Introduction to Reinforcement Learning

Naeemullah Khan
naeemullah.khan@kaust.edu.sa

KAUST Academy
King Abdullah University of Science and Technology

June 30, 2025

Reinforcement Learning

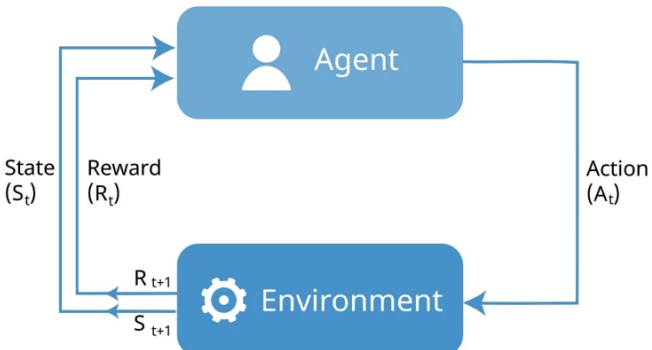
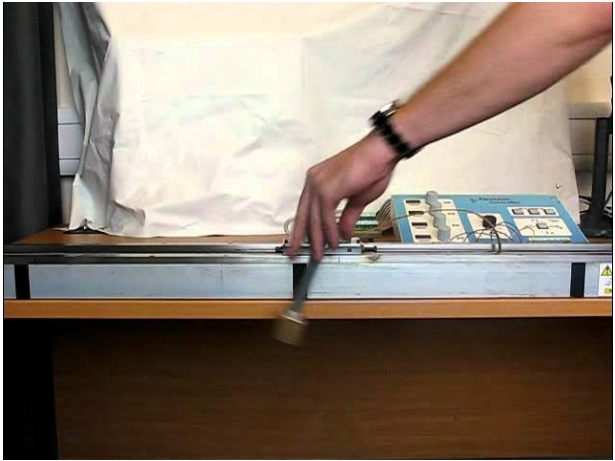


Table of Contents

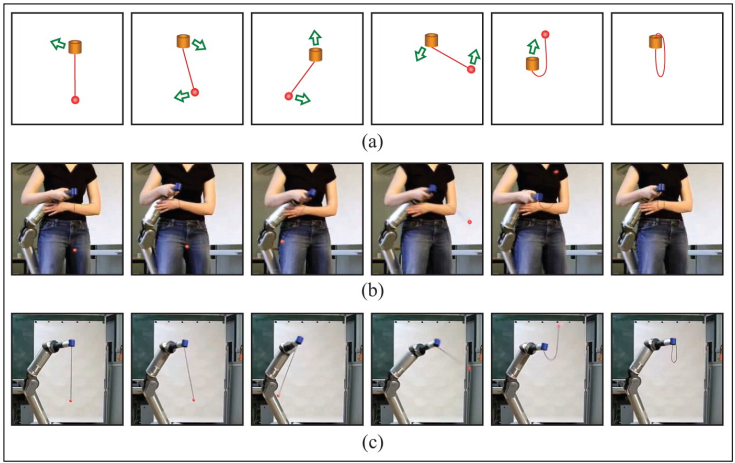
Table of Contents (cont.)

Motivation



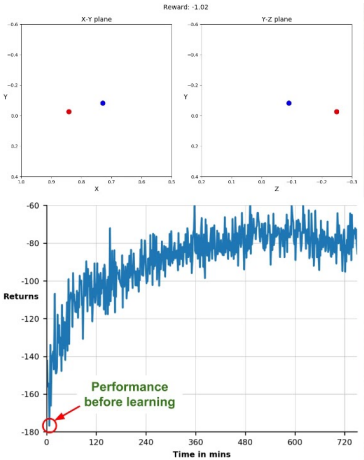
Cart-Pole Swing-Up Task

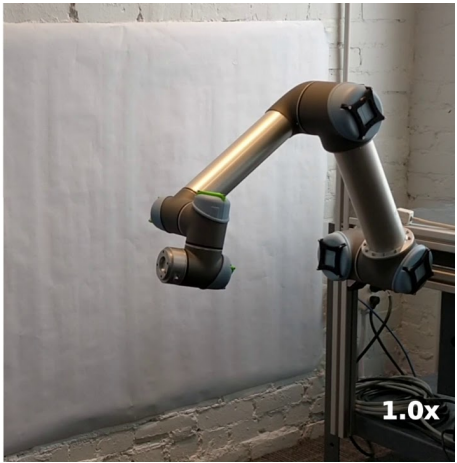
Motivation (cont.)



Robot Ball-in-a-Cup Task (Learning Motor Primitives in Robotics)

Motivation (cont.)



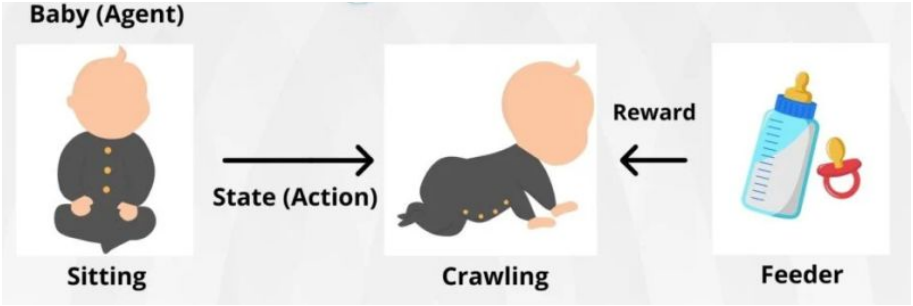


Robot Learning in Real-World Environments

Motivation (cont.)

- Many AI problems involve **sequential decision-making** under uncertainty.
- Inspired by **animal learning** (reward-based adaptation).
- Reinforcement Learning (RL) allows agents to learn optimal behaviors by **trial and error**.
- Real-world success: AlphaGo, Robotics, Game AI, Recommendation Systems.

Motivation (cont.)



Baby Learning to Crawl towards a Milk Bottle (Reward)

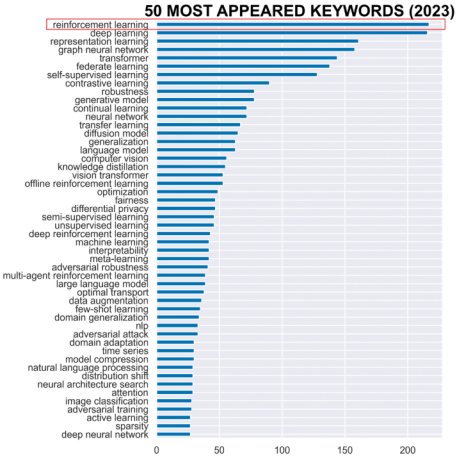
Learning Outcomes

By the end of this session, you will be able to:

- Understand the fundamental **principles and structure of RL**.
- Model problems using **Markov Decision Processes (MDPs)**.
- Implement and apply **Q-learning** to solve control problems.
- Identify **practical applications and challenges** in RL.
- Critically analyze the **limitations and future trends** of RL.

