Reinforcement Learning

AI/ML Bootcamp

Vaibhav Unhelkar

Assistant Professor of Computer Science

Director, Human-centered AI Group

Rice University

vaibhav.unhelkar@rice.edu

Overview

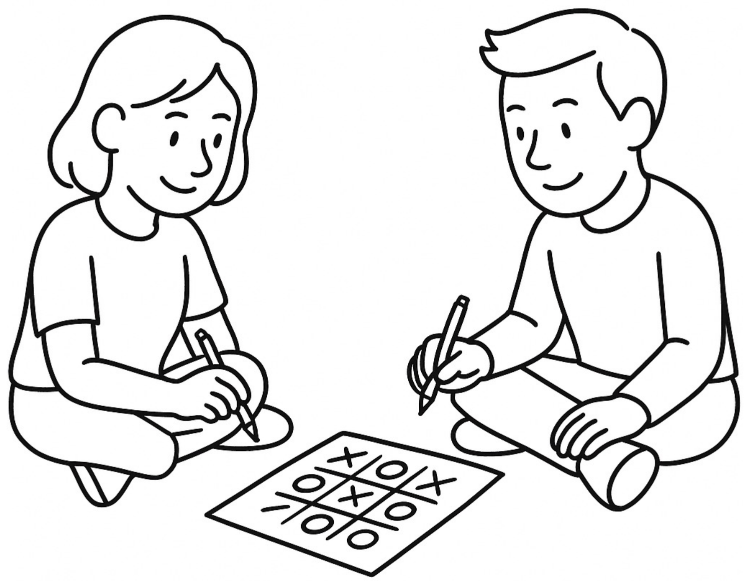
- Mathematical Preliminaries
- Reinforcement Learning vis-à-vis Supervised Learning
- Models for Sequential Decision-Making
  - Multi-armed Bandits (1-Step Decisions)
  - Markov Decision Process (N-Step Decisions)
- RL: Problem Formulation
- RL: Algorithms
  - Q Learning
  - Deep Q Learning
- RL Safety

RL requires Interaction

- Please ask questions (even if they seem tangentially related)
- Please answer questions (even if they seem rhetorical)

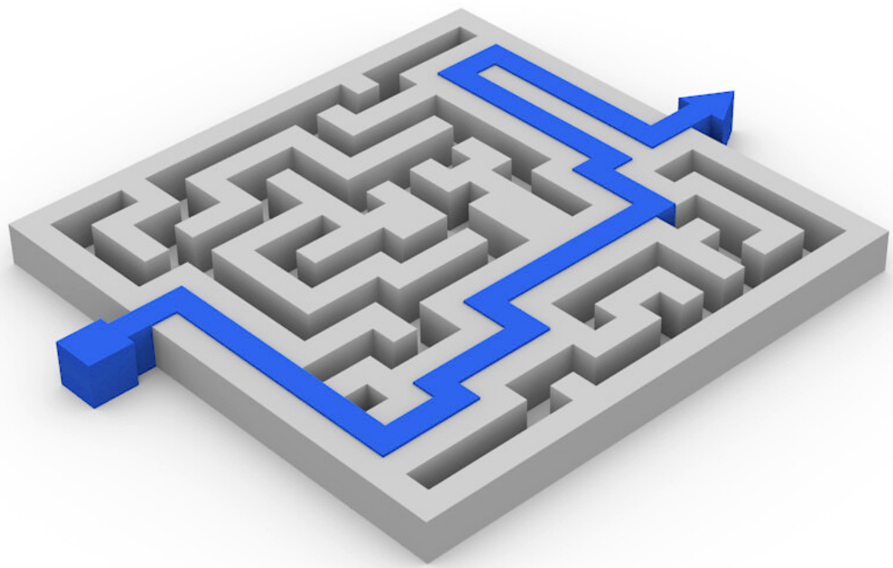
Decision Making

Tic-Tac-Toe



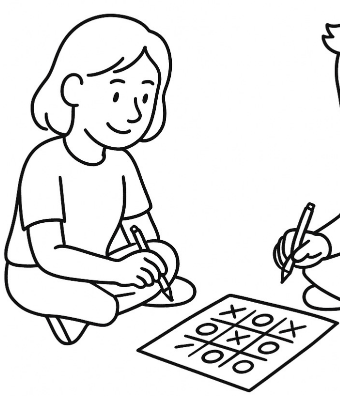
How will you create a program to play Tic-Tac-Toe?

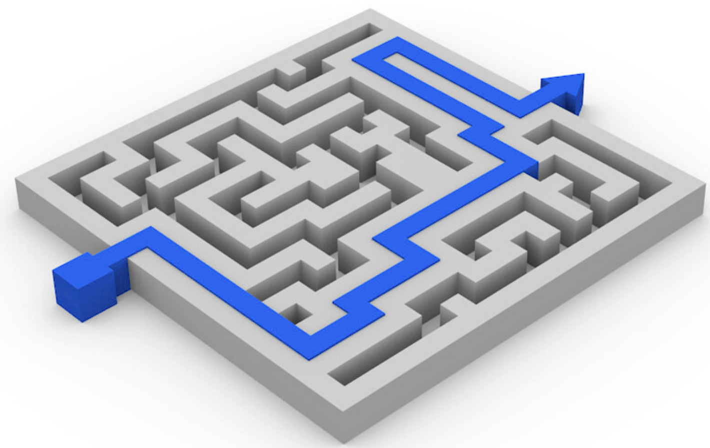
Autonomous Maze Solving



How will you create a program to solve this maze?

Sequential Decision Making





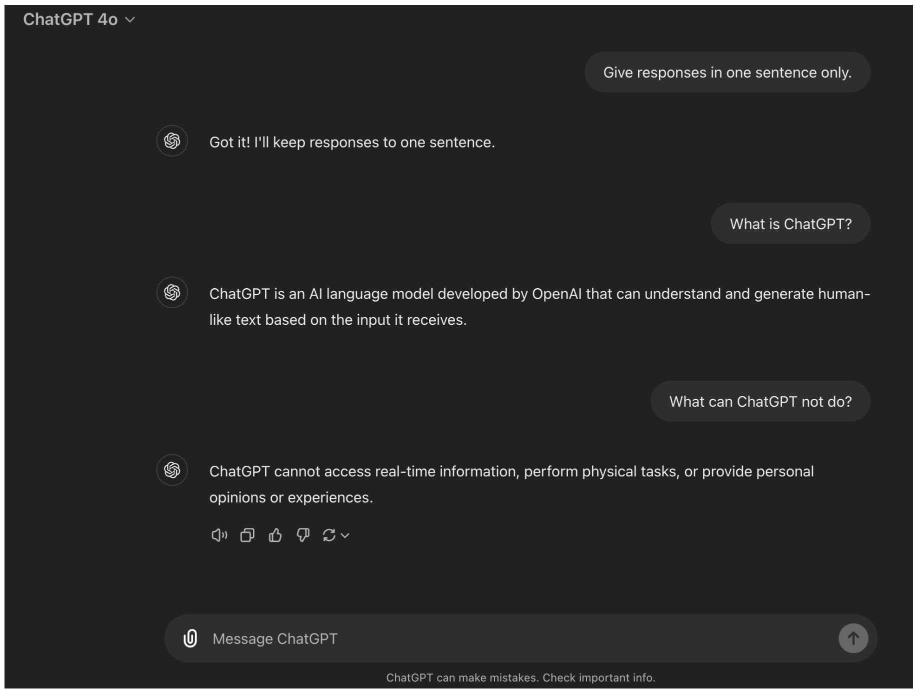
Can you use your Tic-Tac-Toe program to solve the maze?

Can you use your Maze program to solve Tic-Tac-Toe?

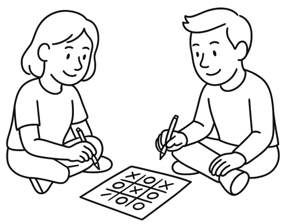
Game Playing

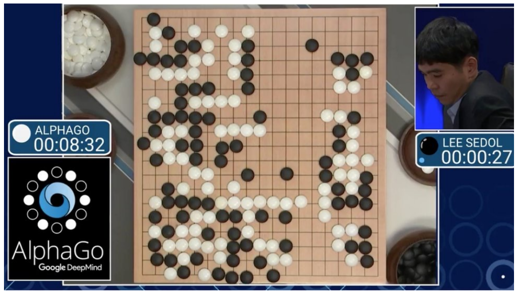
How will you create a program to play this game?

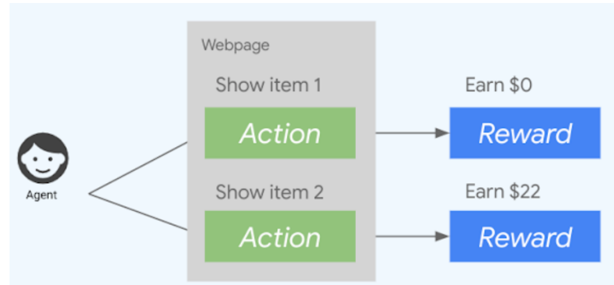
Chat Bots and Language Models

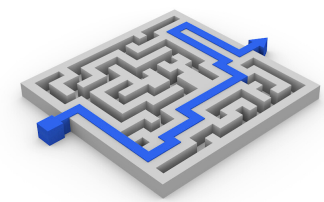


How will you create a program to train a chat bot / language model?

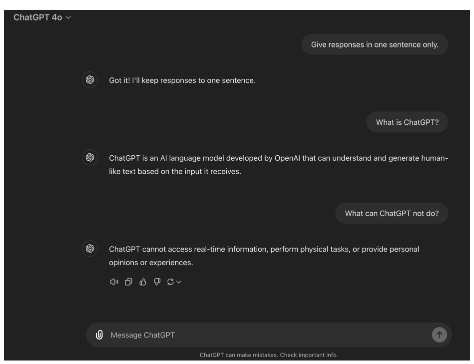


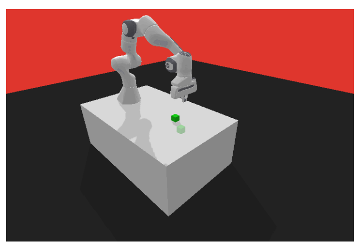


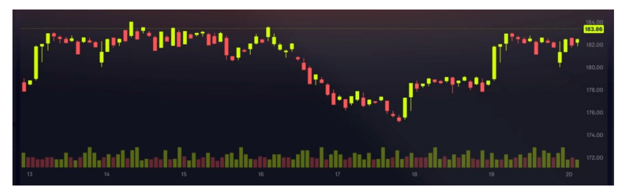


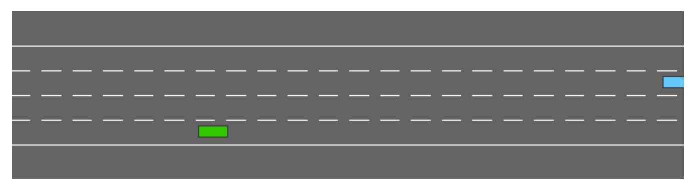


Reinforcement learning offers a **unified approach** to solve a **large class of decision-making problems.**









Sutton and Barto. Reinforcement Learning: An Introduction. MIT Press, 2018.

Reinforcement learning offers a **unified approach** to solve a **large class of decision-making problems**.

