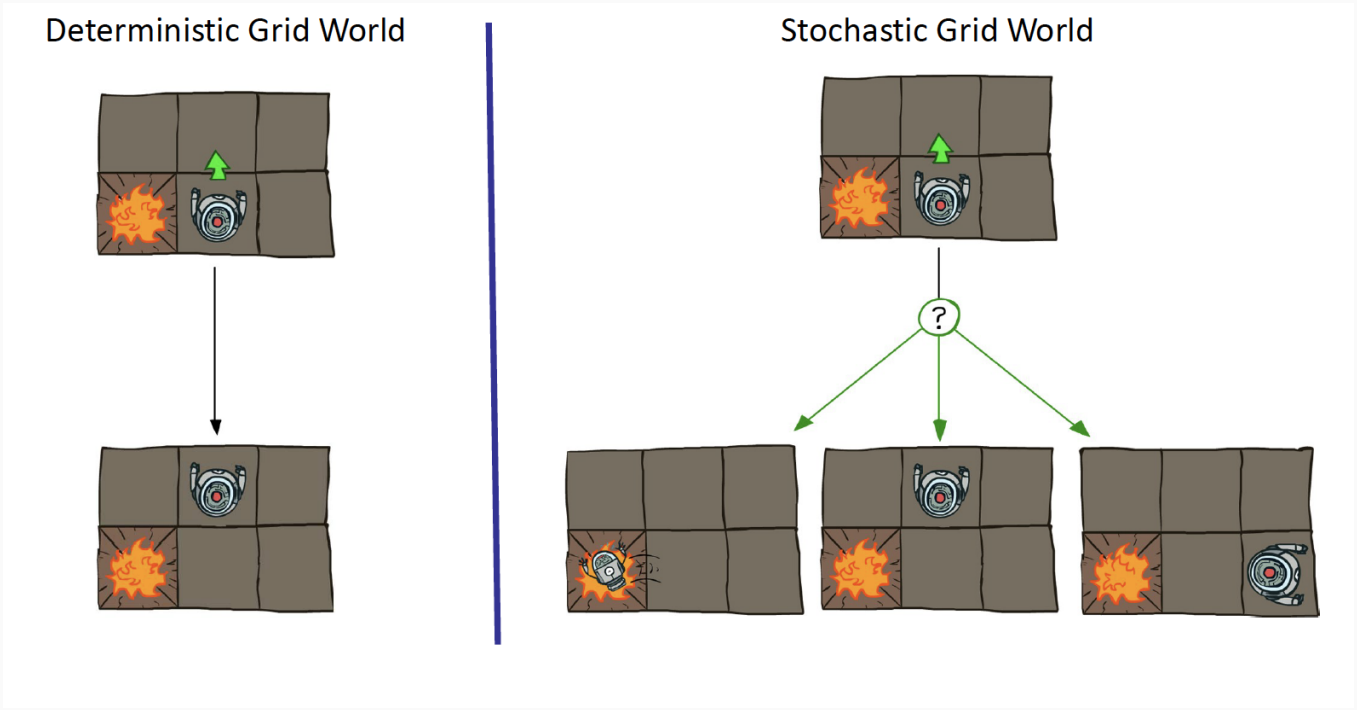


R(s) = -0.4

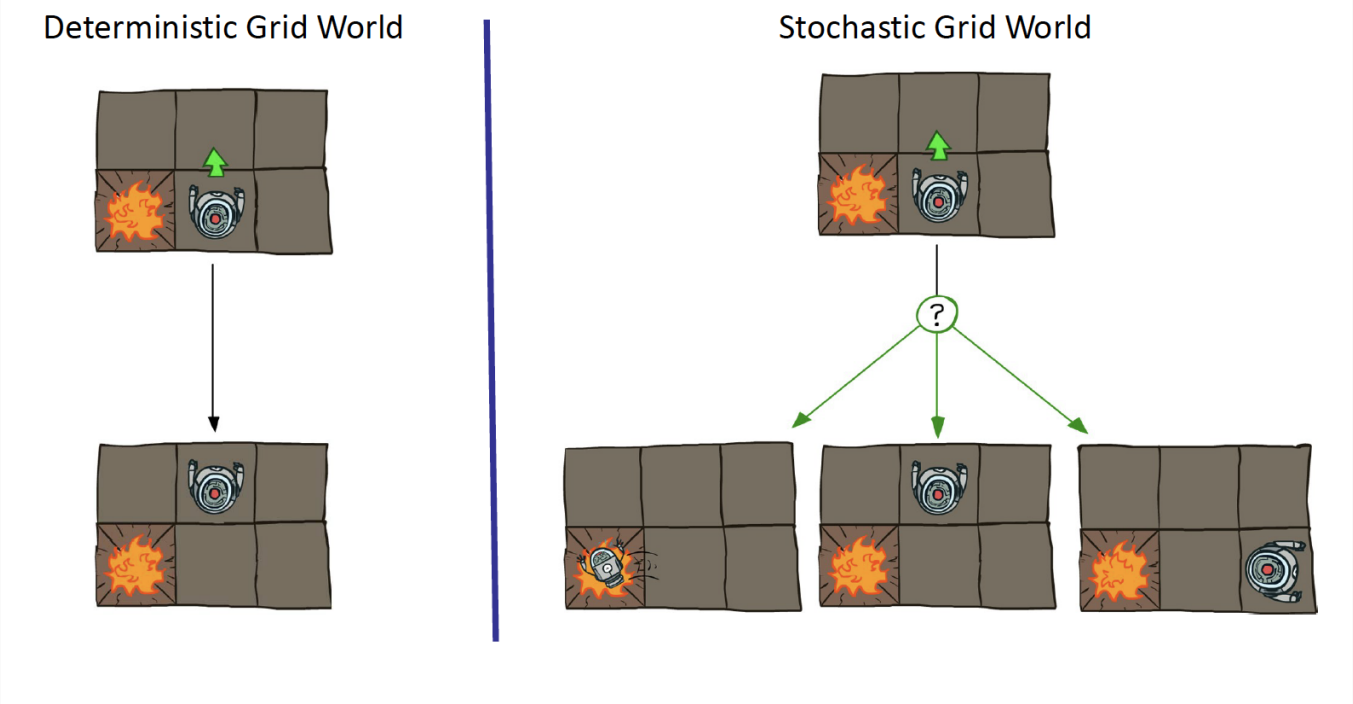


R(s) = -2.0

Goal of RL Agent in Stochastic Environments



Goal of RL Agent in Stochastic Environments



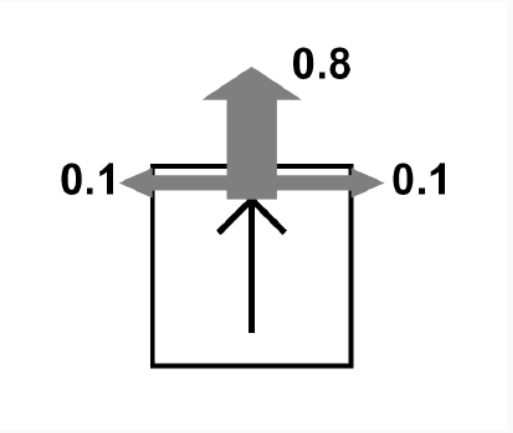
- If the world is ideal, the policy would be a mapping from state to action
- Because of the stochasticity of the environment, we usually deal with non-deterministic policies, i.e. a mapping from a state-action pair to a probability between 0 and 1.

Policy In Stochastic Environment

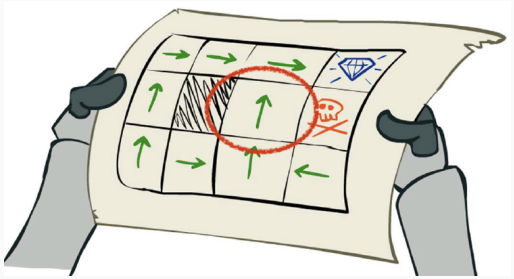
Example

The agent's actions do not always go as planned:

- 80% of the time, the action North takes the agent North (if there is no wall there)
- 10% of the time, North takes the agent West; 10% East
- If there is a wall in the direction the agent would have been taken, the agent stays put

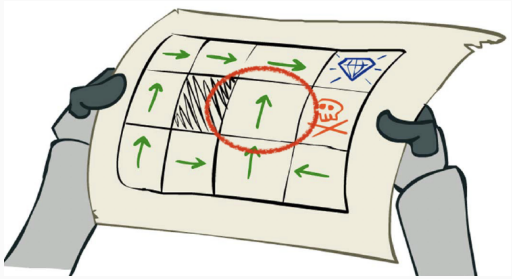


RL Example In Stochastic Environment: Problem



Problem

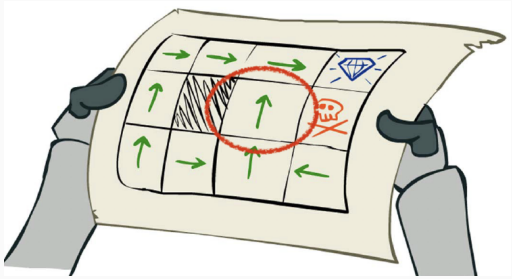
RL Example In Stochastic Environment: Problem



Problem

In state $(2, 3)$ and action "Up",

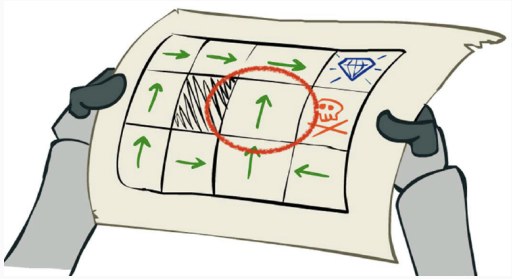
