

Reinforcement Learning (AI644)

Prepared by Dr. Ashraf Elsayed

Lecture 1



The background features a series of vertical panels with rounded ends. The leftmost panel shows a person's hands typing on a laptop. The subsequent panels contain various white line-art icons on a semi-transparent background: a graduation cap, a play button, a target with an arrow, an open book, a magnifying glass, and a group of people. The overall color scheme is warm, with shades of orange and yellow.

Reinforcement Learning

Lecture 1

Prepared by
Dr. Ashraf Elsayed

Introduction to Reinforcement Learning

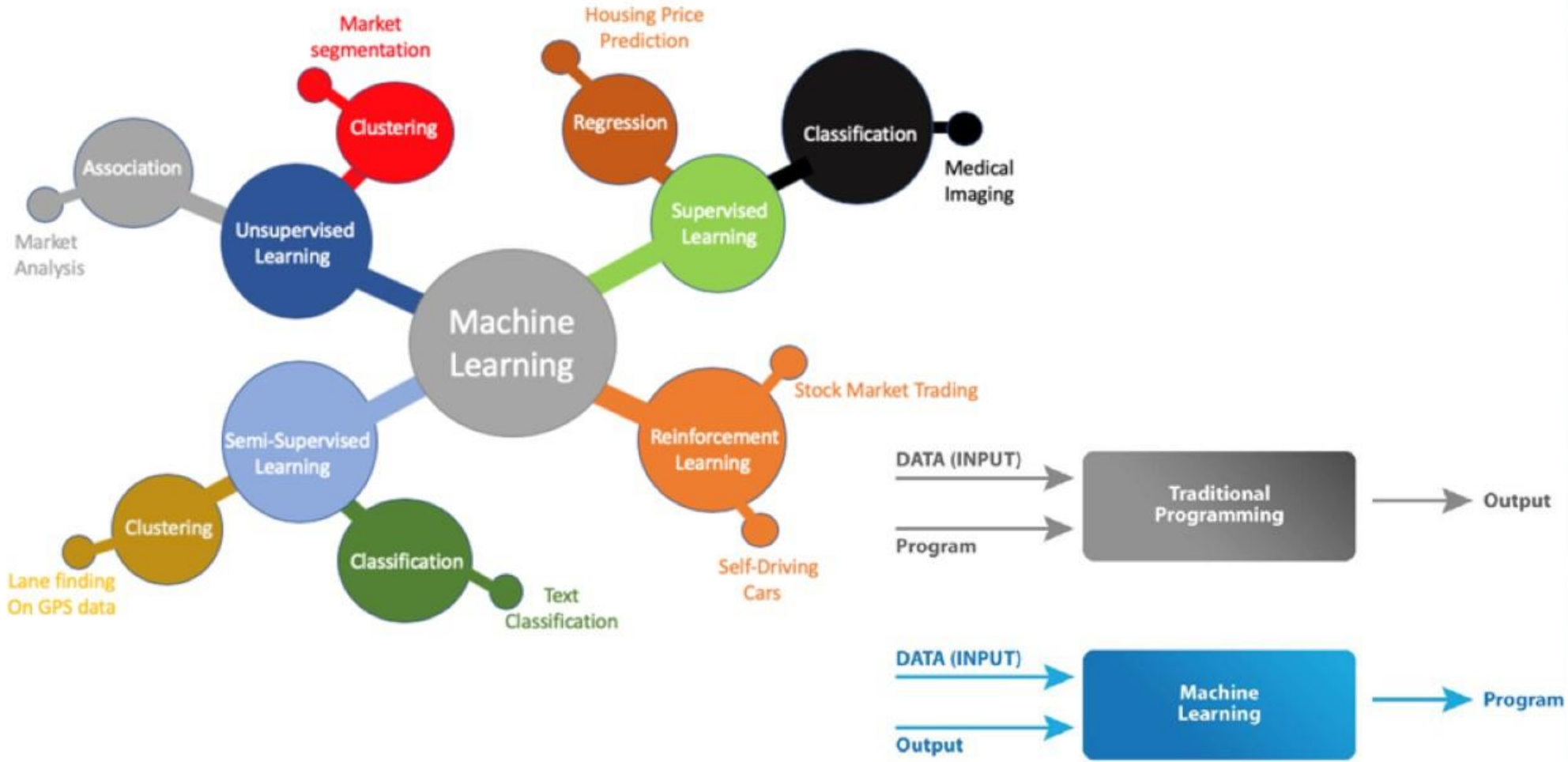
The objectives of the lecture is to:

- Understand Reinforcement Learning Fundamentals
- Define and explain the basic concepts of reinforcement learning, including agents, environments, actions, states, rewards, and policies.
- Understand the difference between supervised learning, unsupervised learning, and reinforcement learning.

What is Reinforcement Learning?

- RL is a type of machine learning paradigm where an agent learns to make decisions by interacting with an environment. The agent takes actions in the environment, and based on these actions, it receives feedback in the form of rewards or penalties. The goal of the agent is to learn a policy, which is a mapping from states to actions, that maximizes the cumulative reward over time.