Reinforcement Learning: **Introduction**

RL - ICLR 2023



Source: ICLR 2023 - Open Review Data

Definition: RL is a type of machine learning where agents learn to make decisions by interacting with an environment.

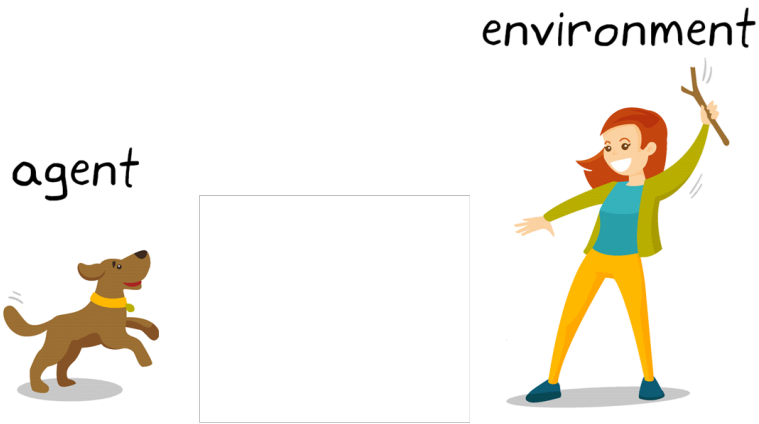
Core Elements:

- **Agent**: The learner or decision maker.
- **Environment**: The external system the agent interacts with.
- **State**: A representation of the current situation.
- **Action**: Choices the agent can make.
- **Reward**: Feedback signal for evaluating actions.

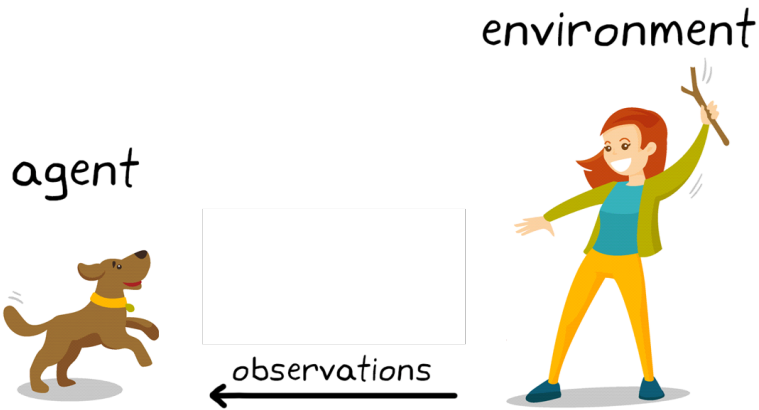
Introduction (cont.)

agent

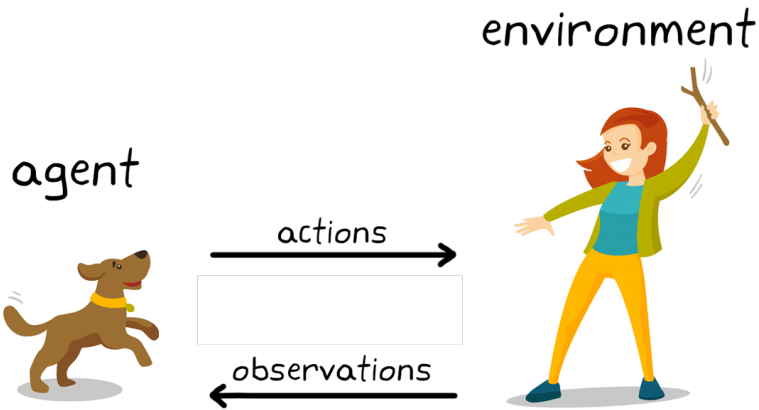
Introduction (cont.)



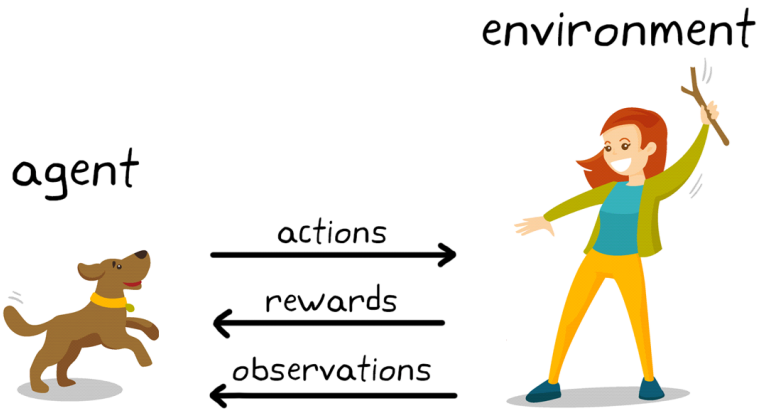
Introduction (cont.)



Introduction (cont.)



Introduction (cont.)



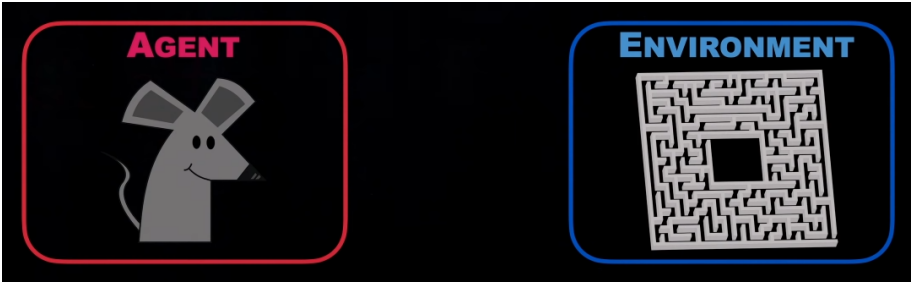
Introduction (cont.)

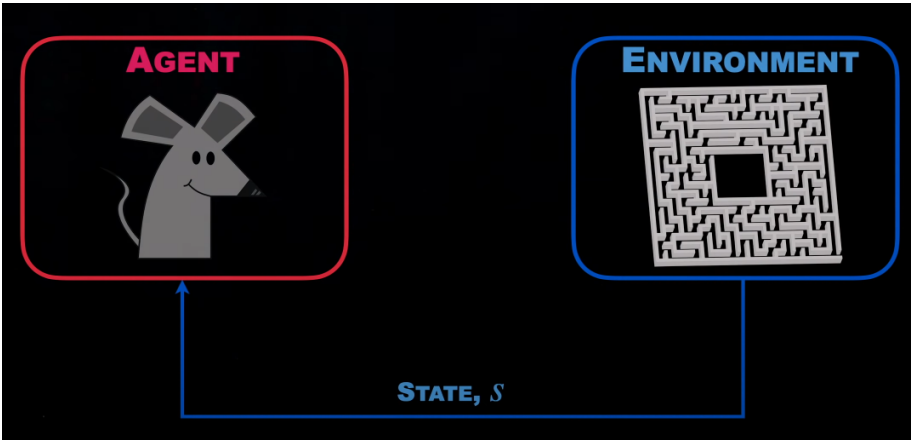
Key Concepts:

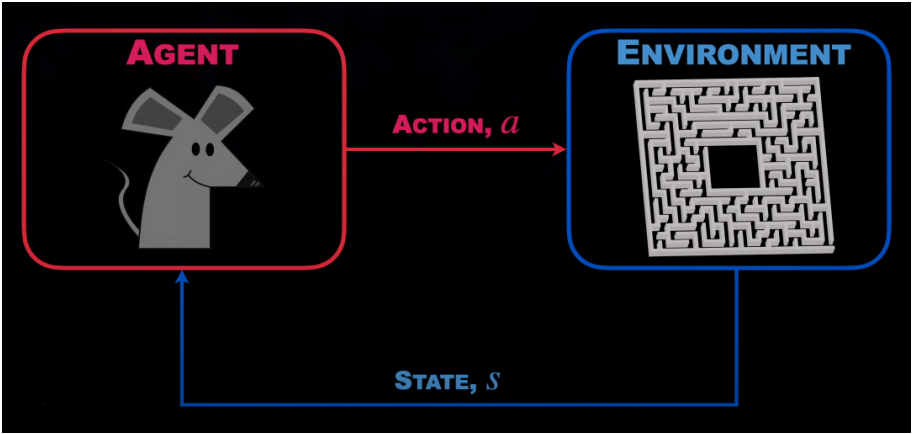
- Agent interacts with environment.
- Learns by trial and error.

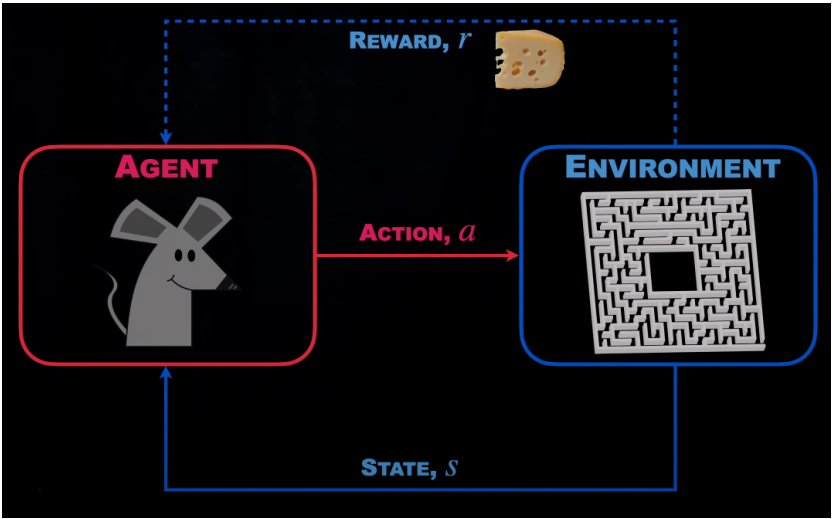
How does it work?

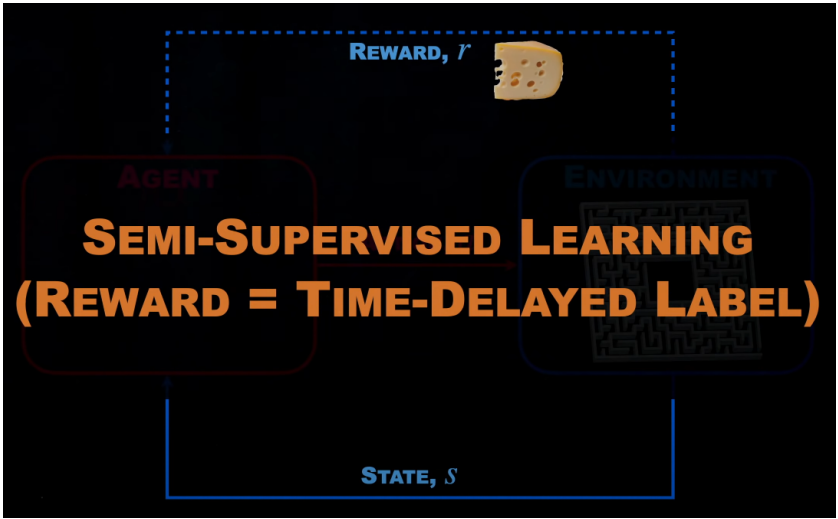
- Agent gets rewards or penalties.
- Goal: Maximize cumulative reward (a.k.a. return).