RL Example In Stochastic Environment: Problem



Problem

In state $(2, 3)$ and action "Up", there is a 10% probability that the agent slips and falls in the "fail" state.

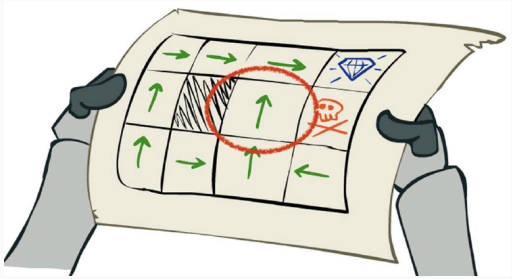
RL Example In Stochastic Environment: Problem



Problem

In state $(2, 3)$ and action "Up", there is a 10% probability that the agent slips and falls in the "fail" state. What's better?

RL Example In Stochastic Environment: Problem

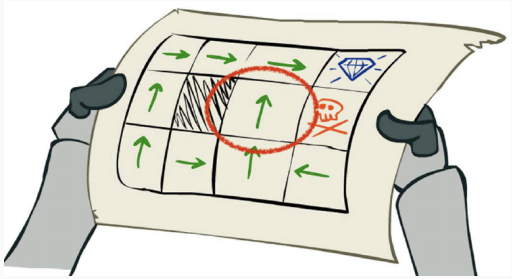


Problem

In state $(2, 3)$ and action "Up", there is a 10% probability that the agent slips and falls in the "fail" state. What's better?

- Go up anyways?

RL Example In Stochastic Environment: Problem

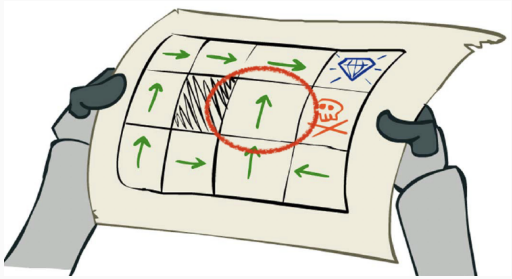


Problem

In state $(2, 3)$ and action "Up", there is a 10% probability that the agent slips and falls in the "fail" state. What's better?

- Go up anyways?
- Go left and stay always at your place?

RL Example In Stochastic Environment: Problem

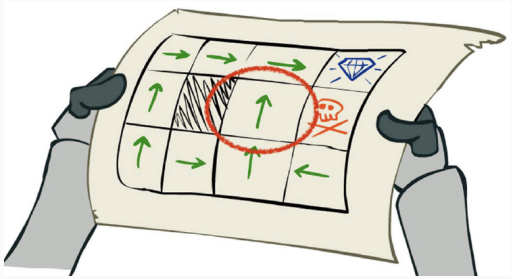


Problem

In state $(2, 3)$ and action "Up", there is a 10% probability that the agent slips and falls in the "fail" state. What's better?