Machine Learning



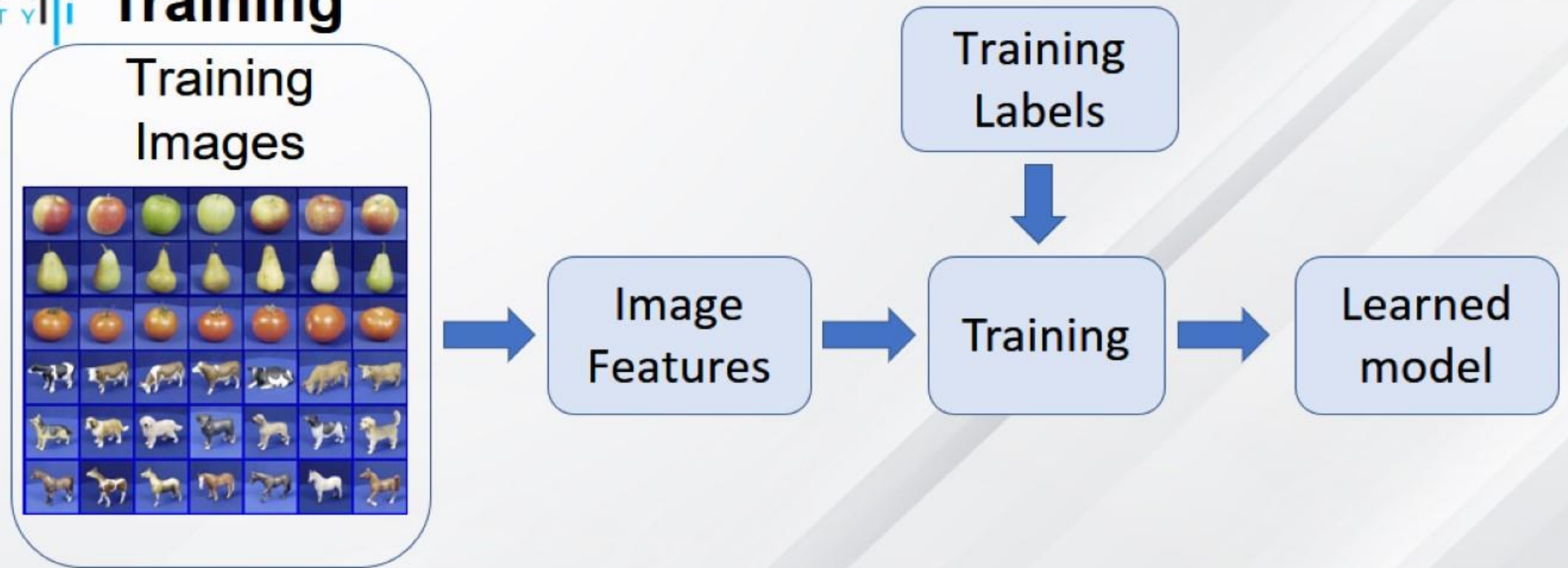
Machine Learning Areas

- **Supervised Learning:** Data and corresponding labels are given
- **Unsupervised Learning:** Only data is given, no labels provided
- **Semi-supervised Learning:** Some (if not all) labels are present
- **Reinforcement Learning:** An agent interacting with the world makes observations, takes actions, and is rewarded or punished; it should learn to choose actions in such a way as to obtain a lot of reward

Supervised Learning

- Learning a discrete function: **Classification**
 - Boolean classification:
 - Each example is classified as true(positive) or false(negative).
- Learning a continuous function: **Regression**

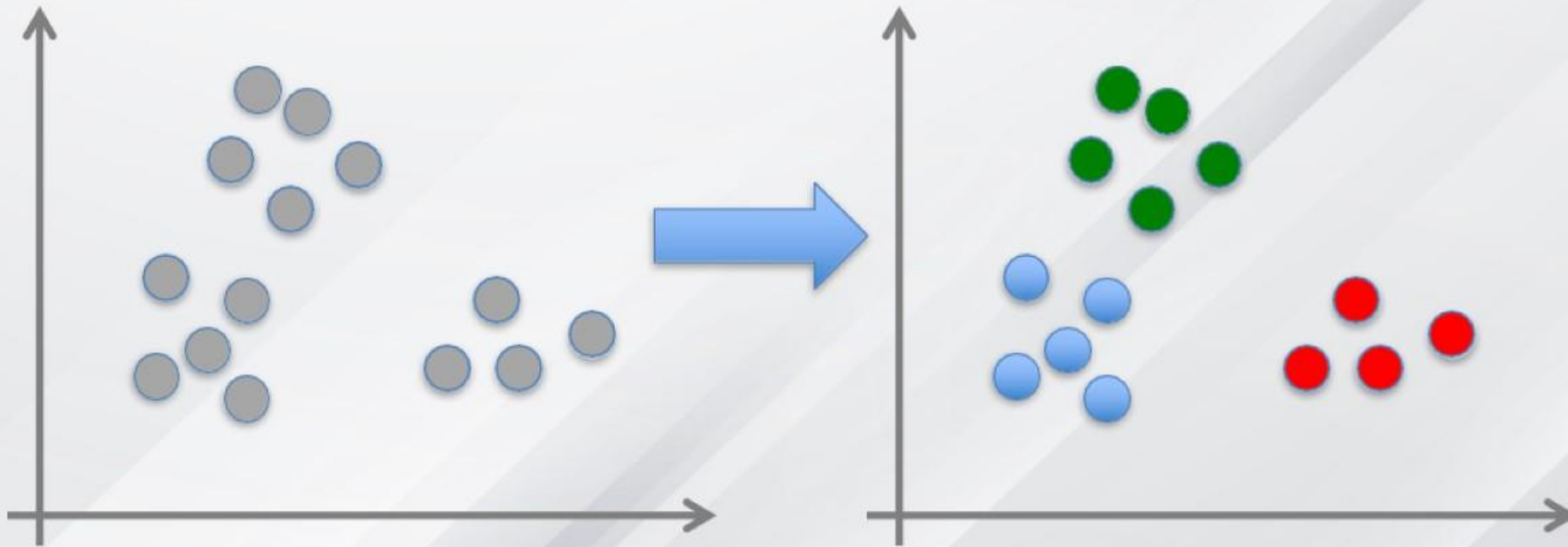
Training



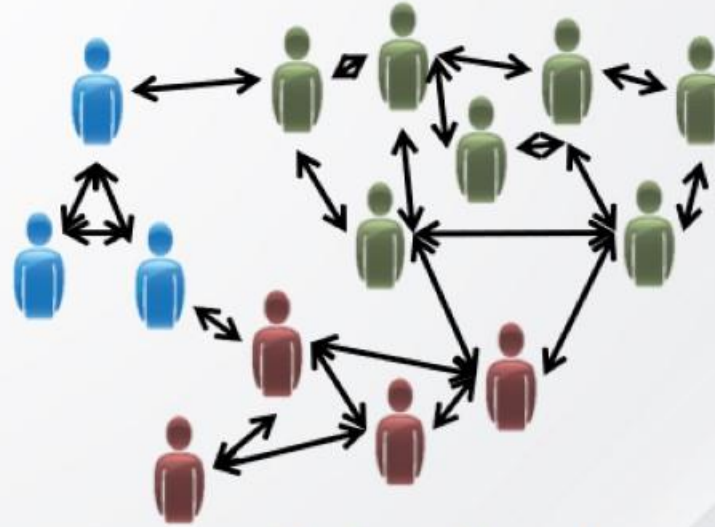
Testing



- Given $x_1, x_2, \dots, x_n$ (without labels)
- Output hidden structure behind the x 's
 - E.g., clustering