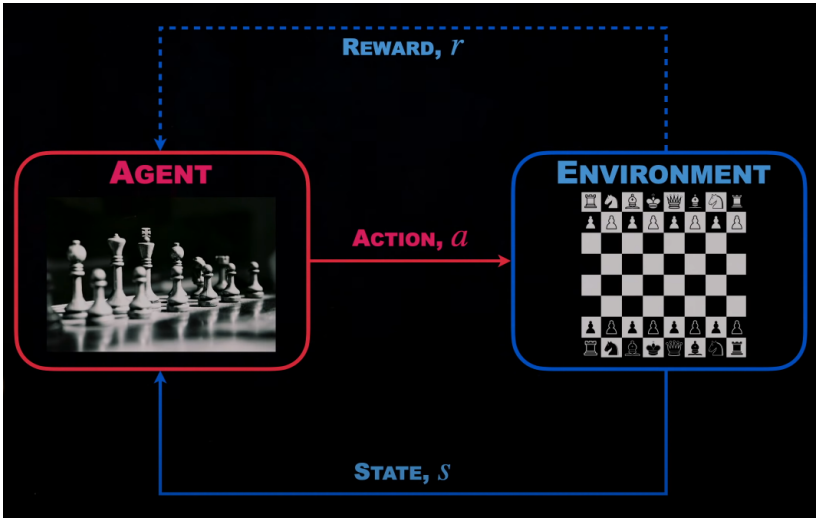

AGENT
ENVIRONMENT

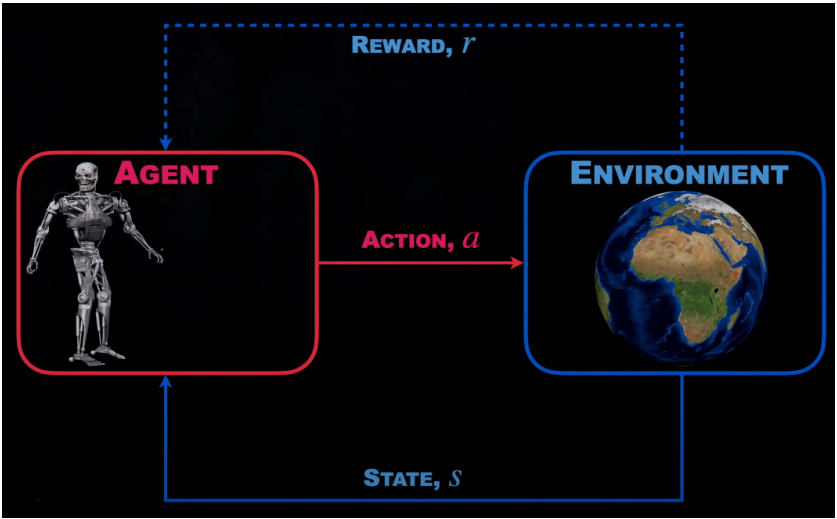

AGENT
ENVIRONMENT
STATE, S

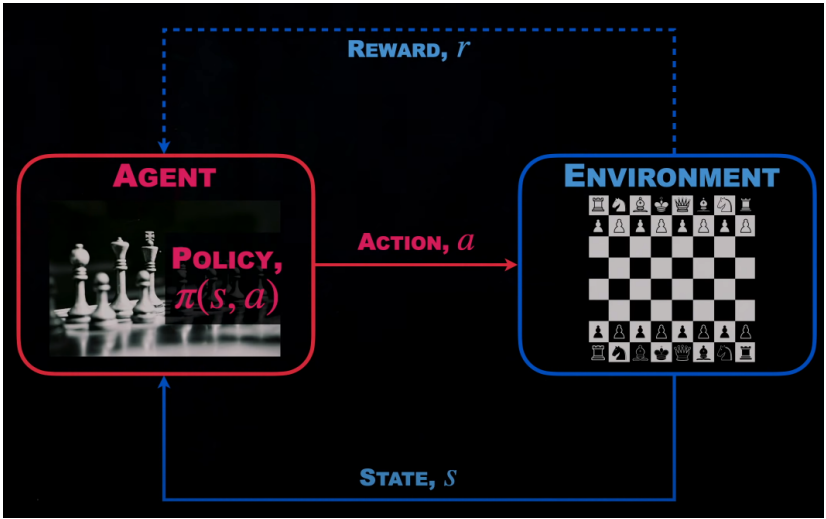

AGENT
ACTION, a
ENVIRONMENT
STATE, s

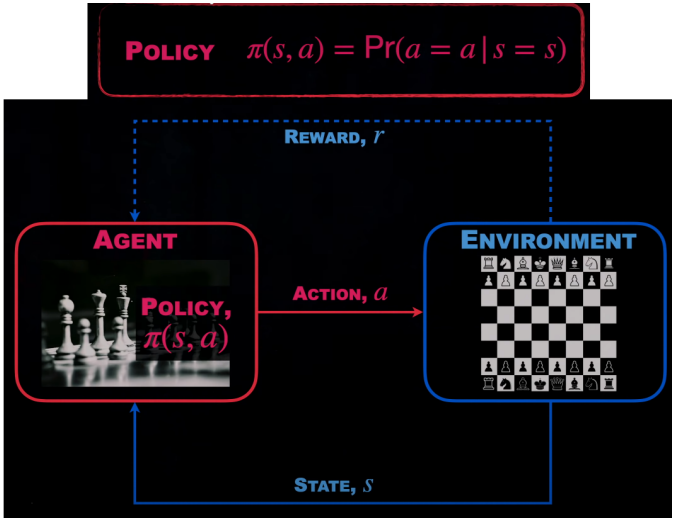

REWARD, r
AGENT
ENVIRONMENT
ACTION, a
STATE, s

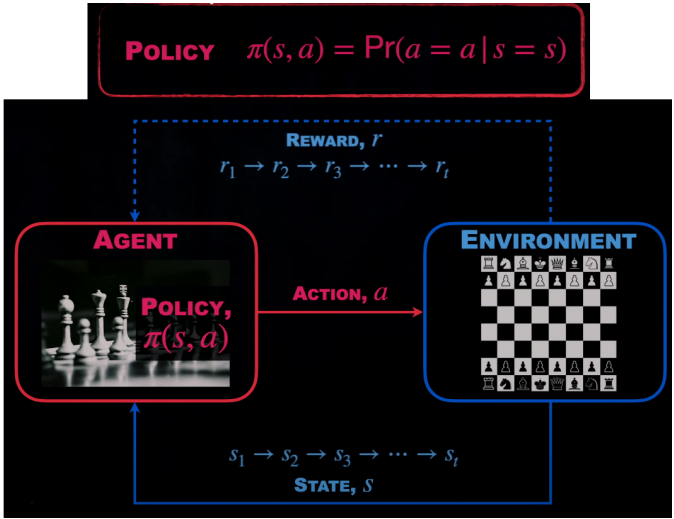

REWARD, r
AGENT
ENVIRONMENT
SEMI-SUPERVISED LEARNING
(REWARD = TIME-DELAYED LABEL)
STATE, s

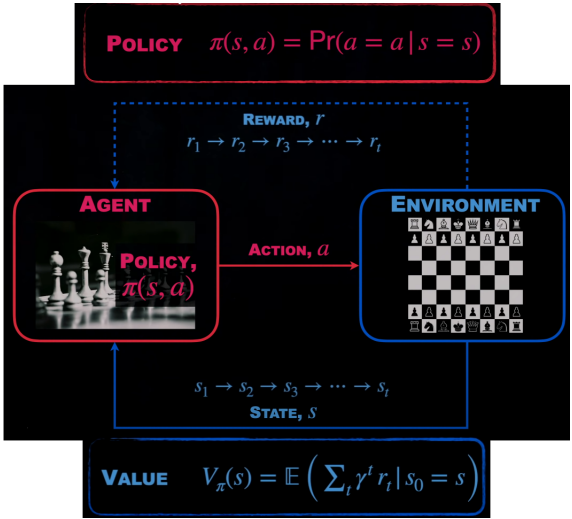

REWARD, r
AGENT
ACTION, a
ENVIRONMENT
STATE, s

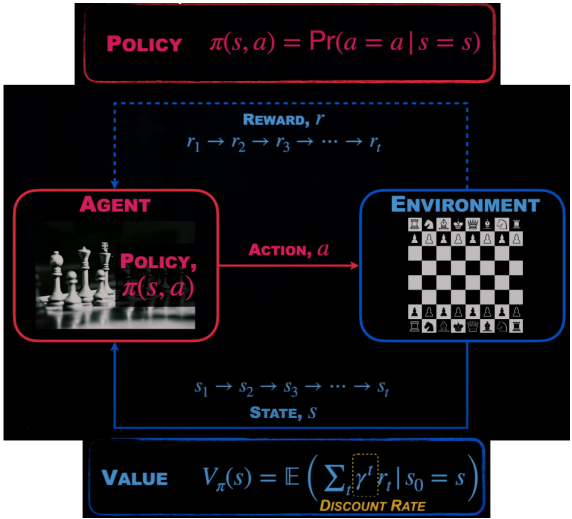

REWARD, r
AGENT
ENVIRONMENT
ACTION, a
STATE, s

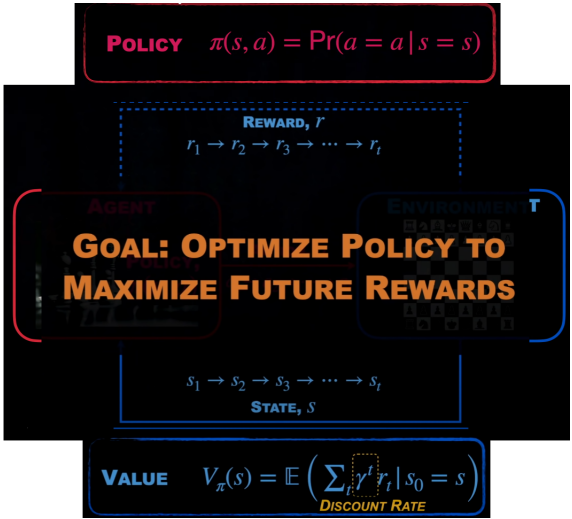

REWARD, r
AGENT
POLICY, $\pi(s, a)$
ACTION, a
ENVIRONMENT
STATE, s


POLICY $\pi(s,a) = \Pr(a = a \mid s = s)$
REWARD, r
AGENT
POLICY, $\pi(s,a)$
ACTION, a
ENVIRONMENT
STATE, s