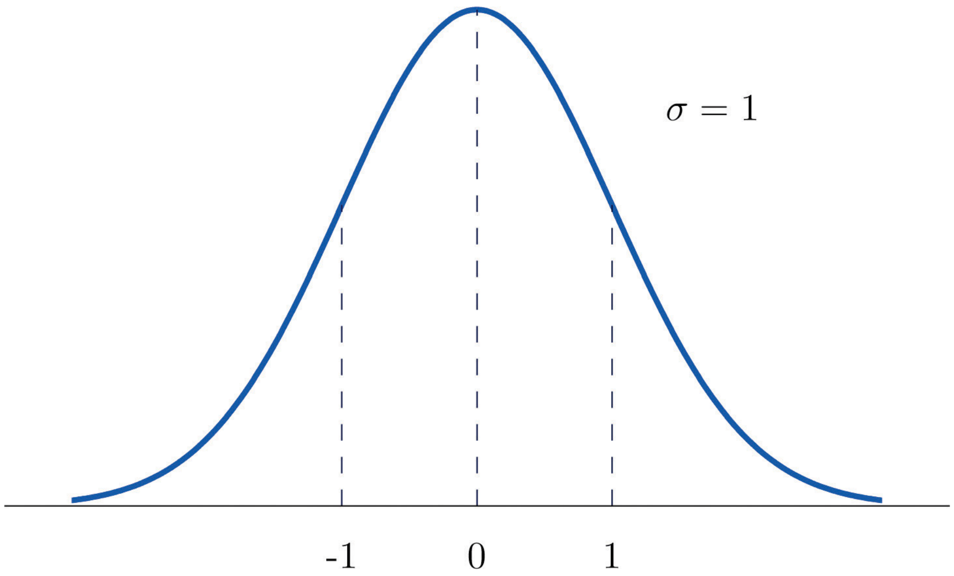
The idea of RL is so general that some even think **Solving RL = Artificial General Intelligence.**

Silver et al. "Reward is enough." *Artificial Intelligence* 299 (2021): 103535.

Preliminaries

Random Variables 101

A variable whose values are numerical outcomes of a random phenomena



Expected Value (1/2)

Discrete random variable

$$\mathbb{E}_{p(x)}[g(x)] = \sum_{\{x \in X\}} g(x)p(x)$$

where $p(x)$ is the probability mass function of the random variable.

When $\mathrm{g}(x) = x$, the expected value corresponds to the mean of the r.v.

Expected Value (2/2)

Continuous random variable

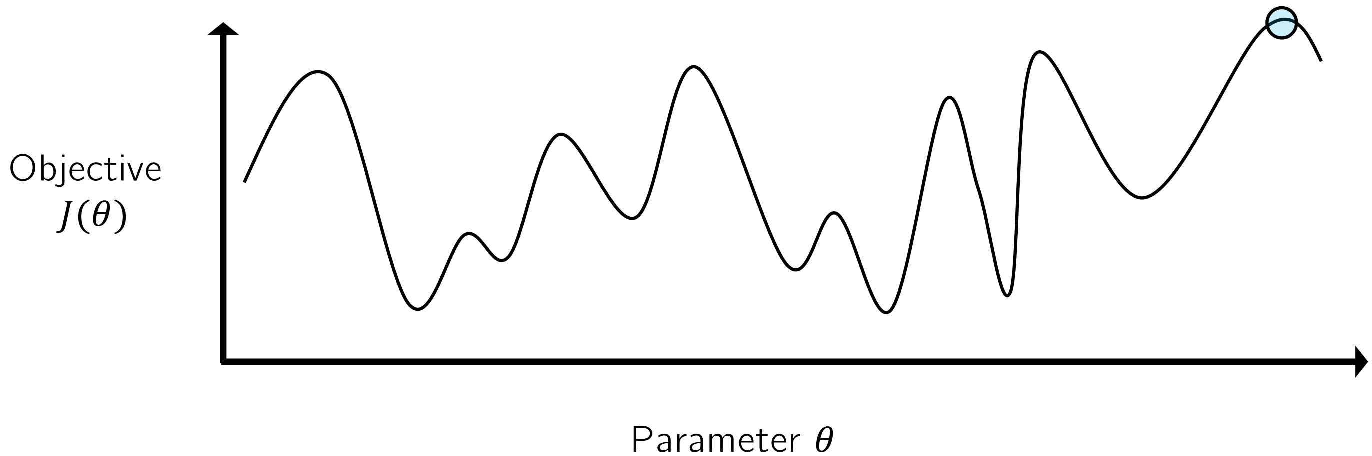
$$\mathbb{E}_{p(x)}[g(x)] = \int_x g(x)p(x)dx$$

where $p(x)$ is the probability density function of the random variable.

For ease of notation, often the subscript is not denoted

$$\mathbb{E}_{p(x)}[g(x)] = \mathbb{E}[g(x)]$$

Optimization 101



The general (unconstrained) optimization problem:

$$\max_{\theta} J(\theta)$$

where $J(\theta)$ is referred to as the objective function.

Gradient Ascent

Optimization Problem:

$$\max_{\theta} J(\theta)$$

Solution: Repeat until convergence

- Compute the gradient of the objective w.r.t. the parameters
- Update the parameters based on the gradient

$$\theta \quad \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$$

Stochastic Gradient Ascent

Optimization Problem:

$$\max_{\theta} J(\theta)$$

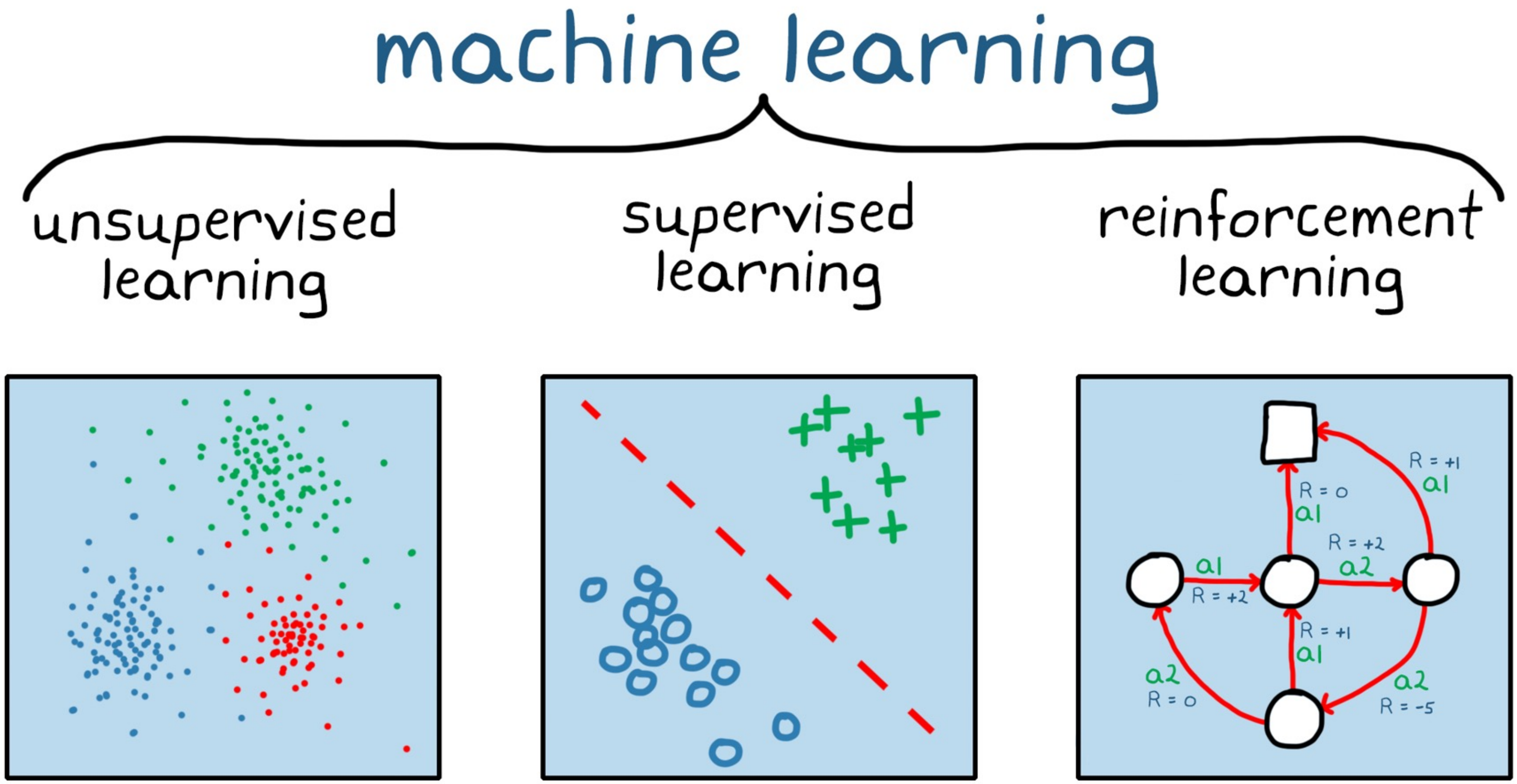
Solution: Repeat until convergence

- Compute the **stochastic** gradient of the objective w.r.t. the parameters
- Update the parameters based on the **stochastic** gradient

$$\nabla_{\theta} J(\theta) = \mathbb{E}[g(\theta)]$$

$$\theta \leftarrow \theta + \alpha g(\theta)$$

Reinforcement Learning vis-à-vis Supervised Learning



Source: https://www.mathworks.com/discovery/reinforcement-learning.html

Supervised Learning

Given

- A finite set of training data, $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$
- A class of models, $f: X \rightarrow Y \in F$
- A loss function, $L: Y \times Y \rightarrow \mathcal{R}$

Find

- The best model $f^* \in F$ that minimizes the empirical loss

$$f^* = \operatorname{argmin}_{f \in F} \frac{1}{m} \sum_i L\,(\hat{y}_i, y_i)$$

Supervised Learning

… assumes training dataset is fixed and given.
However, in practice, we have (some) control over data collection.
We can use this ability to improve data collection and learning.

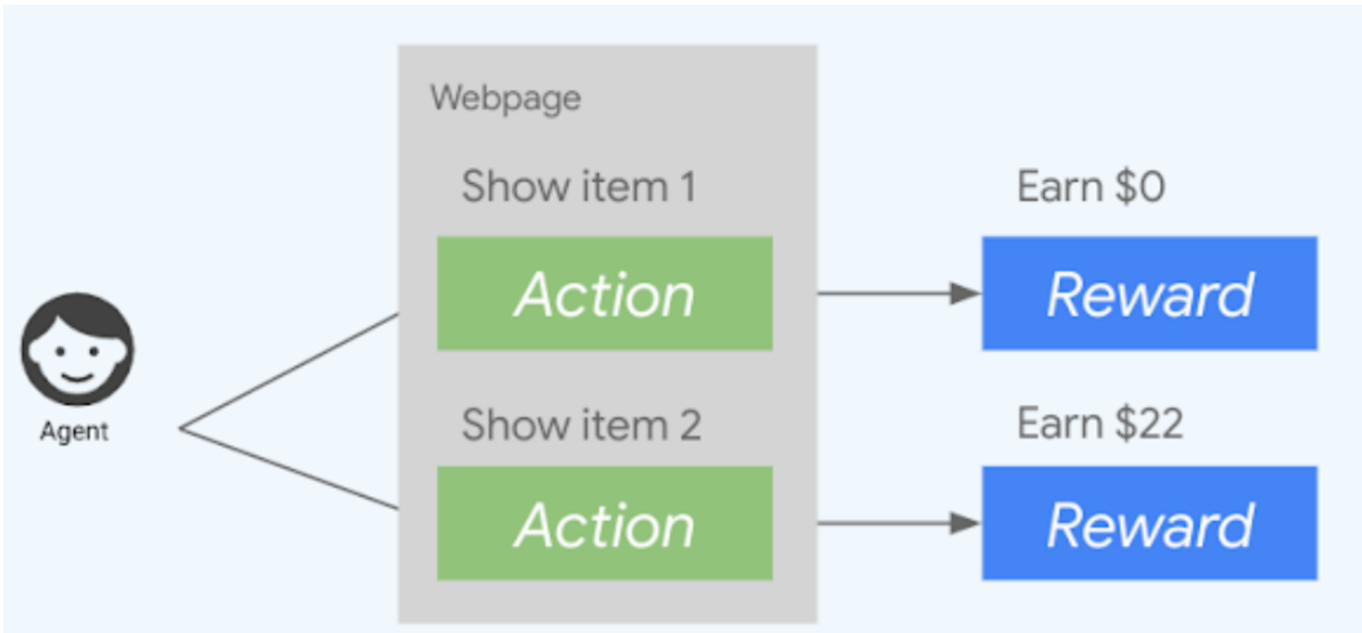
Given

- A finite set of training data, $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$
- A class of models, $f: X \to Y \in F$
- A loss function, $L: Y \times Y \to \mathcal{R}$

Find

- The best model $f^* \in F$ that minimizes the empirical loss

$$f^* = \operatorname{argmin}_{f \in F} \frac{1}{m} \sum_i L\left(\hat{y}_i, y_i\right)$$



Dutta et al. "AutoML for contextual bandits." arXiv preprint arXiv:1909.03212 (2019).