- Go up anyways?
- Go left and stay always at your place?
- Go down and follow the other path to the goal by affording discounts?

RL Example In Stochastic Environment: Problem



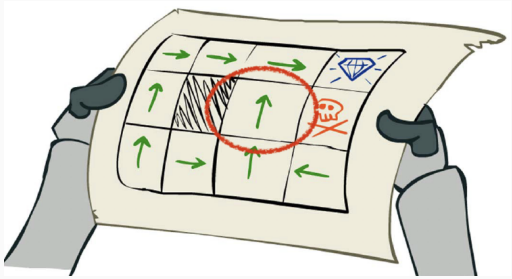
Problem

In state $(2, 3)$ and action "Up", there is a 10% probability that the agent slips and falls in the "fail" state. What's better?

- Go up anyways?
- Go left and stay always at your place?
- Go down and follow the other path to the goal by affording discounts?

What if probability of slipping to the right is 50%?

RL Example In Stochastic Environment: Problem



Problem

In state $(2, 3)$ and action "Up", there is a 10% probability that the agent slips and falls in the "fail" state. What's better?

- Go up anyways?
- Go left and stay always at your place?
- Go down and follow the other path to the goal by affording discounts?

What if probability of slipping to the right is 50%?$\mapsto$ Need a quantization of how good each state is.

RL Algorithms

Solving for Optimal Policy Methods

- The goal is to find the policy that leads to the highest reward.

Solving for Optimal Policy Methods

- The goal is to find the policy that leads to the highest reward.
- Methods:

Solving for Optimal Policy Methods

- The goal is to find the policy that leads to the highest reward.
- Methods:
  - **Value-based:** estimate values of states and go to the state that leads to the highest value (reward).

Solving for Optimal Policy Methods

- The goal is to find the policy that leads to the highest reward.
- Methods:
  - **Value-based:** estimate values of states and go to the state that leads to the highest value (reward).
  - **Policy-based:** directly differentiate a mathematical function

Solving for Optimal Policy Methods

- The goal is to find the policy that leads to the highest reward.
- Methods:
  - **Value-based:** estimate values of states and go to the state that leads to the highest value (reward).
  - **Policy-based:** directly differentiate a mathematical function
  - **Actor-critic:** An agent (Actor) takes an action a according to a policy π and the Critic monitors the reward and informs the actor how to adjust the policy

RL Algorithms

Value-Based Methods

Value-Based Methods: Motivation

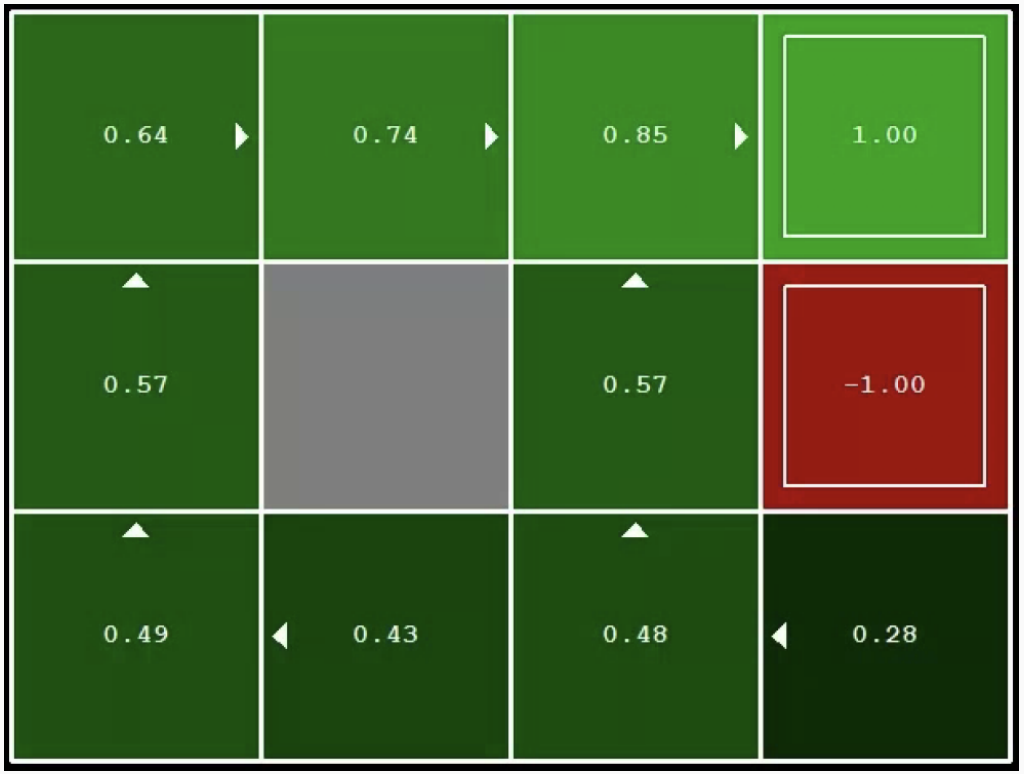


Figure 1: Values of the states

If the agent had a value describing how good a state is, it would always know what action to take.

Value-Based Methods: Definitions

Value-based methods evaluate how good each state is.

The value

- The value of a state represents how good a state is.
- Mathematically, the value of a state s is the expected cumulative reward from following the policy π from state s

$$\underbrace{V^{\pi}(s)}_{\text{Value of states following policy } \pi} = \underbrace{\mathbb{E}}_{\text{Expected value over all possible paths}} \left[\underbrace{\sum_{t \geq 0} \gamma^t r_t}_{\text{Sum of discounted rewards throughout the path}} \underbrace{| s_0 = s, \pi}_{\text{Given that the initial state is } s_0 \text{ and the policy is } \pi} \right]$$

Value-Based Methods: Definitions

The Q-value

We further define, the Q-value of a state s and action a as the expected cumulative reward from taking action a in state s and then following the policy:

$$\underbrace{Q^{\pi}(s,a)}_{\text{Q-value of } (s,a) \text{ pair following policy } \pi} = \underbrace{\mathbb{E}}_{\text{Expected value over all possible paths}} \left[\underbrace{\sum_{t \geq 0} \gamma^t r_t}_{\text{Sum of discounted rewards throughout the path}} \middle| \underbrace{s_0 = s, a_0 = a\pi}_{\text{Given that the initial state is } s_0 \text{ the initial action is } a_0 \text{ and the policy is } \pi} \right]$$

Value-Based Methods: Optimal Policy

The optimal policy π^* corresponds to taking the best action in a state as specified by Q^*.

