CS391L: Machine Learning

Spring 2025

An Introduction to Reinforcement Learning

Nived Rajaraman

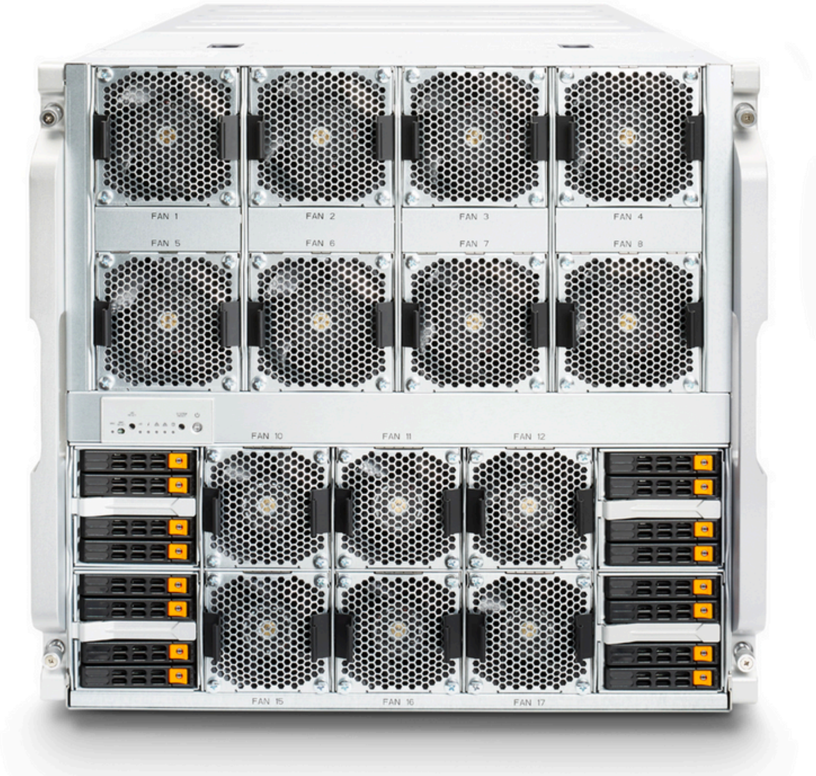
UC Berkeley → Postdoc@MSRNYC

Machine learning vs. Reinforcement learning

DALL·E 2

Learning patterns in a static dataset
Theoretical analog: “Learning from iid data”

Collect a dataset

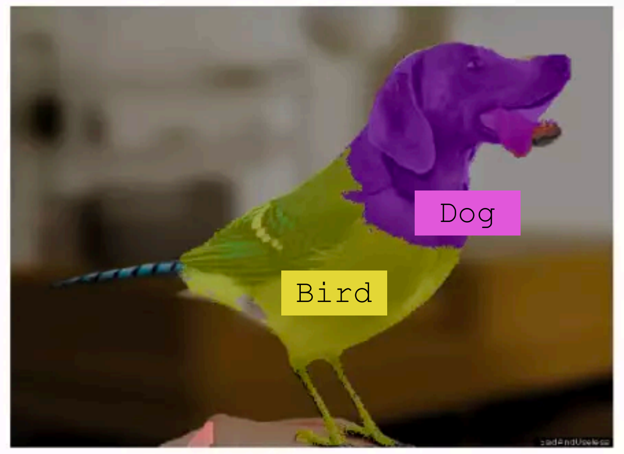


Train on the dataset

Deploy

Input:

Output:



Implementation pipeline

Hope for generalization

Machine learning vs. Reinforcement learning

"Learning interspersed with decision making"

How do we formalize this?

Collect data

Many more questions!

How are decision making and learning connected?

environment

"Generalization?"

Implementation pipeline

Today’s lecture
An introduction to RL

Part 1: A theoretical formulation

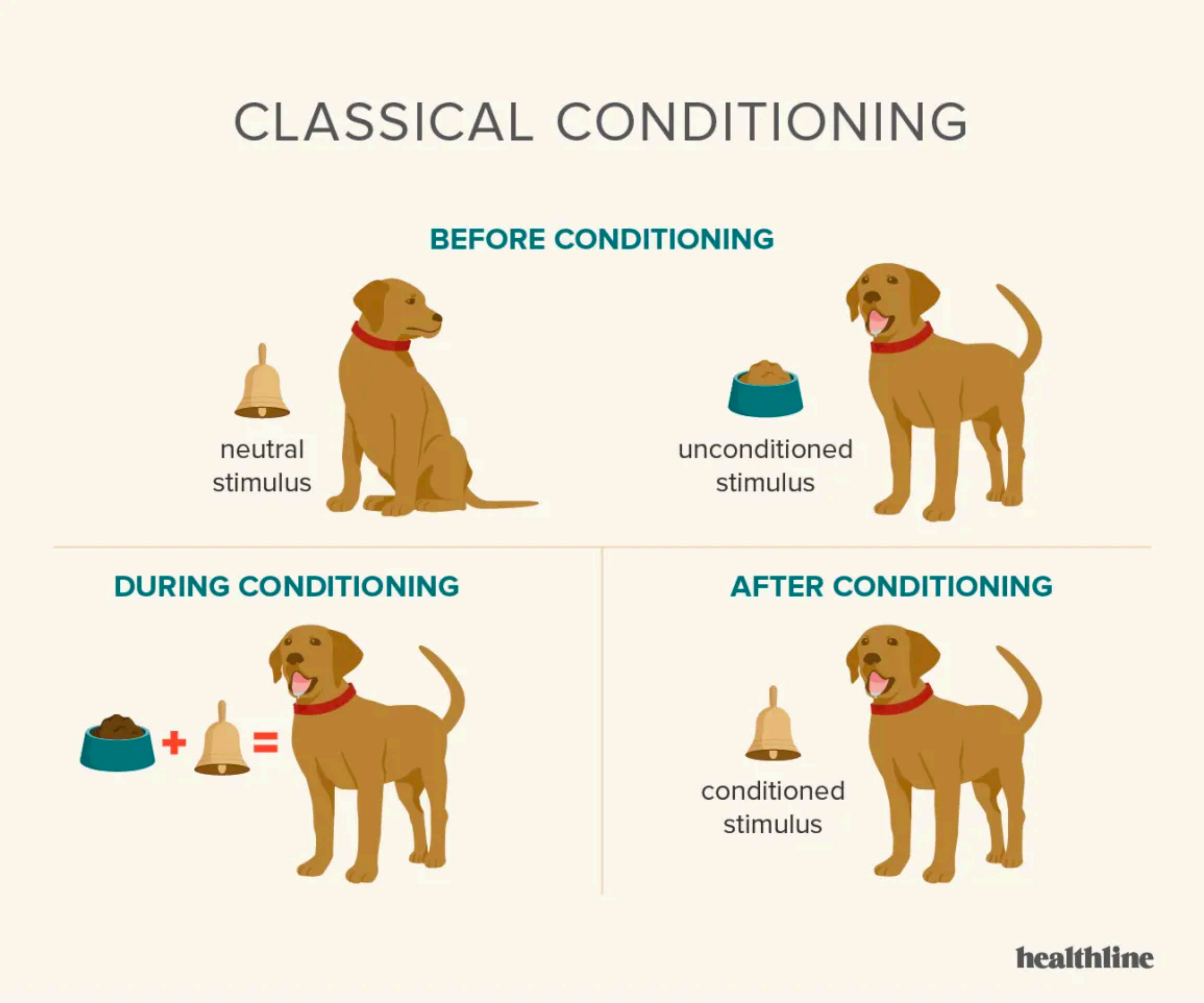
- Markov Decision Processes (MDPs)
- How do we define “optimal” behavior?

