Introduction to RL

AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY
श्रद्धावान् लभते ज्ञानम्

Reinforcement Learning is exactly this — “Learning by interacting with the environment.”

Introduce the Two Core Components- Agents and Environment

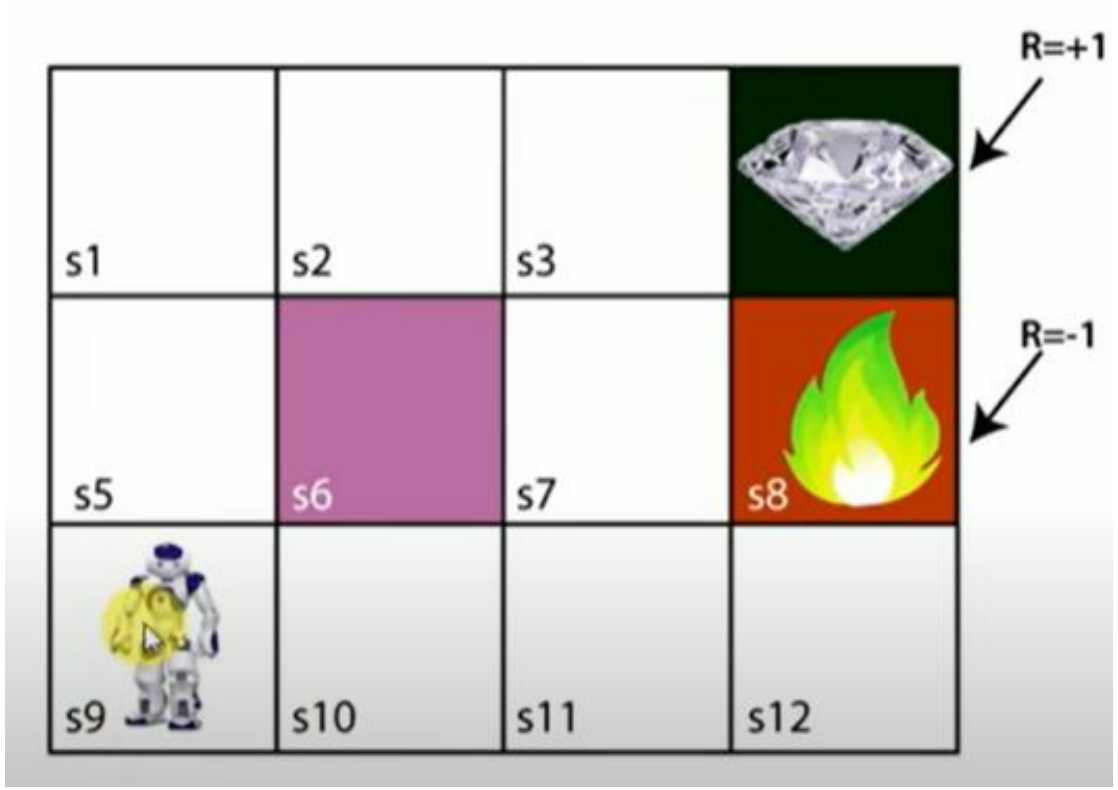
1. Environment

- In this example, the grid is the **environment**.
- Each cell s1,s2,…,s12 are **states**.
- The environment provides **feedback** to the agent after every move in the form of **reward**.

hii

2. Agent

- The little robot is the **agent**.
- It can take **actions**: up, down, left, right.
- Its goal is to learn **which sequence of moves** leads it to good outcomes (i.e., the diamond), and helps it avoid bad ones (i.e., the fire).



The agent's **goal** is to learn a **policy** — a mapping from each state to the best action — that **maximizes total cumulative reward** over time.

"Just like a child learning to walk — this robot learns balance, coordination, and control through repeated trial and error."

No rules. No instructions. Just learning through feedback: fall, try again, adjust

https://www.youtube.com/shorts/Ll7l2yya-bU?feature=share

Robots –opening door
https://www.youtube.com/shorts/aPthvhfAVio?feature=share

https://www.youtube.com/watch?v=M-QUkgk3HyE
Stanford : Andrew NG Teaching a Helicopter to Fly Stunts Autonomously (2000)- Inverse Reinforcement Learning (IRL)

Boston Dynamics is a robotics company known for building highly dynamic, agile, and lifelike robots.

Boston Dynamics- Atlas
https://www.youtube.com/watch?v=LikxFZZO2sk

Their most famous robots include:

- Atlas – a humanoid robot that runs, jumps, does parkour
- Spot – a four-legged robot dog
- Stretch – a box-handling warehouse robot

What is Reinforcement Learning?

Learn to make good sequence of decisions

Reinforcement Learning is a paradigm of **machine learning** where an **agent** learns to take **actions** in an **environment** to maximize a long-term **cumulative reward**.