Supervised Learning

... assumes each model input is independent of the previous ones. However, in certain cases, model inputs depend on the previous model inputs and even the model outputs.

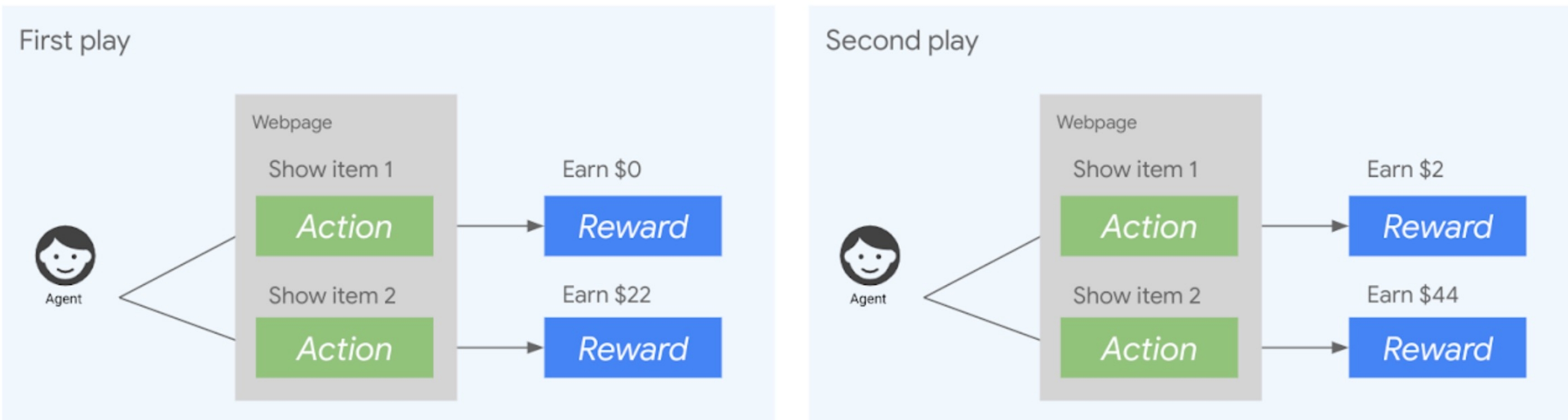
Given

- A finite set of training data, $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$
- A class of models, $f: X \to Y \in F$
- A loss function, $L: Y \times Y \to \mathcal{R}$

Find

- The best model $f^* \in F$ that minimizes the empirical loss

$$f^* = \text{argmin}_{f \in F} \frac{1}{m} \sum_i L\,(\hat{y}_i, y_i)$$



Dutta et al. "AutoML for contextual bandits." arXiv preprint arXiv:1909.03212 (2019).

Towards Reinforcement Learning (1/3)

Given

- The ability to collect training data, $D = \{(x_1, y_1), (x_2, y_2), \ldots\}$
- A class of models, $f: X \rightarrow Y \in F$
- A loss function, $L: Y \times Y \rightarrow \mathcal{R}$

Find

- The best model $f^* \in F$ that minimizes the empirical loss

$$f^* = \operatorname{argmin}_{f \in F} \frac{1}{m} \sum_i L\,(\hat{y}_i, y_i)$$

Towards Reinforcement Learning (2/3)

Given

- The ability to collect training data, $D = \{(x_1, y_1), (x_2, y_2), \ldots\}$ such that
  $x_{t+1} \sim T(\cdot \mid x_t, y_t)$
- A class of models, $f: X \to Y \in F$
- A loss function, $L: Y \times Y \to \mathcal{R}$

Find

- The best model $f^* \in F$ that minimizes the empirical loss

$$f^* = \operatorname{argmin}_{f \in F} \frac{1}{m} \sum_i L\,(\hat{y}_i, y_i)$$

Towards Reinforcement Learning (3/3)

Given

- The ability to collect training data, $D = \{(x_1, y_1), (x_2, y_2), \dots\}$ such that
$$x_{t+1} \sim T(\cdot \mid x_t, y_t)$$
- A class of models, $f: X \rightarrow Y \in F$
- A reward function, $R: X \times Y \rightarrow \mathcal{R}$

Find

- The best model $f^* \in F$ that maximizes the cumulative reward
$$f^* = \operatorname{argmax}_{f \in F} \frac{1}{m} \sum_i R\,(x_i, \hat{y}_i)$$

Reinforcement Learning

Given

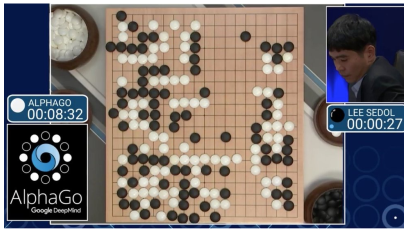
- The ability to collect training data, $D = \{(s_1, a_1), (s_2, a_2), \dots\}$ such that
$$s_{t+1} \sim T(\cdot \mid s_t, a_t)$$
- A class of models, $\pi: S \to \Pr(A) \in \Pi$
- A reward function, $R: S \times A \to \mathcal{R}$

Find

- The best model $\pi^* \in \Pi$ that maximizes the cumulative reward
$$\pi^* = \operatorname{argmax}_\pi \sum_i R\,(s_i, a_i)$$

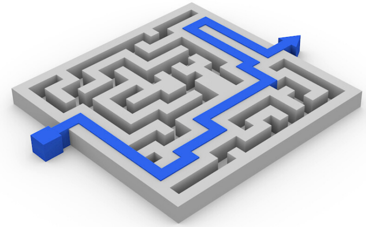
Environment and Agents

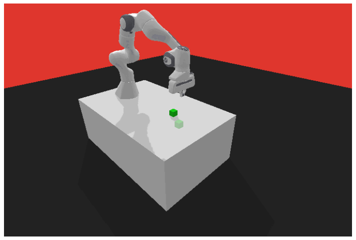
Reinforcement Learning Paradigm

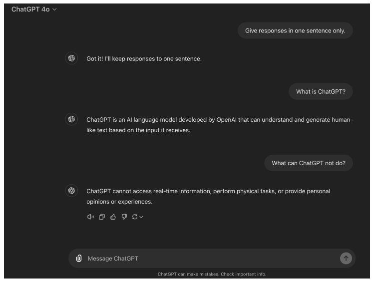


Reinforcement learning (RL) offers a **unified** approach to solve a **large class of problems,** where

- The learner can interact with its environment,
- Trial and error is possible, and
- The learn receives a signal of its success and failures (reward, feedback or reinforcements).





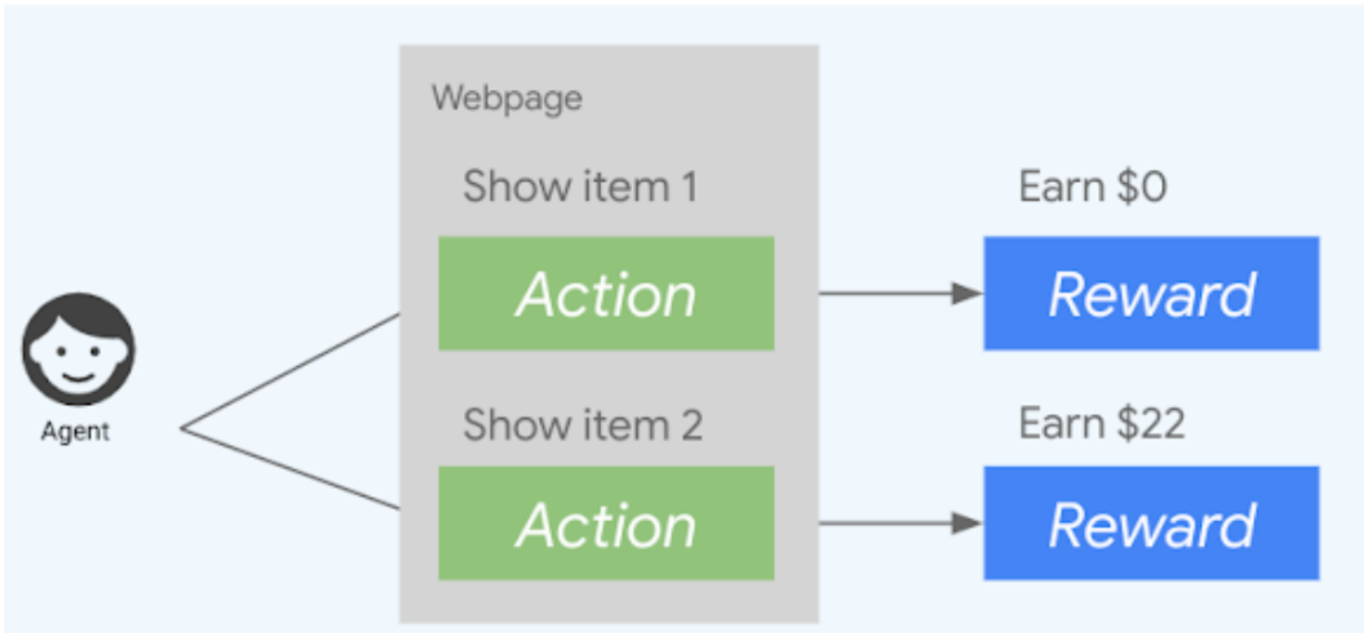


Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *nature* 518.7540 (2015): 529-533.
Animation: https://github.com/kuz/DeepMind-Atari-Deep-Q-Learner

Environment and Agents

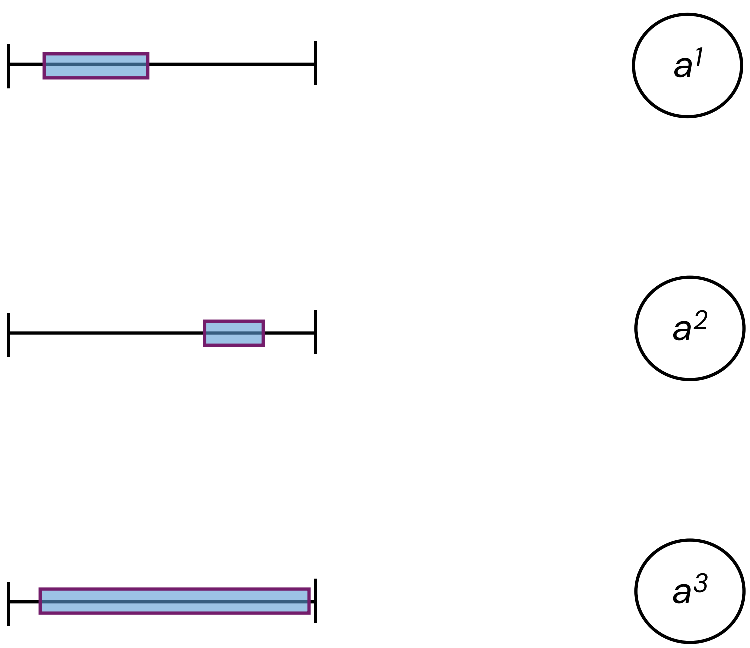
RL is particularly suited for sequential and stochastic problems for which specifying the desired outcome (reward signal) is easier than specifying how to achieve it (policy).