Part 2: Reinforcement Learning in the wild

- Practical challenges in RL
- Exploration vs. Exploitation

Part 3: Conclusion

Reinforcement learning formalism: Pavlov's experiment



Learning by conditioning or "positive reinforcement"
The dog's behavior is inherently reward (food) driven

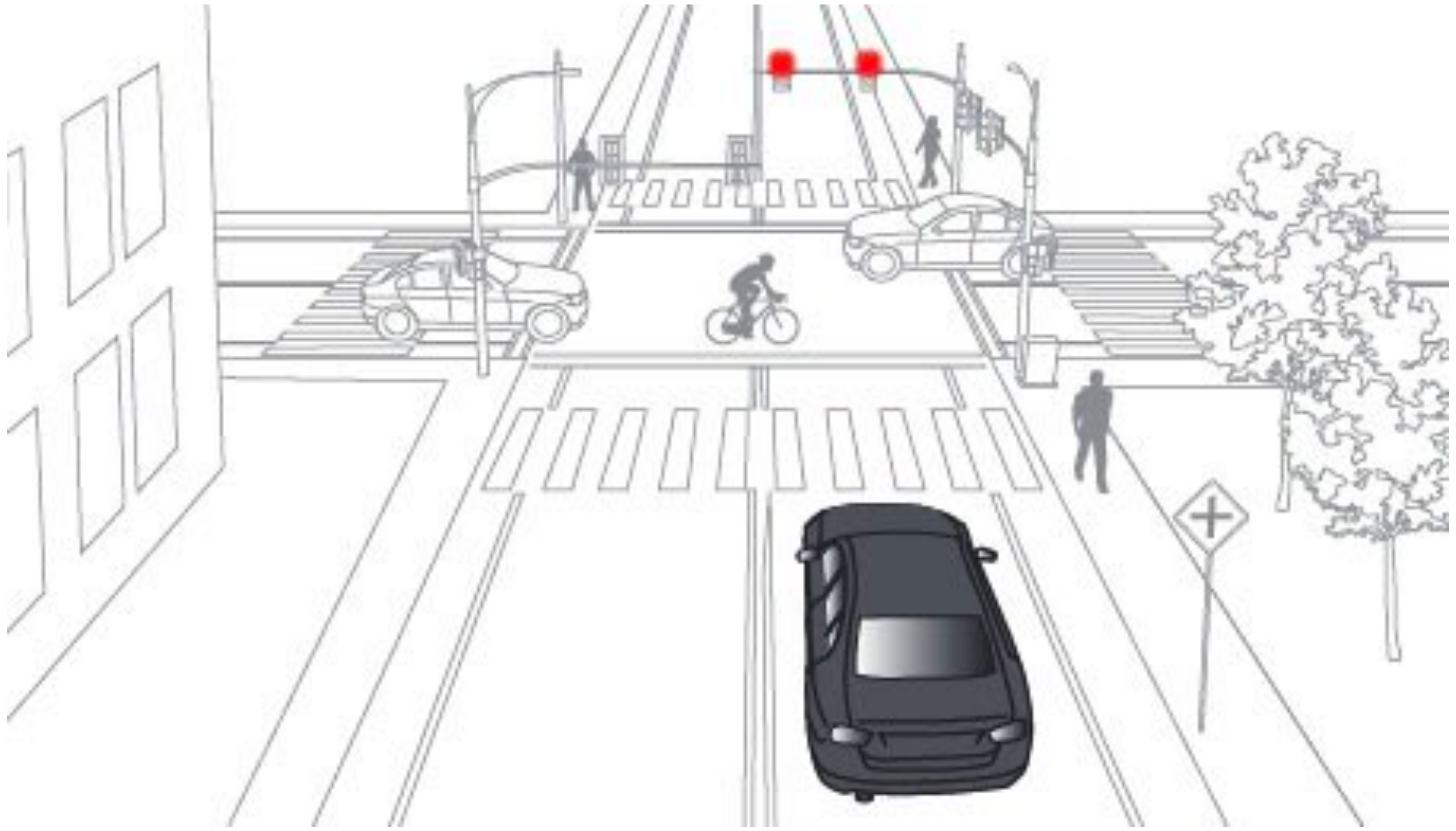
Reinforcement learning formalism: reward maximization

Machine Learning
"Train model to minimizing loss"

Reinforcement Learning
"Train model to maximize reward"

Hypothesis: Reward governs desirable behavior
Do you agree with this?