At its core, RL is about:

- **Trial-and-error learning**
- **Sequential decision-making**
- **Learning from interaction**, not from direct supervision

Eg Analogy : A Child Learning to Crawl

Just like the child:

- Tries different movements (actions)
- Receives feedback (falls down, reaches toy, etc.)
- Learns over time what sequence of movements help them move forward (policy)

This captures the **essence of RL**:

The agent learns from its own experience — not from labeled data, but from **consequences of its actions**.

Sequential Decision Making

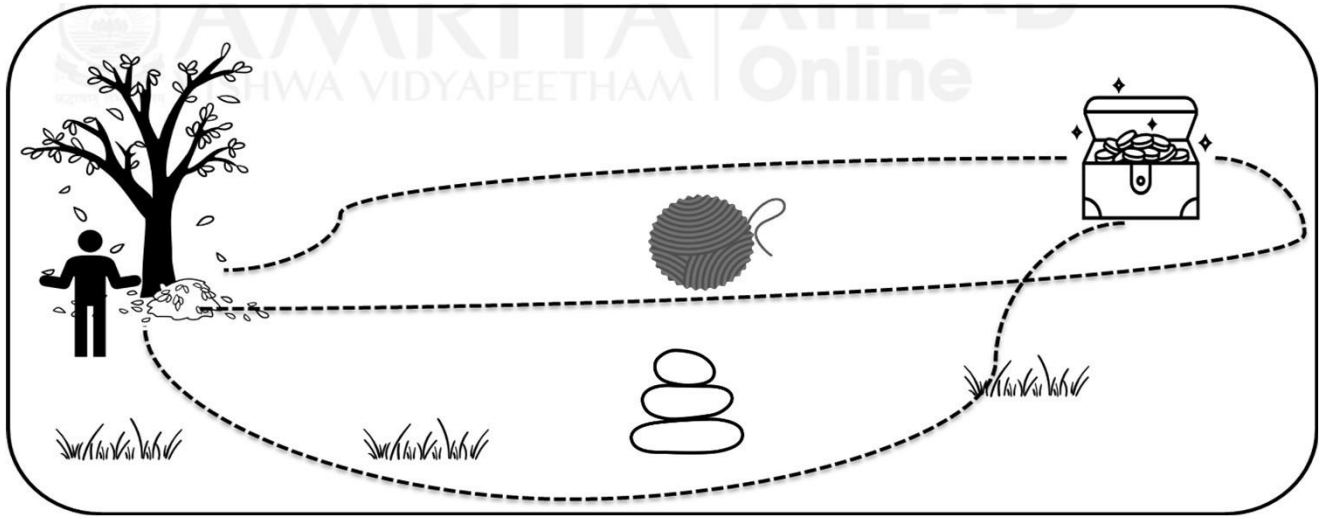
- Series of **decisions over time**
- Decision outcomes may **depend on environmental factors**
- **Final goal** depends on many interactive decisions and their random consequences
- Examples:
  - § Traffic signal control
    - **Objective**: Minimize waiting time and congestion.
    - RL can learn optimal green/red timing based on live traffic patterns.
    - Each signal decision affects the next traffic state → sequential

  - § Communication Network Packet Routing
    - **Objective:** Deliver packets with minimal delay.
    - Routing decisions affect future network load and delay.
    - RL can learn adaptive routing policies depending on congestion and failures.

  - Autonomous Vehicles
    - **Objective: Safe navigation with time and fuel efficiency**
    - Driving involves constant decision-making:
    - Speed up, slow down, change lanes, avoid obstacles
    - These decisions are dependent and sequential.
    - RL agents learn to optimize safety + speed + fuel efficiency over the long term.

What is RL?

- Science of decision making
- Discover the sequence of actions – trial and error
- Learns the optimal behavior through interactions with the environment
- Actions receive a reward or penalty from the environment



- After many attempts the robot learns the best path

Reinforcement Learning – How Learning happens

- In **Reinforcement Learning (RL)**, the **learning** happens as the agent **interacts with the environment** and updates its **policy** or **value estimates** based on the feedback (rewards or penalties) it receives.

- Unlike supervised learning, the agent is **not given the correct answer**; instead, it must **learn through experience** what actions are best.

How Does It Learn Better Over Time?

- In early episodes, the agent tries many actions (exploration).
- It remembers what worked well (via Q-values or policy gradients).
- Gradually favors better actions (exploitation).
- This leads to an improved policy that achieves higher cumulative reward.