Unsupervised Learning



Social network analysis

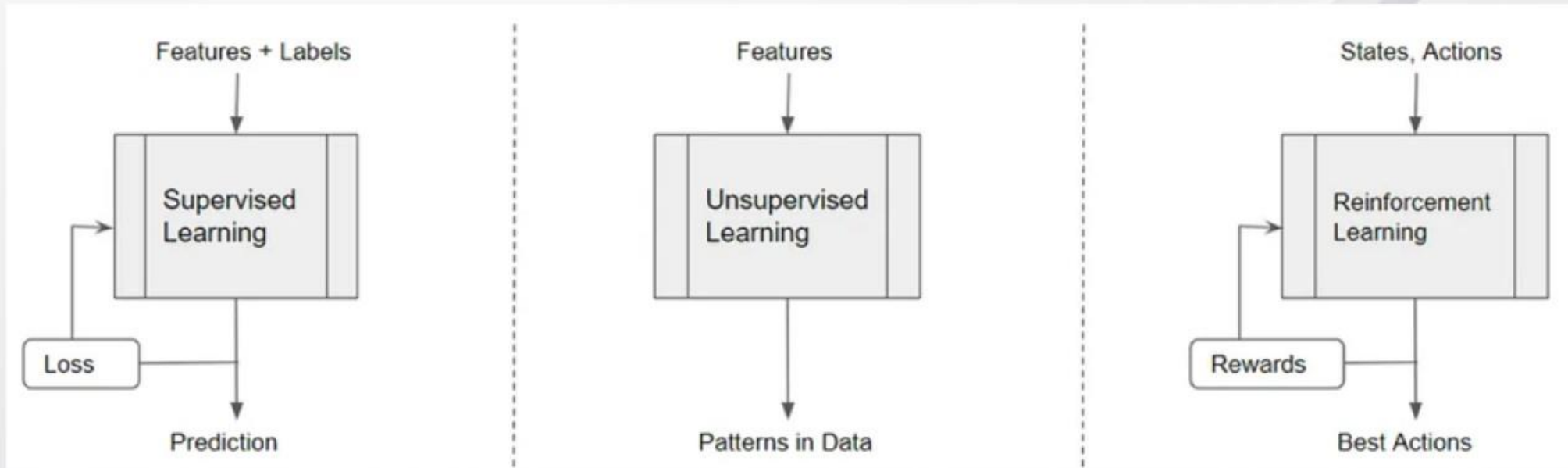


Market segmentation



Astronomical data analysis

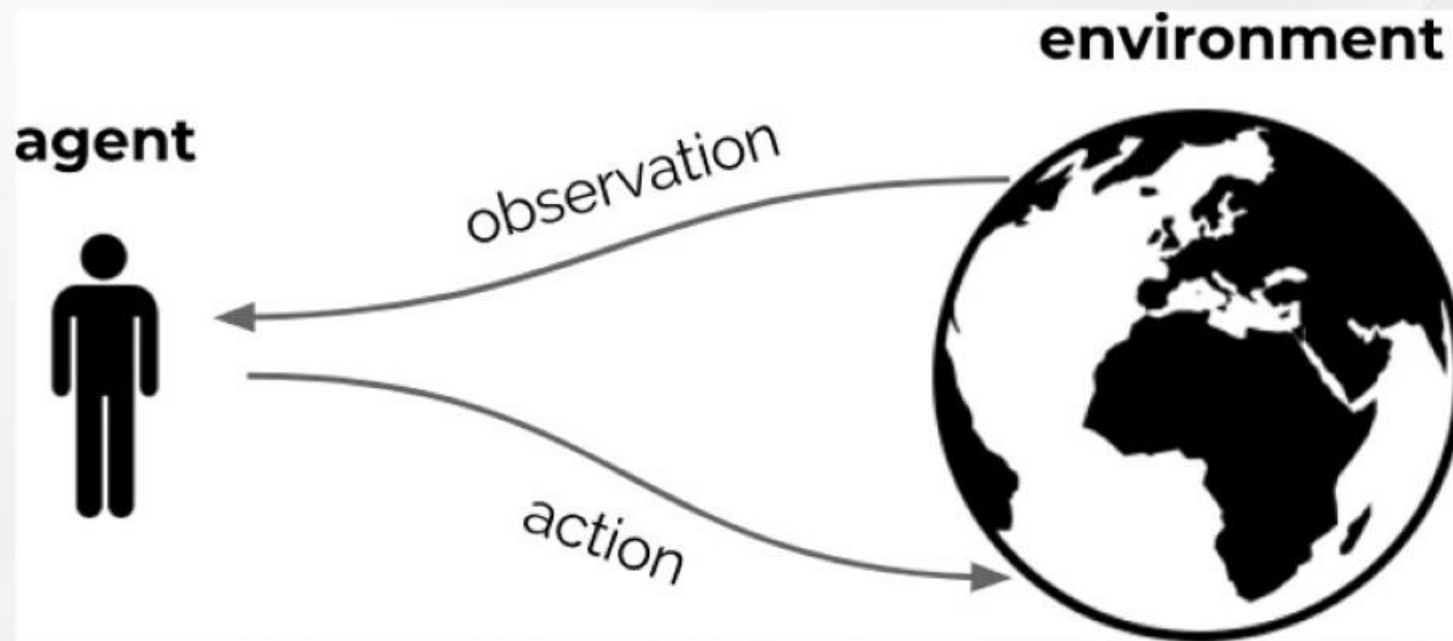
Reinforcement Learning



What is reinforcement learning?

- People and animals learn by **interacting with our environment**
- This differs from certain other types of learning
 - It is **active** rather than passive
 - Interactions are often **sequential** — future interactions can depend on earlier ones
- We are **goal-directed**
- We can learn **without examples** of optimal behaviour
- Instead, we optimize some **reward signal**
- Framework of **learning to make decisions** from **interaction**

The interaction loop



Goal: optimize sum of rewards, through repeated interaction

The reward hypothesis

Reinforcement learning is based on the **reward hypothesis**:

Any goal can be formalized as the outcome of maximizing a cumulative reward

Examples of RL problems

- Fly a helicopter → **Reward**: air time, inverse distance, ...
- Manage an investment portfolio → **Reward**: gains, gains minus risk, ...
- Control a power station → **Reward**: efficiency, ...
- Make a robot walk → **Reward**: distance, speed, ...
- Play video or board games → **Reward**: win, maximize score, ...

If the goal is to learn via interaction, these are all reinforcement learning problems (Irrespective of which solution you use)

What is reinforcement learning?

There are distinct reasons to learn:

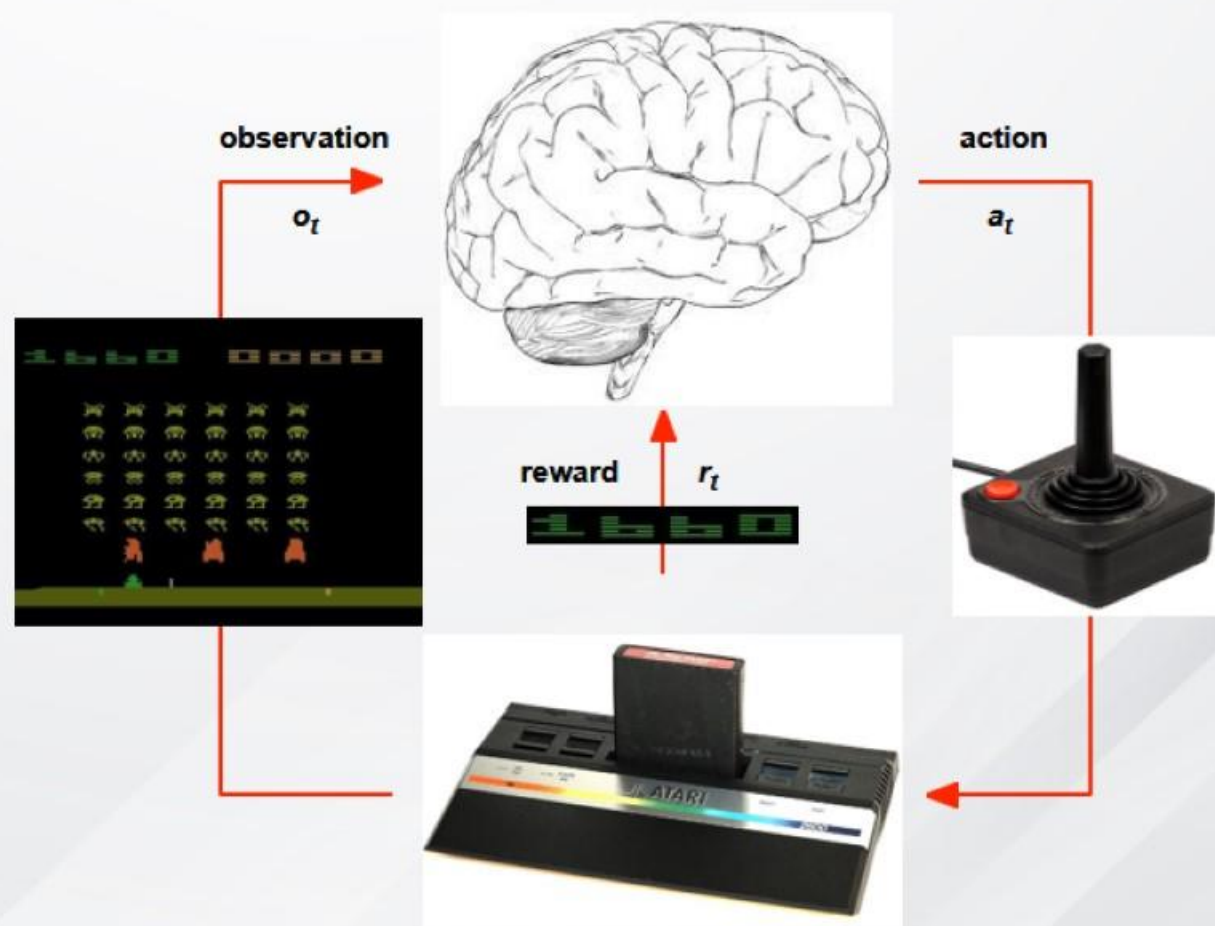
1. Find **solutions**

- A program that plays chess really well
- A manufacturing robot with a specific purpose

2. **Adapt online**, deal with unforeseen circumstances

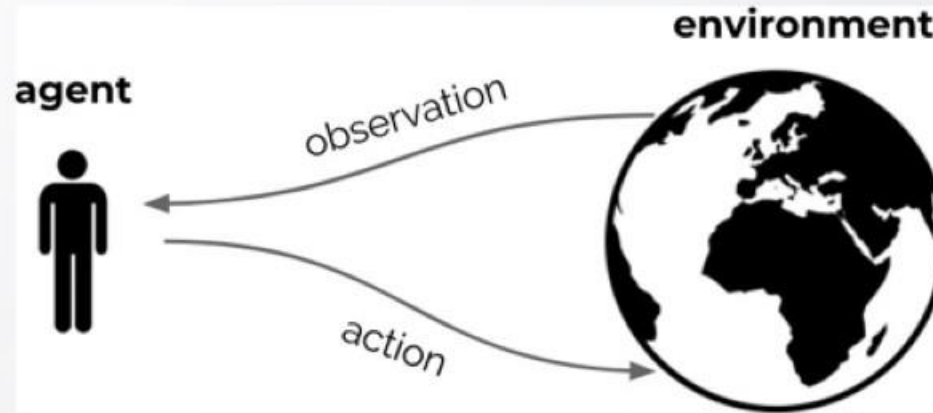
- A chess program that can learn to adapt to you
- A robot that can learn to navigate unknown terrains
- Reinforcement learning can provide algorithms for both cases
- Note that the second point is not (just) about generalization — it is about continuing to learn efficiently online, during operation

Atari Example: Reinforcement Learning



- ▶ Rules of the game are unknown
- ▶ Learn directly from interactive game-play
- ▶ Pick actions on joystick, see pixels and scores

Agent and Environment



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

Rewards

- A **reward** R_t is a scalar feedback signal
- Indicates how well agent is doing at step t — defines the goal
- The agent's job is to maximize cumulative reward

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \dots$$

- We call this the **return**
- Reinforcement learning is based on the **reward hypothesis**:
- *Any goal can be formalized as the outcome of maximizing a cumulative reward*

Values

- We call the expected cumulative reward, from a state s , the **value**

$$v(s) = E[G_t \mid S_t = s]$$

$$= E[R_{t+1} + R_{t+2} + R_{t+3} + \dots \mid S_t = s]$$

Expected future rewards given that the agent is at State s

- The value depends on the actions the agent takes
- Goal is to **maximize value**, by picking suitable actions
- Returns and values can be defined recursively

$$G_t = R_{t+1} + G_{t+1}$$

$$v(s) = E[R_{t+1} + v(S_{t+1}) \mid S_t = s]$$

Maximizing value by taking actions

- Goal: **select actions to maximize value**
- Actions may have long term consequences
- Reward may be delayed
- It may be better to sacrifice immediate reward to gain more long-term reward
- Examples:
 - Refueling a helicopter (might prevent a crash in several hours)
 - Defensive moves in a game (may help chances of winning later)
 - Learning a new skill (can be costly & time-consuming at first)
- A mapping from states to actions is called a **policy**

Action values

- It is also possible to condition the value on **actions**:

$$q_*(a) \doteq \mathbb{E}[R_t | A_t = a]$$

Expected reward given that
the agent take the action a

- We will talk in depth about state and action values later

Core concepts

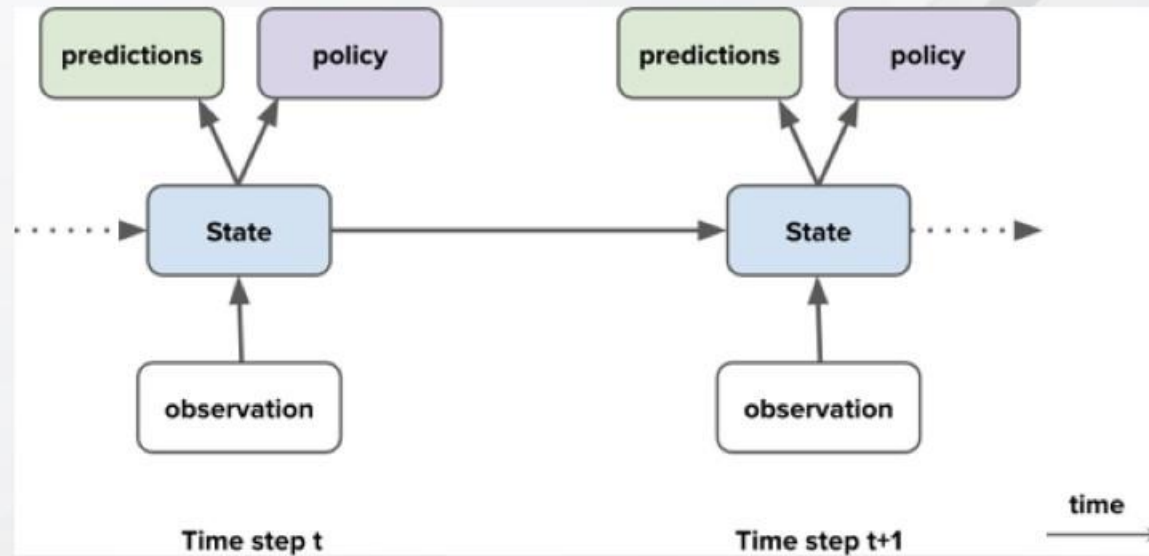
The reinforcement learning formalism includes

- **Environment** (dynamics of the problem)
- **Reward** signal (specifies the goal)
- **Agent**, containing:
 - Agent state
 - Policy
 - Value function estimate?
 - Model?
- We will now go into the agent

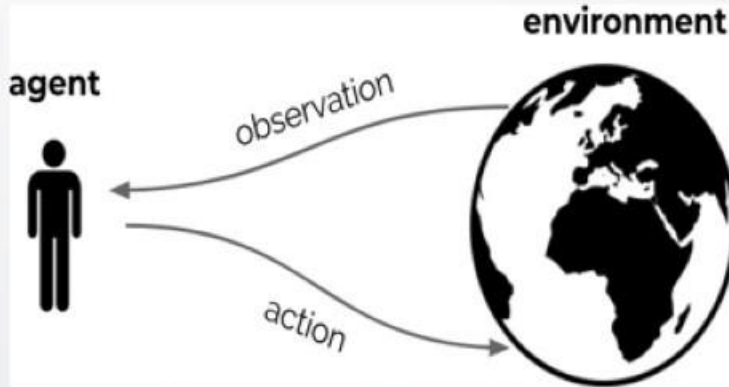
Agent components

Agent components

- **Agent state**
- Policy
- Value functions
- Model



Environment State



- The **environment state** is the environment's internal state
- It is usually invisible to the agent
- Even if it is visible, it may contain lots of irrelevant information

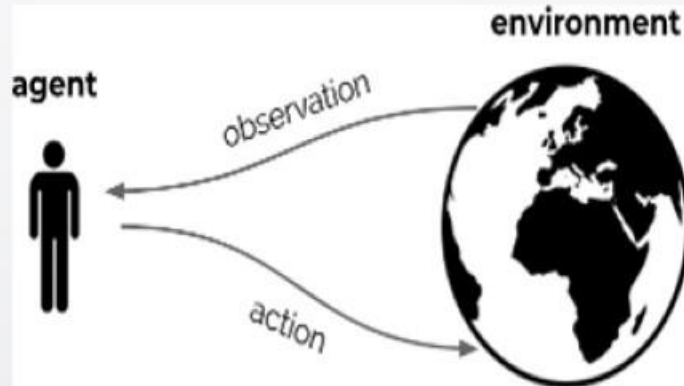
Agent State

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

$$H_t = O_0, A_0, R_1, O_1, \dots, O_{t-1}, A_{t-1}, R_t, O_t$$

- For instance, the sensorimotor stream of a robot
- This history is used to construct the **agent state** S_t

Fully Observable Environments



Full observability

Suppose the agent sees the full environment state

- observation = environment state
- The agent state could just be this observation:

$$S_t = O_t = \text{environment state}$$

Markov decision processes

Markov decision processes (MDPs) are a useful mathematical framework

Definition

A decision process is Markov if

$$p(r, s \mid S_t, A_t) = p(r, s \mid H_t, A_t)$$

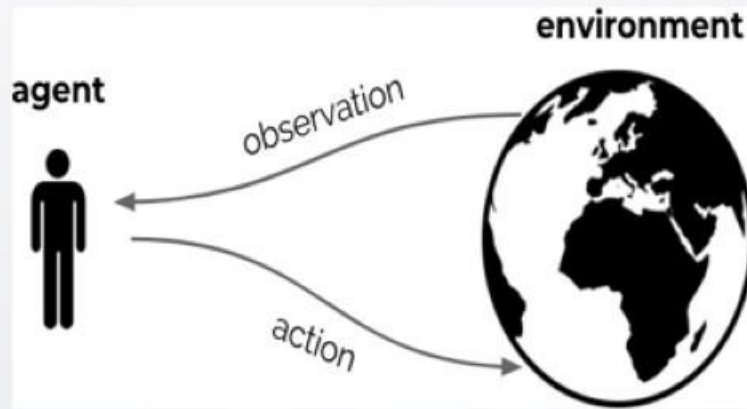
Probability of reward r and state s given that all history of states and action A_t

- This means that the state contains all we need to know from the history
- Doesn't mean it contains everything, just that adding more history doesn't help
- Once the state is known, the history may be thrown away
 - The full environment + agent state is Markov (but large)
 - The full history H_t is Markov (but keeps growing)
- Typically, the agent state S_t is some compression of H_t
- Note: we use S_t to denote the **agent state**, not the **environment state**

Partially Observable Environments

- **Partial observability:** The observations are not Markovian
 - A robot with camera vision isn't told its absolute location
 - A poker playing agent only observes public cards
- Now using the observation as state would not be Markovian
- This is called a **partially observable Markov decision process** (POMDP)
- The **environment state** can still be Markov, but the agent does not know it
- We might still be able to construct a Markov agent state

Agent State



- The agent's actions depend on its state
- The **agent state** is a function of the history
- For instance, $S_t = O_t$
- More generally:

$$S_{t+1} = u(S_t, A_t, R_{t+1}, O_{t+1})$$

where u is a 'state update function'

- The agent state is often **much** smaller than the environment state

Agent State

The full environment state of a maze



Agent State

A potential observation



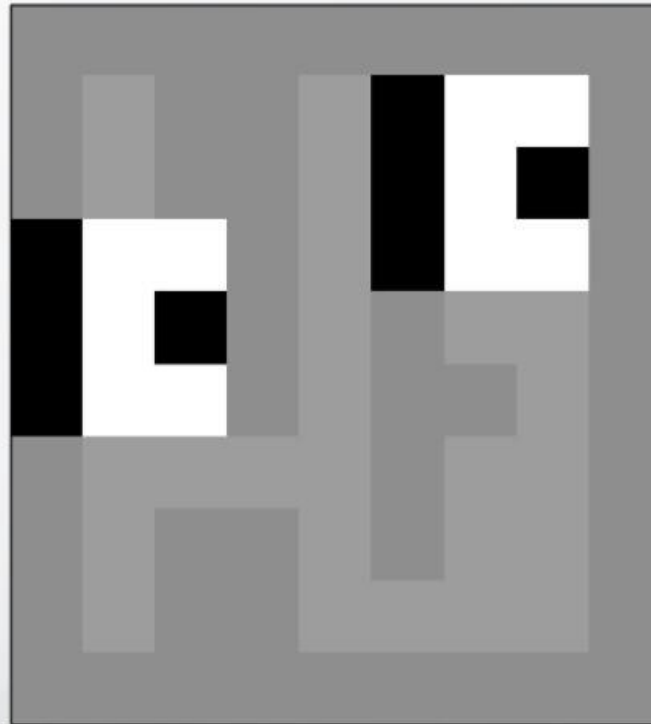
Agent State

An observation in a different location



Agent State

The two observations are indistinguishable



Agent State

These two states are not Markov



How could you construct a Markov agent state in this maze (for any reward signal)?