For an autonomous car:
$+1/-1$ for stopping/not stopping at the stop sign
-1000 for hitting a pedestrian



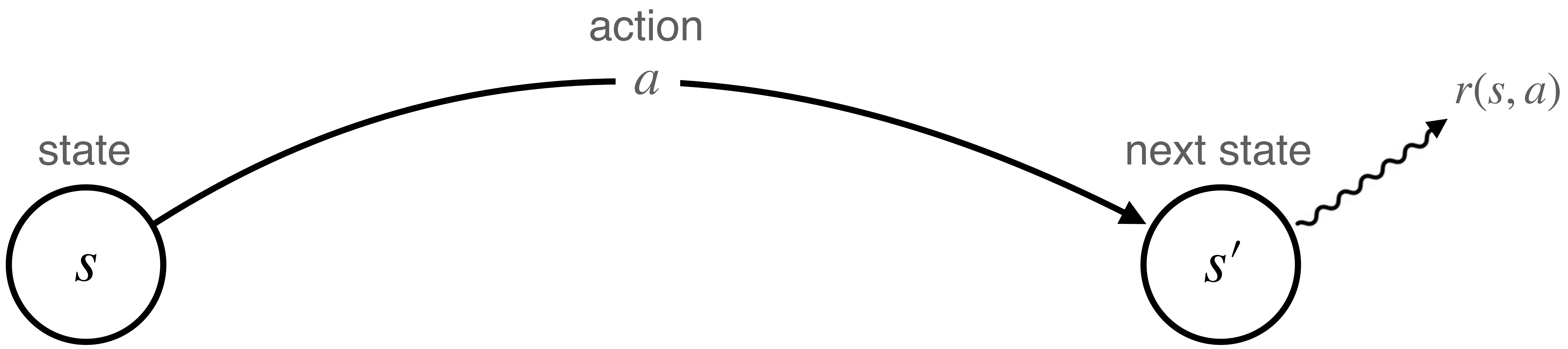
Reinforcement learning formalism: An example

Reward = +0.99

Action

State

Reinforcement learning formalism: Markov Decision Processes

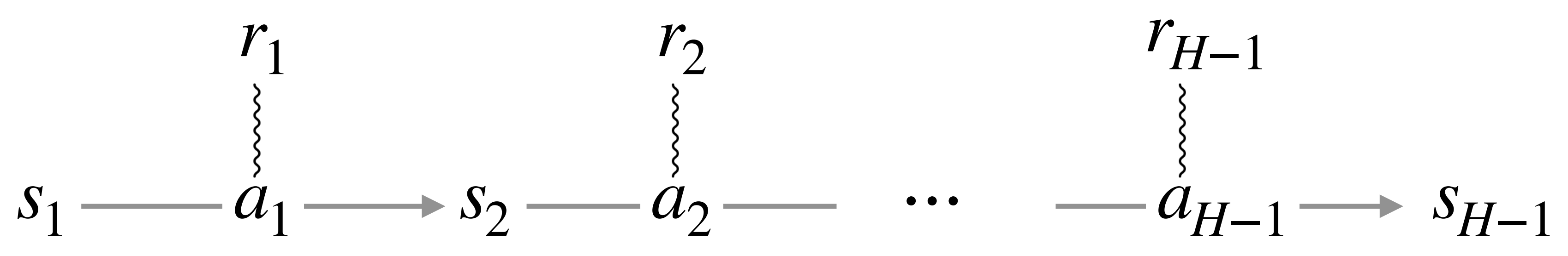


- Learner is initialized in a state s sampled from an *initial state distribution* ρ over states S.

- Picking an action a transitions the learner to a new state s' sampled from the distribution $P(s' \mid s, a)$ depending on the current state and action chosen.

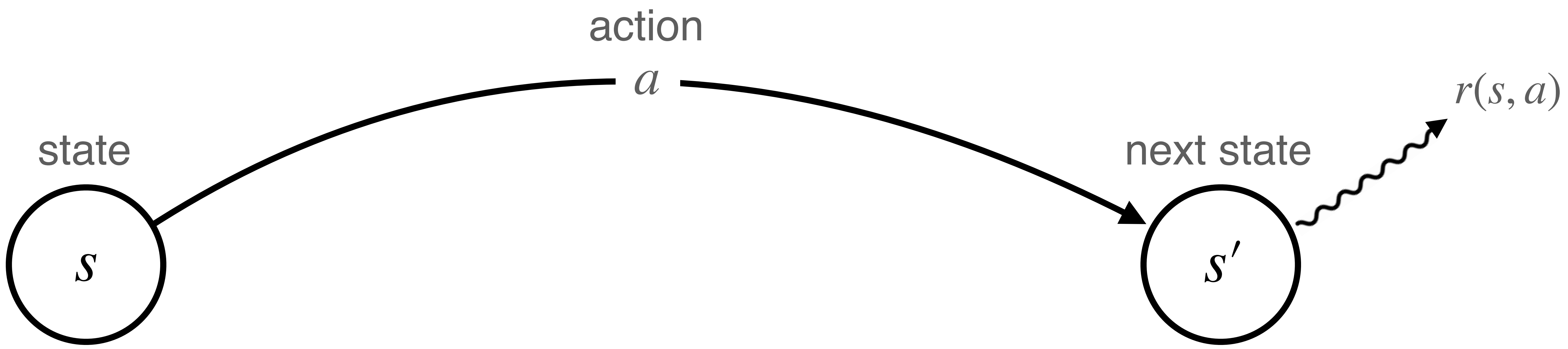
- Learner observes a reward $r(s, a)$ for picking the action a at state s.

