# Reinforcement Learning: Introduction

Lecture 12

Dr. Syed Maaz Shahid

20th May,2024

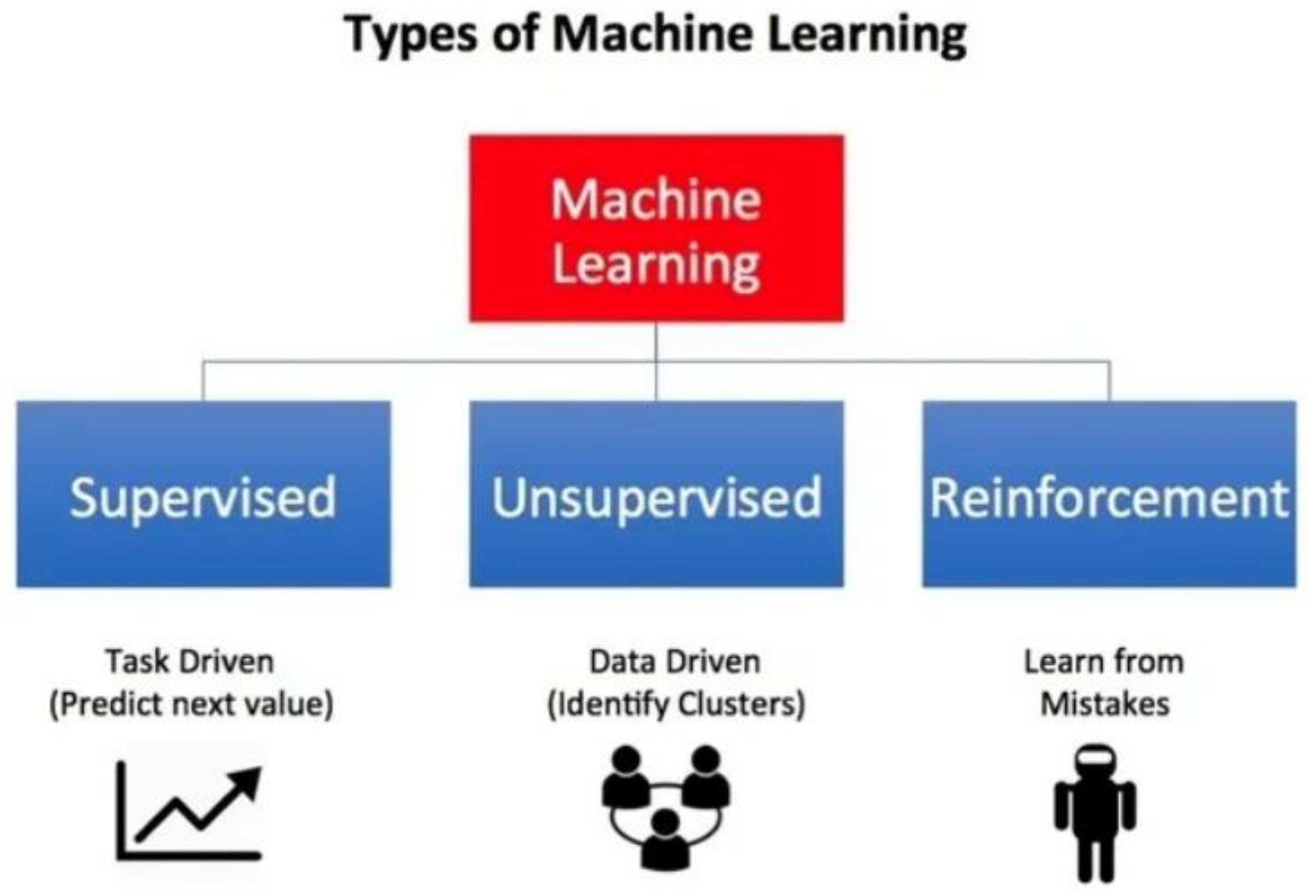
# Contents

# Assignment

- Find a paper that uses HMM to solve a problem in your relevant field.

- Make a report and submit by **31** May, 20204.

# Machine Learning



# Reinforcement Learning

- In supervised machine learning, you focus on predicting what you don’t know→ *learning to predict.*

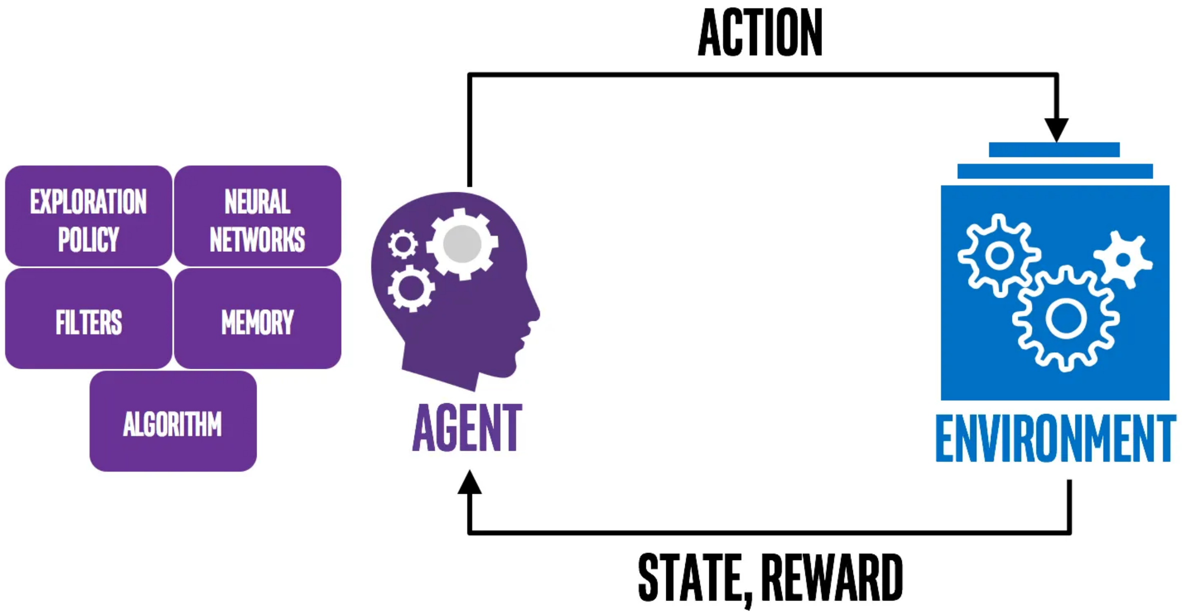
- Reinforcement learning (RL) is a framework for learning-based decision-making→ *learning to act*.