Partially Observable Environments

- To deal with partial observability, agent can construct suitable state representations
- Examples of agent states:
 - Last observation: $S_t = O_t$ (might not be enough)
 - Complete history: $S_t = H_t$ (might be too large)
 - A generic update: $S_t = \mathcal{U}(S_{t-1}, A_{t-1}, R_t, O_t)$ (but how to pick/learn $\mathcal{U}$?)
- Constructing a fully Markovian agent state is often not feasible
- More importantly, the state should allow good policies and value predictions

Agent components

Agent components

- Agent state
- **Policy**
- Value function
- Model

Policy

- A **policy** defines the agent's behavior
- It is a map from agent state to action
- Deterministic policy: $A = \pi(S)$
- Stochastic policy: $\pi(A|S) = p(A|S)$

Agent components

Agent components

- Agent state
- Policy
- **Value function**
- Model

Value Function

- The actual value function is the expected return

$$v_{\pi}(s) = E[G_t \mid S_t = s, \pi]$$

$$= E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s, \pi]$$

- We introduced a **discount factor** $\gamma \in [0, 1]$
 - Trades off importance of immediate vs long-term rewards
- The value depends on a policy
- Can be used to evaluate the desirability of states
- Can be used to select between actions

Value Function

- The return has a recursive form $G_t = R_{t+1} + \gamma G_{t+1}$
- Therefore, the value has as well

$$\begin{aligned} v_{\pi}(s) &= E[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t \sim \pi(s)] \\ &= E[R_{t+1} + \gamma v_{\pi}(S_{t+1}) \mid S_t = s, A_t \sim \pi(s)] \end{aligned}$$

Here $a \sim \pi(s)$ means a is chosen by policy π in state s

- This is known as a **Bellman equation** (Bellman 1957)
- A similar equation holds for the optimal (=highest possible) value:

$$v_*(s) = \max_a E[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a]$$

This does **not** depend on a policy

Agent components

Agent components

- Agent state
- Policy
- Value function
- **Model**

Model

- A **model** predicts what the environment will do next
- E.g., $\mathcal{P}$ predicts the next state

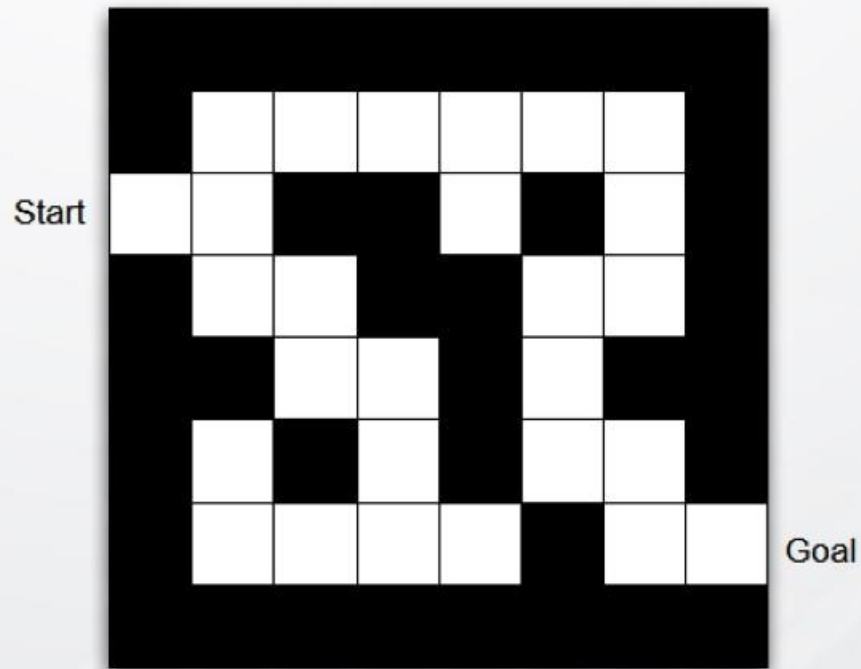
$$\mathcal{P}(s, a, s') \approx p(S_{t+1} = s' \mid S_t = s, A_t = a)$$

- E.g., $\mathcal{R}$ predicts the next (immediate) reward

$$\mathcal{R}(s, a) \approx E[R_{t+1} \mid S_t = s, A_t = a]$$

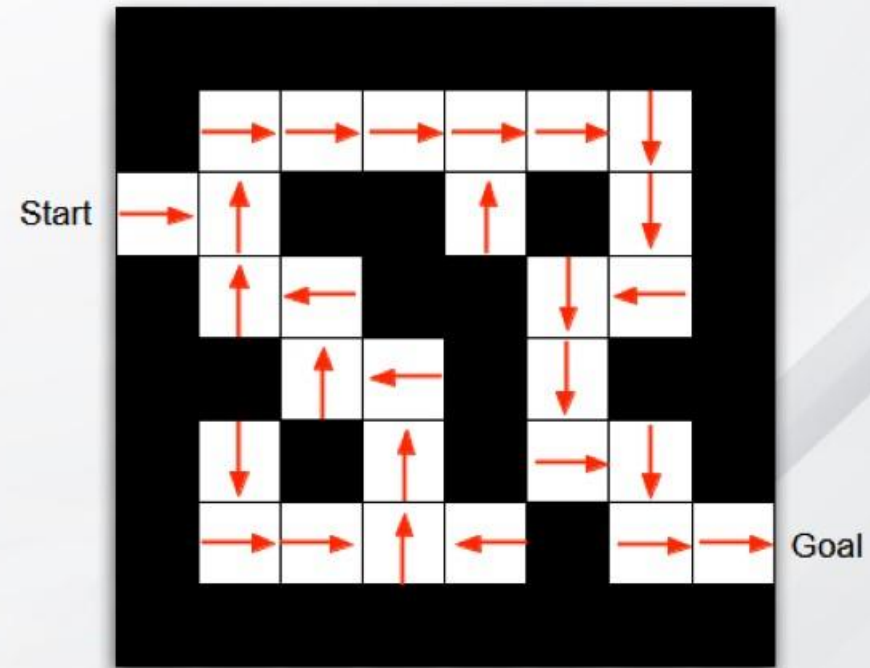
- A model does not immediately give us a good policy - we would still need to plan
- We could also consider **stochastic** (**generative**) models