POLICY $\pi(s,a) = \Pr(a = a \mid s = s)$
REWARD, r
$r_1 \to r_2 \to r_3 \to \cdots \to r_t$
AGENT
POLICY, $\pi(s,a)$
ACTION, a
ENVIRONMENT
$s_1 \to s_2 \to s_3 \to \cdots \to s_t$
STATE, s


POLICY $\pi(s,a) = \Pr(a = a \mid s = s)$
REWARD, r
$r_1 \to r_2 \to r_3 \to \cdots \to r_t$
AGENT
POLICY, $\pi(s,a)$
ACTION, a
ENVIRONMENT
$s_1 \to s_2 \to s_3 \to \cdots \to s_t$
STATE, s
VALUE $V_\pi(s) = \mathbb{E}\left(\sum_t \gamma^t r_t \mid s_0 = s\right)$


POLICY $\pi(s,a) = \Pr(a = a \mid s = s)$
REWARD, r
$r_1 \to r_2 \to r_3 \to \cdots \to r_t$
AGENT
POLICY, $\pi(s,a)$
ACTION, a
ENVIRONMENT
$s_1 \to s_2 \to s_3 \to \cdots \to s_t$
STATE, s
VALUE $V_\pi(s) = \mathbb{E}\left(\sum_t \gamma^t r_t \mid s_0 = s\right)$
DISCOUNT RATE


POLICY $\pi(s,a) = \Pr(a = a \mid s = s)$
REWARD, r
$r_1 \to r_2 \to r_3 \to \cdots \to r_t$
AGENT
ENVIRONMENT
GOAL: OPTIMIZE POLICY TO MAXIMIZE FUTURE REWARDS
$s_1 \to s_2 \to s_3 \to \cdots \to s_t$
STATE, s
VALUE $V_\pi(s) = \mathbb{E}\left(\sum_t \gamma^t r_t \mid s_0 = s\right)$
DISCOUNT RATE

Supervised Learning vs. Reinforcement Learning

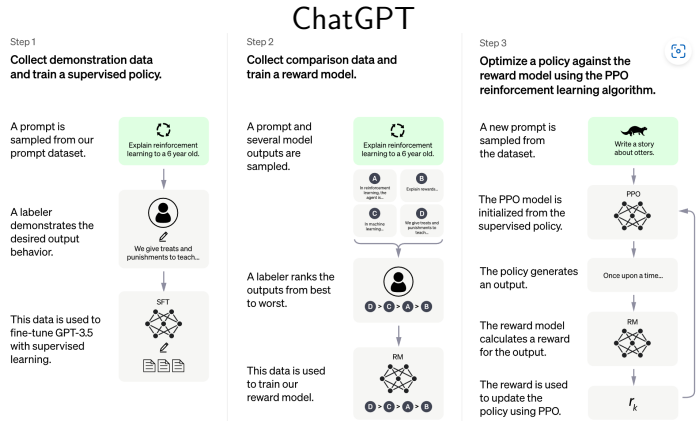
Supervised Learning

- Given labeled data: $\{(x_i, y_i)\}$, learn $f(x) \approx y$
- Directly told what to output
- Inputs x are independently, identically distributed (i.i.d.)

Reinforcement Learning

- Learn behavior $\pi(a|s)$
- Learn from experience, indirect feedback
- Data not i.i.d.: actions affect future observations

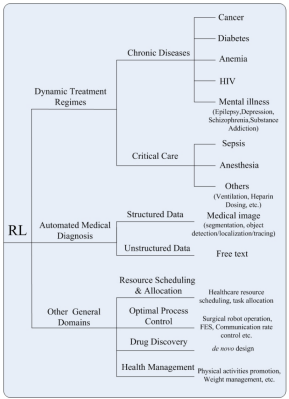
Examples of Reinforcement Learning



https://openai.com/blog/chatgpt

Examples of Reinforcement Learning (cont.)

Healthcare

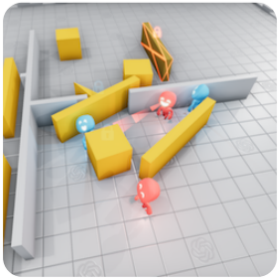


Yu, et al. Reinforcement Learning in Healthcare: A Survey

Examples of Reinforcement Learning (cont.)

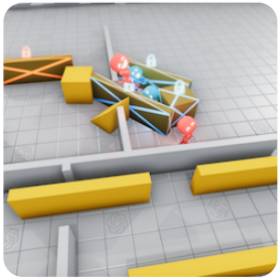
Multiagent Hide & Seek

(a) Running and Chasing



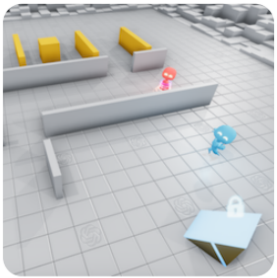
(b) Fort Building

(c) Ramp Use

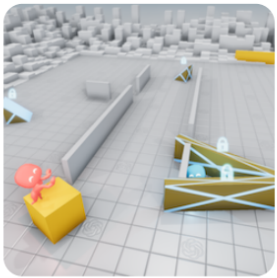


Examples of Reinforcement Learning (cont.)

Multiagent Hide & Seek



(d) Ramp Defense



(e) Box Surfing

(f) Surf Defense

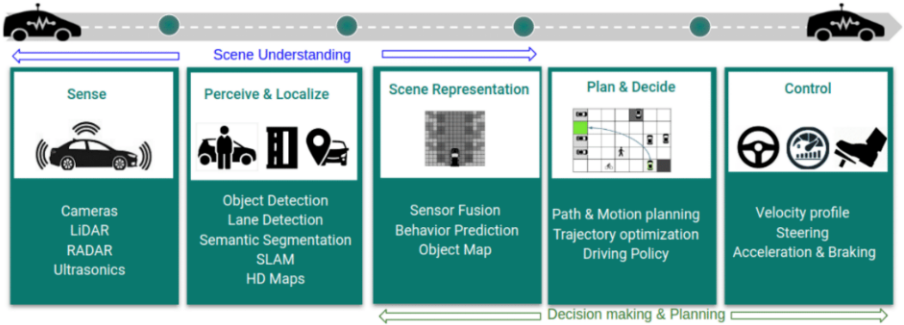
Examples of Reinforcement Learning (cont.)