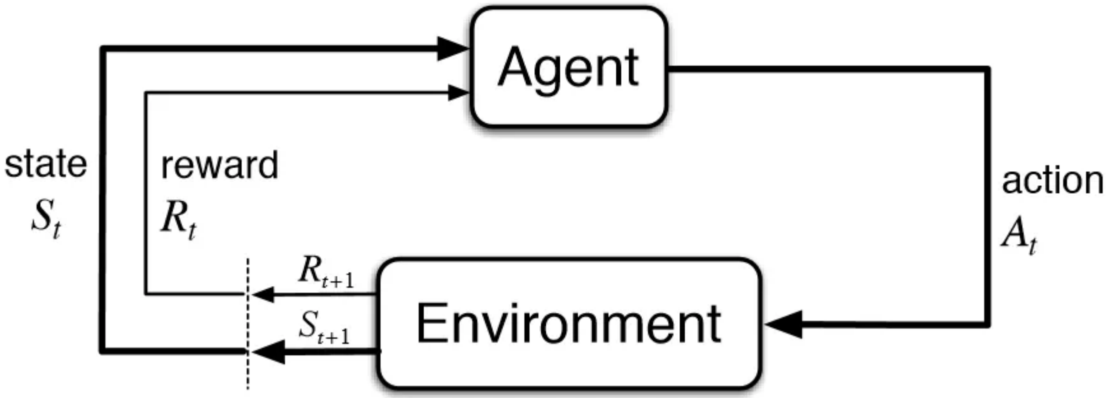
- Reinforcement learning uses **rewards and punishments** as signals for positive and negative behavior.

# Reinforcement Learning: Elements

- **Agent/ Policy:** The program you train for specific task/Method to map agent’s state to actions.
- **Environment/State:** The world, real or virtual, in which the agent performs actions.
- **Action.** A move made by the agent, which causes a status change in the environment.
- **Rewards.** The evaluation of an action (positive or negative)/Feedback from the environment..