- Repeat this process H times (an "episode")



a trajectory

Reinforcement learning formalism: Markov Decision Processes



Learner's defines **a policy** (π): Distribution over actions to play at a time.

$\pi_t(a \mid s, \text{History}_{t-1})$: probability learner picks action a at state s at time t
given the history $\text{History}_{t-1} = \{(s_1, a_1, r_1), \cdots, (s_{t-1}, a_{t-1}, r_{t-1})\}$

Learner's objective: To find a policy which maximizes the ***value:*** the expected total reward.

$$V(\pi) = \mathbb{E}\left[\sum_{t=1}^{H} r_t(s_t, a_t) \,\middle|\, \pi\right]$$

Expectation is over the random trajectory $\{s_1, a_1, s_2, a_2, \cdots, s_H\}$ by "rolling-out" π

A theory of RL

What can we tell about the optimal policy?

$$V(\pi) = \mathbb{E}\left[\sum_{t=1}^{H} r_t(s_t, a_t) \,\middle|\, \pi\right]$$

Optimal policy is the one which maximizes the expected cumulative reward,

$$\pi^\star \in \arg\max_{\pi} V(\pi)$$

Theorem 1. There exists an optimal policy $\pi^\star(a \mid s, \text{History}_{t-1})$ which is not a function of, History_{t-1}.
$\pi_t^\star(a \mid s)$ is Markovian: picks actions only based on the current (s, t).

Example: It doesn't matter how you got to the current board state while playing chess. Playing the best move only depends on the current board state, not the moves played to get there.

Reinforcement learning formalism: discounted MDPs

In practice, often *discounted / infinite horizon* MDPs are considered.
Here, the objective is to maximize the discounted value,

$$V(\pi) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r(s_t, a_t) \middle| \pi \right]$$

The rewards are summed up with geometric decay, γ^{t-1} where γ is the discount factor.

$H_{\text{eff}} = \dfrac{1}{1-\gamma}$ is the "effective horizon". Rewards become insignificant after roughly H_{eff} timesteps (analog of H)

Rest of the lecture focuses on the discounted setting (results can be extended to the episodic setting as well)

Q functions

These will be useful in characterizing the optimal policy.

Q function:
"If I am at state s and played action a thereafter, how much reward would I collect if I continued playing after that?"

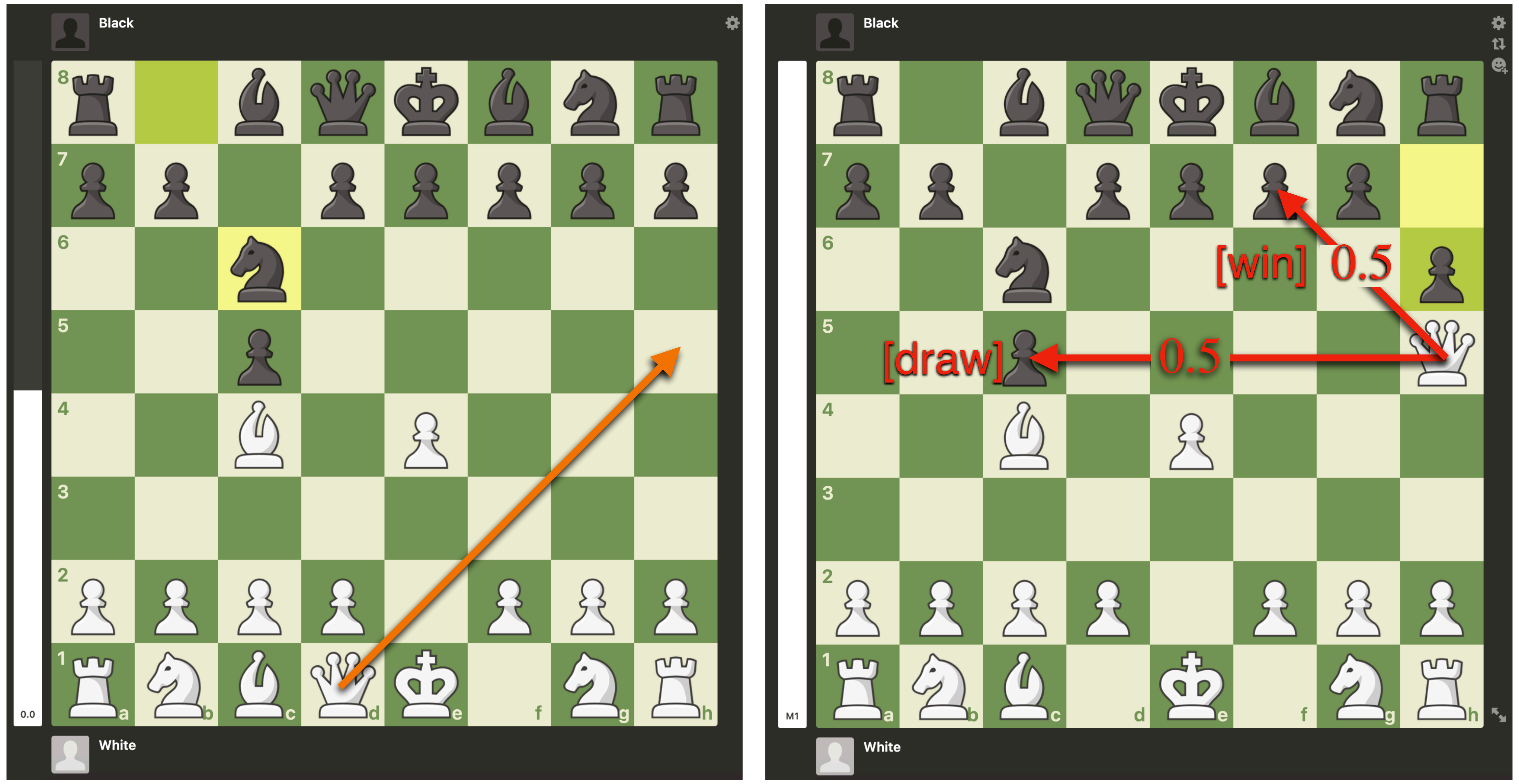
Definition 1. (Q-function): Expected discounted reward starting from the state s, playing the action a and rolling out π subsequently,