Gran Turismo GTSophie

https://www.gran-turismo.com/us/gran-turismo-sophy/technology/

Examples of Reinforcement Learning (cont.)

Autonomous Driving



Kiran, et al. Deep Reinforcement Learning for Autonomous Driving: A Survey

Examples of Reinforcement Learning (cont.)

AlphaGo

'AlphaGo' AI Scores Narrow Win Against Ke Jie, World's Top 'Go' Player

By Lucian Armasu published May 23, 2017

Comments (4)

https://www.tomshardware.com/news/alphago-narrow-win-ke-jie,34486.html

Examples of Reinforcement Learning (cont.)

https://www.deepmind.com/blog/muzero-mastering-go-chess-shogi-and-atari-without-rules

Examples of Reinforcement Learning (cont.)

Minecraft

AI learns how to play Minecraft by watching videos

Open AI has trained a neural network to play Minecraft by Video PreTraining (VPT) on a massive unlabeled video dataset of human Minecraft play, while using just a small amount of labeled contractor data.

With a bit of fine-tuning, the AI research and deployment company is confident that its model can learn to craft diamond tools, a task that usually takes proficient humans over 20 minutes (24,000 actions). Its model uses the native human interface of keypresses and mouse...

29 June 2022 | Artificial Intelligence

https://www.artificialintelligence-news.com/2022/06/29/ai-learns-how-to-play-minecraft-by-watching-videos/

Examples of Reinforcement Learning (cont.)

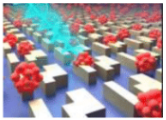
More Examples

Reinforcement learning: From board games to protein design

Scientists have successfully applied reinforcement learning to a challenge in molecular biology. The team of researchers developed powerful new protein design software adapted from a strategy proven adept at board games like ...

BIOTECHNOLOGY

APR 20, 2023 0 37



Chiral detection of biomolecules based on reinforcement learning

As one of the basic physical properties, chirality plays an important role in many fields. Especially in biomedical chemistry, the discrimination of enantiomers is a very important research subject. Most biomolecules exhibit ...

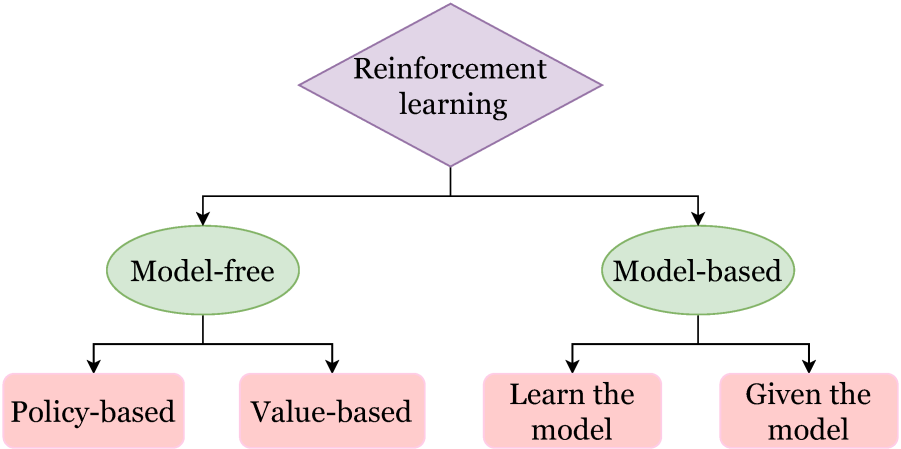
OPTICS & PHOTONICS

FEB 22, 2023 0 4

https://phys.org/tags/reinforcement+learning/

Reinforcement Learning: **Types**

Types of Reinforcement Learning



Types of Reinforcement Learning (cont.)

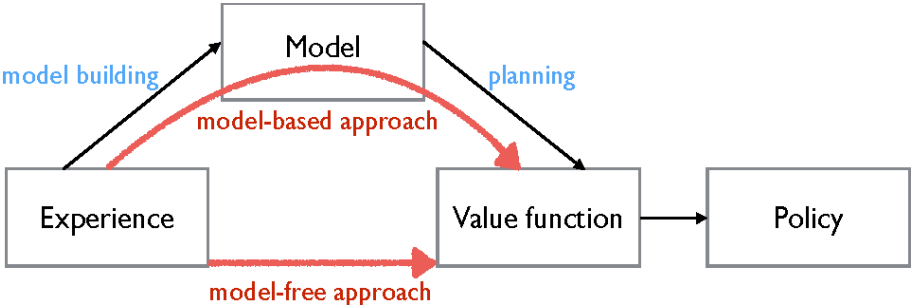
Model-based vs. Model-free

- **Model-based:** The agent builds or uses a model of the environment to plan actions.
- **Model-free:** The agent learns to act without explicitly modeling the environment.

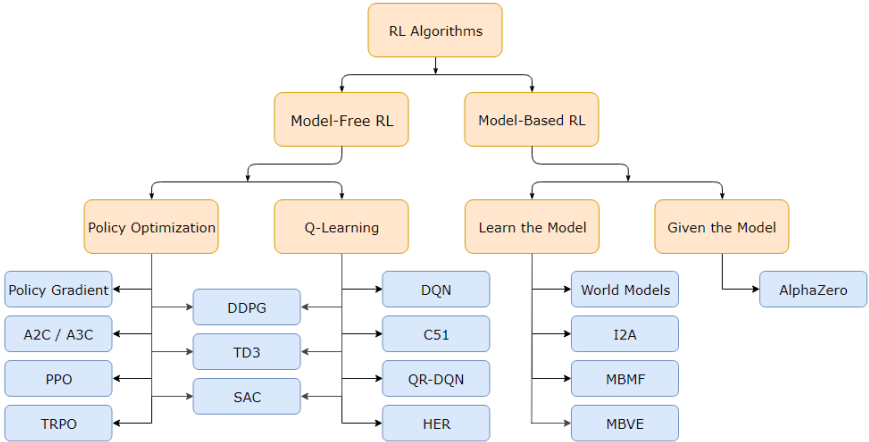
Value-based vs. Policy-based

- **Value-based:** The agent learns a value function (e.g., Q-learning) to evaluate actions or states.
- **Policy-based:** The agent directly learns a policy that maps states to actions (e.g., Policy Gradient methods).

Types of Reinforcement Learning (cont.)



Types of Reinforcement Learning (cont.)