The optimal Q-value function

The optimal Q-value function Q* is the maximum expected cumulative reward achievable from a given (state, action) pair:

$$\underbrace{Q^*(s,a)}_{\substack{\text{Q optimal of } (s,a) \text{ pair} \\ \text{following policy } \pi}} = \underbrace{\max_{\pi}}_{\substack{\text{Maximum over} \\ \text{possible policies}}} \underbrace{\mathbb{E}}_{\substack{\text{Expected value} \\ \text{over all} \\ \text{possible paths}}} \left[\underbrace{\sum_{t \geq 0} \gamma^t r_t}_{\substack{\text{Sum of} \\ \text{discounted rewards} \\ \text{throughout the path}}} \underbrace{| s_0 = s, a_0 = a\pi}_{\substack{\text{Given that} \\ \text{the initial state is } s_0 \\ \text{the initial action is } a_0 \\ \text{and the policy is } \pi}} \right]$$

Value Iteration Algorithm

Value Iteration

- Start with $V_0^*(s) = 0$ for all the states s.
- Iterate for H times ($i = 1, \ldots, H$)
  - Given V_i^*, for all the states, compute:

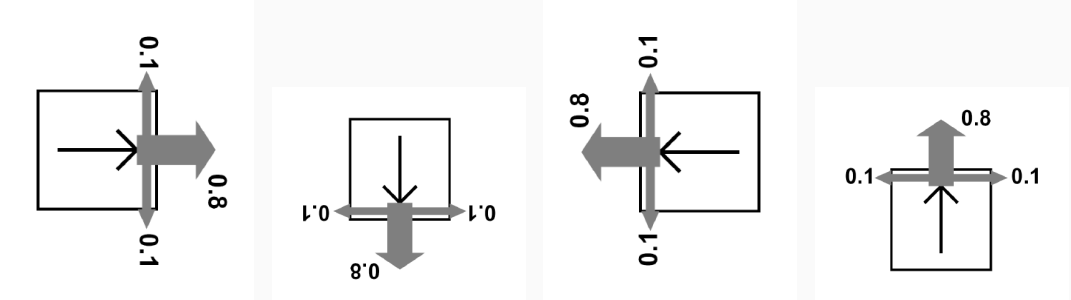
$$\underbrace{V_{i+1}^*(s)}_{\substack{\text{Value of } s \\ \text{at iteration } i+1}} = \max_a \times \underbrace{\sum_{s'}}_{\substack{\text{sum over possible next states } s'}} \underbrace{\mathcal{P}(s,a,s')}_{\substack{\text{of the probability of ending in } s'}} \left[\underbrace{R(s,a,s')}_{\substack{\text{reward of the transition}}} + \gamma \underbrace{V_i^*(s')}_{\substack{\text{value of } s' \\ \text{computed at iteration } i}} \right] \quad (1)$$

- V_i will converge to V^* when $i \mapsto \infty$
- This is called a value update or Bellman update/back-up

Value Iteration Algorithm: Example

Example: Same grid

- Noise $= 0.2$: if agent executes action a, such as go up
  - with probability 80% it will end up in the desired destination
  - with probability 10% it will move to a neighbouring cell
  - with probability 0% it will end up in the opposite direction.
- If agent hits a wall or boundary, it stays in the original position.
- Discount $\gamma = 0.9$
- Agent receives a zero reward for every move unless it hits the terminal state which is $+1$ for the green and -1 for the red spot.

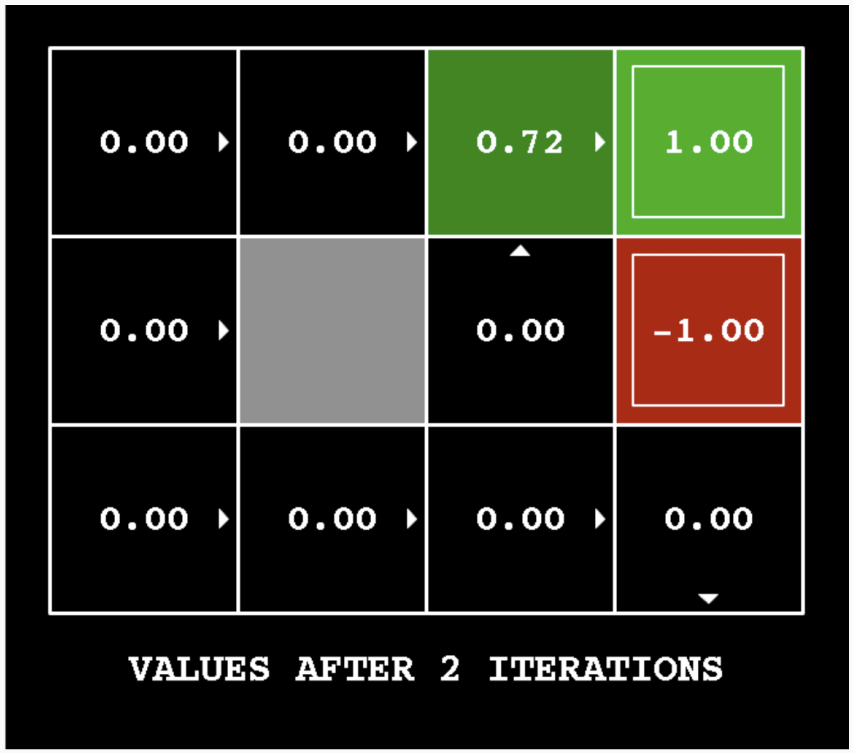


Value Iteration Algorithm: Example (Iteration 1)

$$V^*_{i+1}(s) = \max_a \times \sum_{s'} \mathcal{P}(s, a, s') \left[R(s, a, s') + \gamma V^*_i(s') \right]$$