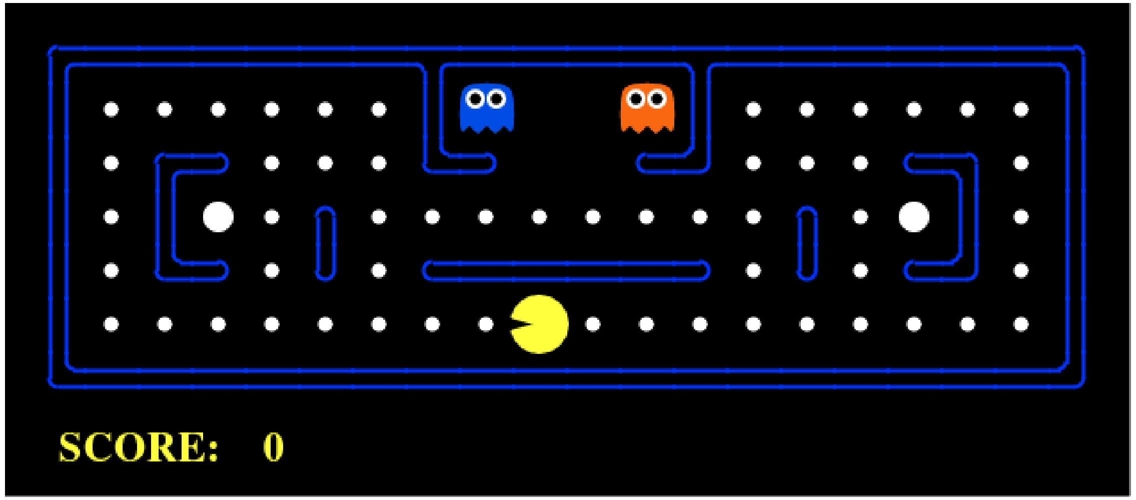
# Reinforcement Learning: Elements



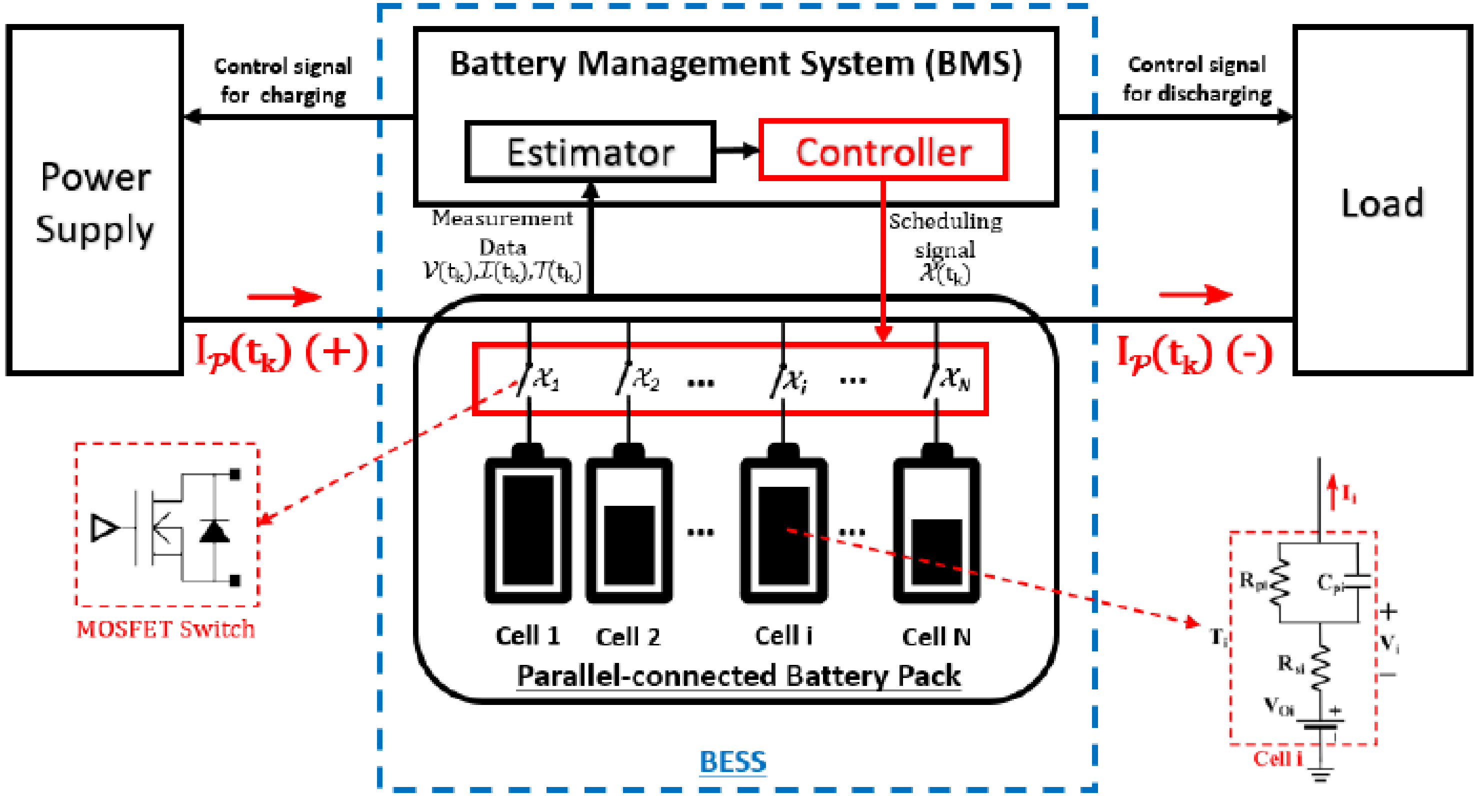
# Reinforcement Learning: Use Case

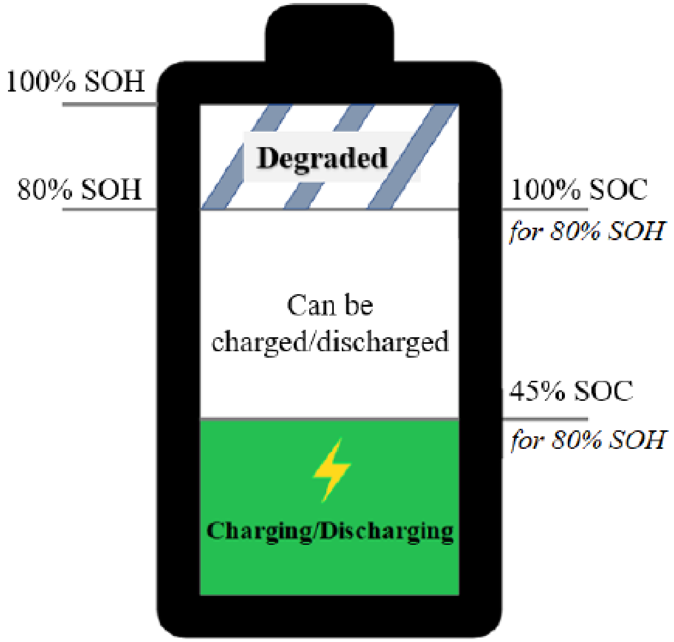
- **PacMan** game
  - *Agent* → PacMan
  - *Environment*: Grid world
  - *Reward* for eating food & *Punishment* if gets killed by the ghost
  - *State*→ Location
  - *Cumulative reward*→ winning the game



# Reinforcement Learning: Use Case

- Battery Management for Retired Electric Vehicle Batteries




Doan, Nhat Quang, et al. "Deep Reinforcement Learning-Based Battery Management Algorithm for Retired Electric Vehicle Batteries with a Heterogeneous State of Health in BESSs." Energies 17.1 (2023): 79.