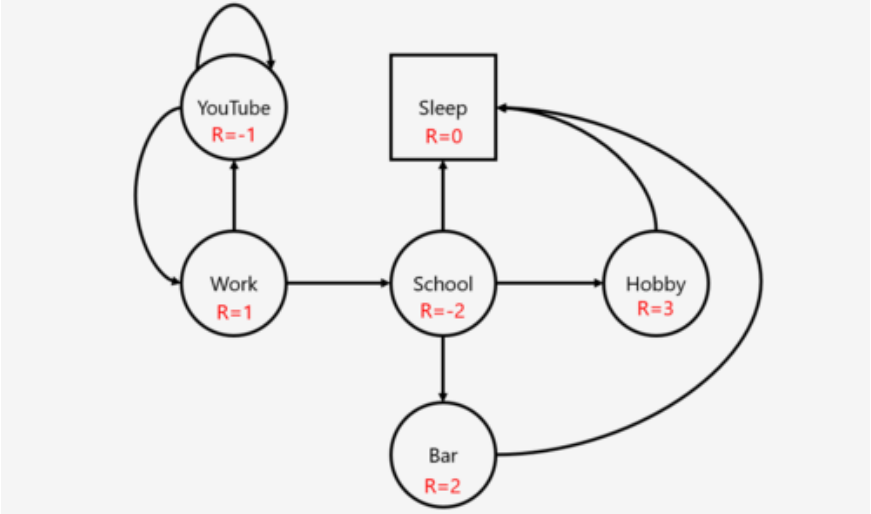


Reinforcement Learning: **Markov Decision Processes (MDPs)**

Mathematical Formulation of the RL Problem

- **Markov property:** The current state completely characterizes the state of the world; that is, the future is independent of the past given the present.
- **Defined by:** $(\mathcal{S}, \mathcal{A}, R, P, \gamma)$
  - $\mathcal{S}$: set of possible states
  - $\mathcal{A}$: set of possible actions
  - $R(s, a)$: reward distribution given a (state, action) pair
  - $P(s' \mid s, a)$: transition probability, i.e., distribution over next state given a (state, action) pair
  - γ: discount factor
- Assumes the **Markov property**: the next state depends only on the current state and action.

Mathematical Formulation of the RL Problem (cont.)



Mathematical Formulation of the RL Problem (cont.)

- At time step $t = 0$, the environment samples the initial state $s_0 \sim p(s_0)$.
- Then, for $t = 0$ until termination:
  - The agent selects action a_t.
  - The environment samples reward $r_t \sim R(\cdot|s_t, a_t)$.
  - The environment samples the next state $s_{t+1} \sim P(\cdot|s_t, a_t)$.
  - The agent receives reward r_t and next state s_{t+1}.
- A policy π is a function from $\mathcal{S}$ to $\mathcal{A}$ that specifies which action to take in each state.
- **Objective:** Find a policy $\pi^\star$ that maximizes the cumulative discounted reward: $\sum_{t \geq 0} \gamma^t r_t$.

Mathematical Formulation of the RL Problem (cont.)

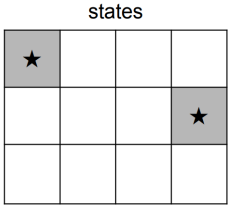
- The agent and the environment interact in a time-sequenced loop:
  - The agent observes the current state and reward, then selects an action.
  - The environment responds by producing the next state and a scalar reward.
- Each state satisfies the **Markov property**:
  - The next state and reward depend only on the current state and action.
  - The current state captures all relevant information from the history.
  - The current state is a sufficient statistic for predicting the future (given the action).
- The goal of the agent is to maximize the expected sum of all future rewards by controlling the policy function $\pi : \mathcal{S} \rightarrow \mathcal{A}$.
- This setup forms a dynamic, time-sequenced control system under uncertainty.

A Simple MDP: Grid World

actions = {

1. right
2. left
3. up
4. down

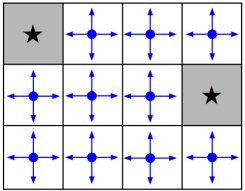
}



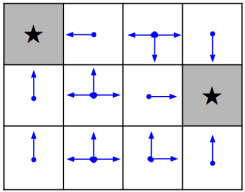
Set a negative "reward" for each transition (e.g. r = -1)

Objective: Reach one of the terminal states (greyed out) in the least number of actions.

A Simple MDP: Grid World (cont.)



Random Policy



Optimal Policy

Real-World Problems that Fit the MDP Framework

- Self-driving vehicles: choosing speed and steering to optimize safety and travel time
- Playing chess: receiving a reward only at the end of the game
- Complex logistical operations, such as managing movements in a warehouse
- Enabling a humanoid robot to walk or run on challenging terrains
- Managing an investment portfolio over time
- Controlling a power station efficiently
- Making optimal decisions during a football match
- Developing strategies to win an election (a high-complexity MDP)

Why Are These Problems Hard?

- The state space can be large or complex (involving many variables).
- Sometimes, the action space is also large or complex.
- There is no direct feedback on "correct" actions (the only feedback is the reward).
- Time-sequenced complexity: actions influence future states and actions.
- Actions can have delayed consequences (delayed rewards).
- The agent often does not know the model of the environment.
- The "model" refers to the probabilities of state transitions and rewards.
- Thus, the agent has to learn the model and solve for the optimal policy.
- Agent actions need to trade off between "exploration" and "exploitation".

Reinforcement Learning: **Policy**

Policy

- Policy π determines how the agent chooses actions
- $\pi : S \rightarrow A$, mapping from states to actions
- Deterministic Policy:

$$\pi(s) = a$$

- Stochastic Policy:

$$\pi(a|s) = Pr(a_t = a|s_t = s)$$

The Optimal Policy

- We want to find optimal policy $\pi^\star$ that maximizes the sum of reward
- But, how do we handle the randomness (initial state, transition probability...)?

The Optimal Policy

- We want to find optimal policy $\pi^\star$ that maximizes the sum of reward
- But, how do we handle the randomness (initial state, transition probability...)?
- **Solution**: Maximize the expected sum of rewards!
- Formally,

$$\pi^\star = \arg\max_\pi \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | \pi\right] \quad \text{with} \quad s_0 \sim p(s_0), a_t \sim \pi(.|s_t), s_{t+1} \sim p(.|s_t, a_t)$$

Reinforcement Learning: **Value Function**

Value Function

- How good is a state?

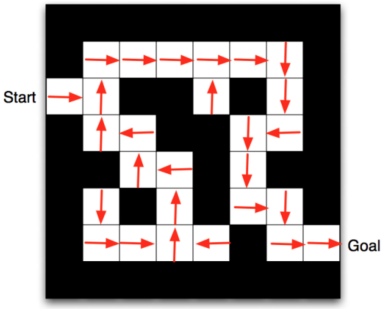
Value Function

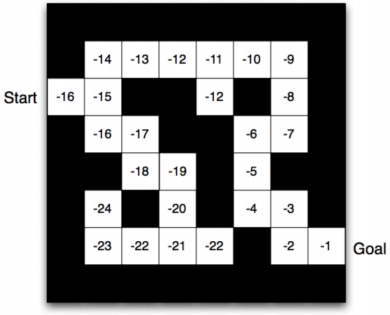
- How good is a state?
- The **value function** at state s, is the expected cumulative reward from following the policy from state s:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^{t} r_{t} | s_{0} = 0, \pi\right]$$

- Computing the value function is generally impractical, but we can try to approximate (learn) it
- The benefit is credit assignment: see directly how an action affects future returns rather than wait for rollouts

Value Function





Q-Value Function

- Can we use a value function to choose actions?

$$\arg\max_{a} r(s_t, a) + \gamma \mathbb{E}_{p(s_{t+1}|s_t,a_t)} \left[v^{\pi}(s_{t+1}) \right]$$

Q-Value Function

- Can we use a value function to choose actions?

$$\arg\max_a r(s_t, a) + \gamma \mathbb{E}_{p(s_{t+1}|s_t,a_t)} \left[v^\pi(s_{t+1}) \right]$$

- **Problem:** this requires taking the expectation with respect to the environment's dynamics, which we don't have direct access to!
- Instead, learn a Q-value function.

