AI VIETNAM
All-in-One Course
(TA Session)

Introduction Reinforcement to Learning and Q-Learning

Extra Class: RL

AI VIET NAM
@aivietnam.edu.vn

Nguyen-Thuan Duong – TA

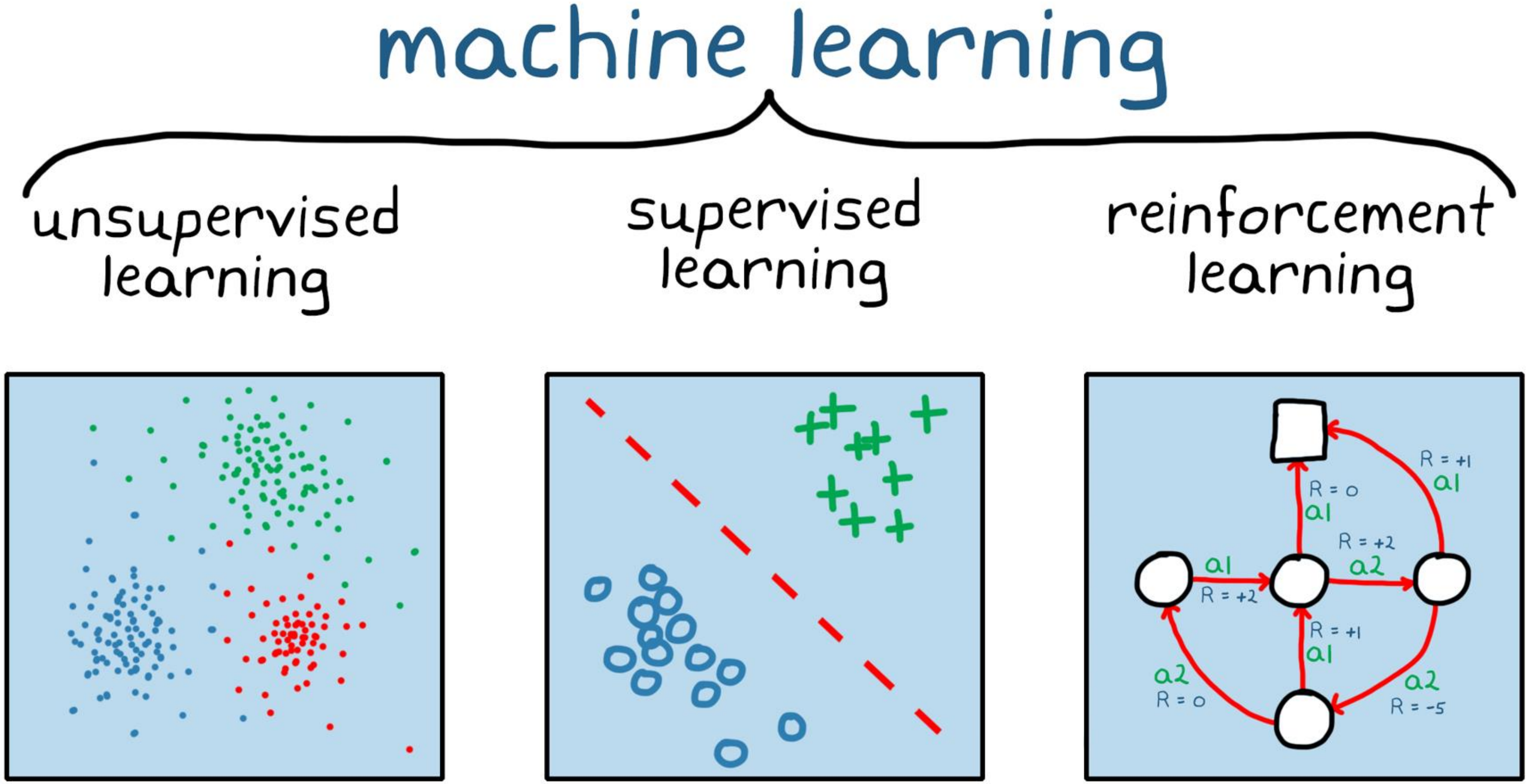
Year 2024

Outline

Introduction

Introduction

❖ Getting Started



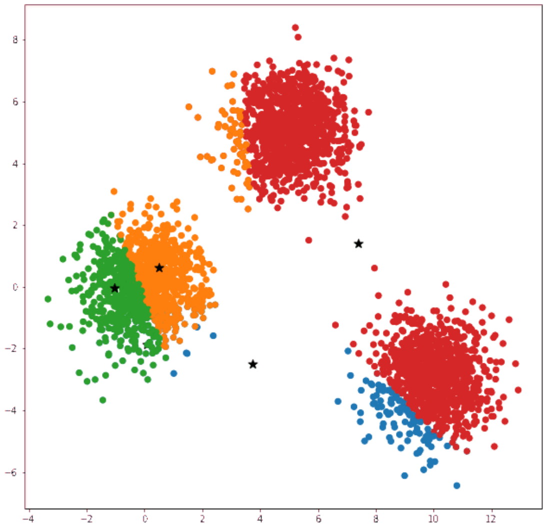
Introduction

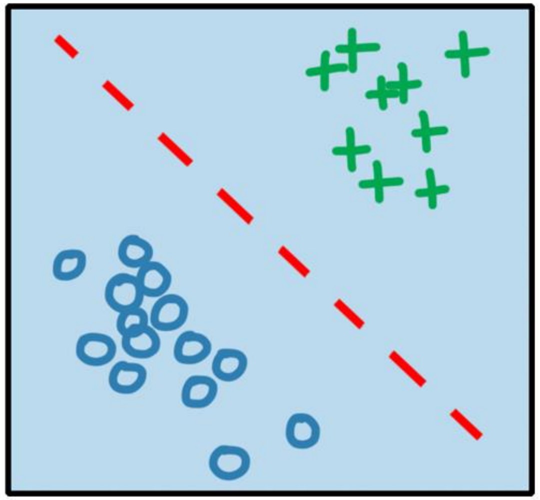
❖ Unsupervised vs Supervised vs Reinforcement Learning

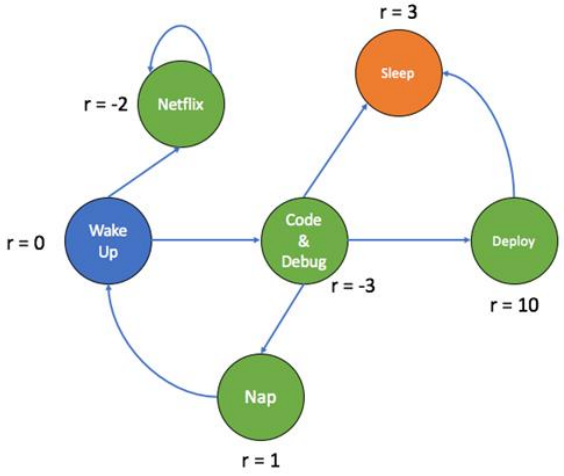
Unspervised Learning is used to train machines using labeled data.

Supervised Learning uses unlabeled data to train machines.

Reinforcement Learning uses an agent and an environment to produce actions and rewards.