# Reinforcement Learning: Use Case

- **Deep Q-learning playing Atari**

- **AlphaGo - The Movie**

# Differences: Supervised, Unsupervised, and Reinforcement Learning

- Static vs. Dynamic
- No explicit right answer→ agent learns by trial and error
- RL requires exploration
- RL is a multiple-decision process→ decision-making chain through the time in RL

# Reinforcement Learning: Agent & Environment

- At each step t the agent
  - Executes action $A_t$
  - Receives observation $O_t$
  - Receives scalar reward $R_t$
- The environment:
  - Receives action $A_t$
  - Emits observation $O_{t+1}$
  - Emits scalar reward $R_{t+1}$

# Reinforcement Learning: History & State

- The **history** $H_t$ is the sequence of observations, actions, rewards

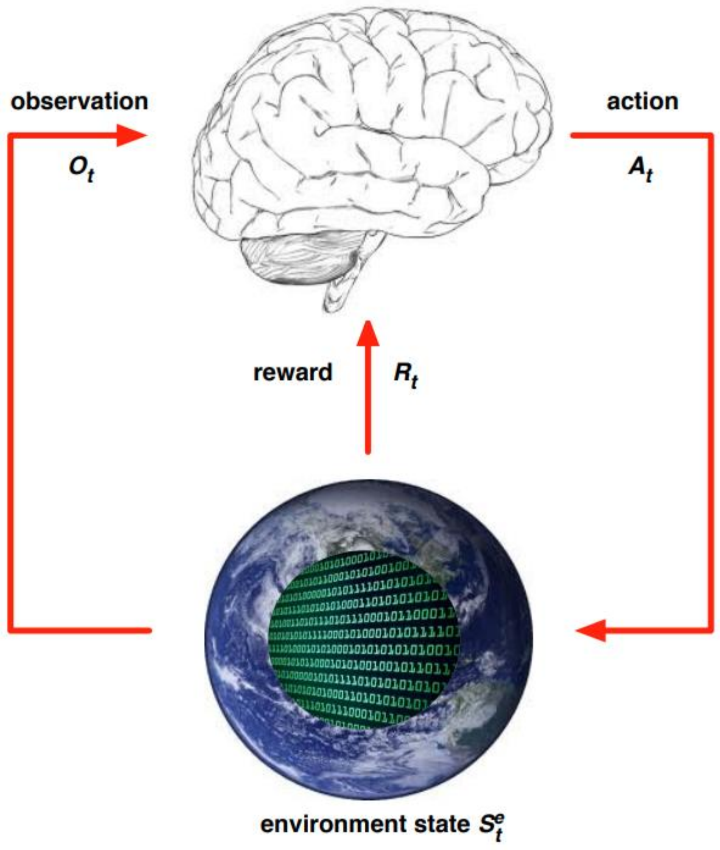
- **State** is the information used to determine what happens next.

- Formally, state is a function of the history:

$$S_t = f(H_t)$$

# Reinforcement Learning: Environment State

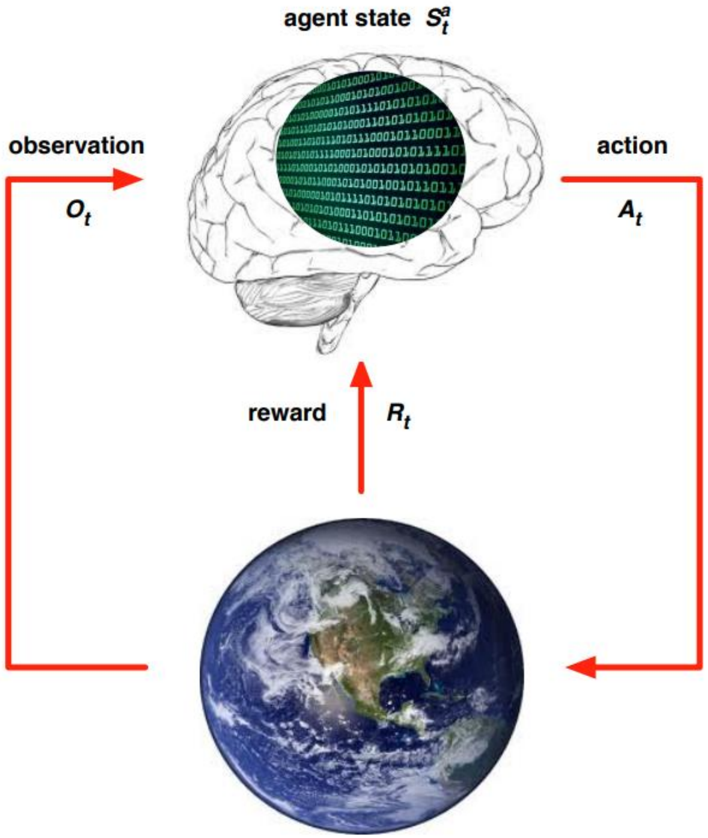
- The **environment state** $S_t^e$ is the environment's private representation.
- Usually, $S_t^e$ is not visible to the agent.
- Even if $S_t^e$ is visible to agent, it may contain irrelevant information



# Reinforcement Learning: Agent State

- The **agent state** $S_t^a$ is the agent's internal representation→ uses to pick the next action.
- 
- It can be any function of history:

$$S_t^a = f(H_t)$$



# Reinforcement Learning: Information State

- An **information state** (a.k.a. *Markov state*) contains all useful information from the history.
  - Markov chain is to assume that $X_k$ captures all *the relevant information* for predicting the future