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r(s_t, a_t) \,\middle|\, \pi, s_1 = s, a_1 = a\right]$$

Q functions

These will be useful in characterizing the optimal policy.

In this $s =$ board state, playing the $a =$ red move action results in a checkmate. $Q^{\pi}(s, a) = 1$



Result of $s =$ board state, playing $a =$ orange move **depends on future moves.**

$$Q^{\pi}(s, a) = 0.5 \times 1 + 0.5 \times 0 = 0.5$$

Value functions

These will be useful in characterizing the optimal policy.

Value function:
"If I am at state s, how much reward would I collect if I continued playing my policy π after that?"

Definition 2. (V-function): Expected discounted reward starting from the state s and rolling out π after,

$$V^{\pi}(s) = \mathbb{E}_{a \sim \pi(\cdot|s)}\left[Q^{\pi}(s, a) \middle| s\right]$$

A recurrence for Q functions: Bellman equation

Theorem 1. (Q-function): Recurrence relation for the Q-function of a policy π,

$$Q^\pi(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \mathbb{E}_{a' \sim \pi(\cdot|s')} \left[Q^\pi(s',a') \right]$$

$= \quad 0 \; + \; 0.5 \times$

$+ \; 0.5 \times$

$Q^\pi(s,a)$

$r(s,a)$
(only at last step)

$Q^\pi(s',a'_1)$

$Q^\pi(s',a'_2)$

Characterizing the optimal policy for discounted MDPs

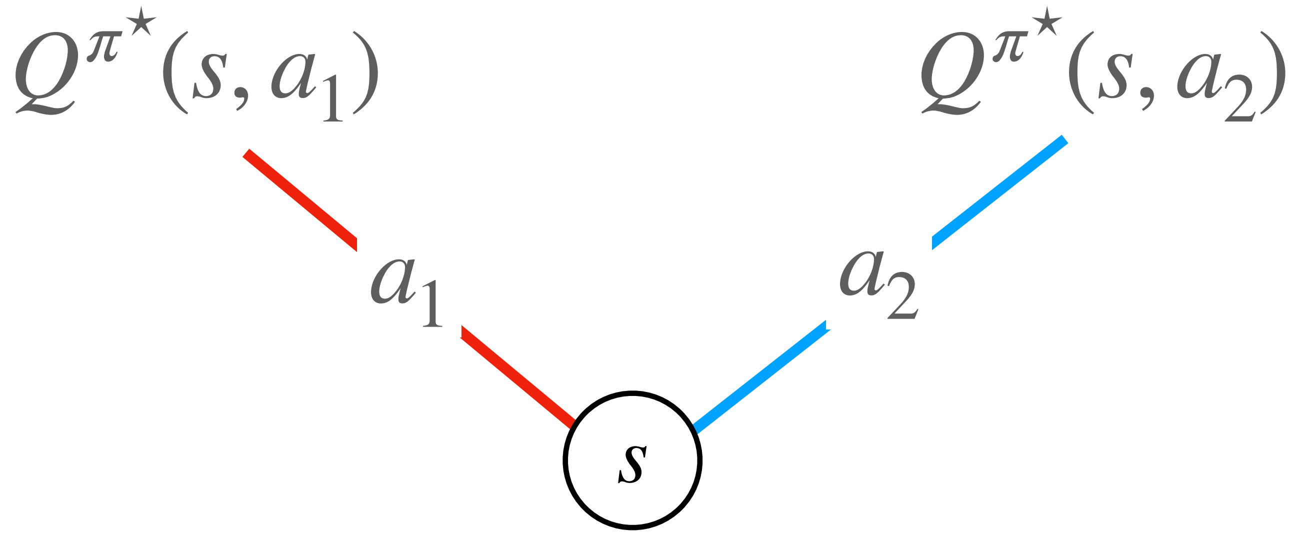
Theorem 2 (a). (Bellman optimality equation, part 1)
There exists an optimal policy which takes the form,

$$\pi^\star(\,\cdot\mid s) = \delta_{a^\star}, \text{ where } a^\star = \arg\max_{a \in A} Q^{\pi^\star}(s,a)$$

In words, the expert policy is deterministic and picks the action with the largest Q value

Optimal policy plays greedily with respect to $Q^{\pi^\star}$.

Proof sketch. If $Q^{\pi^\star}$ were known, what would be the optimal thing to do at a state?



Characterizing the optimal policy for discounted MDPs

$Q^{\pi^\star}(s, a)$

=

$Q^{\pi^\star}(s', a_1') = 1$ (win)

$Q^{\pi^\star}(s', a_2') = 0$ (draw)

Optimal policy would always pick this action

Characterizing the optimal policy for discounted MDPs

Theorem 2 (b). (Bellman optimality equation, part 2)
Plugging in the optimal policy from part 1 into the recurrence relation for $Q^{\pi^\star}$,

$$Q^{\pi^\star}(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}\left[\max_{a \in A} Q^{\pi^\star}(s',a)\right]$$

There is a very simple algorithm (value-iteration) which can solve this recurrence relation approximately.