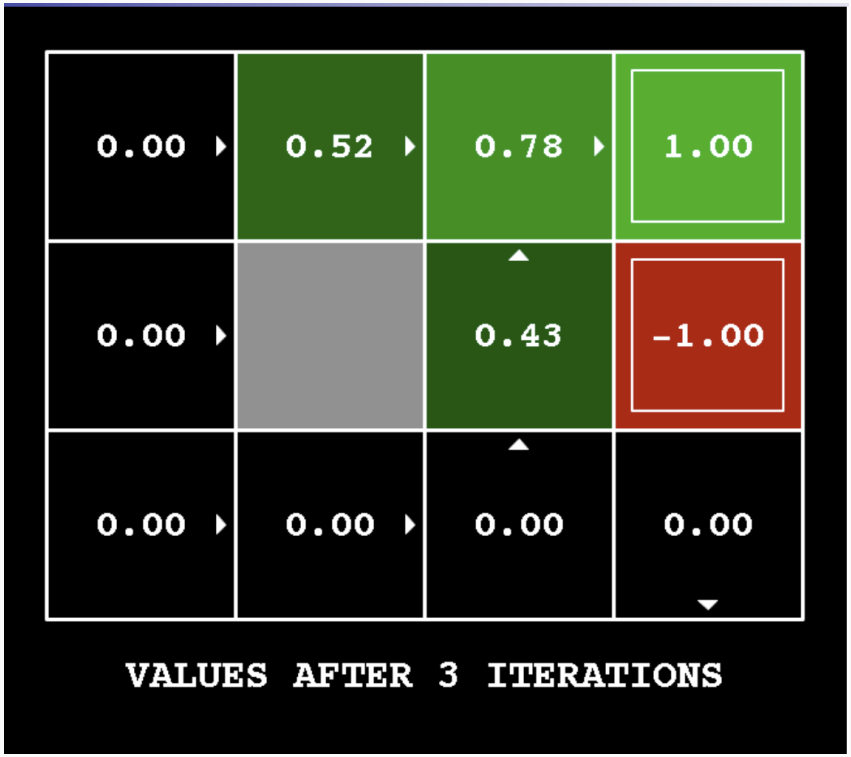
Value Iteration Algorithm: Example (Iteration 2)

$$V^*_{i+1}(s) = \max_a \times \sum_{s'} \mathcal{P}(s, a, s') \left[R(s, a, s') + \gamma V^*_i(s') \right]$$



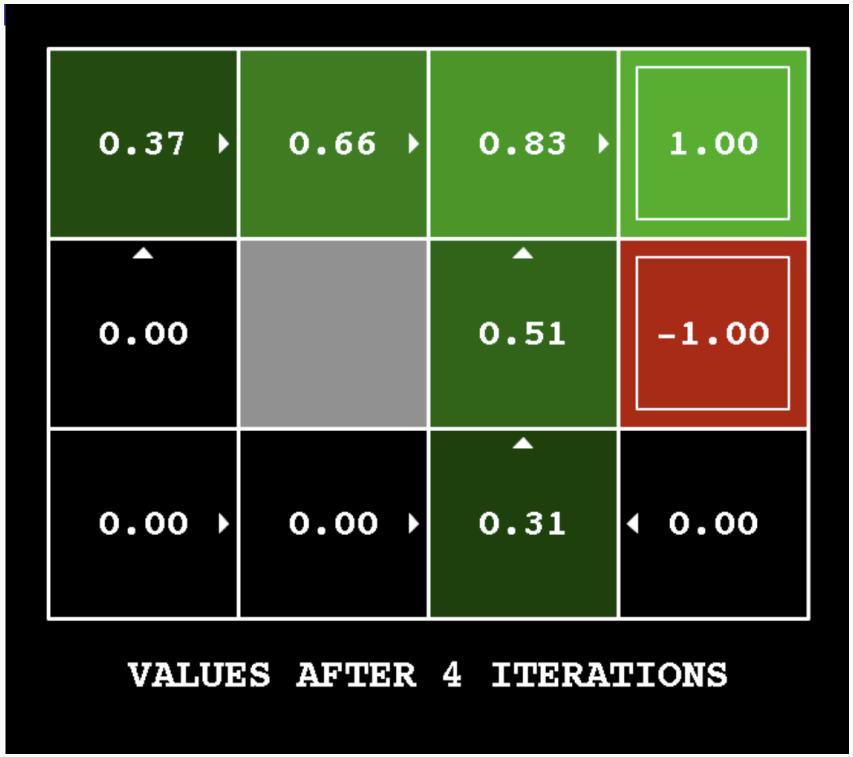
Value Iteration Algorithm: Example (Iteration 3)

$$V^*_{i+1}(s) = \max_a \times \sum_{s'} \mathcal{P}(s, a, s') \left[R(s, a, s') + \gamma V^*_i(s') \right]$$



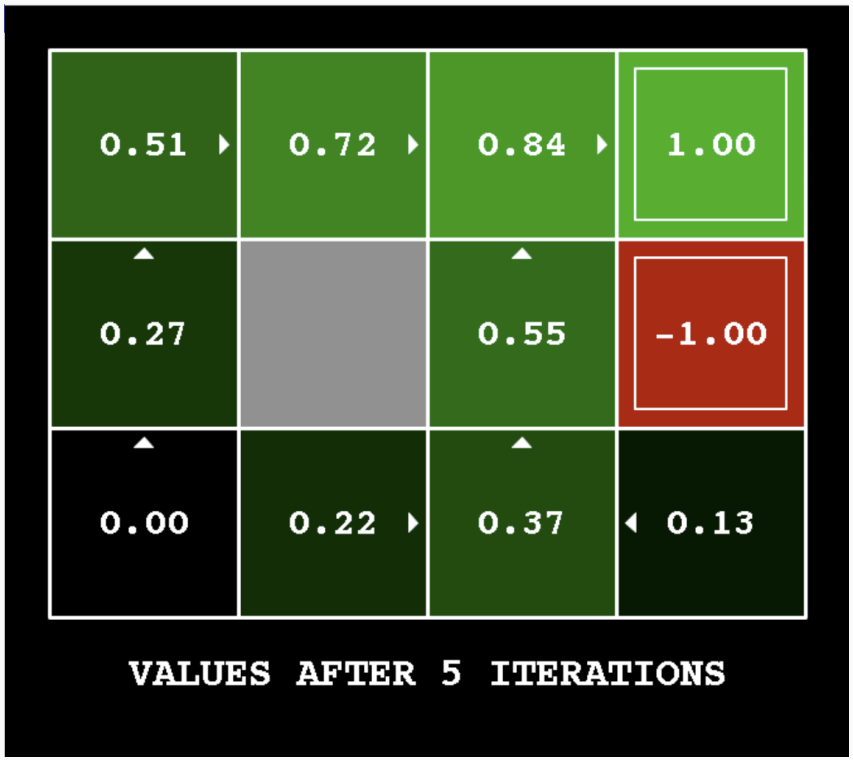
Value Iteration Algorithm: Example (Iteration 4)