- A state St is Markov if and only if

$$P[S_{t+1}|S_t] = P[S_{t+1}|S_1, \dots, S_t]$$

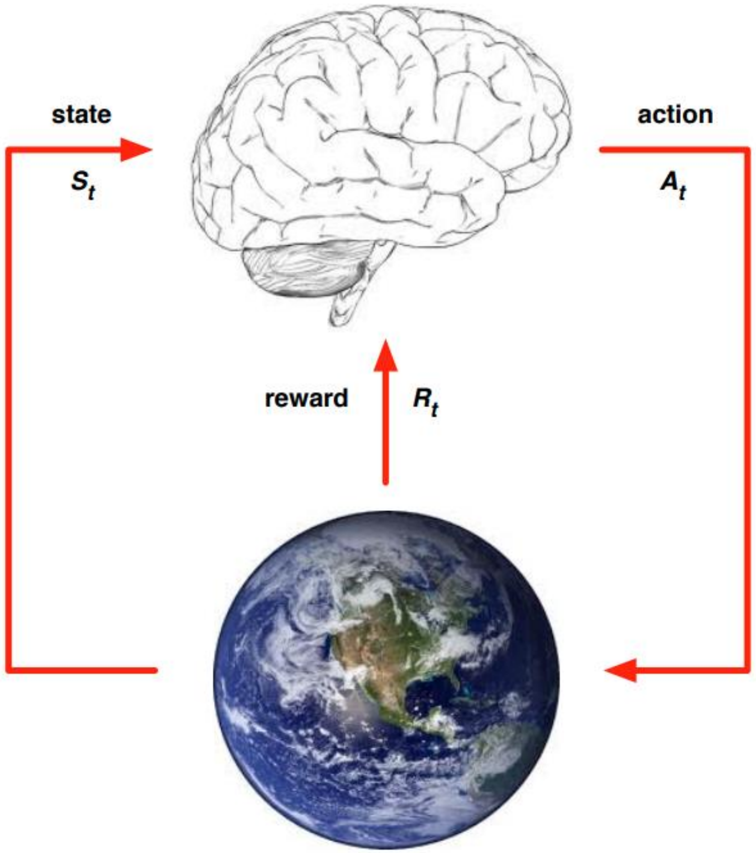
  - The future is independent of the past given the present
  - Once the state is known, the history may be thrown away

# Reinforcement Learning: Fully Observable Environments

- Agent directly observes environment state

$$O_t = S_t^a = S_t^e$$

- This is a Markov decision process (MDP).



# Reinforcement Learning: Partially Observable Environments

- Agent indirectly observes environment

$$S_t^a \neq S_t^e$$

→A poker playing agent only observes public cards.

- This is a partially observable Markov decision process (POMDP).

- Agent must construct its own state representation $S_t^a = H_t$

# Components of an RL Agent

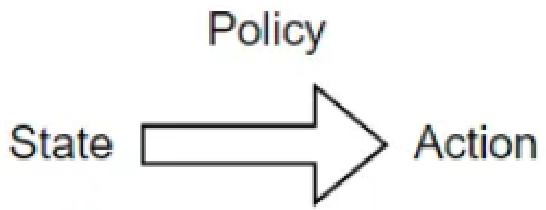
- **Policy**: agent’s behavior function

- **Value function**: how good is each state and/or action

- **Model**: agent’s representation of the environment

# Policy

- A policy is the agent's behavior.




- It is a map from state to action,
  - **Deterministic** policy: $a = \pi(s)$→same action give a state
  - **Stochastic** policy: $\pi(a|s) = P[A_t = a|S_t = s]$→ choose an action randomly based on the probability distribution

# Value Function

- Value function $v_\pi(s)$ is a prediction of future reward

- Used to evaluate the goodness/badness of states

- By following a policy $\pi$, the value function is defined as
$$v_\pi(s) = E[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots | S_t = s]$$

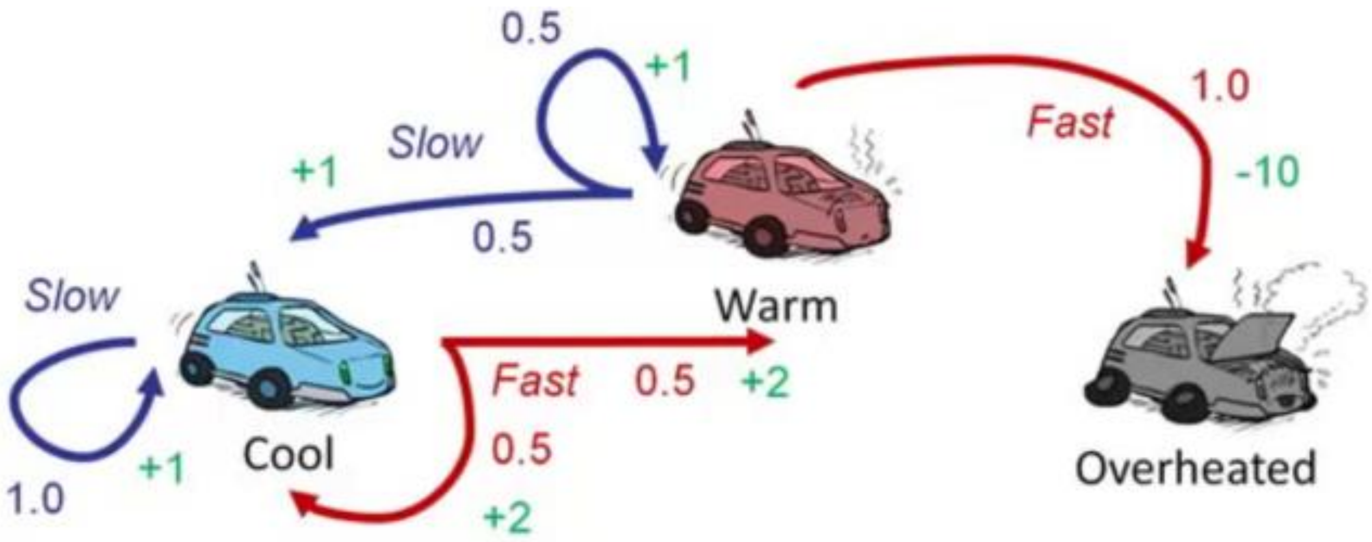
  - $0 \leq \gamma \leq 1$: *discount rate*
  - $\gamma$ close to 1: rewards further in the future count more → agent is farsighted