Introduction

❖ Algorithms

Unsupervised Learning
- K-Mean Clustering
- PCA
- DBSCAN

Supervised Learning
- Linear Regession
- Logistic Regression
- SVM
- KNN

Reinforcement Learning
- Monte Carlo
- Q-Learning
- Deep Q-Learning
- SARSA

Introduction

❖ Training data

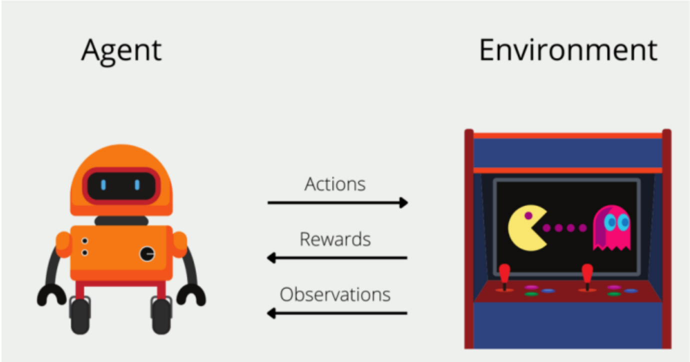
Unsupervised Learning

Input data is not labeled

Supervised Learning

Input data is labeled

Reinforcement Learning

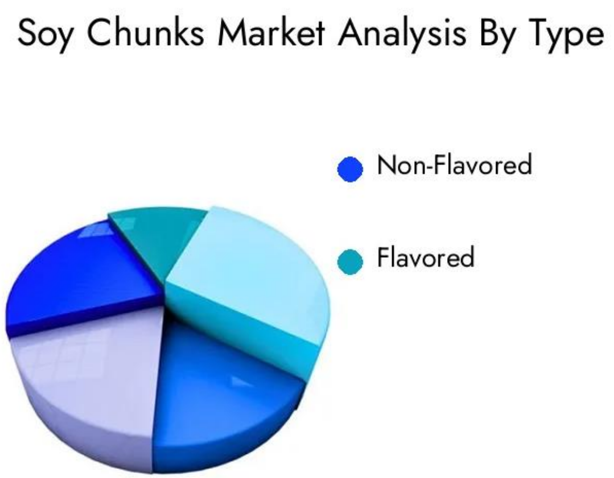


No need input data

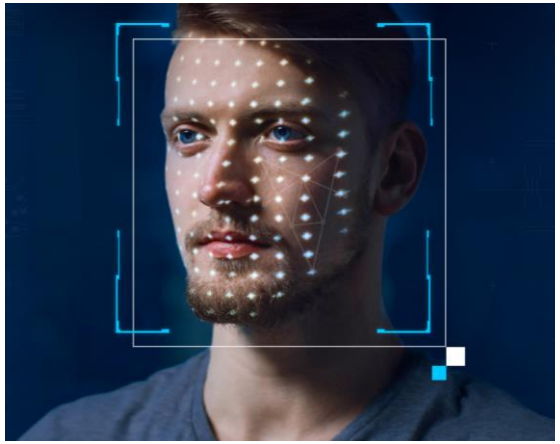
Introduction

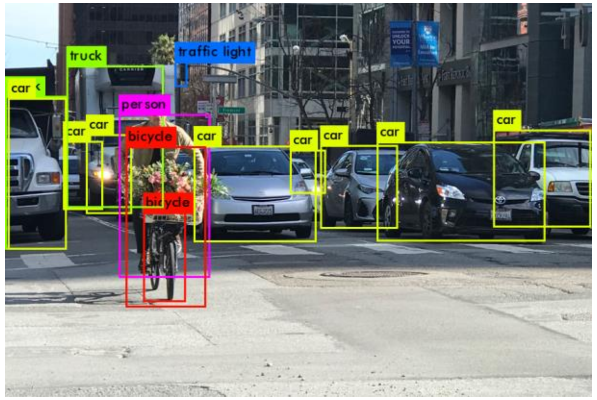
❖ Applications

Unsupervised Learning

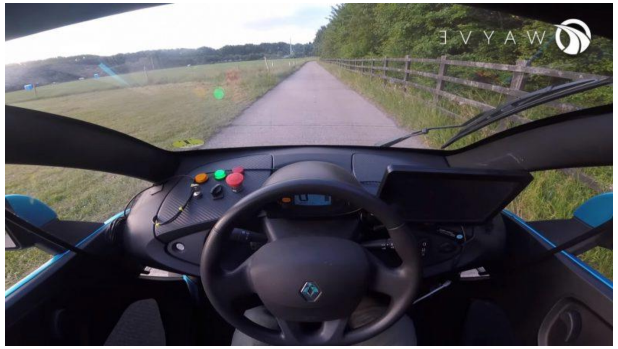


Supervised Learning





Reinforcement Learning

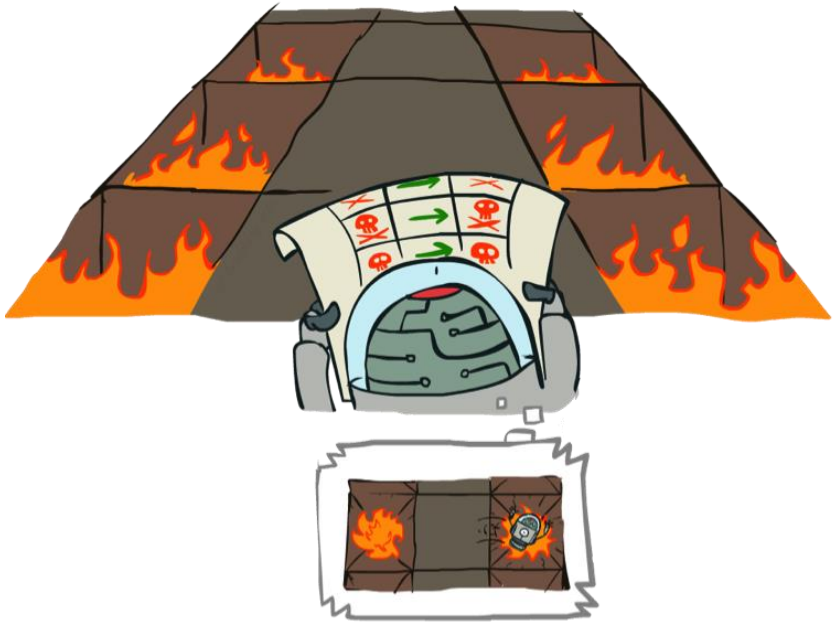


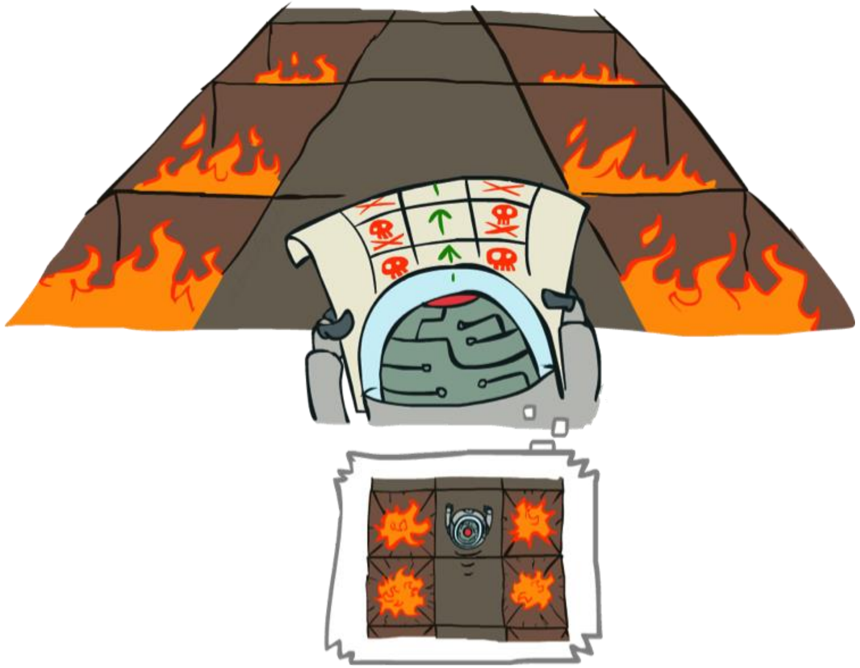
Reinforcement Learning

Reinforcement Learning

❖ Getting Started

- Reinforcement learning is a framework for solving control tasks (also called decision problems)
- By building agents that learn from the environment by interacting with it through trial and error.
- And receiving rewards (positive or negative) as unique feedback.



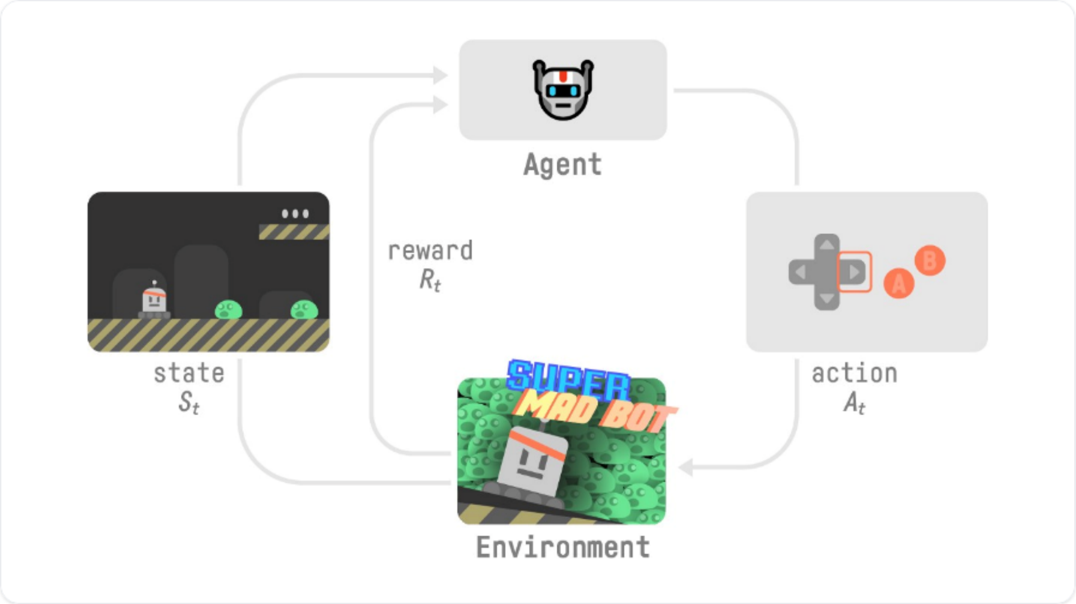


Learning from interactions with the environment comes from our natural experiences.

Reinforcement Learning

❖ Getting Started

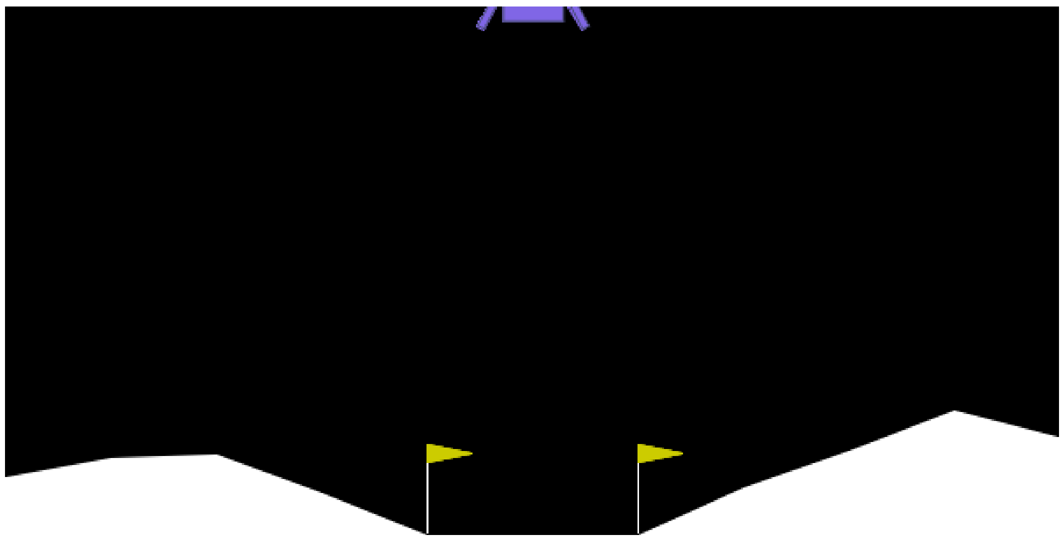
1. Agent receives **state** $\boldsymbol{S_0}$ from environment.
2. Base on S_0, Agent take a **action** $\boldsymbol{A_0}$
3. Environment goes to next **state** $\boldsymbol{S_1}$
4. Environment give some **rewards** $\boldsymbol{R_1}$ to the Agent