Bandits



Dutta et al. "AutoML for contextual bandits." arXiv preprint arXiv:1909.03212 (2019).

The Multi-armed Bandit Model

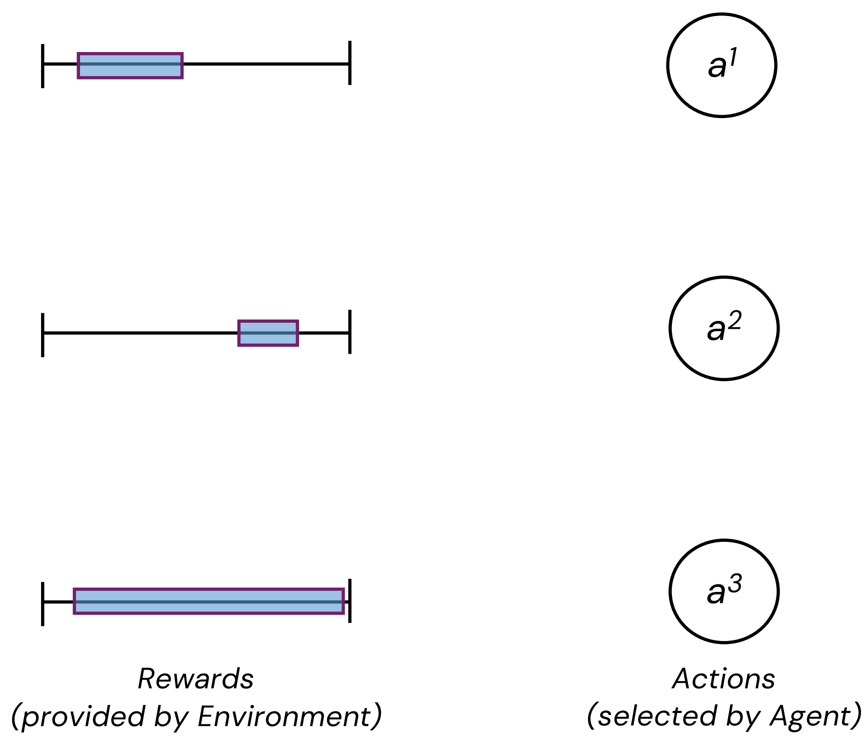


Rewards
(provided by Environment)

Actions
(selected by Agent)

- Agent selects an action *(a^i).*
- Environment provides a reward *(r).*
  - Environment does not change.
  - Conditional distribution of reward is fixed.
  - Reward distribution is unknown to the agent.
- Agent's goal is to maximize its cumulative reward.

Bandit: Key Terms



- Action, a
- Reward, r
- Conditional distribution of reward, $Pr(r|a)$
- Policy, $\pi(a)$

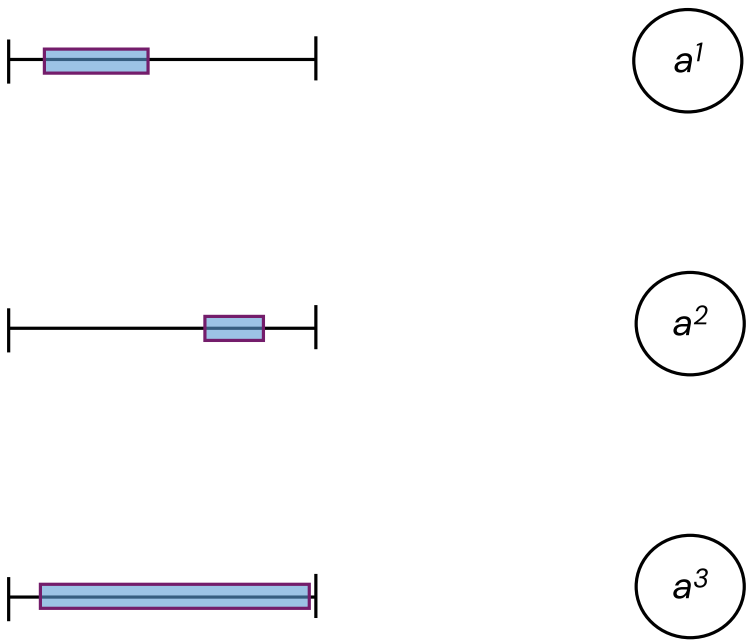
Q. Select the Optimal Action

Action	Pr(Reward | Action)
a_1	5 with probability 0.5 0 with probability 0.5
a_2	5 with probability 0.9 0 with probability 0.1
a_3	5 with probability 0.6 -1 with probability 0.4

Q. Select the Optimal Action

Action	Pr(Reward | Action)
a_1	2 with probability 0.5 0 with probability 0.5
a_2	10 million w.p. 10^{-7} 0 otherwise
a_3	1 always

Bandits: Problem Formulation



Rewards
(provided by Environment)

Actions
(selected by Agent)

Inputs

- Sequence of actions and rewards, $(a_1, r_1, a_2, r_2, a_3, r_3, a_4, r_4, \ldots)$.
- Not all inputs available at once.
- Agent's past actions influence the future inputs.

Output

- Policy, π_t *(a),* a probability distribution over which action to pick next that maximizes the objective.

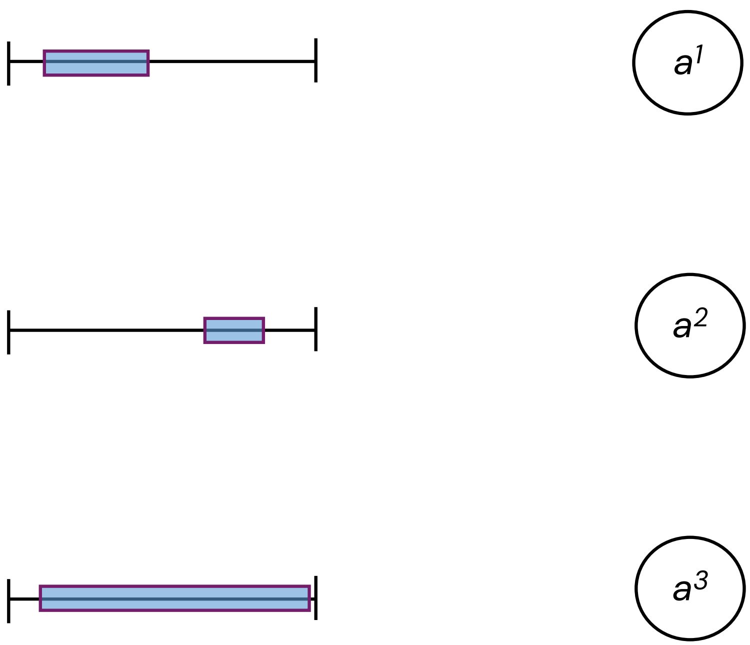
Objective

$$\max_{\{a_1, a_2, \ldots\}} \mathbb{E}[\Sigma_t r_t] = \max_{\pi} \mathbb{E}[\Sigma_t r_t]$$

Q. Select the Optimal Action

Action	Pr(Reward | Action)
a_1	2 with probability 0.5 0 with probability 0.5
a_2	10 million w.p. 10^{-7} 0 otherwise
a_3	10 million w.p. 0.5 -9 million w.p. 0.5

Exploration v/s Exploitation Tradeoff



Rewards
(provided by Environment)

Actions
(selected by Agent)

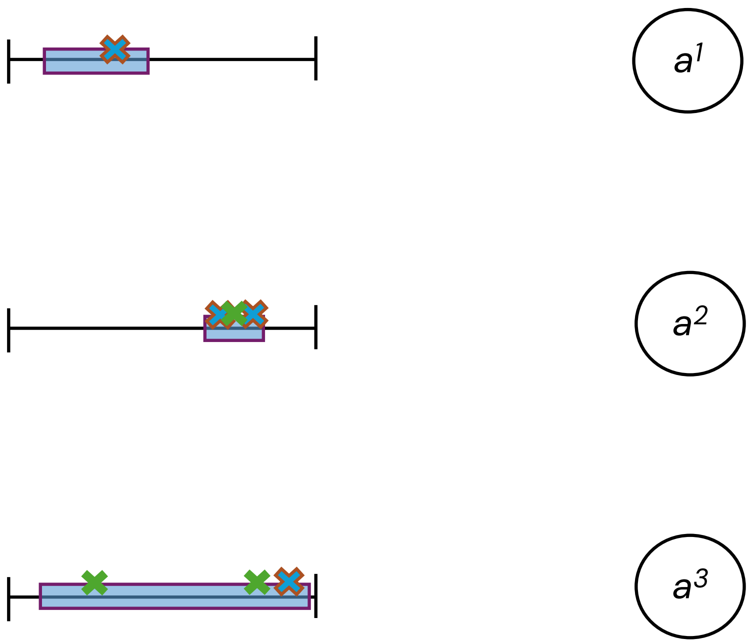
Bandits
(general case)

Exploration + Exploitation

Need to **learn a policy** that balances exploration and exploitation to maximize cumulative reward.

- Exploration to ensure that we do not miss out on high but previously unseen rewards.
- Exploitation to ensure that we do not miss out on previously seen high rewards.