$$V^*_{i+1}(s) = \max_a \times \sum_{s'} \mathcal{P}(s, a, s') \left[R(s, a, s') + \gamma V^*_i(s') \right]$$



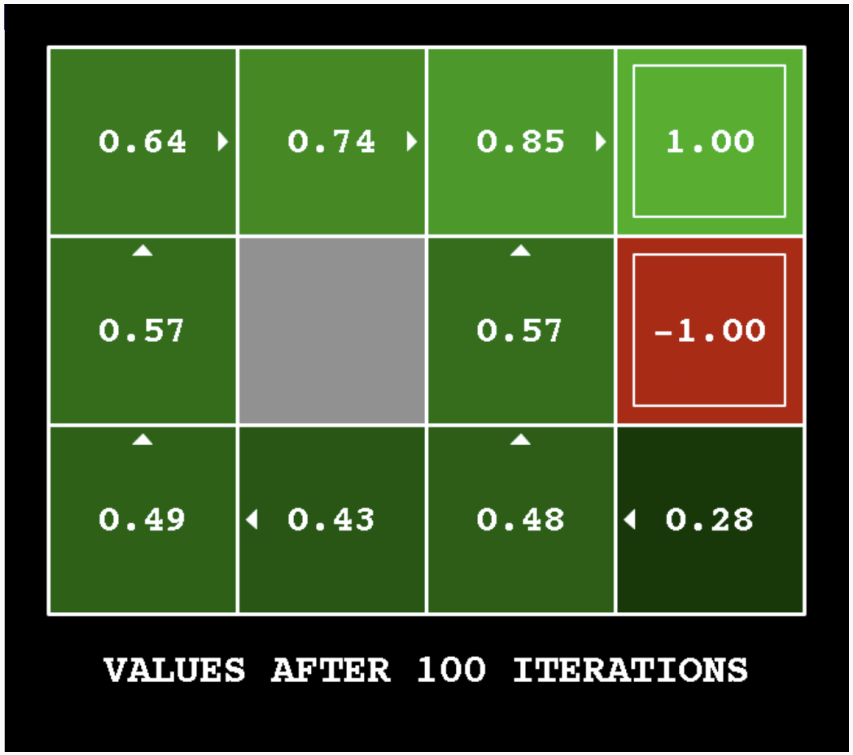
Value Iteration Algorithm: Example (Iteration 5)

$$V^*_{i+1}(s) = \max_a \times \sum_{s'} \mathcal{P}(s, a, s') \left[R(s, a, s') + \gamma V^*_i(s') \right]$$

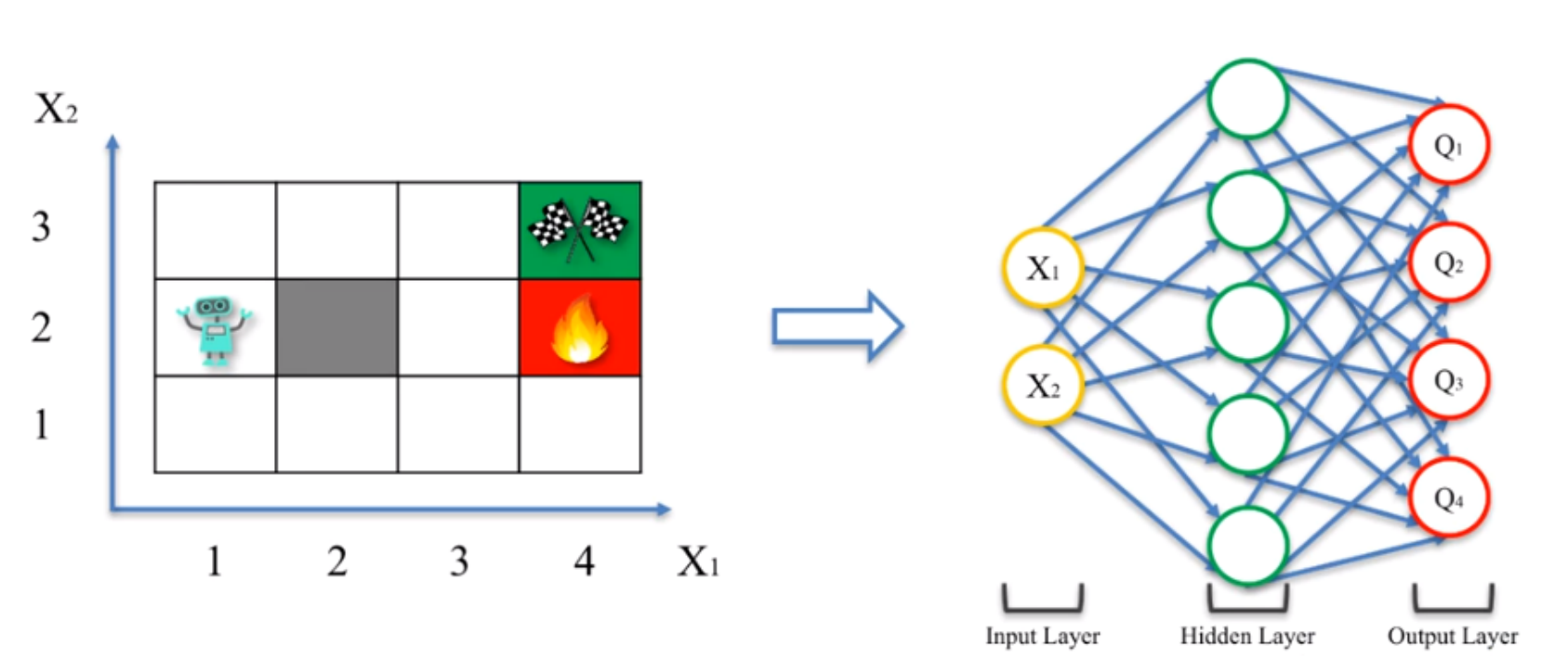


Value Iteration Algorithm: Example (Iteration 100)

$$V^*_{i+1}(s) = \max_a \times \sum_{s'} \mathcal{P}(s, a, s') \left[R(s, a, s') + \gamma V^*_i(s') \right]$$