Reinforcement Learning

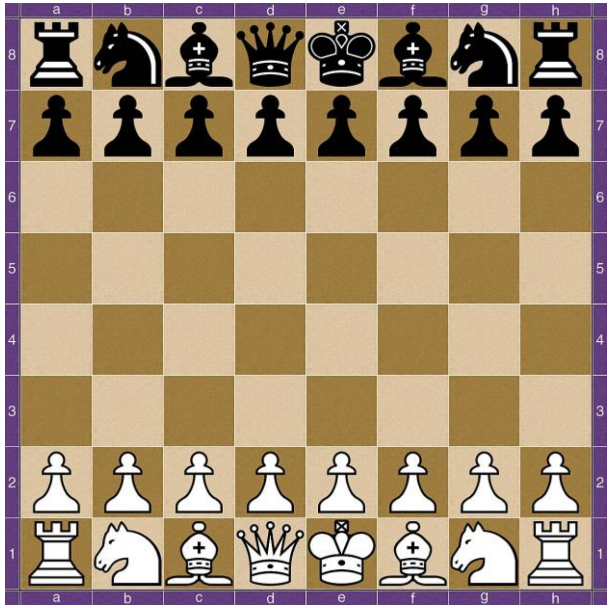
❖ Environment

The environment for testing RL algorithms is often games

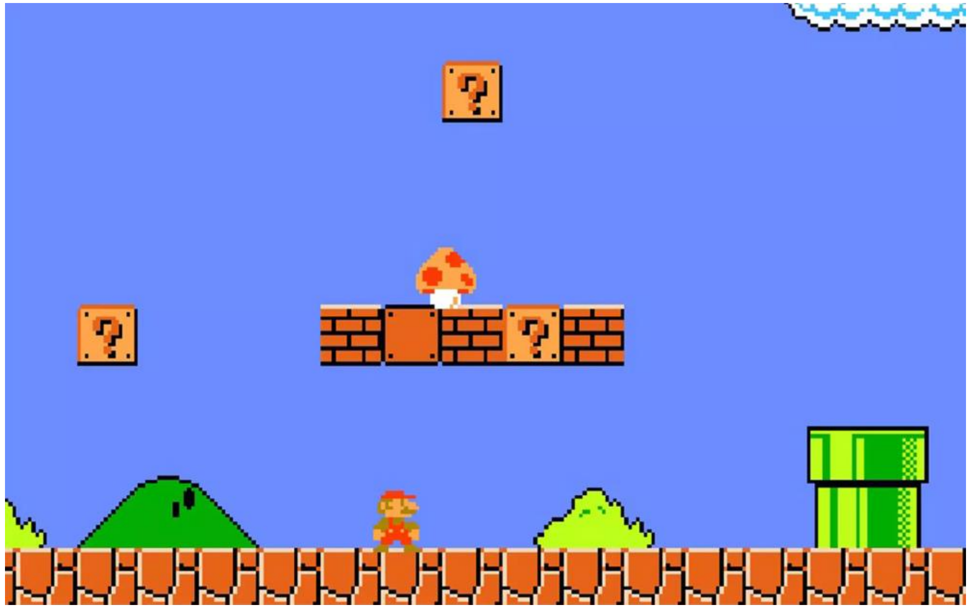


Reinforcement Learning

❖ Observation / State Space



State s: is a complete description of the state of the world.



Observation o: is a partial description of the state.

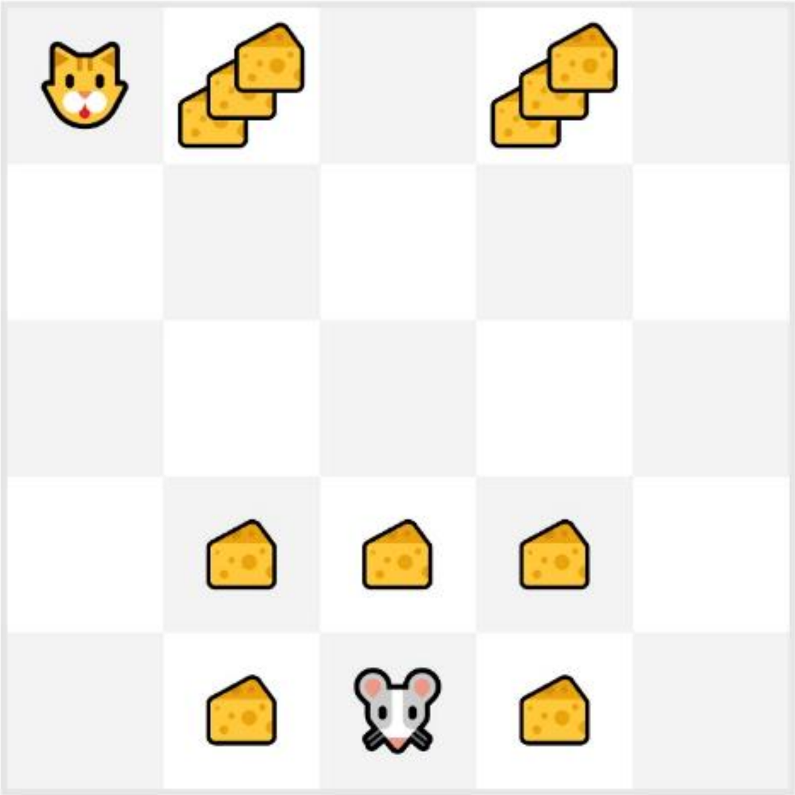
Reinforcement Learning

❖ Reward

The goal is maximizing the **expected return** (expected cumulative reward).