…IF the MDP transition and rewards are known, which may not always be the case in practice.

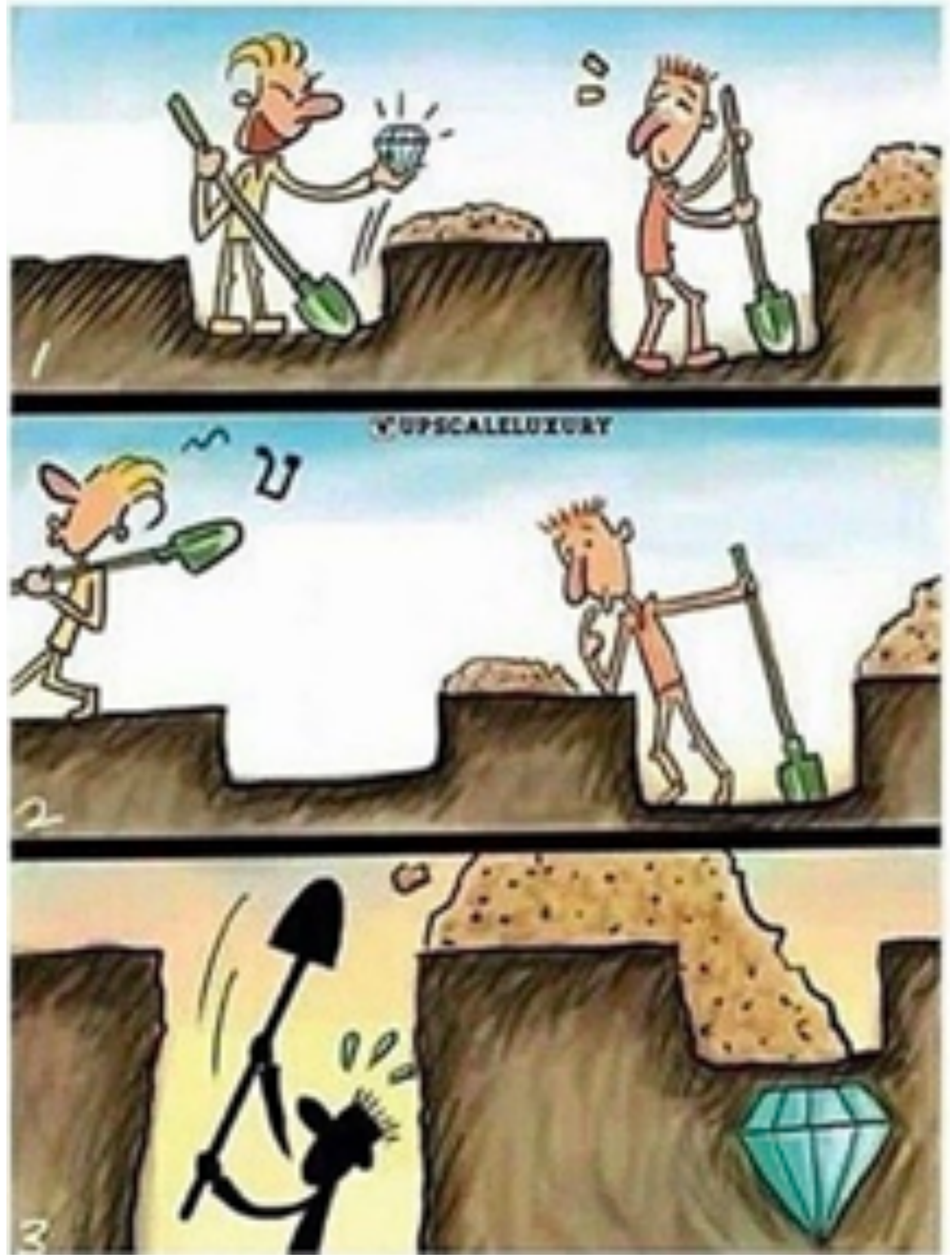
Reinforcement learning in the wild

Practical challenges in deploying RL

In practice, several challenges are present:

1. The reward functions and transitions are often not known explicitly: they must be learned.
2. State/action spaces can be massive: how do we generalize from a small number of interactions?
3. Rewards observed are often noisy or not aligned with the human's objectives. How can algorithms be robust to this?

Exploration vs. Exploitation dilemma



Reinforcement learning in the wild: exploration

Want to explore the state space to discover new states which may be good. (Exploration)
At the same time want to pick actions greedily with respect to our current Q-function estimates. (Exploitation)

Simplest idea:
ϵ-greedy exploration: Exploit with probability $1 - \epsilon$ and explore a uniformly random action with probability ϵ

Reinforcement learning in the wild: A very brief survey

Value-based methods:

1. The RL agent trains a model (usually a neural network) to learn $Q^{\pi^\star}$
2. Uses “Bellman backups” to update the value function.
3. Usually used with a technique called “experience replay” to smoothen training.

Eg.
Deep Q-Network (DQN) [Mnih et al. 2015]
Double DQN [van Hasselt et al. 2015]

Policy-based methods:

1. The RL agent trains a model (usually a neural network) to approximate the policy $\pi^\star(\cdot \mid s)$
2. Uses the “policy gradient theorem” to update the policy.
3. Usually implemented in continuous or large action-space environments.

Eg.
REINFORCE [Sutton 1999]
Trust Region Policy Optimization (TRPO) [Schulman 2015]
Proximal Policy Optimization (PPO) [Schulman 2017]

These approaches are being used extensively in training LLM reasoning models

A shallow dive: LLM reasoning

Initial state = prompt

Problem: Suppose a and b are positive real numbers with $a > b$ and $ab = 8$. Find the minimum value of $\frac{a^2+b^2}{a-b}$.