Intuition: Making Decision in Bandits



Rewards
(provided by Environment)

Actions
(selected by Agent)

Bandits
(general case)

Exploration + Exploitation

First initialize action values, then iteratively

- Select
  - Greedy actions with high probability **(exploitation)**
  - Random actions with low but non-zero probability **(exploration)**
- Receive rewards from the environment
- Update estimates of action values

A Value-Based Bandit Algorithm

A simple bandit algorithm

Initialize, for $a = 1$ to k:

$Q(a) \leftarrow 0$

$N(a) \leftarrow 0$

Loop forever:

$$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$

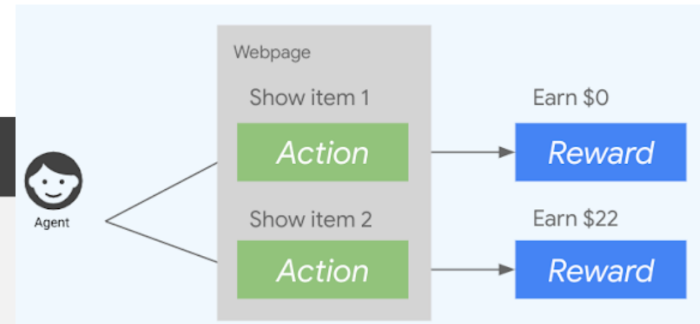
$R \leftarrow bandit(A)$

$N(A) \leftarrow N(A) + 1$

$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\left[R - Q(A)\right]$

Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

A Value-Based Bandit Algorithm



A simple bandit algorithm

Initialize, for $a = 1$ to k:

$$Q(a) \leftarrow 0$$
$$N(a) \leftarrow 0$$

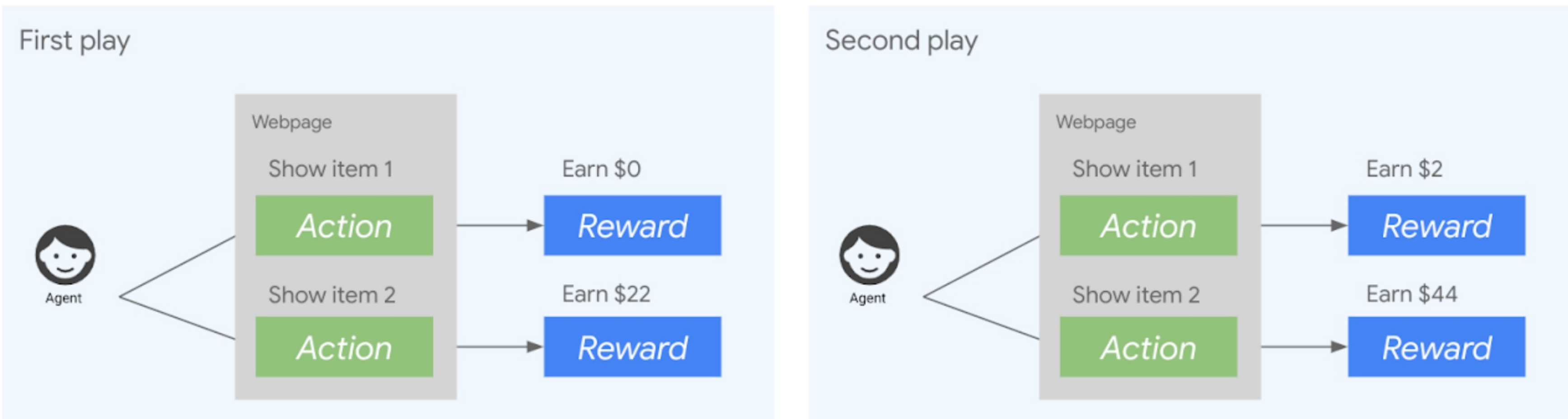
Loop forever:

$$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$
$$R \leftarrow bandit(A)$$
$$N(A) \leftarrow N(A) + 1$$
$$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\big[R - Q(A)\big]$$

Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

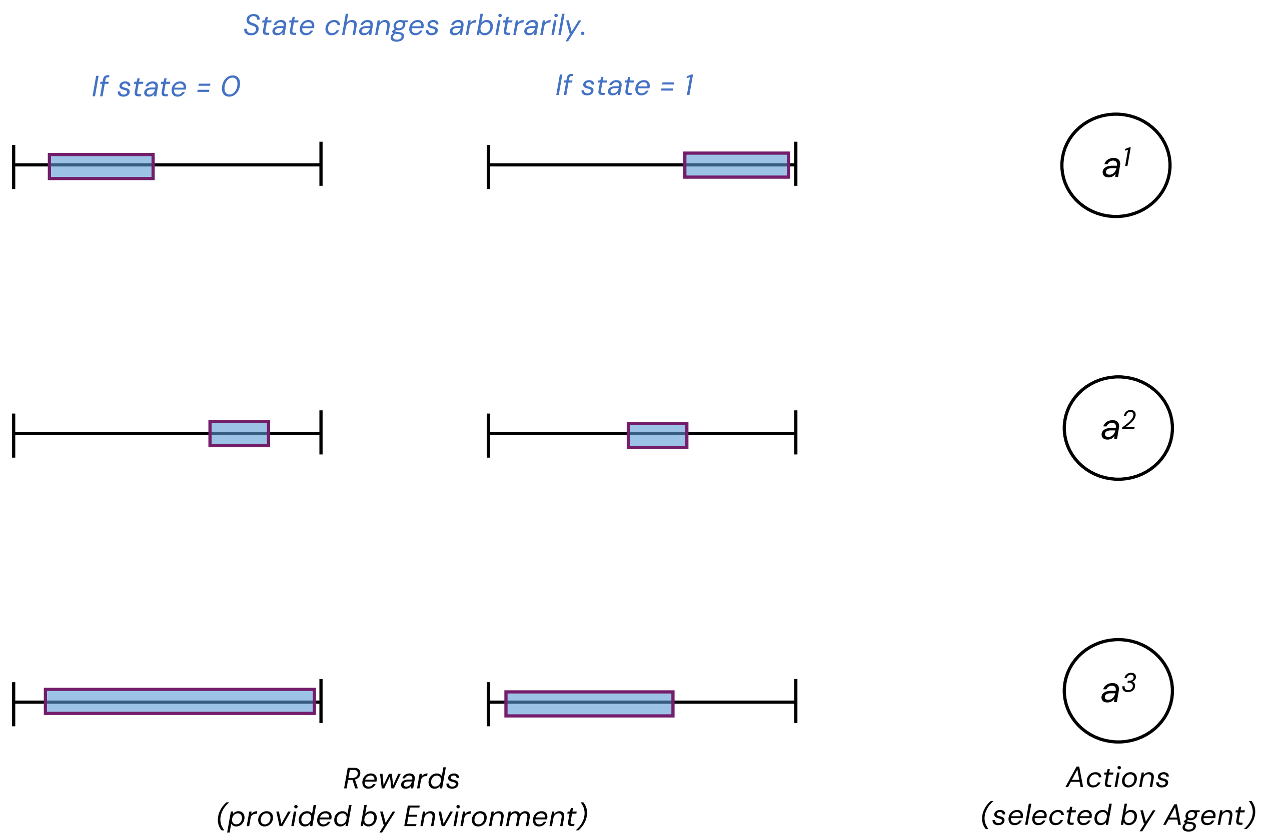
Q. Select the Optimal Action

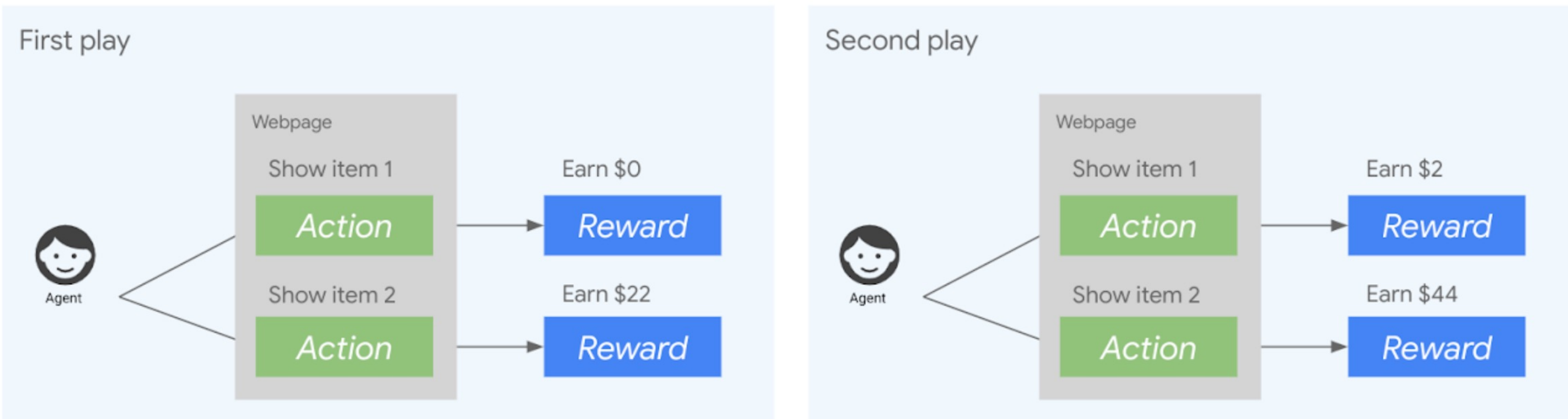
Action	Pr(Reward | Action)
a_1	2 with probability 0.5 0 with probability 0.5
a_2	10 million w.p. 10^{-7} 0 otherwise
a_3	10 million w.p. 0.5 -9 million w.p. 0.5



Dutta et al. "AutoML for contextual bandits." arXiv preprint arXiv:1909.03212 (2019).

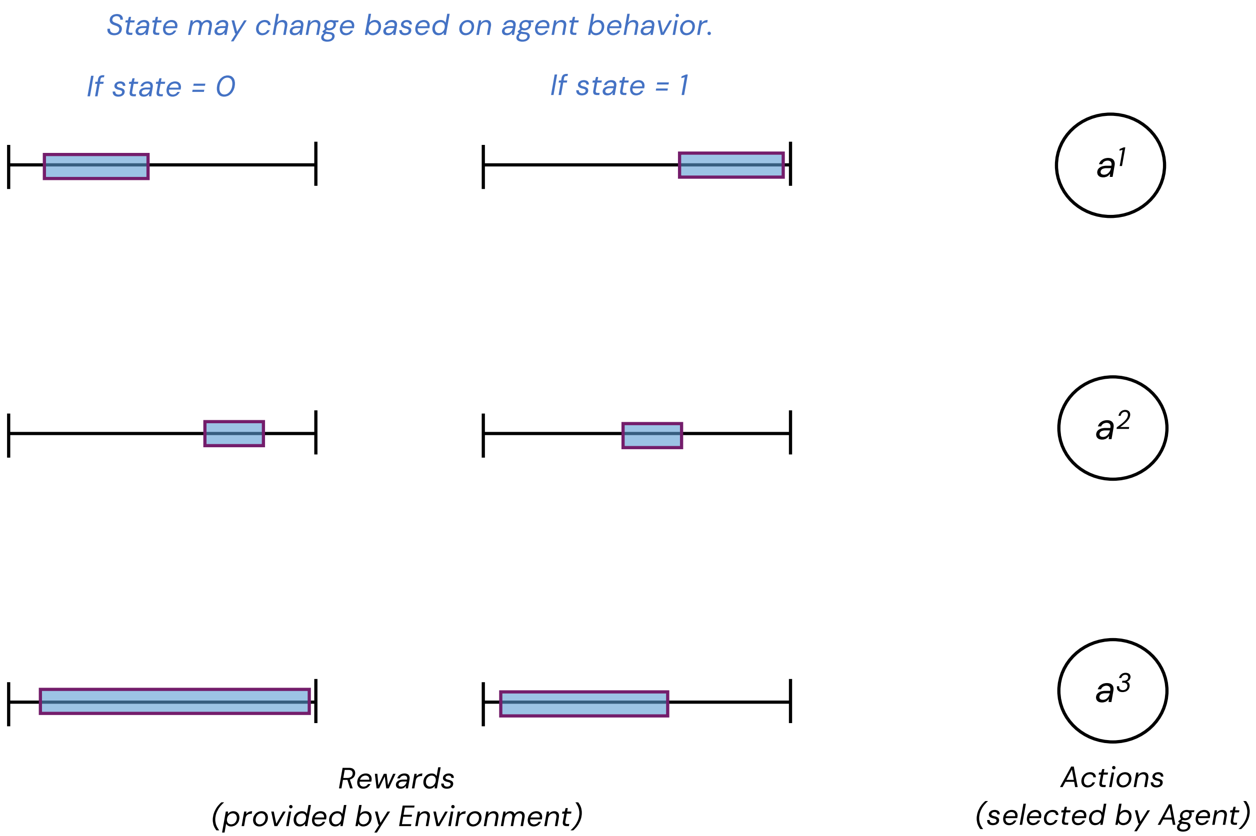
Contextual Bandits





Dutta et al. "AutoML for contextual bandits." arXiv preprint arXiv:1909.03212 (2019).

Sequential Decision Making



Sequential Decision Making

In contrast to Bandits, in the full RL problem
(general sequential decision-making)
the environment may change based on agent’s actions.