$$R = r_t + r_{t+1} + r_{t+2} + r_{t+3} + \cdots$$

$$R = \sum_{k=0}^{n} r_{t+k+1}$$



❖ Discounting

It’s also reasonable to prefer rewards now to rewards later

$$R = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots$$

$$R = \sum_{k=0}^{n} \gamma^k r_{t+k+1}$$

1

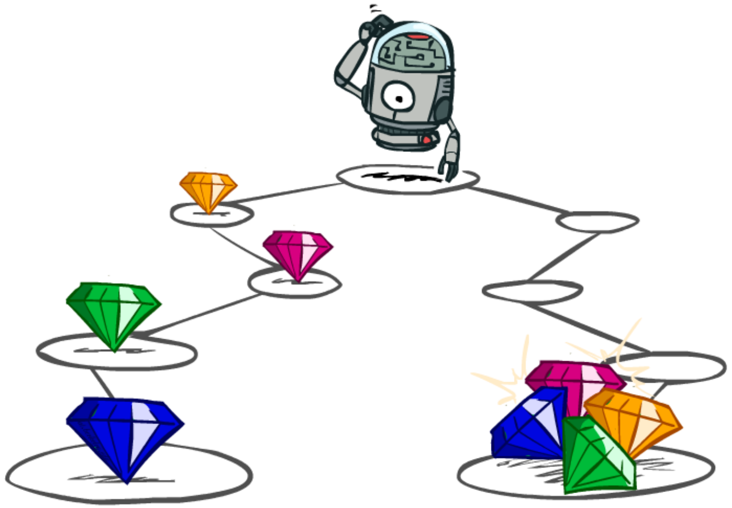
Worth Now

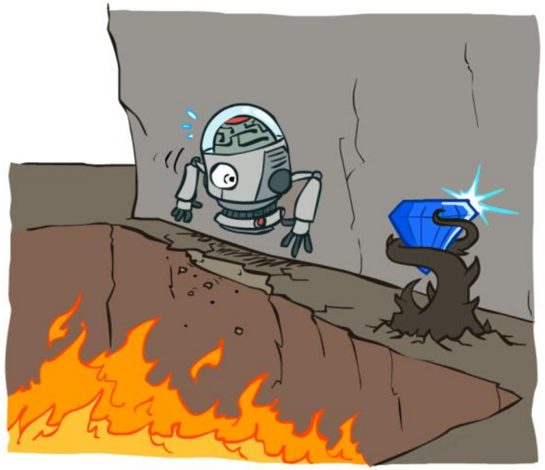
γ

Worth Next Step

γ^2

Worth In Two Steps



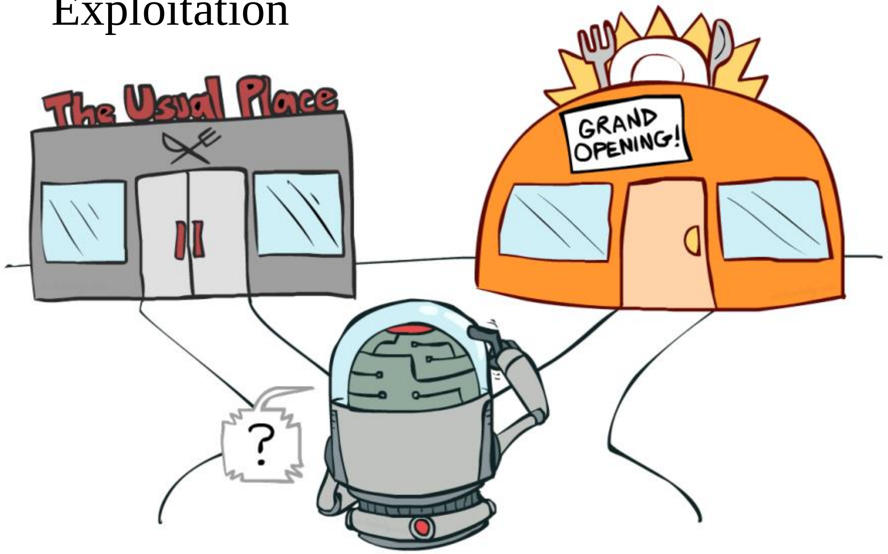


Reinforcement Learning

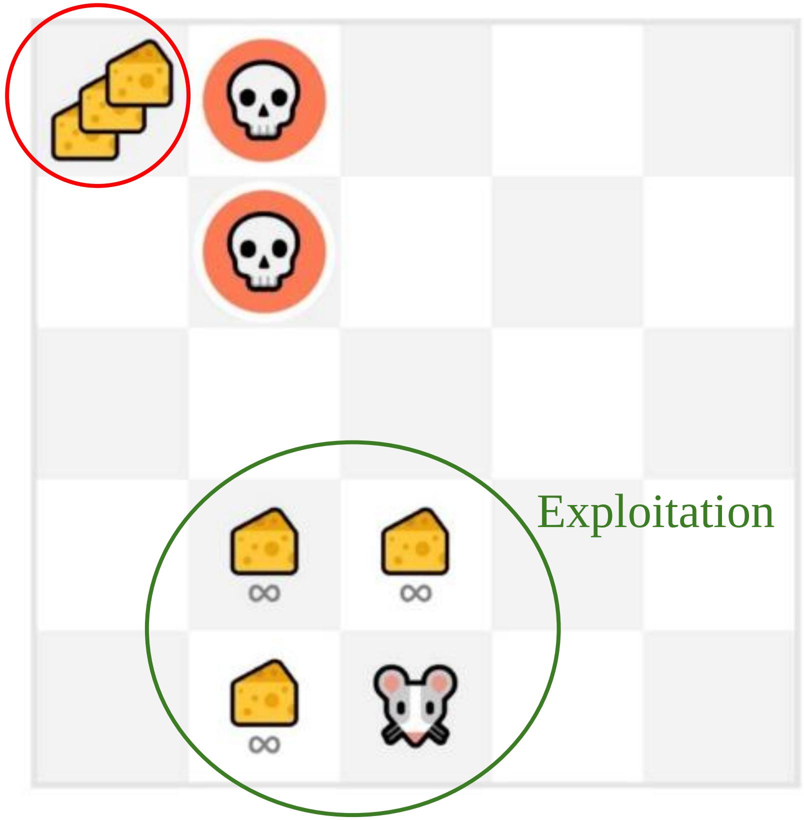
❖ Exploration / Exloitation trade-off

Exploitation

Exploration



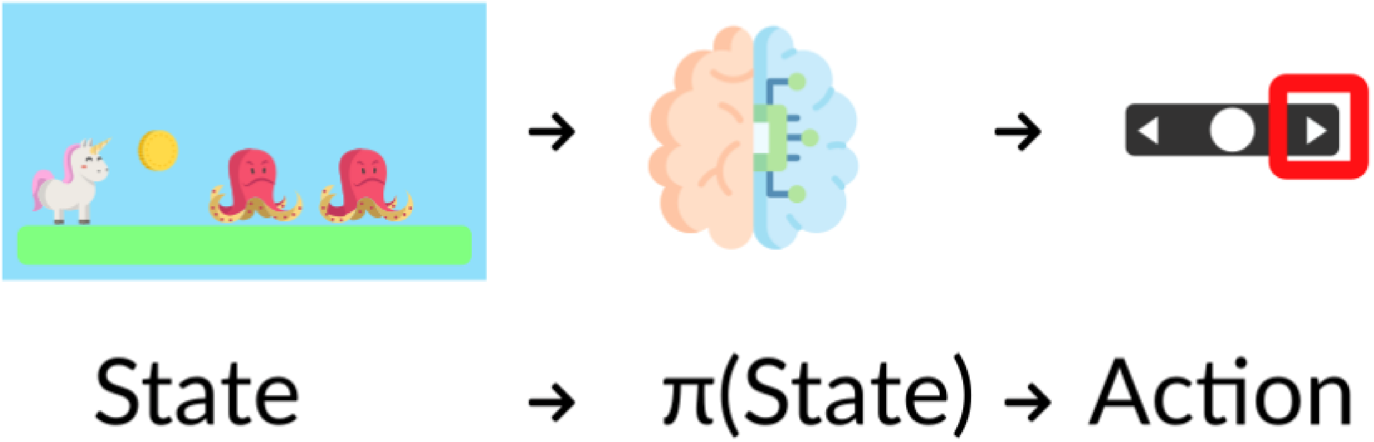
Exploration



Exploitation

❖ Appoachs for solving RL problem

The **Policy π** is the **brain** of our Agent, it’s the function that tells us **what action** to take given the state we are in. So it defines the **agent’s behavior** at a given time.



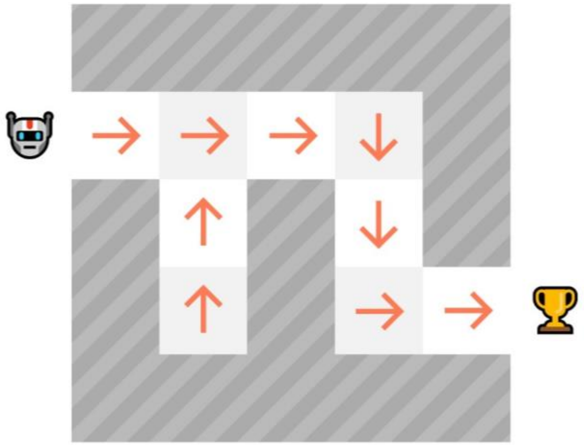
This Policy is the function we want to learn, our goal is to find the **optimal policy π*,** the policy that **maximizes expected return** when the agent **acts according to it**. We find this π* through training.

Reinforcement Learning

❖ How to find optimal policy π^*

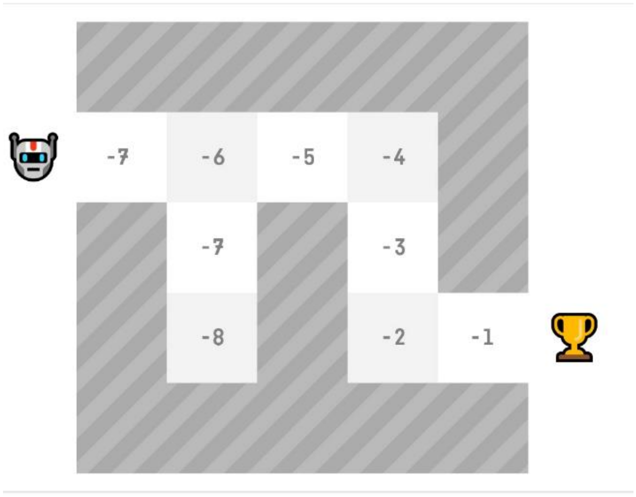
Directly, by teaching the agent to learn which **action to take**, given the **current state**.

Policy-Based Methods



Indirectly, teach the agent to learn which state is **more valuable** and then take the action that **leads to the more valuable** states

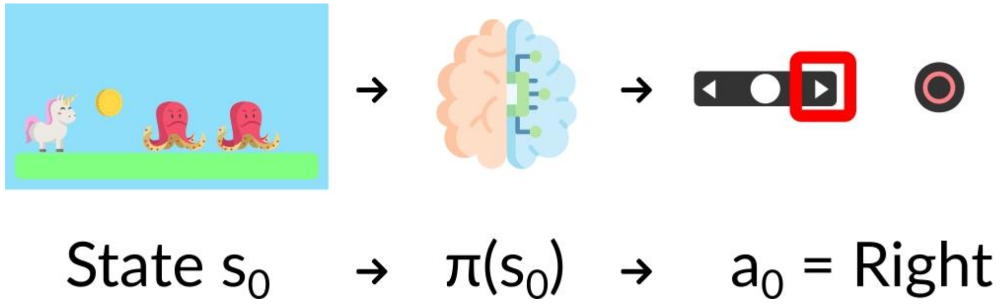
Value-Based Methods



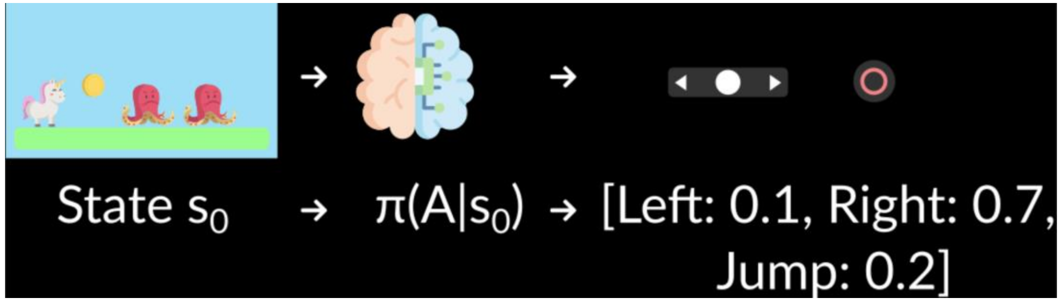
Reinforcement Learning

❖ Policy-based methods

Deterministic: action $a_0 = \pi(s_0)$



Stochastic: $\pi(A|s_0) = P[A|s_0]$

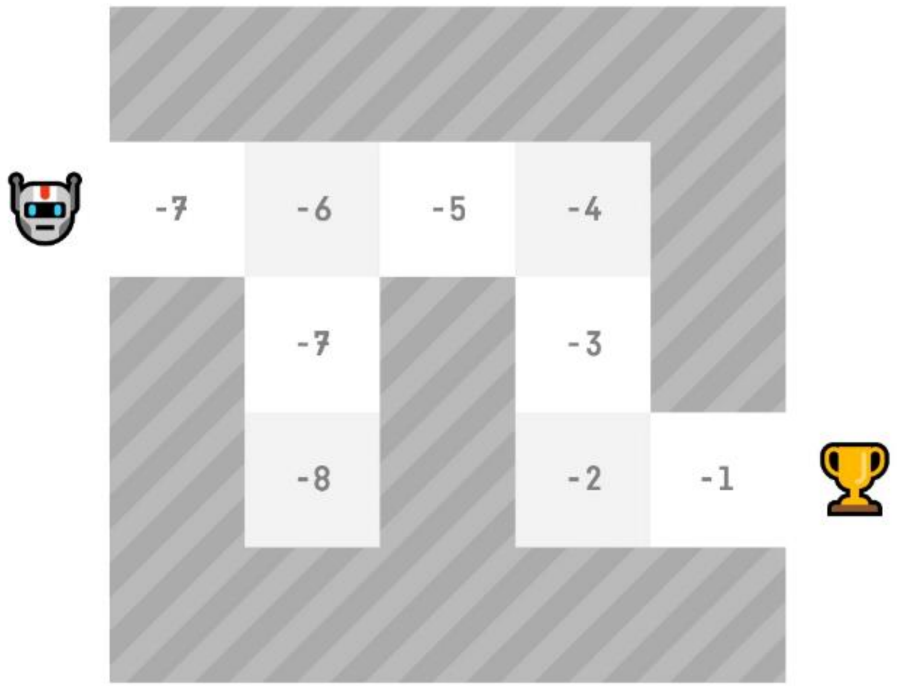


Policy π is a Neural Network

❖ Value-based methods

We learn a **value function** that **maps** a **state** to the **expected value** of being at that state.

$$\underbrace{v_\pi(s)}_{\text{Value function}} = \underbrace{\mathbb{E}_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \right.}_{\text{Expected discounted return}} \mid \underbrace{\left. S_t = s \right]}_{\text{Starting at state s}}$$