Q-Value Function

- The Q-value function at state s and action a, is the expected cumulative reward from taking action a in state s and then following the policy:

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^{t} r_{t} | s_{0} = 0, a_{0} = a, \pi\right]$$

- Relationship:

$$V^{\pi}(s) = \sum_{a} \pi(a|s) Q^{\pi}(s,a)$$

- Optimal Action:

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

Bellman Equation

- The Bellman Equation is a recursive formula for the Q-value function:

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

- There are various Bellman equations, and most RL algorithms are based on repeatedly applying one of them.

Bellman Equations:

- For the value function $V^{\pi}(s)$:

$$V^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma V^{\pi}(s')\right]$$

- For the Q-value function $Q^{\pi}(s,a)$:

$$Q^{\pi}(s,a) = \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma \sum_{a'} \pi(a'|s') Q^{\pi}(s',a')\right]$$

Optimal Bellman Equation

- The optimal policy $\pi^\star$ is the one that maximizes the expected discounted reward, and the optimal Q-value function $Q^\star$ is the Q-value function for $\pi^\star$
- The Optimal Bellman Equation gives a recursive formula for $Q^\star$:

$$Q^\pi(s,a) = r(s,a) + \gamma \mathbb{E}_{p(s'|s,a)} \left[\max_{a'} Q^\pi(s_{t+1}, a') | s_t = s, a_t = a \right]$$

- This system of equations characterizes the optimal Q-value function. So maybe we can approximate $Q^\star$ by trying to solve the optimal Bellman equation!
- **Intuition**: If the optimal state-action values for the next time-step Q*(s',a') are known, then the optimal strategy is to take the action that maximizes the expected value of $r(s,a) + \gamma Q^\pi(s', a')$

Reinforcement Learning: **Q-Learning**

Q-Learning

- Q-learning is a **model-free** reinforcement learning algorithm
- It **learns the value of actions in states**, known as the **Q-value function**
- The goal is to find the optimal policy that maximizes the expected cumulative reward
- Q-learning is off-policy, meaning it learns the value of the optimal policy independently
- It can be used in environments with unknown dynamics, making it suitable for a wide range of problems

Q-Learning (cont.)

- Let Q be an action-value function which hopefully approximates Q^*
- Q-learning is an algorithm that repeatedly adjusts Q to minimize the Bellman error
- Each time we sample consecutive states and actions $(s_t, a_t, s_{t+1}$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \underbrace{\left[r(s, a) + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right]}_{\text{Bellman Error}}$$

Exploration-Exploitation Tradeoff

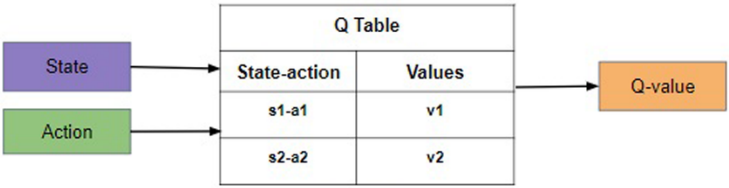
- Notice: Q-learning only learns about the states and actions it visits.
- Exploration-exploitation tradeoff: the agent should sometimes pick suboptimal actions in order to visit new states and actions.
- Simple solution: ϵ-greedy policy
  - With probability $1 - \epsilon$, choose the optimal action according to Q
  - With probability ϵ, choose a random action
- ϵ-greedy policy still widely used today regardless of simplicity

Q-Learning Algorithm

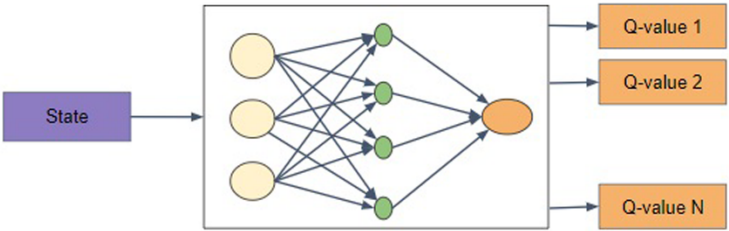
Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, 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

Convergence Properties

- Q-learning converges to the optimal action-value function Q^* under certain conditions:
  - **Sufficient exploration:** Every state-action pair is visited infinitely often.
  - **Learning rate:** The learning rate α decays appropriately, i.e., $\alpha \to 0$ as the number of visits increases, but $\sum_t \alpha_t = \infty$ and $\sum_t \alpha_t^2 < \infty$.
  - **Finite MDP:** The environment is a finite Markov Decision Process.
- Under these conditions, Q-learning is guaranteed to converge to Q^* with probability 1.



Q-Learning



Deep Q-Learning

Reinforcement Learning: **Limitations**

Limitations of Reinforcement Learning

- **Sample inefficiency**: Requires a large number of interactions with the environment to learn effectively.
- **Stability issues with function approximation**: Training with neural networks can be unstable and sensitive to hyperparameters.
- **Sparse or delayed rewards**: Learning is difficult when rewards are infrequent or delayed.
- **Requires careful reward design**: Poorly designed rewards can lead to unintended behaviors.
- **Poor generalization across tasks**: Policies often do not transfer well to new or slightly different tasks.

Reinforcement Learning: **Summary**

Summary

- Reinforcement Learning (RL) is a powerful paradigm for **sequential decision-making**.
- Markov Decision Processes (MDPs) provide a **mathematical foundation**.
- Q-learning enables **model-free control**.
- Despite its power, RL has **significant limitations**.
- Research is ongoing to make RL **scalable, robust, and safe**.

Reinforcement Learning: **References**

References

[1] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.).

[2] Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). Human-level control through deep reinforcement learning. *Nature*.

[3] Silver, D., Huang, A., Maddison, C. J., et al. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*.

[4] Levine, S., Kumar, A., Tucker, G., & Fu, J. (2020). Offline Reinforcement Learning: Tutorial, Review, and Perspectives. *arXiv:2005.01643*.

[5] David Silver's RL Course: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html

References (cont.)

[6] Berkeley CS285: Deep RL
https://rail.eecs.berkeley.edu/deeprlcourse/

[7] Emma Brunskill, Stanford CS234: Reinforcement Learning
https://web.stanford.edu/class/cs234/

Credits

Dr. Prashant Aparajeya

Computer Vision Scientist — Director(AISimply Ltd)

p.aparajeya@aisimply.uk

This project benefited from external collaboration, and we acknowledge their contribution with gratitude.