Ground truth solution: We can write $\frac{a^2+b^2}{a-b} = \frac{a^2+b^2-2ab+16}{a-b} = \frac{(a-b)^2+16}{a-b} = a - b + \frac{16}{a-b}$. By AM-GM, $a - b + \frac{16}{a-b} \geq 2\sqrt{(a-b) \cdot \frac{16}{a-b}} = 8$. Equality occurs when $a - b = 4$ and $ab = 8$. We can solve these equations to find $a = 2\sqrt{3} + 2$ and $b = 2\sqrt{3} - 2$. Thus, the minimum value is $\boxed{8}$.

reward = 1 if answer is correct

Actions = tokens

State =

A sparse-reward MDP with deterministic dynamics (like

Policy Optimization methods (REINFORCE/PPO/GRPO)

We want to maximize $V(\pi_\theta) = \mathbb{E}_{\pi_\theta}[r(s_H, a_H)]$**.** Here θ: weights of network.

Why not just run some gradient-based optimizer? (SGD/Rmsprop/Adam?)

The **Policy gradient theorem** tells us how to.

Policy Optimization methods (REINFORCE/PPO/GRPO)

Theorem 3. (Policy gradient theorem) Suppose learner's policy is parameterized as π_θ. Then,

$$\nabla_\theta V(\pi_\theta) = \mathbb{E}_{\pi_\theta}[Z], \text{ where,}$$

$$Z = \sum_{t=1}^{H} \nabla_\theta \log(\pi_\theta(a_t \mid s_t)) \cdot Q^{\pi_\theta}(s_t, a_t)$$

Looks complicated, what is going on?

Nicely motivated by the REINFORCE algorithm:

1. Generate a trajectory by rolling out π_θ.
2. Compute $\theta_{n+1} = \theta_n - \eta \cdot \widehat{Z}$ [stochastic gradient descent]

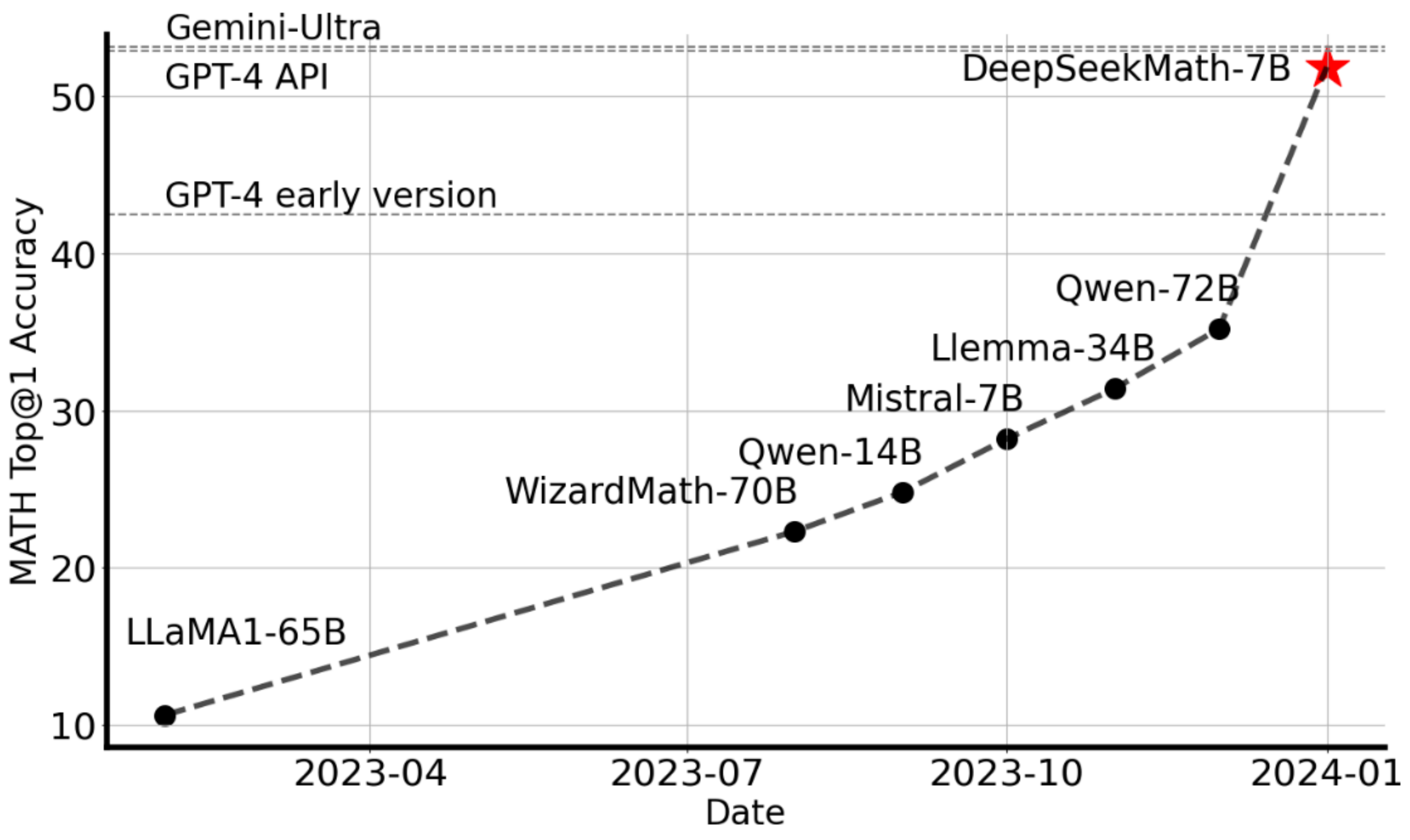
Where $\widehat{Z}$ is the value of Z computed on the trajectory sampled from π_θ // we may also run with minibatch size > 1.