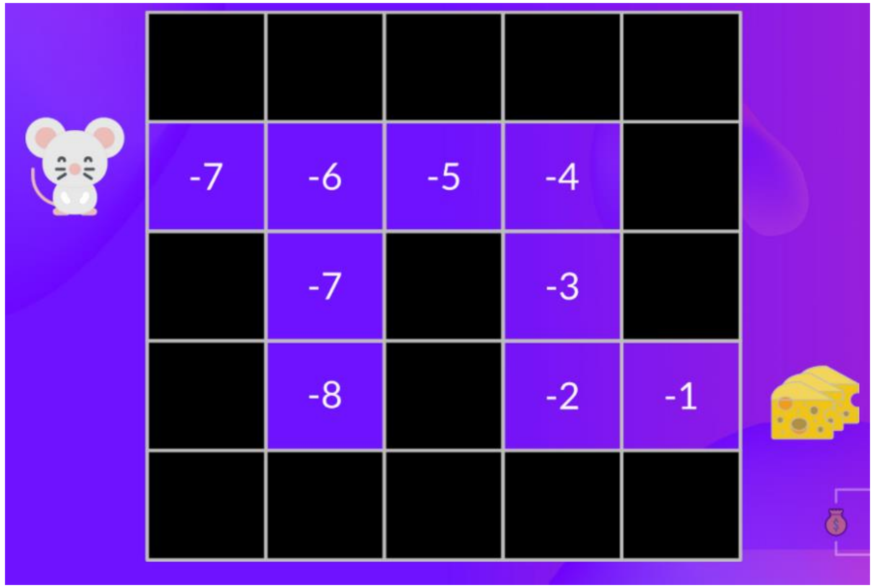


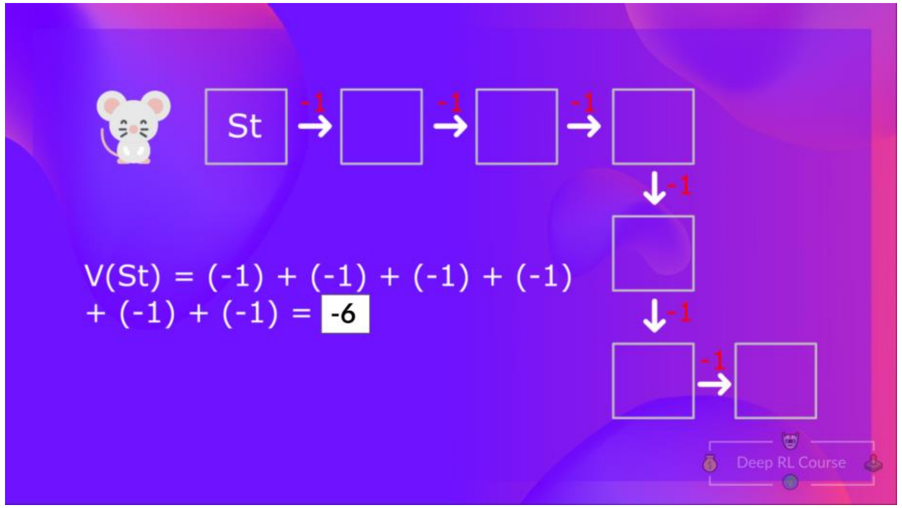
$V_\pi(s)$: value of a state is the expected discounted return the agent can get if it starts in that state, and then acts according to our policy.

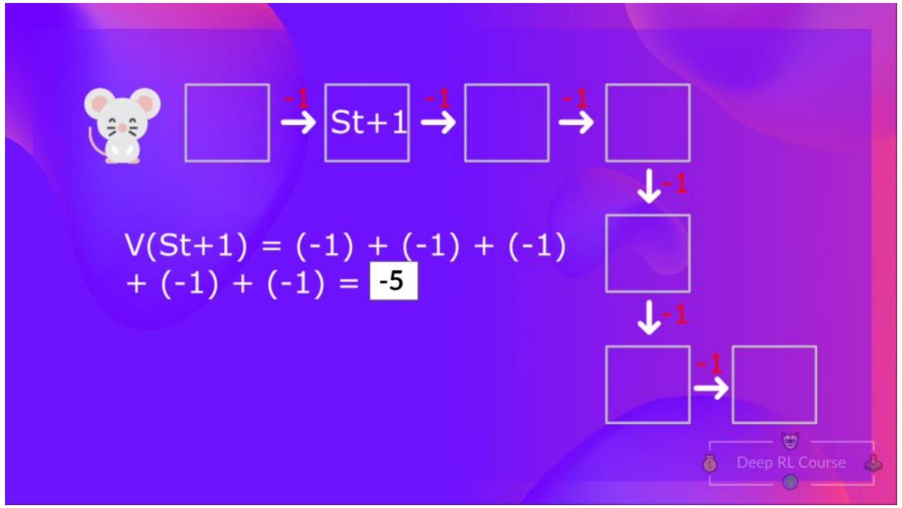
Bellman Equation

Bellman Equation

❖ Getting Started

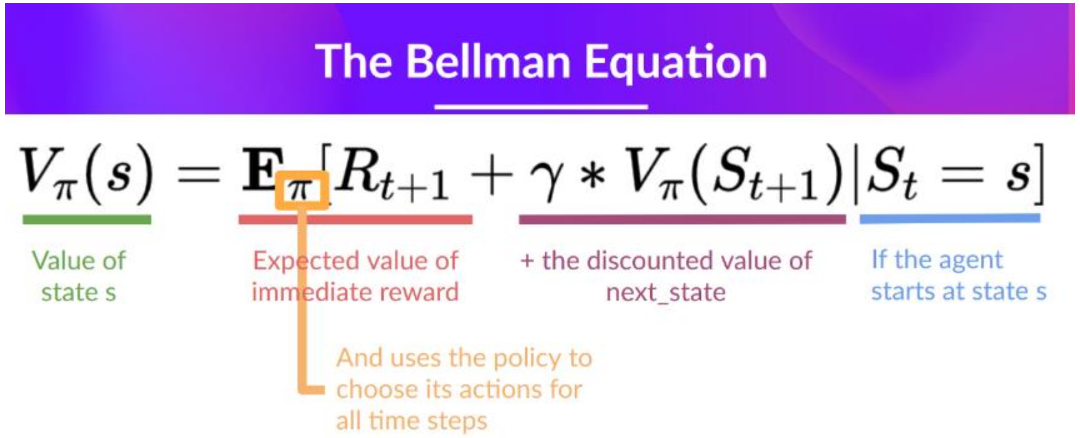


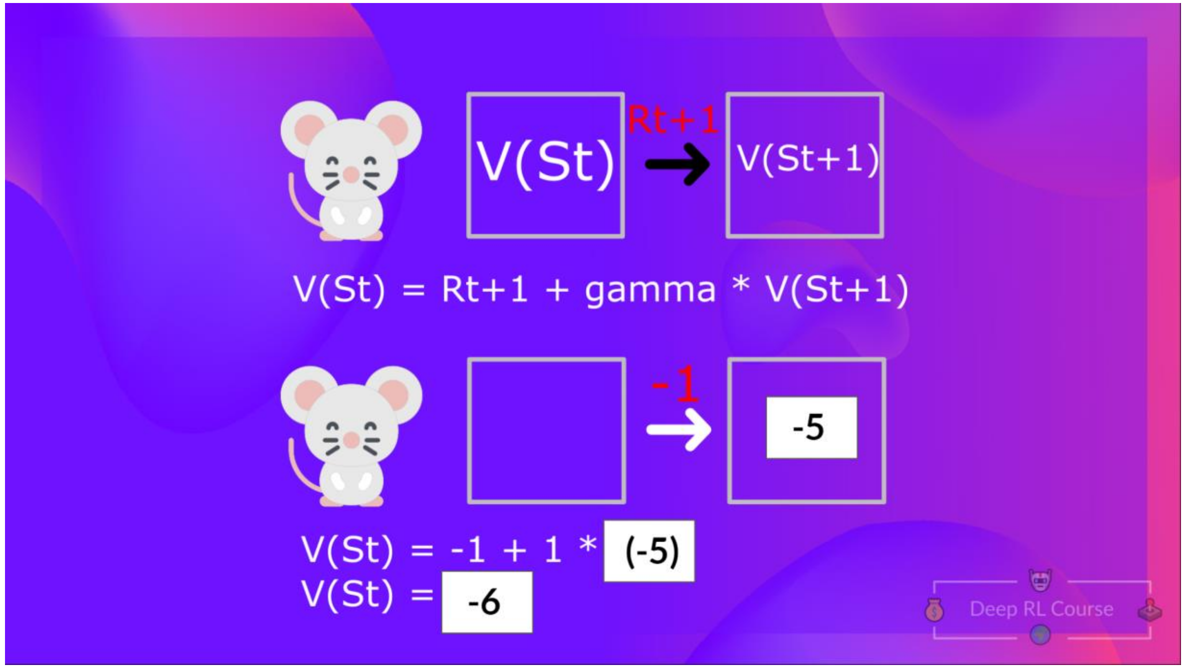




Bellman Equation

❖ Getting Started



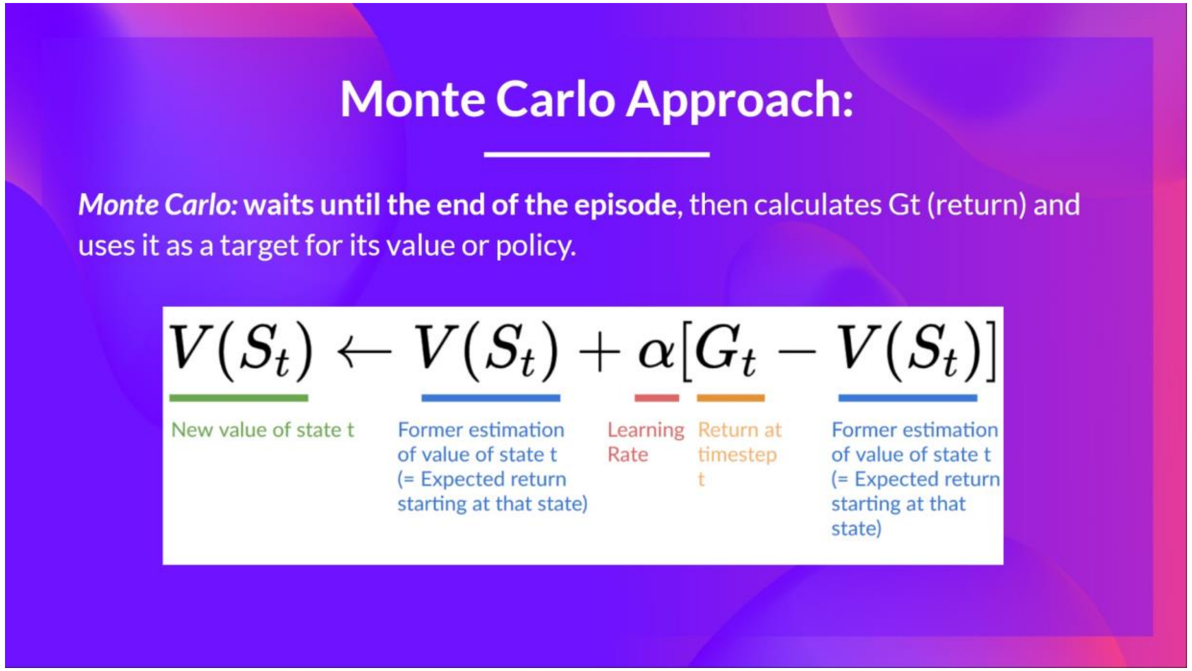


Monte Carlo

Monte Carlo

❖ Getting Started

- **Monte Carlo** is a strategy to train our **value function function**.
- It use **experience to solve** the RL problem.