Maze Example



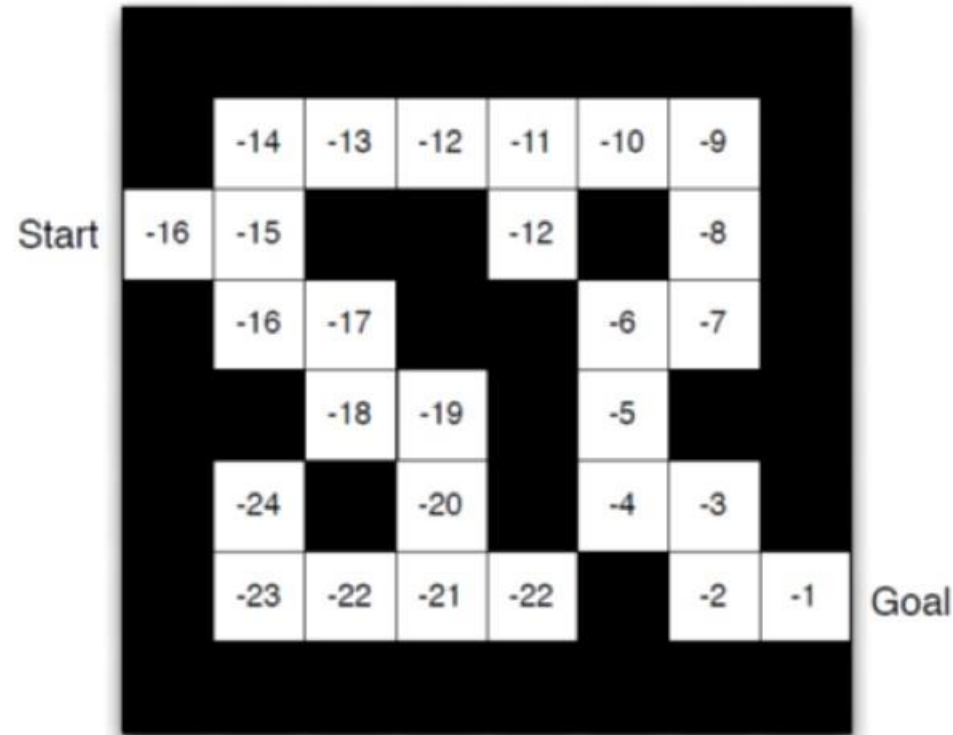
- ▶ Rewards: -1 per time-step
- ▶ Actions: N, E, S, W
- ▶ States: Agent's location

Maze Example: Policy



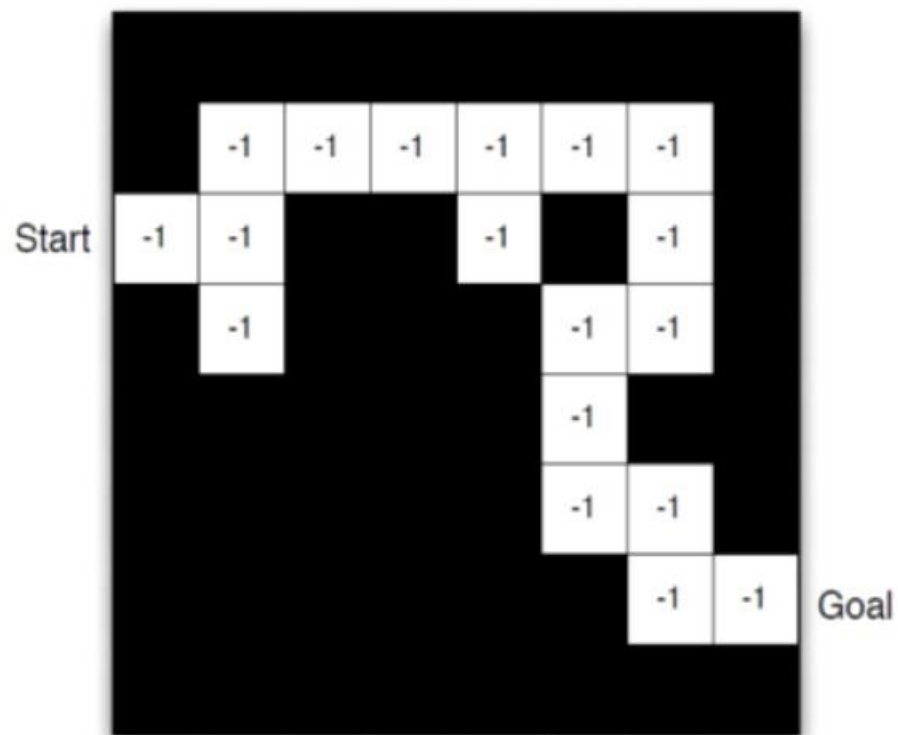
- Arrows represent policy $\pi(s)$ for each state s

Maze Example: Value Function



- Numbers represent value $v_{\pi}(s)$ of each state s

Maze Example: Model



- ▶ Grid layout represents partial transition model $\mathcal{P}_{ss'}^a$
- ▶ Numbers represent immediate reward $\mathcal{R}_{ss'}^a$ from each state s (same for all a and s' in this case)

Agent Categories

- ▶ Value Based
 - ▶ No Policy (Implicit)
 - ▶ Value Function
- ▶ Policy Based
 - ▶ Policy
 - ▶ No Value Function
- ▶ Actor Critic
 - ▶ Policy
 - ▶ Value Function

Agent Categories

- ▶ **Model Free**
 - ▶ Policy and/or Value Function
 - ▶ No Model
- ▶ **Model Based**
 - ▶ Optionally Policy and/or Value Function
 - ▶ Model

Learning Agent Components

- All components are functions
 - Policies: $\pi: S \rightarrow A$ (or to probabilities over A)
 - Value functions: $v: S \rightarrow \mathbb{R}$
 - Models: $m: S \rightarrow S$ and/or $r: S \rightarrow \mathbb{R}$
 - State update: $u: S \times O \rightarrow S$
- E.g., we can use neural networks, and use **deep learning** techniques to learn
- Take care: we do often violate assumptions from supervised learning (iid, stationarity)
- Deep learning is an important tool
- Deep reinforcement learning is a rich and active research field

Reinforcement Learning Applications



Reinforcement Learning Applications

1 Control Systems

RL learns optimal control policies for dynamic systems.

Examples:

- Adaptive cruise control in autonomous vehicles
- Power grid voltage and frequency control
- Smart HVAC temperature regulation

Reinforcement Learning Applications

2 Game Playing

RL agents learn strategies by interacting with the game environment.