Policy Optimization methods (REINFORCE/PPO/GRPO)

Q1. To compute $\widehat{Z}$, we need an estimate of $Q^{\pi_\theta}(s_t, a_t)$. How to do this?
Different approaches do it differently. PPO uses generalized advantage estimation (GAE). GRPO uses leave-one-out estimate

Q2. Every gradient update relies on generating a new trajectory from current policy π_θ.
Reuse data from the past via importance weights: PPO / GRPO

GRPO is one of the main drivers behind DeepSeek R1



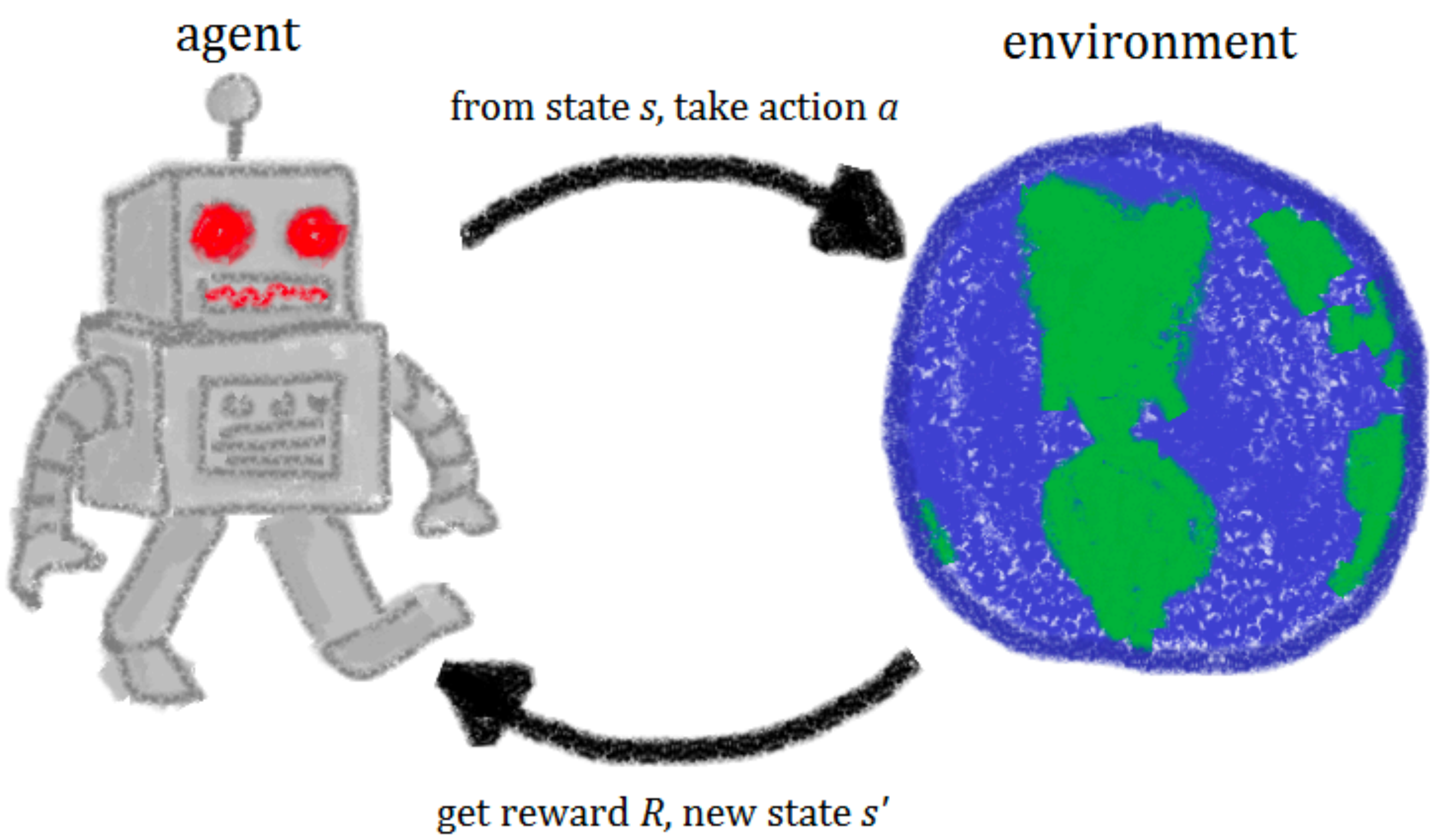
Smaller DeepSeekMath models were also trained to SOTA performance via GRPO

Conclusion

1. **Large Language Models:** Reasoning models, alignment to reflect human preferences (RLHF) have been transformed by RL
2. **Game solving:** Current RL agents are 1000x better than the best humans at Chess, Go, Shogi, Poker and many other games
3. **Autonomous driving and planning:** Have already been deployed successfully in many major cities, including Austin.
4. **Robotics:** many challenging open problems, especially in grasping and manipulation.

2024 Turing Award: Sutton and Barto (Pioneers of RL)

Thank you!



Many interesting avenues for research:
Offline Reinforcement Learning,
Imitation Learning,
Safe / Constrained Reinforcement Learning,
Partially observed MDPs,
Meta RL,

In many interesting domains:
Healthcare,
Finance,
Geology and climate prediction,
Robotics + self-driving

Some references:
CS394R @UT: Reinforcement Learning: Theory and Practice
Reinforcement Learning: An Introduction (Sutton & Barto),
Algorithms for Reinforcement Learning (Csaba)
Bandit Algorithms (Lattimore and Csaba)
Lil'log (Lilian Weng's blog)
Many many Huggingface tutorials…

You can reach me at: nived.rajaraman@berkeley.edu if you have any questions.