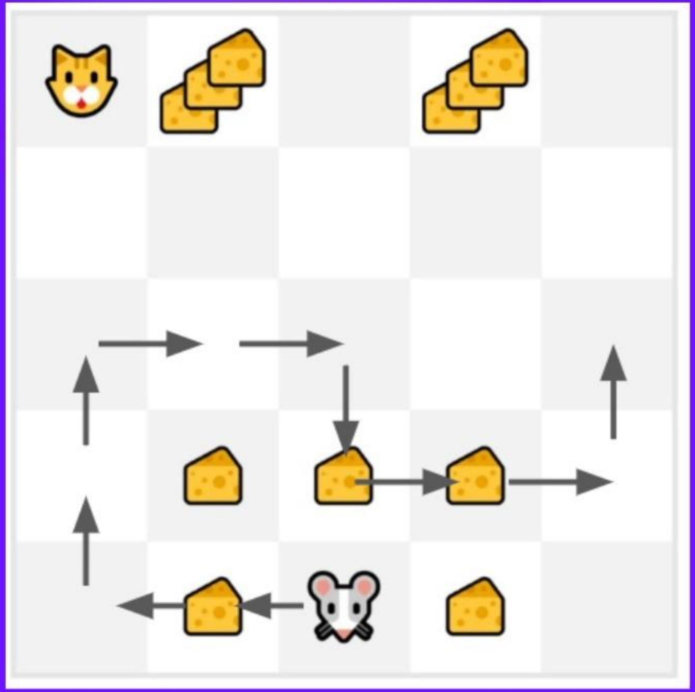


- Monte Carlo uses an **entire episode** of experience **before learning.**
- So it requires a **complete episode** of interaction before updating our value function.

Monte Carlo

❖ Training

- Initialize value function to zero for each state
- Learning rate (lr) is 0.1 and our discount rate is 1 (= no discount)
- The mouse explores the environment and takes random actions



- Calculate the return Gt.

Gt = Rt+1 + Rt+2 + Rt+3...

Gt = 1 + 0 + 0 + 0+ 0 + 0 + 1 + 1+ 0 + 0

Gt= 3

- We can now update V(S0).

$$\boxed{V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]}$$

New V(S0) = V(S0) + lr * [Gt-V(S0)]

New V(S0) = 0 + 0.1 * [3 –0]

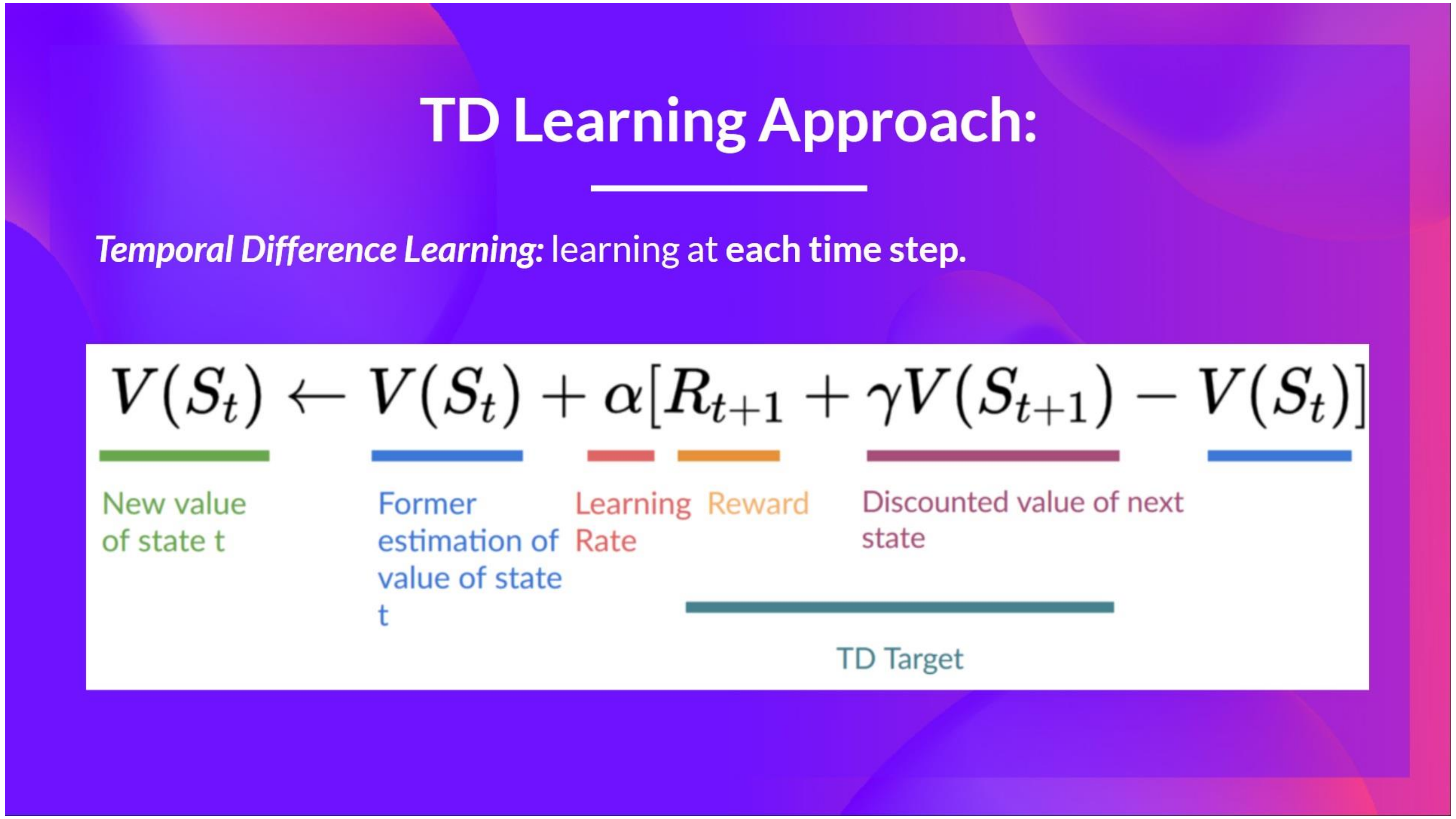
New V(S0) = 0.3

Temporal Difference

Temporal Difference

❖ Getting Started

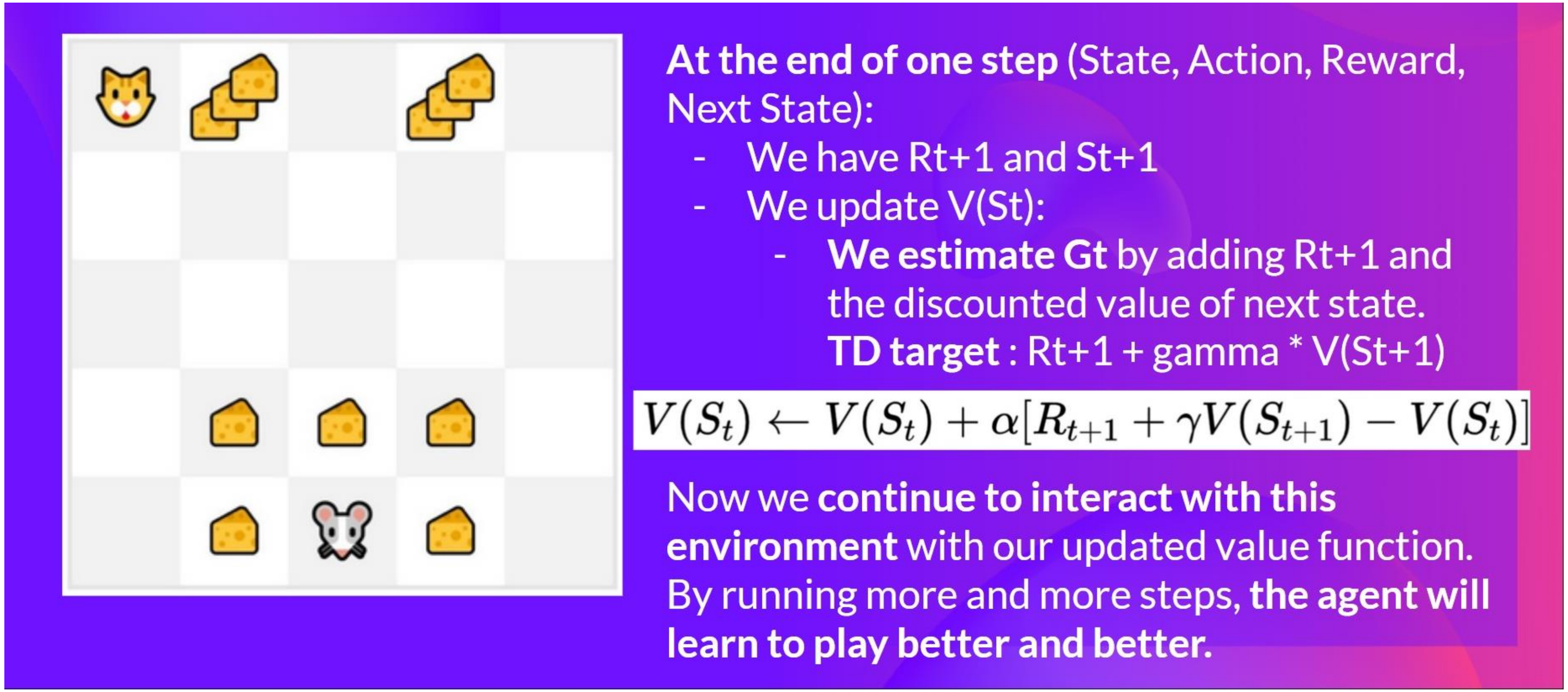
- The idea with TD is to update at each step.
- Estimate G_t by R_{t+1} and discounted value of next state.