Markov Decision Processes

Markov Property

Consider a sequence of random variables

$$s_1, s_2, s_3, \cdots, s_n$$

The sequence is Markovian if

$$\Pr(s_n | s_1, s_2, \cdots, s_{n-1}) = \Pr(s_n | s_{n-1}) \quad \forall \quad n$$

Markov Decision Processes (MDPs)

MDPs are a special case of the general sequential decision-making problem, where

- Agent's observation completely describe the current state of the problem,
- Sequence of observations conditioned on the agent's action is Markovian, and
- A scalar signal (reward) is sufficient to describe the problem objective.

MDP: Definition

MDPs are defined by the tuple (S, A, T, R, γ), where

- S is the state space, A is the action space,
- T is the transition function, R is the reward function, and
- γ is the discount factor.

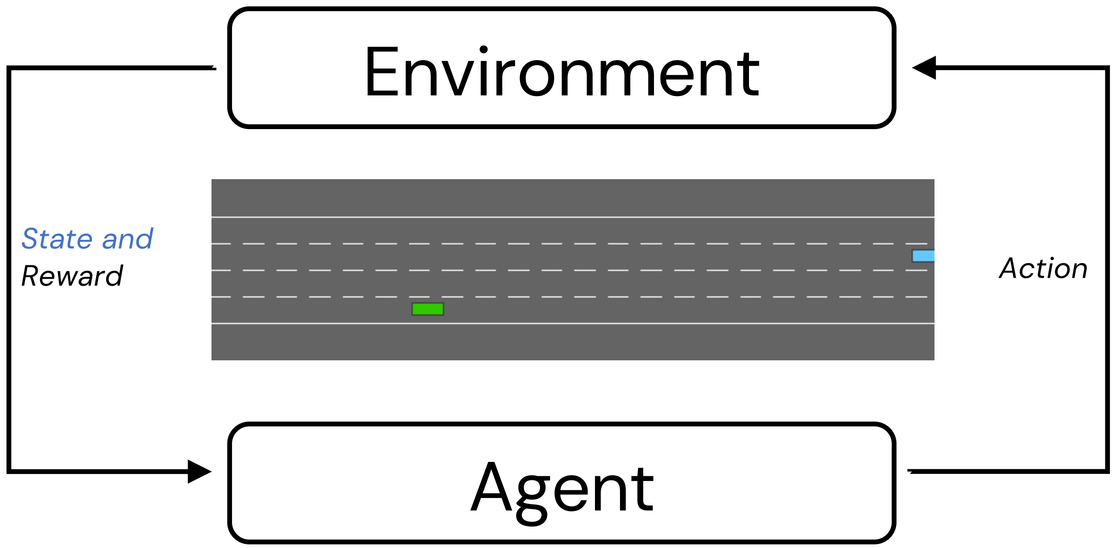
MDP: Definition

Note: In practice, an important step for employing RL is to convert *the real-world problem that you want to solve to* an *MDP model that is both useful and computationally tractable.*

MDPs are defined by the tuple (S, A, T, R, γ), where

- S is the state space, A is the action space,
- T is the transition function, R is the reward function, and
- γ is the discount factor.

MDP: Example #1



MDPs are defined by the tuple (S, A, T, R, γ), where

- S is the state space, A is the action space,
- T is the transition function, R is the reward function, and
- γ is the discount factor.

MDP: Example #2

MDPs are defined by the tuple (S, A, T, R, γ), where

- S is the state space, A is the action space,
- T is the transition function, R is the reward function, and
- γ is the discount factor.

Discount Factor: Motivation

Let us say the agent receives rewards as money and has two actions:

1. Action 1 gives \$100 today.
2. Action 2 gives \$50 today and \$60 a year from now.

Which action should the agent choose?

Discount Factor: Motivation

Let us say the agent receives rewards as money and has two actions:

1. Action 1 gives $100 today.
2. Action 2 gives $50 today and $60 a year from now.

Which action should the agent choose?

1. Return 1 = $100
2. Return 2 = $50 + γ60

Reward Design

Note: RL requires specification of sound reward functions. Rewards should convey "what" is the Agent's task, and not "how" the Agent should do this task.

MDPs are defined by the tuple (S, A, T, R, γ), where

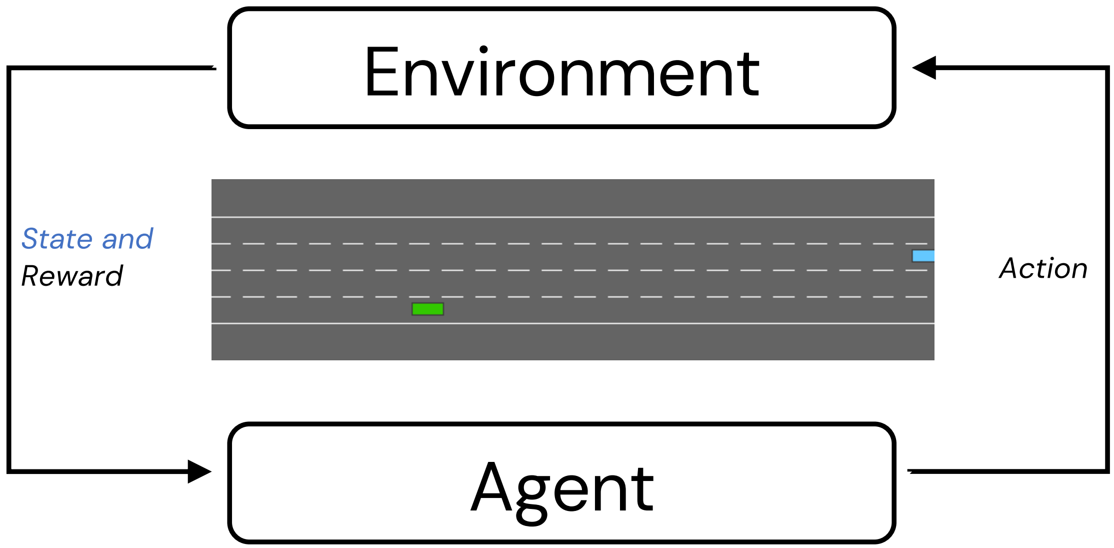
- S is the state space, A is the action space,
- T is the transition function, R is the reward function, and
- γ is the discount factor.

Reinforcement Learning: Problem Formulation

Decision-Making Policy

Given a real-world decision-making problem described as an Markov Decision Process,
the Agent needs a "policy" to make decisions and collect rewards.

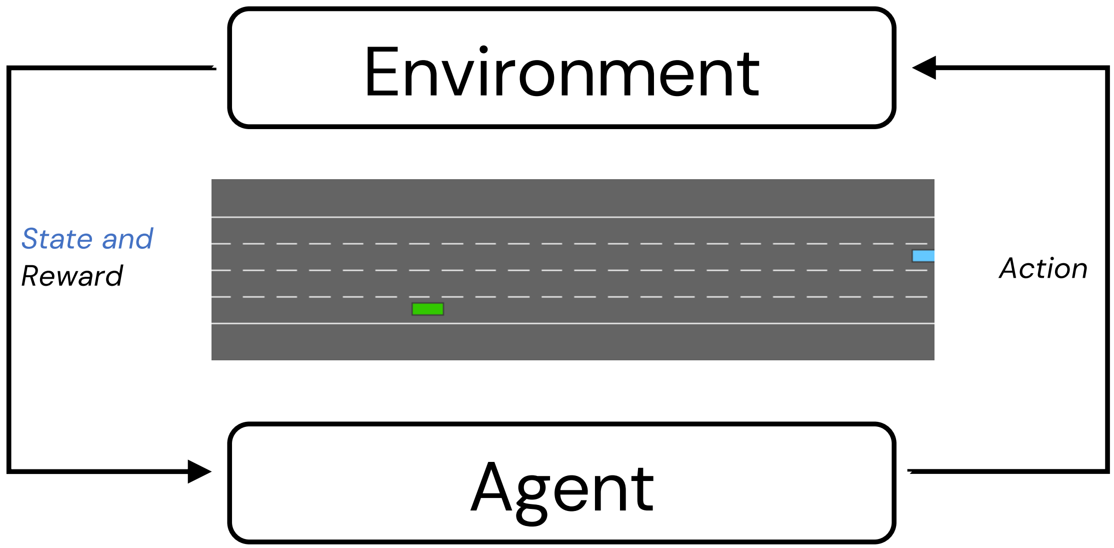
Decision-Making Policy



The policy for an MDP can be described as either sequence of actions $a_1, a_2, a_3, \cdots$

Or, more compactly, as the "Markov policy function" $\pi(a|s)$

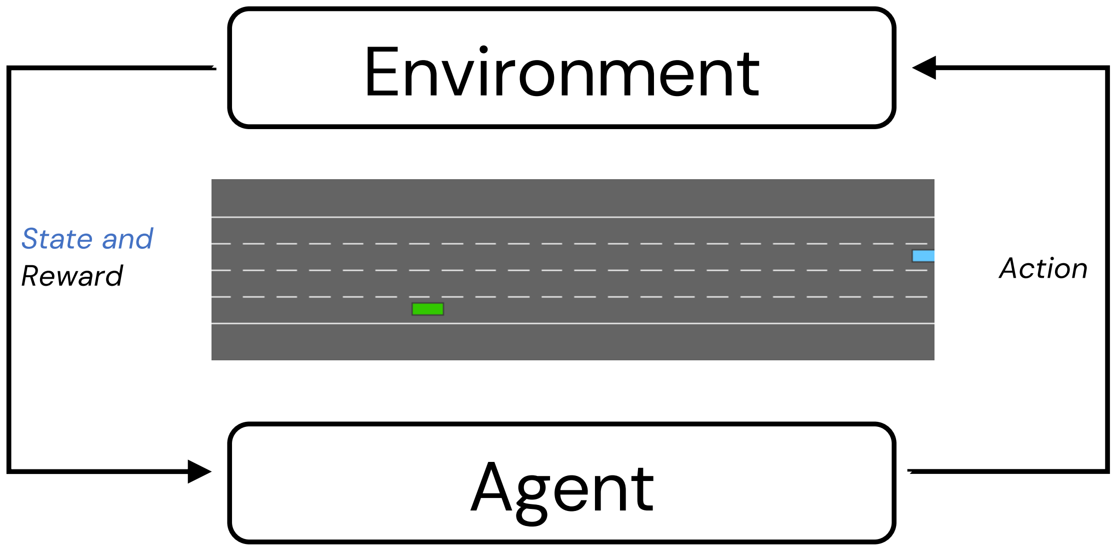
Decision-Making Objective



For an MDP, we can have many policies. Some good, some not so good.

We seek the optimal policy (π^*) that provides the highest return: $J = \Sigma_i R(s_i, a_i)$

Reinforcement Learning: Problem



Find the optimal policy π^* for an MDP (S, A, T, R, γ) given the knowledge of (S, A, γ) and ability to interact with the environment.