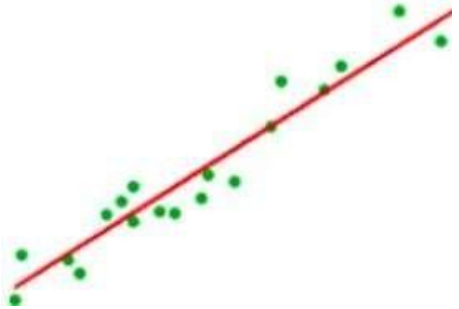
Reinforcement Learning is like a student trying a maze, hitting dead-ends, and learning which path leads to the exit — **that's learning through experience.**"

https://www.youtube.com/shorts/0opKhTKxXX0?feature=share

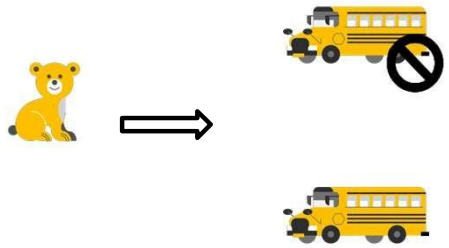
Type of Learning Problems

1Supervised

A. Regression

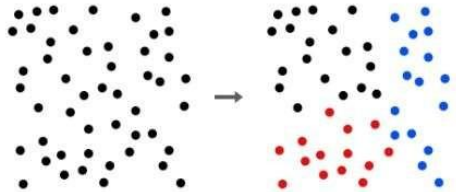


B. Classification



2. Unsupervised Learning

A. Clustering



B. Association

If customer purchased an item "A" ⟹ Then recommend item "B"