Temporal Difference

❖ Training

- Initialize value function to zero for each state
- Learning rate (lr) is 0.1 and our discount rate is 1 (= no discount)
- The mouse explores the environment and takes random actions **(going to the left)**



Monte Carlo vs TD Learning

❖ Summary

Monte Carlo: $V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$

TD Learning: $V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$

Q-Learning

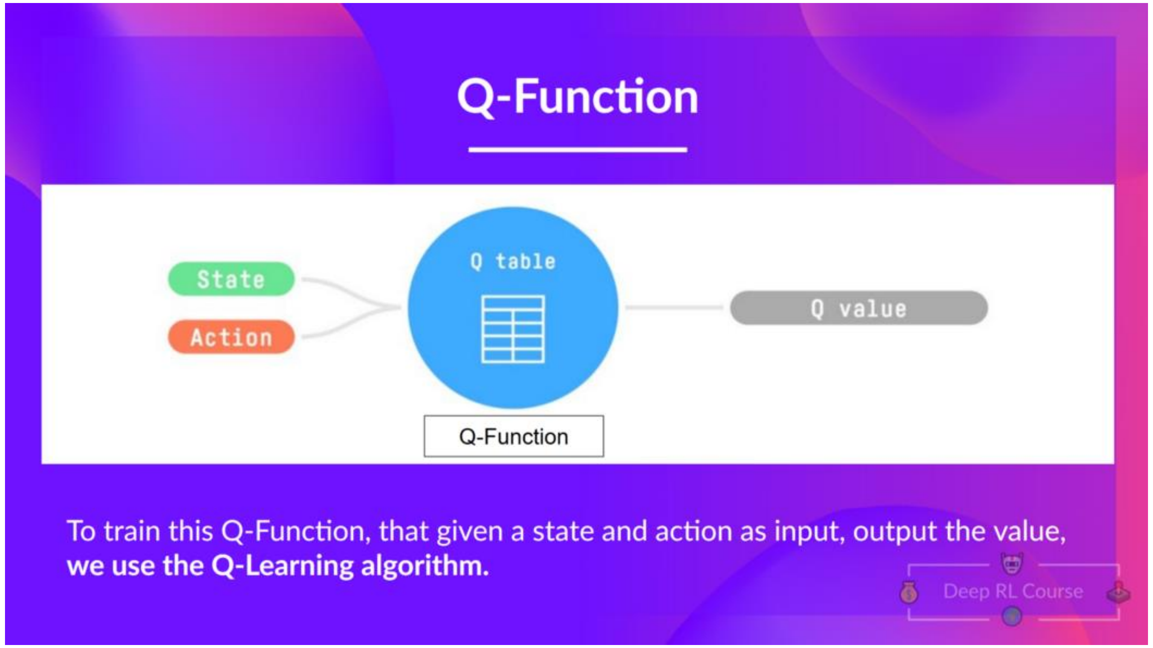
Q-Learning

❖ Getting Started

Q-learning is a simple Reinforcemen Learning algorithm

Q-learning is a value-based methods.

Q-learning uses a TD approach to train its action-value function.

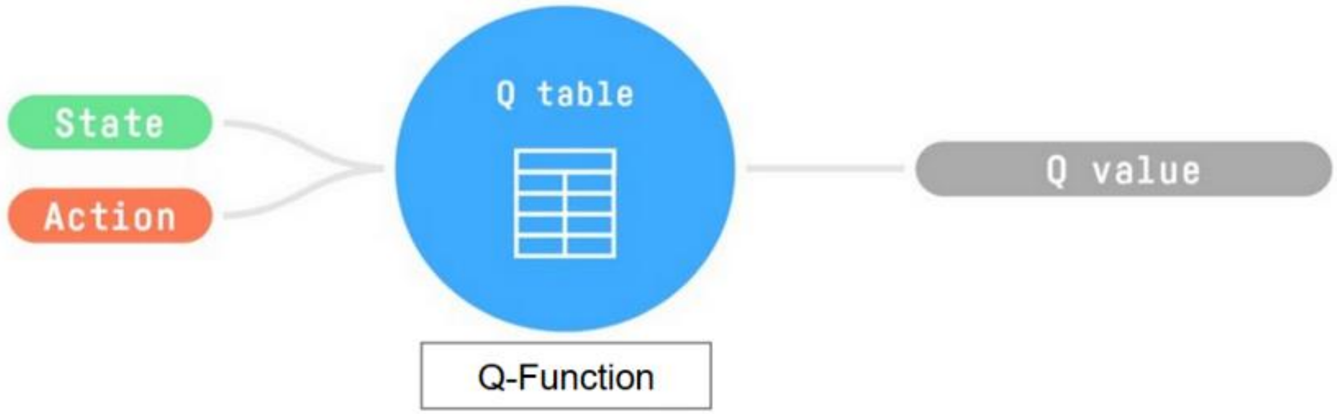


Q-Learning is the algorithm we use to train our Q-function, an action-value function that determines the value of being at a particular state and taking a specific action at that state.

Q-Learning

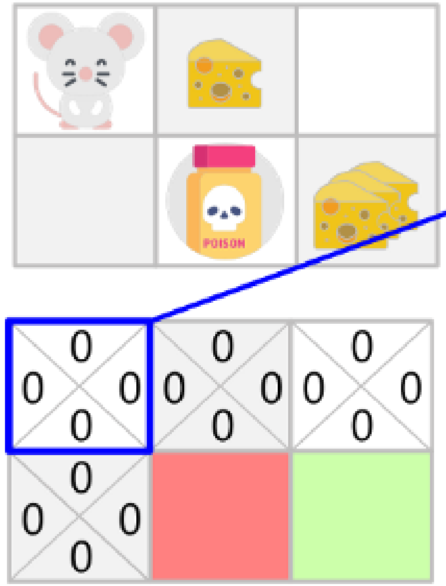
❖ Getting Started

- The Q-function uses a Q-table that has the value of each state-action pair.
- Given a state and action, our Q-function will search inside its Q-table to output the value.
- Q(s, a) is the the value by starting from s and take action a.



Q-Learning

❖ Getting Started



Actions

States	←	→	↑	↓
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

Q-Learning

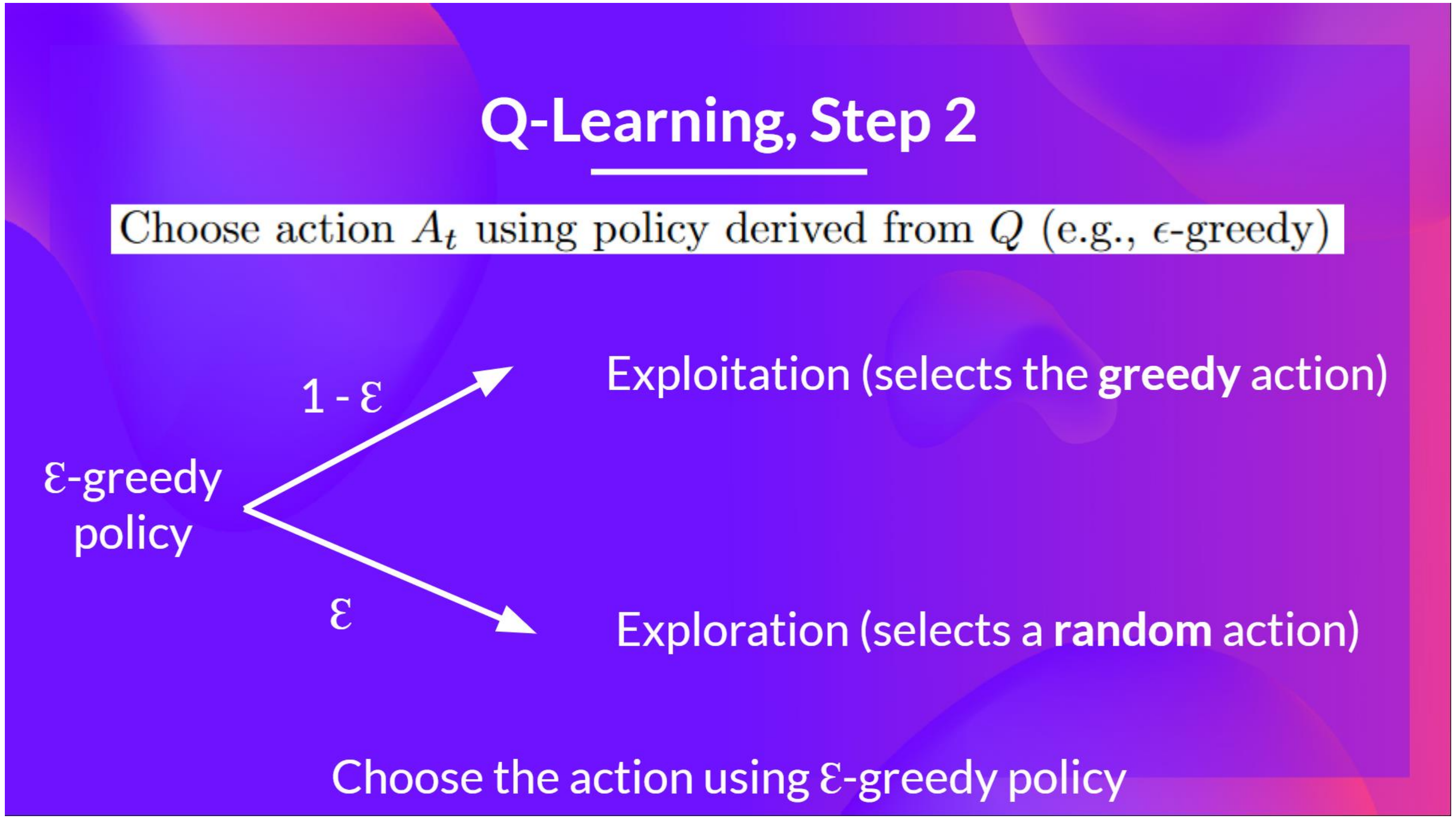
❖ Training

- Initialize Q(s, a) = 0 for each s, a pair
- Select action and observe an experience (s, a, r, s').
- Update $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \boldsymbol{max}\, Q(s', a') - Q(s, a)]$