Deep Q-Networks



RL Algorithms

Policy Gradient Methods

Policy Gradient Methods: Policy Search

- Learn policies that maximize rewards, not the values that predict them

Policy Gradient Methods: Policy Search

- Learn policies that maximize rewards, not the values that predict them
- Directly differentiate the RL objective

Policy Gradient Methods: Policy Search

- Learn policies that maximize rewards, not the values that predict them
- Directly differentiate the RL objective
- Search space is infinite $\mapsto$ Use parametrized policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{a}_t)$ where $\boldsymbol{\theta} \in \mathbb{R}^d$

Policy Gradient Methods: Policy Search

- Learn policies that maximize rewards, not the values that predict them
- Directly differentiate the RL objective
- Search space is infinite $\mapsto$ Use parametrized policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{a}_t)$ where $\boldsymbol{\theta} \in \mathbb{R}^d$
- θ could be control parameters for example
- Goal becomes to find *theta* taht maximize $\sum R_{\pi_\theta}$

RL Algorithms

RL Challenges

Challenges in Advancing Reinforcement Learning

- **Scalability**: Extending RL to large, complex, real-world problems.

Challenges in Advancing Reinforcement Learning

- **Scalability**: Extending RL to large, complex, real-world problems.
- **Safety and Robustness**: Ensuring reliable performance under uncertainty.

Challenges in Advancing Reinforcement Learning

- **Scalability**: Extending RL to large, complex, real-world problems.
- **Safety and Robustness**: Ensuring reliable performance under uncertainty.
- **Interpretability**: Understanding and trusting agent behavior.

Challenges in Advancing Reinforcement Learning

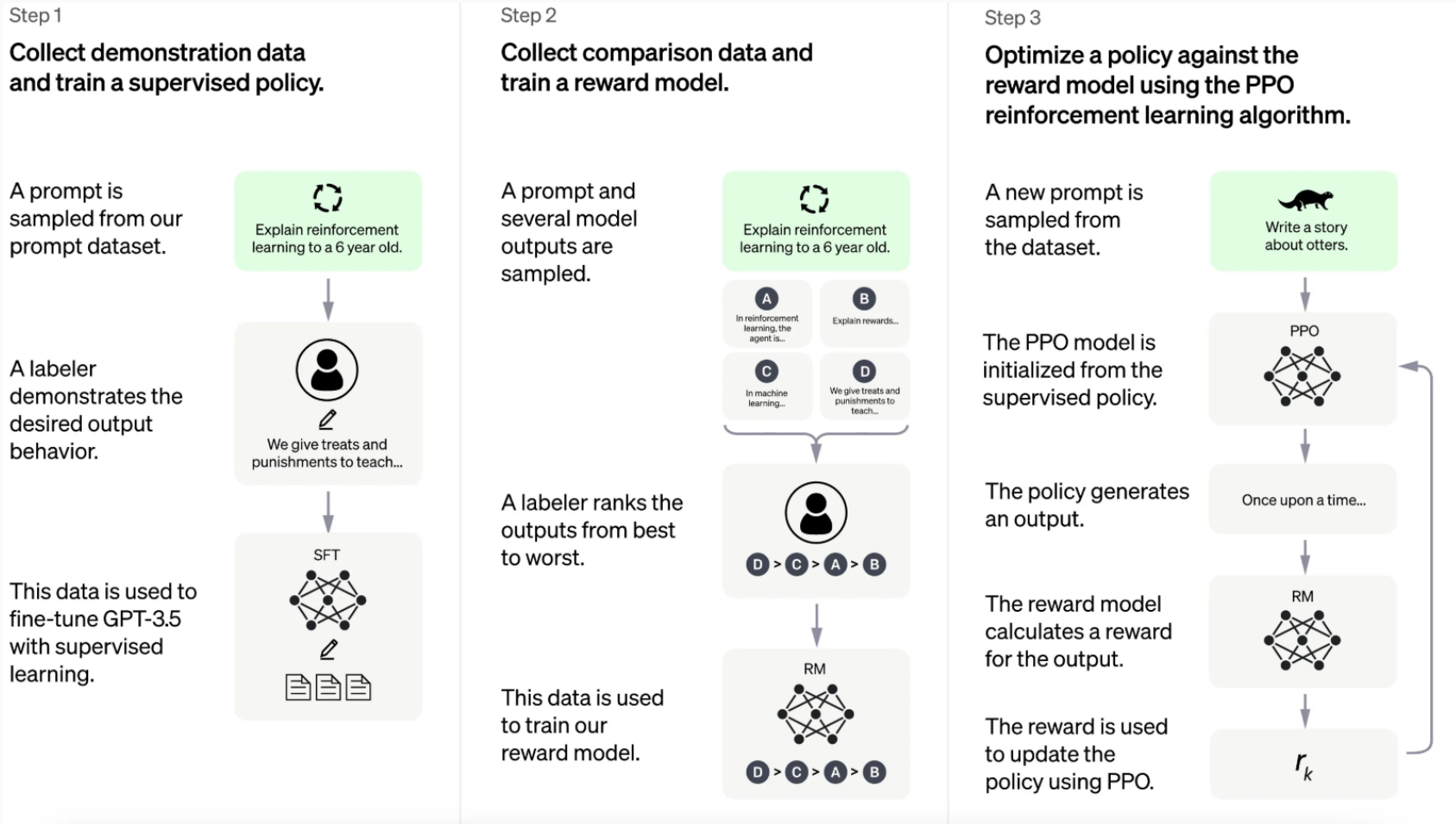
- **Scalability**: Extending RL to large, complex, real-world problems.
- **Safety and Robustness**: Ensuring reliable performance under uncertainty.
- **Interpretability**: Understanding and trusting agent behavior.
- **Reward Engineering**: Designing appropriate reward functions remains non-trivial.