# Model

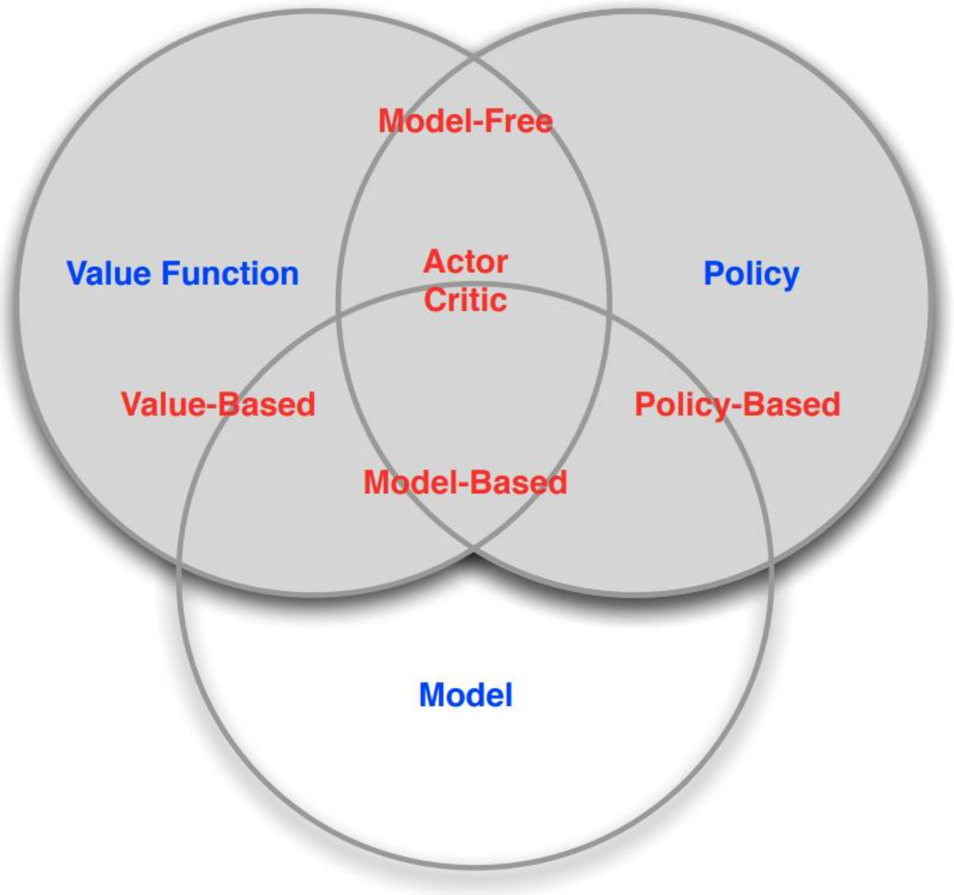
- The model describes the **environment** by a distribution over rewards and state transitions.