Reinforcement Learning: Problem

Given

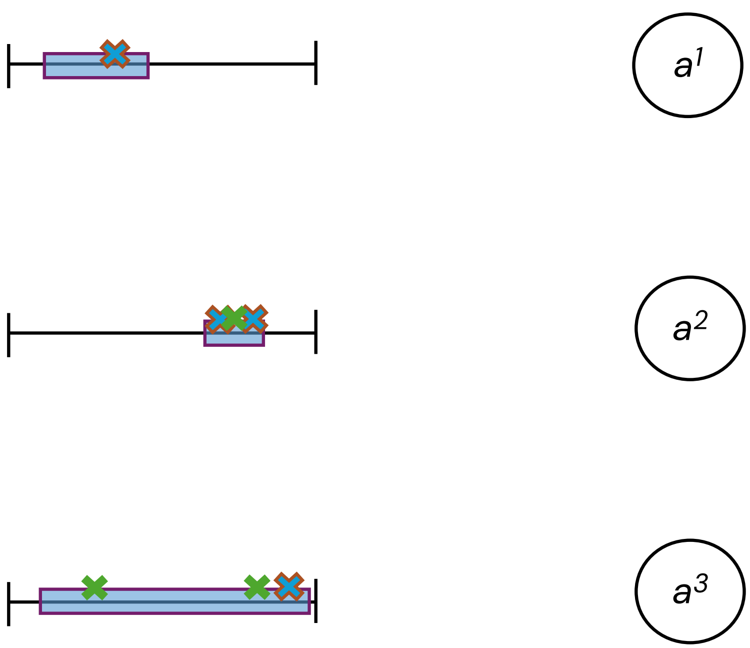
- The ability to collect training data, $D = \{(s_1, a_1), (s_2, a_2), \dots\}$ such that
$$s_{t+1} \sim T(\cdot\,|s_t, a_t)$$
- A class of models, $\pi: S \to \Pr(A) \in \Pi$
- A reward function, $R: S \times A \to \mathcal{R}$

Find

- The best model $\pi^* \in \Pi$ that maximizes the cumulative reward
$$\pi^* = \operatorname{argmax}_{\pi} \sum_i R\,(s_i, a_i)$$

Q Learning

Intuition: Making Decision in Bandits



Rewards
(provided by Environment)

Actions
(selected by Agent)

Bandits
(general case)

Exploration + Exploitation

First initialize action values, then iteratively

- Select
  - Greedy actions with high probability **(exploitation)**
  - Random actions with low but non-zero probability **(exploration)**
- Receive rewards from the environment
- Update estimates of action values

Intuition: Making Decisions in RL

Algorithm sketch

1. Start with a policy, π
2. Collect experience ($\mathcal{D}$) using the policy, π
3. Estimate value using data, $Q \leftarrow PolicyEvaluationUsingData(\pi, \mathcal{D})$
4. Improve policy based on the value, $\pi \leftarrow Policy\ Improvement(\pi, Q)$
5. Repeat 2-4, Stop when policy no longer improves

This is just one type of algorithmic sketch for RL, which results in many algorithms based on how each subroutine (2,3,4) is designed.

Q-Learning: Pseudocode

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

Q-Learning: Reducing Sample Complexity

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\textit{terminal}, \cdot) = 0$

Loop for each episode:
    Initialize S
    Loop for each step of episode:
        Choose A from S using policy derived from Q (e.g., ε-greedy)
        Take action A, observe R, S'
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
        $S \leftarrow S'$
    until S is terminal

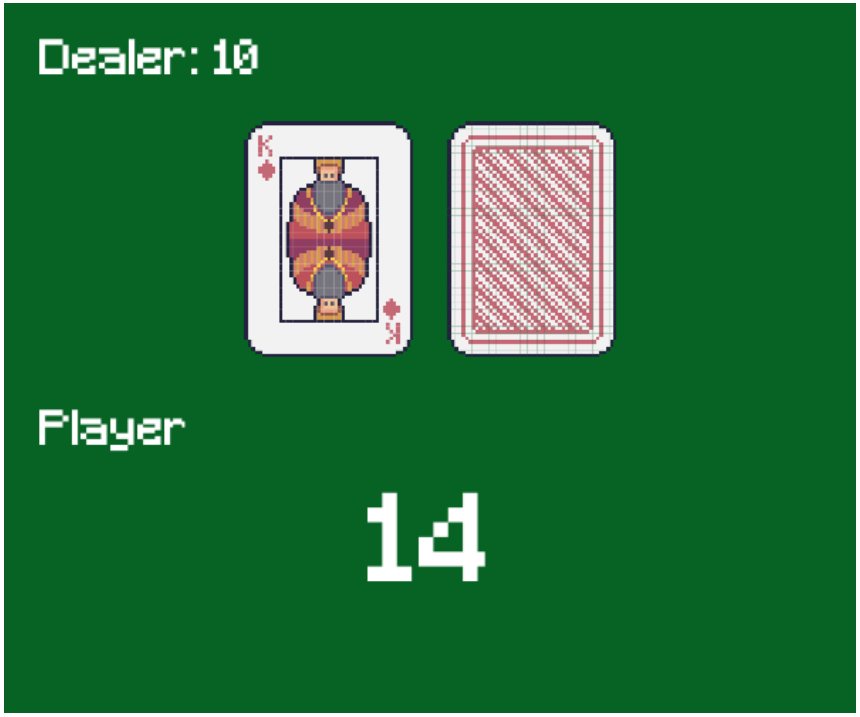
Q-Learning: Challenges in Scaling Up

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

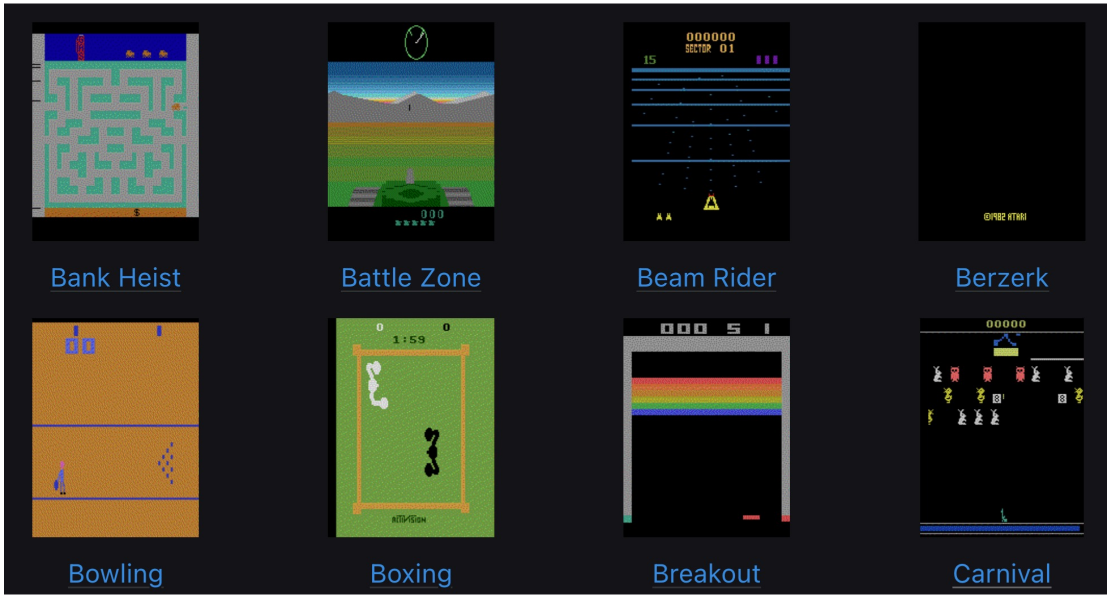
Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

Blackjack: `tiny.cc/rl-blackjack`

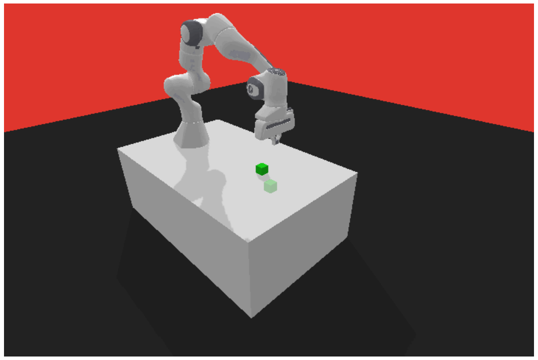


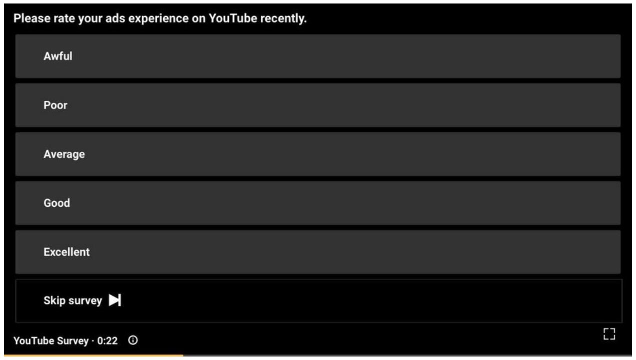
Deep Q Learning

Tabular RL approaches are intractable even for modestly complex games



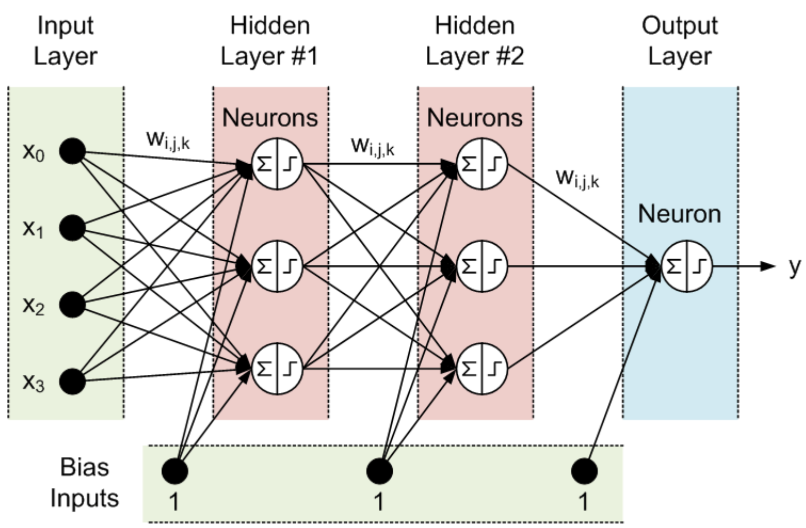
Tabular RL is almost always intractable for applications in the real world





How can we make RL tractable?

Function Approximation: Deep Learning + RL



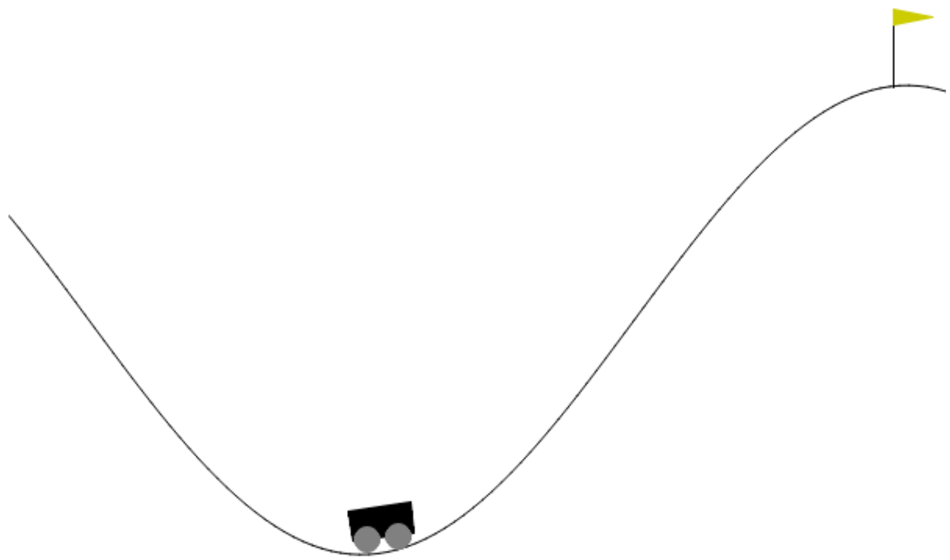
General Recipe for Deep Q Learning

1. Initialize optimal policy, value, and target network: π, $Q_{*,\phi}$, Q_{*,ϕ^T}
2. Collect J experiences using behavior policy and update replay buffer $\mathcal{D}_b$
3. Estimate value using data, $Q_{*,\phi} \leftarrow PolicyEvaluationUsingData(\pi, \mathcal{D}_b)$ by sampling a mini-batch of I experiences and updating Q
$$(s_i, a_i, r_i, s_i') \sim \mathcal{D}_b$$
$$\hat{G}_i = r_i + \max_a Q_{*,\phi^T}(s_i', a)$$
$$\phi \leftarrow \operatorname*{argmin}_{\phi} \mathcal{L}\left(Q_{*,\phi}(s_i, a_i) - \hat{G}_i\right)$$
4. Improve the policy based on value, $\pi \leftarrow Policy\ Improvement(\pi, Q_*)$
5. Update target network every K steps, $\phi^T \leftarrow \phi$
6. Repeat 2-5, Stop when policy no longer improves