Examples:

- AlphaGo defeating human champions
- AI agents playing Atari games
- Real-time strategy games (e.g., StarCraft AI)

Reinforcement Learning Applications

3 Robotics

RL enables robots to learn tasks through trial and error.

Examples:

- Robotic arm grasping and manipulation
- Humanoid robot walking
- Drone navigation and obstacle avoidance

Reinforcement Learning Applications

4 Business

RL optimizes sequential business decisions to maximize profit or engagement.

Examples:

- Dynamic pricing in e-commerce
- Personalized advertisement selection
- Customer recommendation systems

Reinforcement Learning Applications

5 Manufacturing

RL improves production efficiency and quality.

Examples:

- Job-shop scheduling optimization
- Predictive maintenance of machines
- Industrial process parameter tuning

Reinforcement Learning Applications

6 Finance Sector

RL supports intelligent financial decision-making under uncertainty.

Examples:

- Algorithmic stock trading
- Portfolio rebalancing strategies
- Risk-aware investment management

Reinforcement Learning Applications

7 Chemistry

RL accelerates discovery and optimization in chemical systems.

Examples:

- Drug molecule design
- Chemical reaction optimization
- Material property discovery

Supervised vs. Unsupervised vs. Reinforcement Learning



Criteria	Supervised ML	Unsupervised ML	Reinforcement ML
Definition	Learns by using labelled data	Trained using unlabelled data without any guidance.	Works on interacting with the environment
Type of problems	Regression and classification	Association and Clustering	Exploitation or Exploration
Algorithms	Linear Regression, Logistic Regression, SVM, KNN etc.	K – Means, C – Means, Apriori	Q – Learning, SARSA
Aim	Calculate outcomes	Discover underlying patterns	Learn a series of action
Application	Risk Evaluation, Forecast Sales	Recommendation System, Anomaly Detection	Self Driving Cars, Gaming, Healthcare

Term Used in Reinforcement Learning

Agent

→ An entity that can perceive/explore the environment and act upon it.

Environment

→ A situation in which an agent is present or surrounded by.

Action

→ Actions are the moves taken by an agent within the environment.

State

→ State is a situation returned by the environment after each action taken by the agent.

Reward

→ A feedback returned to the agent from the environment to evaluate the action of the agent.

Policy

→ Policy is a strategy applied by the agent for the next action based on the current state.

Value

→ It is expected long-term return with the discount factor and opposite to the short-term reward.

Elements of Reinforcement Learning

There are four main elements of Reinforcement Learning, which are given below

1. Policy:

A policy can be defined as a way how an agent behaves at a given time. It maps the perceived states of the environment to the actions taken on those states.

The policy-based approach has mainly two types of policy:

- **Deterministic:** The same action is produced by the policy (π) at any state.
- **Stochastic:** In this policy, probability determines the produced action.

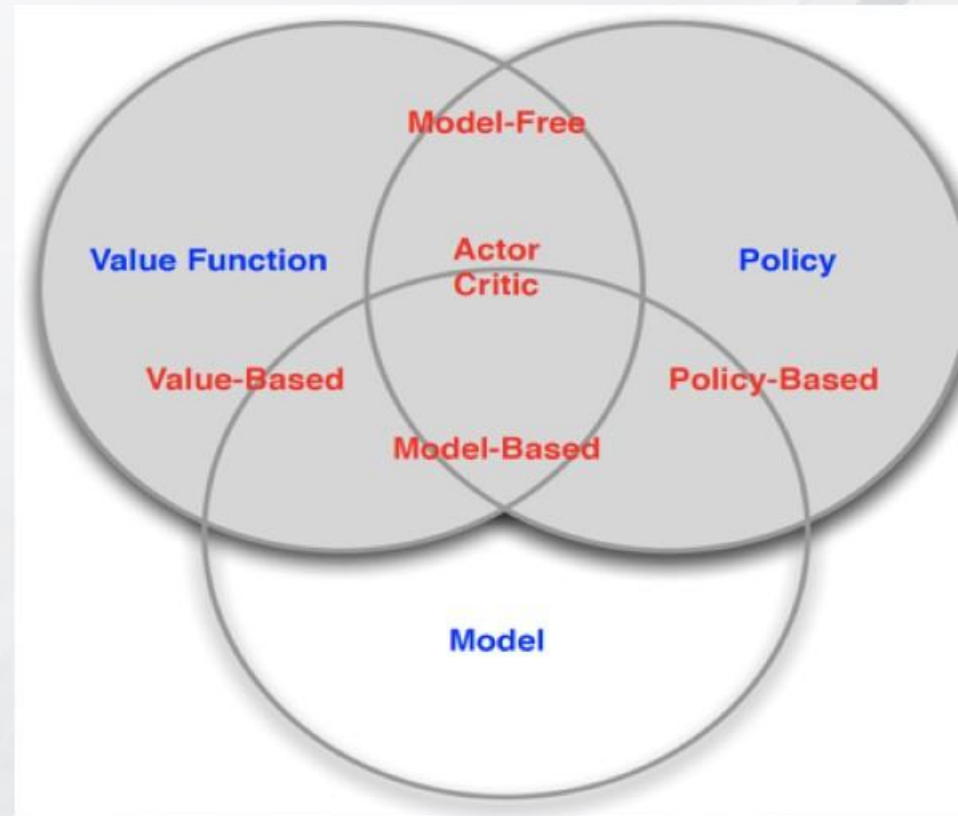
2. Reward Signal: The goal of RL is defined by the reward signal. Reward signals are given according to the good and bad actions taken by the learning agent. The main objective is to maximize the total number of rewards for good actions.

3. Value Function: Gives information about how good the situation and action are and how much reward an agent can expect.

4. Model: Mimics the behavior of the environment.

Reinforcement Learning Categories

- **Value Based**
 - No Policy
 - Value Function
- **Policy Based**
 - Policy
 - No Value Function
- **Actor Critic**
 - Policy
 - Value Function
- **Model Free**
 - Policy and/or Value Function
 - No Model
- **Model Based**
 - Policy and/or Value Function
 - Model



Passive vs. Active learning

- Passive learning
 - The agent has a fixed policy and tries to learn the utilities of states by observing the world go by
 - Analogous to policy evaluation
 - Often serves as a component of active learning algorithms
 - Often inspires active learning algorithms
- Active learning
 - The agent attempts to find an optimal (or at least good) policy by acting in the world
 - Analogous to solving the underlying MDP, but without first being given the MDP model

Model-Based vs. Model-Free RL

- *Model based approach to RL:*
 - learn the MDP model, or an approximation of it
 - use it for policy evaluation or to find the optimal policy
- *Model free approach to RL:*
 - derive the optimal policy without explicitly learning the model
 - useful when model is difficult to represent and/or learn
- We will consider both types of approaches



Thank You