$$P_{s'}^{a} = P[S_{t+1} = s' | S_t = s, A_t = a]$$

$$R_{s'}^{a} = E[R_{t+1} | S_t = s, A_t = a]$$



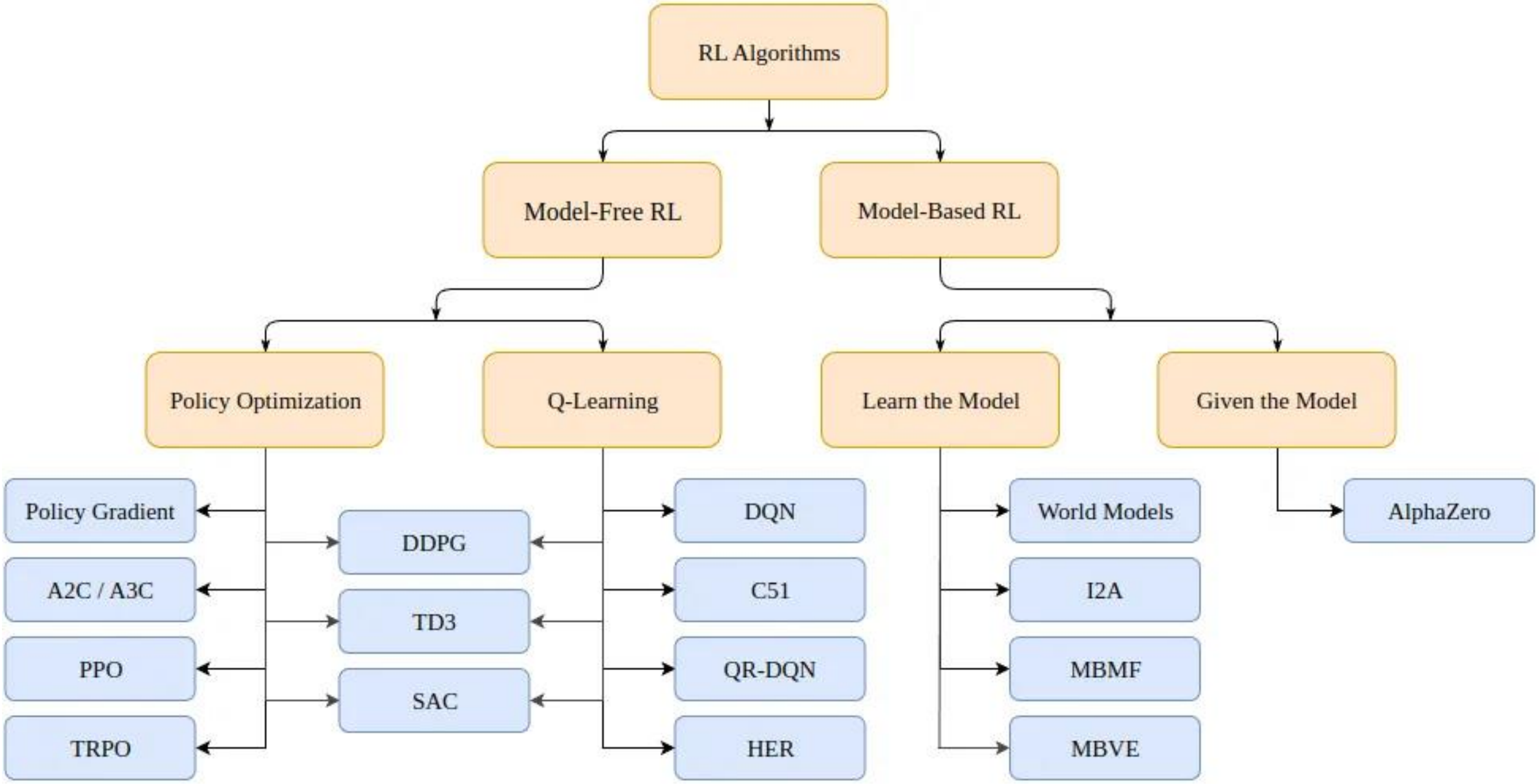
# RL Agent Taxonomy



# RL Agent Taxonomy

- DQN→ value-based method
- A2C→ policy-based method (actor) & value-based method (critic)
- Policy Optimization (PPO)

# Taxonomy of Reinforcement Learning algorithms




OpenAI spinning up

# Reinforcement Learning algorithms

- **Model-based** algorithms use a model of the environment.
  - Model is used to predict future states and rewards.
  - The model is either given (e.g. a chessboard) or learned.

- **Model-free** algorithms directly learn how to act for the states encountered during training
  - Which state-action pairs yield good rewards (Q-Learning).

# Reinforcement Learning: Fundamental Problems

- **Learning:**
  - The environment is initially unknown → which states are good or what the actions do.
  - The agent interacts with the environment
  - The agent improves its policy

- **Planning:**
  - A model of the environment is known→ Markov decision problem
  - The agent performs computations with its model
  - The agent improves its policy

# Exploration and Exploitation

- Reinforcement learning is like **trial-and-error** learning.
  - The agent should discover a good policy from its experiences of the environment → without losing too much reward along the way.

- *Exploration*: finds more information about the environment.
- *Exploitation*: exploits known information to maximize reward.

- Interesting trade-off:
  - immediate reward (**exploitation**) vs. gaining knowledge that might enable higher future reward (exploration)

# Exploration and Exploitation: Examples

- Restaurant Selection
  - Exploitation: Go to your favorite restaurant
  - Exploration: Try a new restaurant

- Oil Drilling
  - Exploitation: Drill at the best-known location
  - Exploration: Drill at a new location

# Major Challenges of RL

- Sample efficiency
  - RL algorithms require a large amount of data and experience to learn effectively → costly and time-consuming.
- State and action spaces
  - Exponential growth of state and action spaces as problem complexity increases.
  - For example, in the game of Go, the number of possible board configurations is estimated to be $10^{170}$.

# Major Challenges of RL

- Exploration and exploitation
  - Both exploration and exploitation are essential for learning, but they can also conflict with each other.

# Multi-Agent Reinforcement Learning

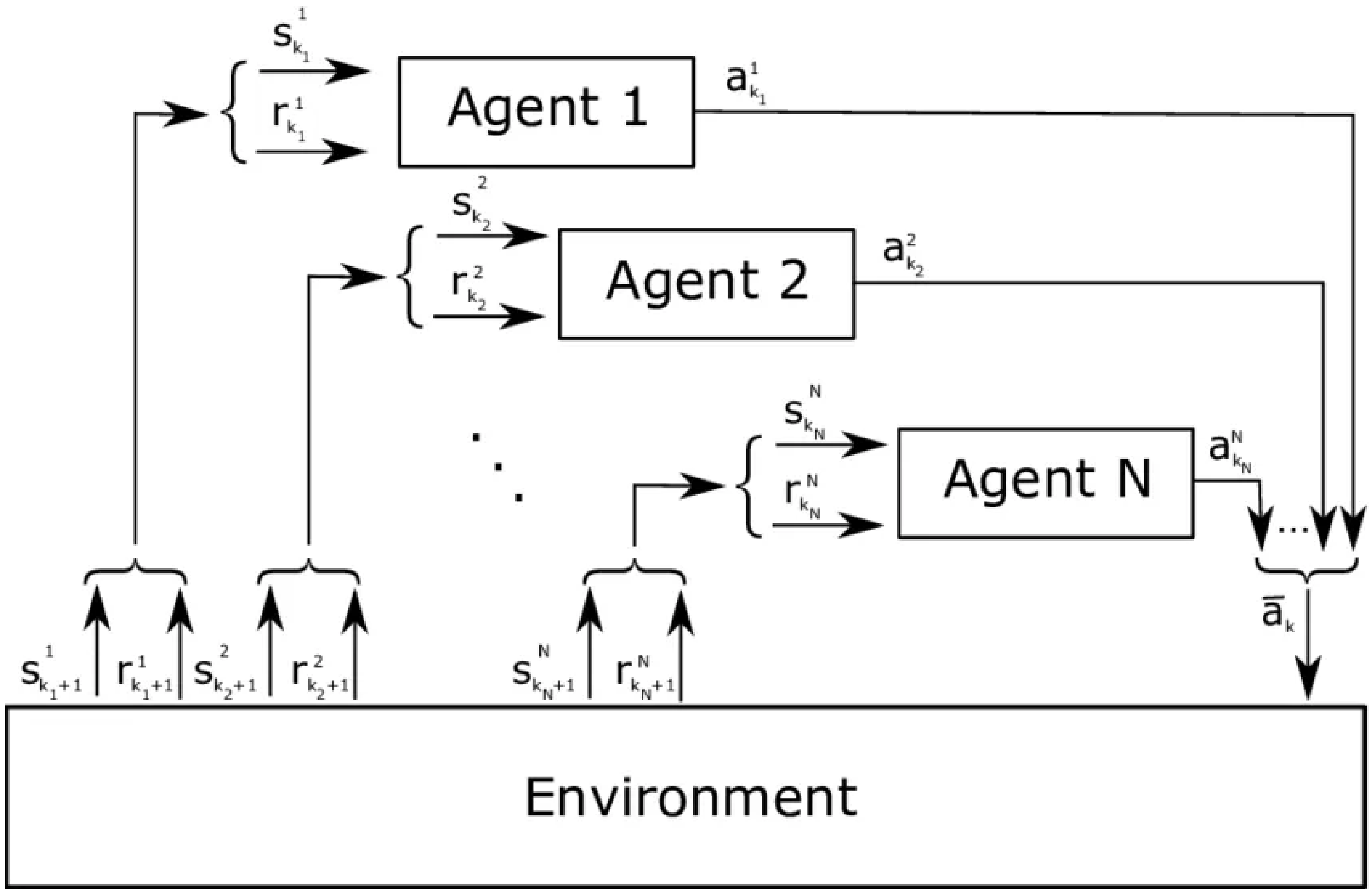
- **Vanilla reinforcement learning** is concerned with a single agent.