Deep Q Network

1. Initialize optimal policy, value, and target network: π, $Q_{*,\phi}$, Q_{*,ϕ^T}
2. Collect J=1 experiences using behavior policy & update replay buffer $\mathcal{D}_b$
3. Estimate value using data, $Q_{*,\phi} \leftarrow PolicyEvaluationUsingData(\pi, \mathcal{D}_b)$ by sampling a mini-batch of I>1 experiences and updating Q
$$(s_i, a_i, r_i, s_i') \sim \mathcal{D}_b$$
$$\hat{G}_i = r_i + \max_a Q_{*,\phi^T}(s_i', a)$$
$$\phi \leftarrow \operatorname*{argmin}_{\phi} \mathcal{L}\left(Q_{*,\phi}(s_i, a_i) - \hat{G}_i\right)$$
4. Improve the policy based on value, $\pi \leftarrow Policy\ Improvement(\pi, Q_*)$
5. Update target network every K>1 steps, $\phi^T \leftarrow \phi$
6. Repeat 2-5, Stop when policy no longer improves

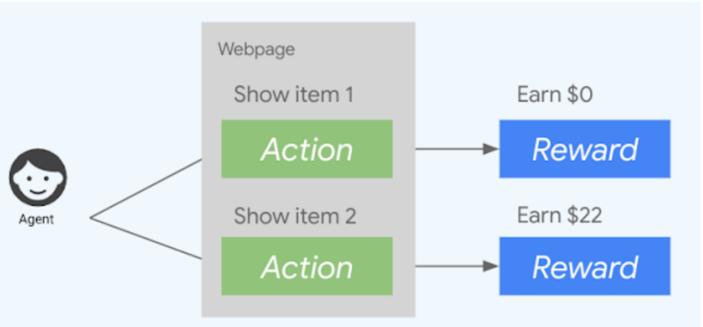
Blackjack: `tiny.cc/rl-mountain-car`



RL Safety

For more details, please see: Amodei, Dario, et al. "Concrete problems in AI safety." arXiv preprint arXiv:1606.06565 (2016).

Expected Values and Scalar Rewards



A simple bandit algorithm

Initialize, for $a = 1$ to k:

$$Q(a) \leftarrow 0$$
$$N(a) \leftarrow 0$$

Loop forever:

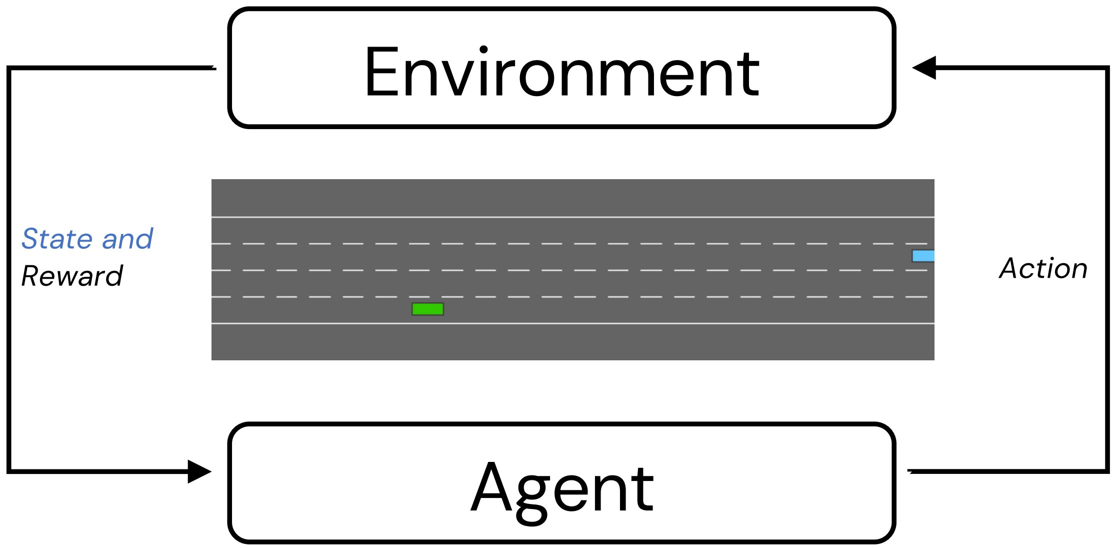
$$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$
$$R \leftarrow bandit(A)$$
$$N(A) \leftarrow N(A) + 1$$
$$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\big[R - Q(A)\big]$$

Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Q. Select the Optimal Action

Action	Pr(Reward | Action)
a_1	2 with probability 0.5 0 with probability 0.5
a_2	10 million w.p. 10^{-7} 0 otherwise
a_3	10 million w.p. 0.5 -9 million w.p. 0.5

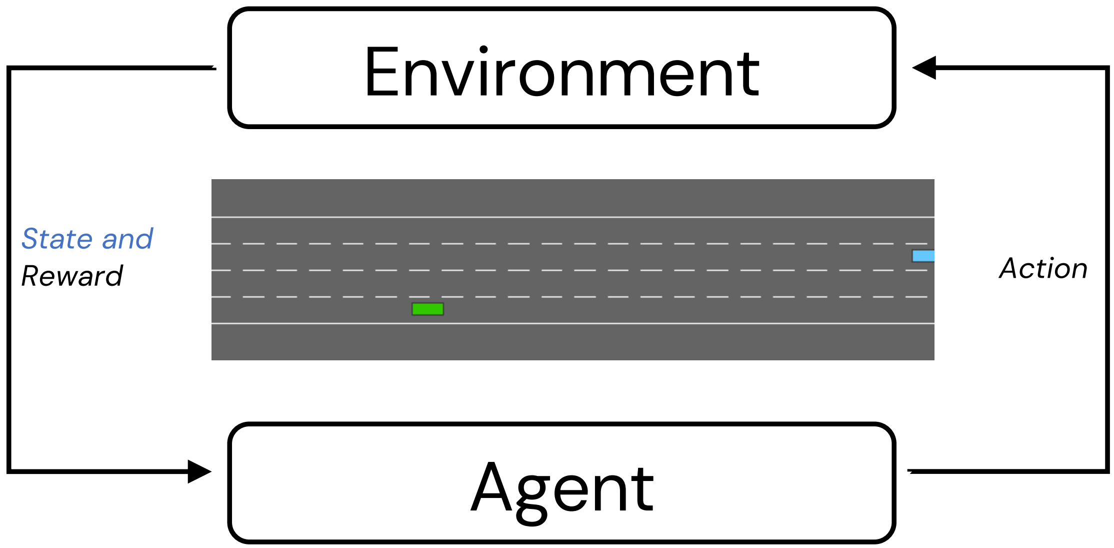
Ensuring Safe Exploration



Ensuring Safe Exploration

\

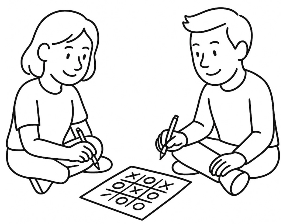
Preventing Reward Hacking

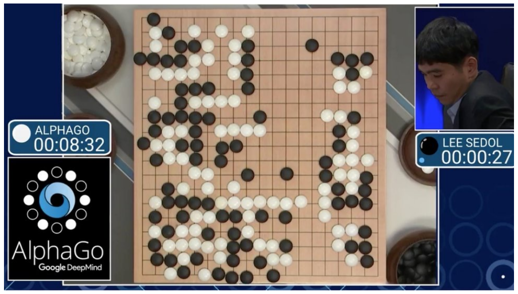


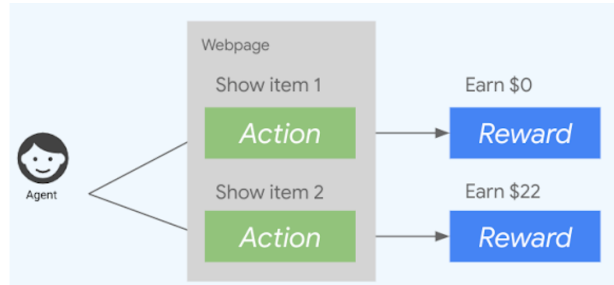
Improving Robustness to Distribution Shifts

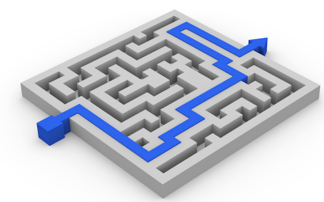
\

Concluding Remarks

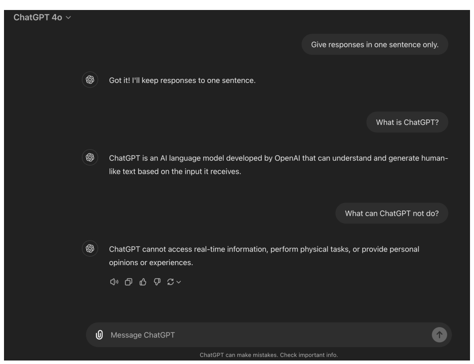


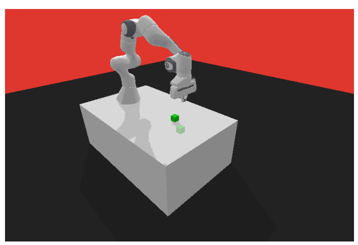


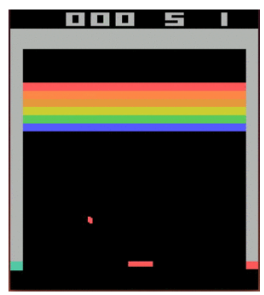


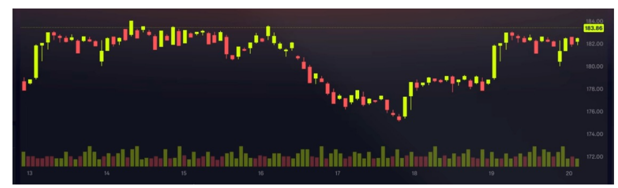


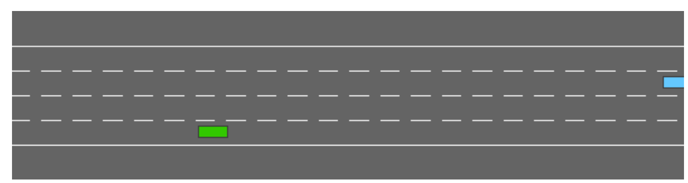
Reinforcement learning offers a **unified approach** to solve a **large class of decision-making problems.**











Sutton and Barto. Reinforcement Learning: An Introduction. MIT Press, 2018.

Environment and Agents

RL is particularly suited for sequential and stochastic problems for which specifying the desired outcome (reward signal) is easier than specifying how to achieve it (policy).