	←	→	↑	↓
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

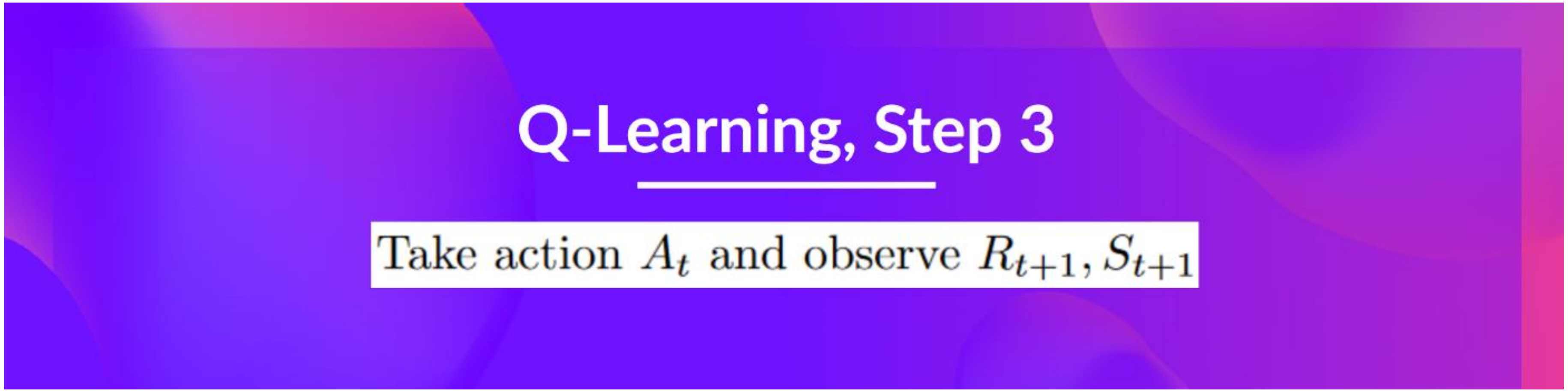
Q-Learning

❖ Training step 2



Q-Learning

❖ Training step 3



Q-Learning

❖ Training step 4

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

New Q-value estimation

Former Q-value estimation

Learning Rate

Immediate Reward

Discounted Estimate optimal Q-value of next state

Former Q-value estimation

TD Target

TD Error

Q-Learning

❖ Rules

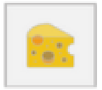
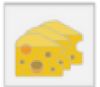
- You’re a mouse in this tiny maze. You always start at the same starting point.
- The goal is to eat the big pile of cheese at the bottom right-hand corner and avoid the poison. After all, who doesn’t like cheese?
- The episode ends if we eat the poison, eat the big pile of cheese, or if we take more than five steps.
- The learning rate is 0.1
- The discount rate (gamma) is 0.99

- +0: Going to a state with no cheese in it.
- +1: Going to a state with a small cheese in it.
- +10: Going to the state with the big pile of cheese.
- -10: Going to the state with the poison and thus dying.
- +0 If we take more than five steps.

Q-Learning

❖ Training

$$lr = 0.1$$
$$\gamma = 0.99$$

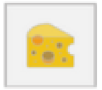
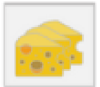
	←	→	↑	↓
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

Q-Learning

❖ Training

$$\alpha = 0.1$$
$$\gamma = 0.99$$

	←	→	↑	↓
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

Q-Learning

❖ Training

$\alpha = 0.1$
$\gamma = 0.99$

	←	→	↑	↓
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

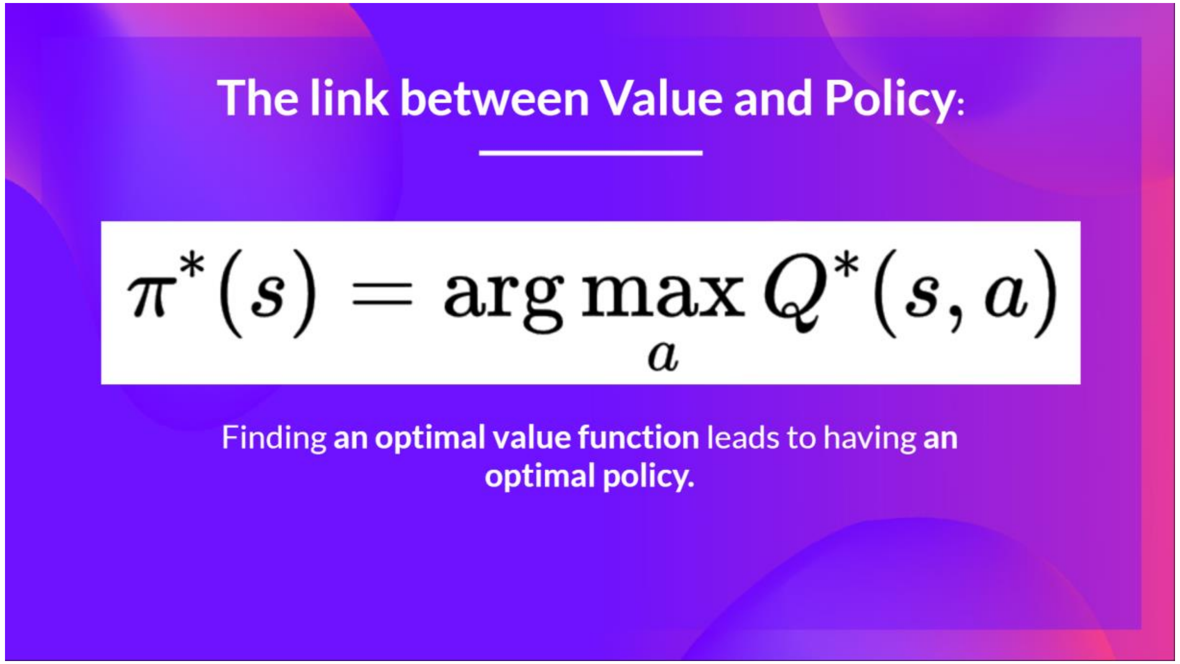
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

Q-Learning

❖ Training

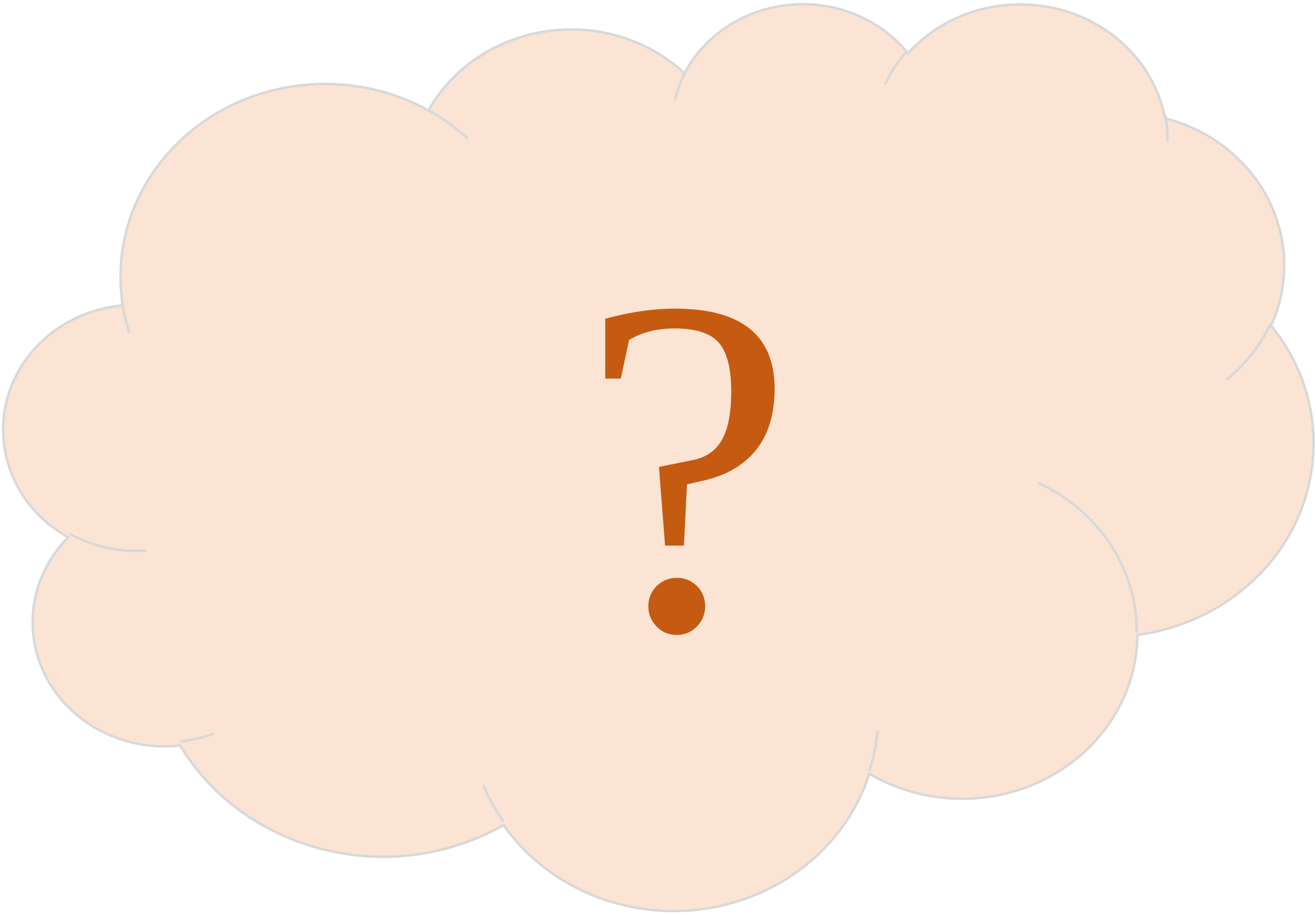
At the end of the training, we'll get an estimate of the optimal Q-function.

	←	→	↑	↓
	0	10.8	0	0
	0	9.9	0	-10
	0	0	0	10
	0	-10	0	0
	0	0	0	0
	0	0	0	0



Demo

Question




Thank you!