- Multi-agent reinforcement learning (MARL) studies how multiple agents interact in a common environment.

  - **Cooperative**: All agents working towards a common goal

  - **Competitive**: Agents competing with one another to accomplish a goal

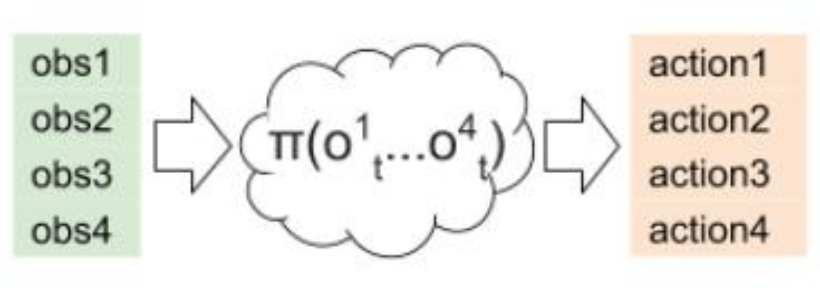
# Multi-Agent Reinforcement Learning



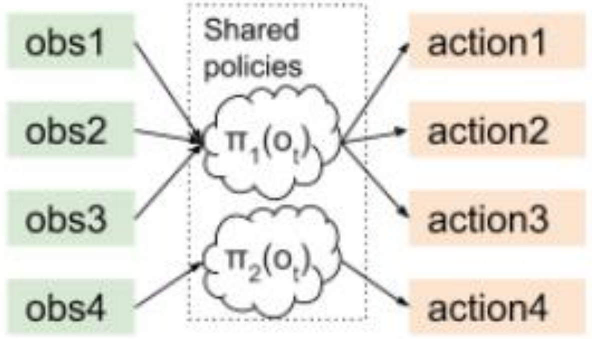
# Why MARL?

- *A more natural decomposition of the problem*: Need to train policies for **cellular antenna tilt control**.
  - Instead of training a single *super-agent* that controls all the cellular antennas in a city→ it is more natural to model each antenna as a separate agent in the environment.

- *Potential for more scalable learning*




Sing super-agent




Multi-agent

# MARL: Use case

- Dota 2: AI agents are trained to coordinate with each other to compete against humans.

# MARL: Challenges

- Environment non-stationarity

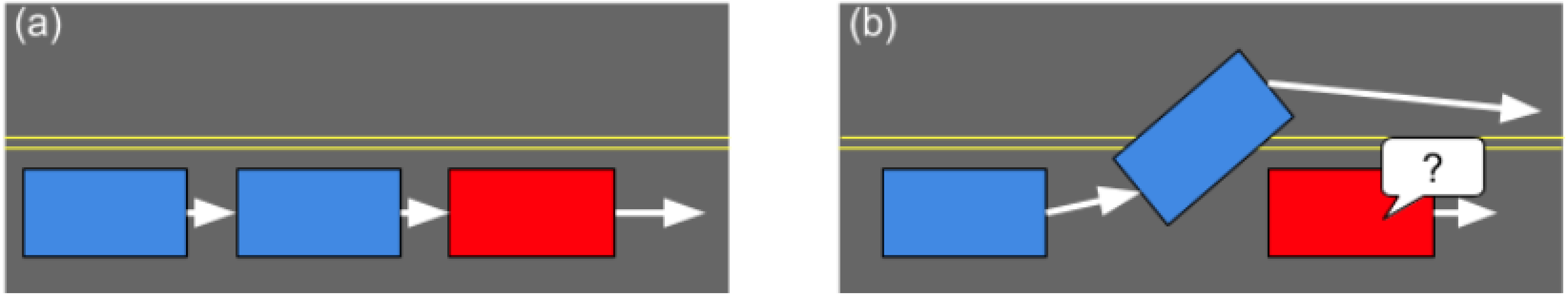



**Fig:** Non-stationarity of environment: Initially **(a)**, the red agent learns to regulate the speed of the traffic by slowing down. However, over time the blue agents learn to bypass the red agent **(b)**, rendering the previous experiences of the red agent invalid.

# References


- Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto.
- David Silver Course on RL