Challenges in Advancing Reinforcement Learning

- **Scalability**: Extending RL to large, complex, real-world problems.
- **Safety and Robustness**: Ensuring reliable performance under uncertainty.
- **Interpretability**: Understanding and trusting agent behavior.
- **Reward Engineering**: Designing appropriate reward functions remains non-trivial.
- **Multi-Agent Coordination**: Learning in environments with multiple interacting agents.

RL in generative AI

RLHF: a Breakthrough in LLM Training



RLHF Challenges

- Collecting quality human feedback
  - Modeling human preferences is difficult due to their fluidity, context dependence, and complexity.
  - Selecting a representative sample of annotators who can provide consistent and accurate feedback is challenging.
- Accurate learning of a reward model from human feedback
  - Evaluating reward models is difficult and expensive, especially when the true reward function is unknown.
  - Reward hacking can occur, leading to unintended consequences and undesirable behavior.
- Optimizing AI policy using the imperfect reward model
  - RLHF trained models can suffer from issues such as hallucinations, bias, and susceptibility to adversarial attacks.
  - Fine-tuning reduces the diversity of samples produced by a model, leading to "mode collapse".

Newer Alternatives to RLHF

Direct Preference Optimization (DPO)

- Learns directly from human preference pairs without training a reward model.
- Optimizes a simple supervised loss to prefer better responses.
- Removes the complexity of reward model of RLHF + faster, simpler, and more stable.

Reinforcement Learning from AI Feedback (RLAIF)

- Trains a reward model using AI-generated labels instead of human rankings.
- Feedback is simulated using stronger models or ensembles.
- Greatly reduces human labeling costs of RLHF and scales up feedback generation.

Even More: Emerging Methods

Implicit Preference Optimization (IPO)

- Learns from implicit signals (e.g., user clicks, engagement) rather than explicit ranking data.
- Trains preferences in a "self-supervised" style.
- Makes it possible to adapt models from live user behavior without costly annotation.

KL-Regularized Training Objectives (KTO)

- Adds a KL divergence penalty to optimization to keep the model close to its original supervised policy.
- Can be used during PPO, DPO, or other preference-based learning.
- Prevents catastrophic model drift; balances reward-seeking behavior and model safety.

RL Implementation

How to Approach RL Implementation

- **1. Environment:**
  - **Use an existing environment** (e.g., Gym, PettingZoo, DM Control).
  - **Or build your own environment** (define states, actions, rewards).

- **2. Algorithm:**
  - **Use an off-the-shelf algorithm** (e.g., DQN, PPO, SAC from libraries like Stable-Baselines3).
  - **Or code your own algorithm** (for deeper understanding or research).

In short: Choose an environment & choose (or build) an agent!

Best Tools for Reinforcement Learning Implementation i

Environments

- **OpenAI Gym** (now *Gymnasium*): Standard benchmark environments for RL.
- **PettingZoo**: Multi-agent environment library.
- **DM Control Suite**: Continuous control tasks by DeepMind.

Libraries:

- **Stable-Baselines3**: Reliable PyTorch implementations of popular RL algorithms.
- **RLlib** (Ray): Scalable, distributed RL library.
- **CleanRL**: Minimal, single-file PyTorch RL implementations.

Best Tools for Reinforcement Learning Implementation ii

Visualization and Debugging:

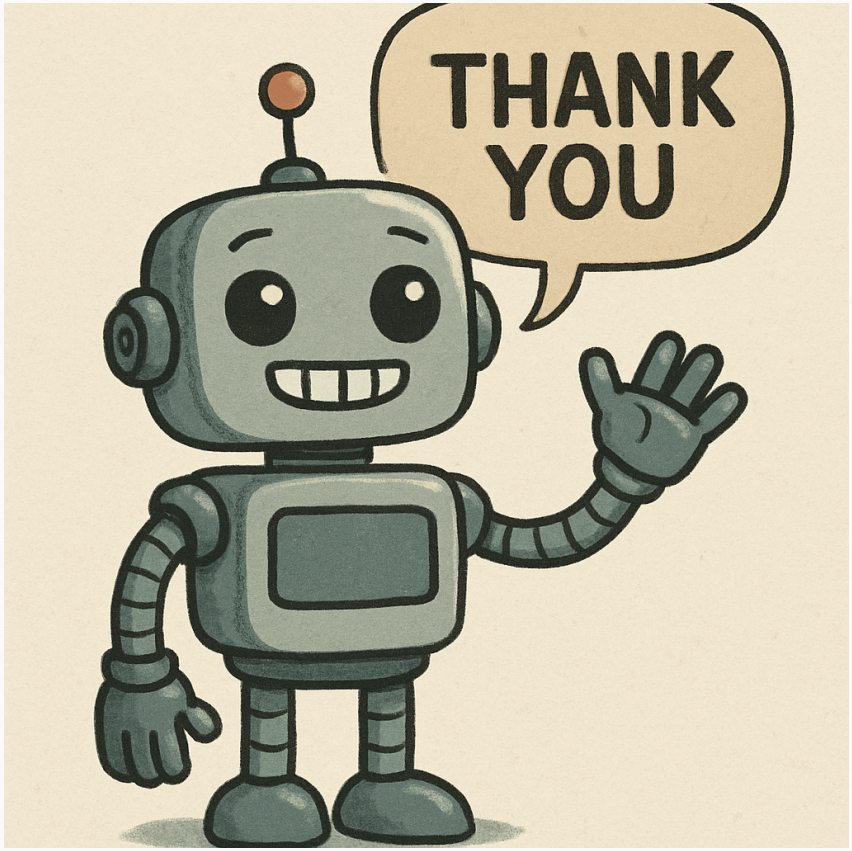
- **TensorBoard**: Monitor and visualize training metrics.
- **Weights & Biases**: Experiment tracking and visualization.

Bonus:

- **Hugging Face Hub for RL**: Pretrained RL models and model sharing.

Today...

- Introduction to OpenAI gymnasium library
- Q-learning Implementation



Questions?

https://www.linkedin.com/in/juliaelzini/