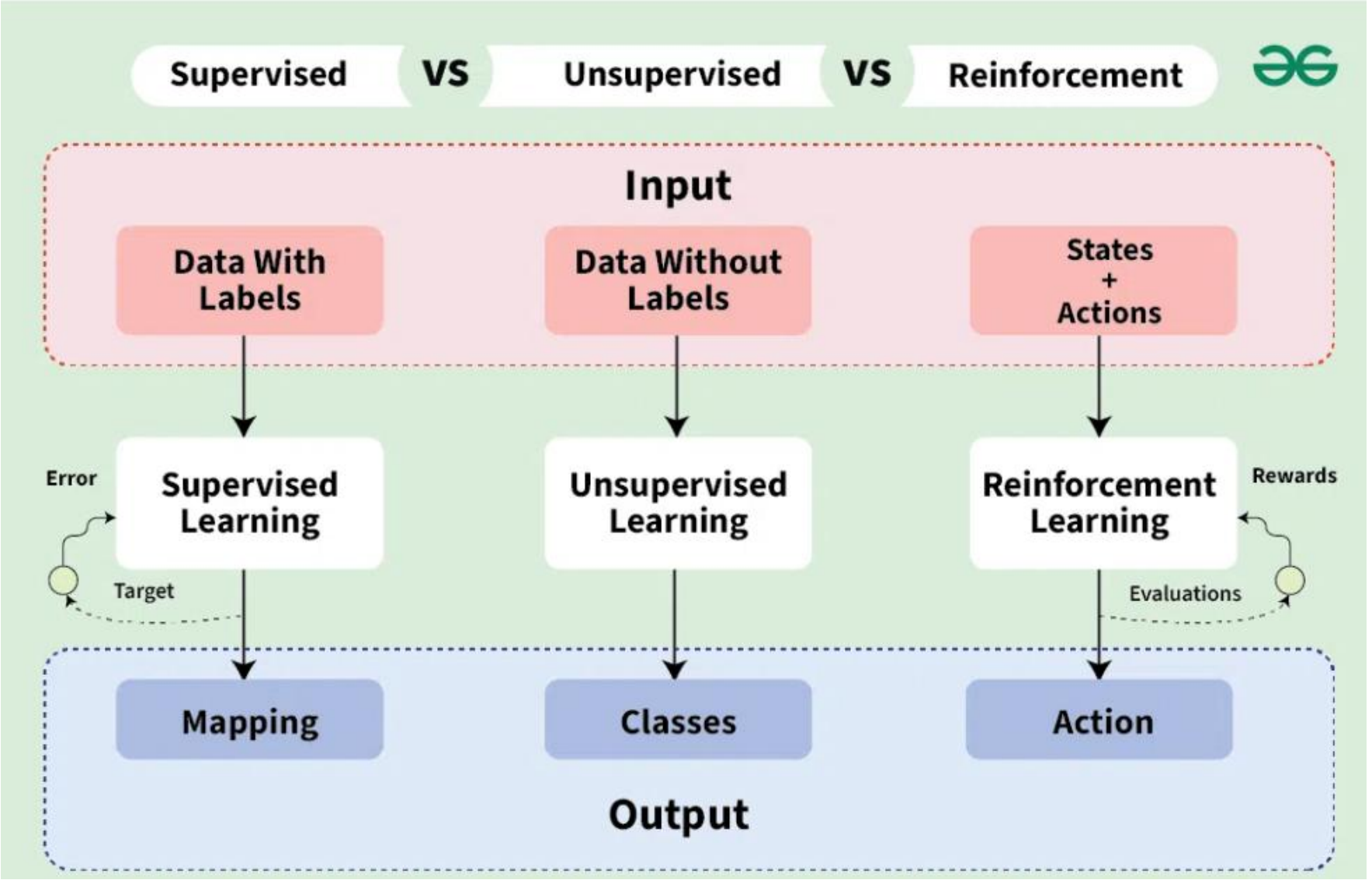
3. Reinforcement Learning

A. Reward Based

1.Supervised Learning: Learning from labelled data.

2.Unsupervised Learning: Discovering patterns in unlabeled data.

3.Reinforcement Learning: Learning through interactions with an environment.



Note the feedback arrows:

•**Supervised**: error is calculated from known targets **RL**: reward is given based on actions and environment response

How RL different from SL and USL?

- There is no supervisor to guide the training
- Not required to train with a large (labeled or unlabeled) dataset.
- Data is provided dynamically via feedback environment with which you are interacting.
- Make decisions over a sequence of time-steps
- Work in dynamic and uncertain environments

•Not **learning from labels** (as in supervised learning)
•Not **learning patterns or clusters** (as in unsupervised learning)
•Instead, it is **learning through experience** and delayed feedback

Comparison of SL, USL and RL

Supervised Learning	Unsupervised Learning	Reinforcement Learning
Labelled data with target	Unlabelled data without target	Input data not predefined: learns from environment using rewards and penalty
External Supervision	No supervision	
Learn pattern in data and its labels	Learn to group data	Compute best reward to reach goal from start state
Map input data to known labels	Find similar features in data and understand patterns	Maximize rewards following trail and error approach
Model training prior to testing	Model training prior to testing	Model training and testing simultaneously
E.g. Regression and classification problems	E.g. Association mining and clustering	E.g. Reward based problems planning, control

A full interaction between Agent and Environment

- **S0**: The current state observed by the agent.
- **A0**: The action chosen by the agent in that state.
- **R1**: The reward received from the environment for taking action A0.
- **S1**: The resulting next state of the environment

Represents **one full interaction** between the **agent and the environment**.

It continues into a **trajectory**:

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, \ldots$$

Through many such interactions, the agent learns an optimal policy.

Convergence - It **learns a stable policy** that no longer changes significantly with more interaction — and that **maximizes expected cumulative reward** over time.

Symbol	Meaning	Role in RL
$\mathbf{S_0}$	Initial **state** at time step 0	The agent **observes the environment** — this is its current situation
$\mathbf{A_0}$	Action taken at time step 0	The agent **chooses an action** based on policy π
$\mathbf{R_1}$	Reward received **after** taking A_0	The **environment gives feedback**: was the action good or bad?
$\mathbf{S_1}$	The new state at time step 1	The agent is **moved to a new state** based on the environment's dynamics

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, \ldots$$

Real World Applications of RL

Smart Vehicles	• Self Driving Cars • Autonomous Helicopters
Games	• Atari • Alpha Go
Robotics	• Navigation • Surveillance
Healthcare	• Manage Critical Diseases • Adaptive Treatment Plans
Finance	• Stock market • Portfolio optimizations
Smart Ads	• Personalized Ads • Recommendation Systems
Chatbots	• Siri • Alexa

Smart Vehicles

Application: Self-Driving Cars

Problem: Learn to drive safely by observing surroundings and taking real-time actions (steering, acceleration, braking)

How RL is used:

1.State: Camera/LiDAR inputs, current speed, position

2.Actions: Turn left/right, accelerate, decelerate, brake

3.Reward:

1. +1 for staying in lane
2. –1 for lane deviation or close to collision
3. –100 for crash

4.Goal: Learn driving policy that avoids crashes and follows traffic rules

Example:

•**Wayve** and **Tesla Autopilot** use variants of **deep reinforcement learning** to improve navigation and decision-making.

Games

Application: AlphaGo, Atari Games

Problem: Play games better than humans using RL agents

How RL is used:

1.State: Game screen pixels or board configuration

2.Actions: Legal moves (left, right, place stone, fire)

3.Reward:

1. +1 for winning
2. –1 for losing
3. Intermediate rewards based on game points

4.Goal: Maximize expected game score or win probability

Example:

•**AlphaGo** (by DeepMind): Defeated world champion in Go using **Monte Carlo Tree Search + Policy Gradients**

•**Atari** (by Deep Q-Network, DQN): RL agents learn to play from pixel input by trial-and-error.

AlphaGo movie Award Winning documentary
https: //www.youtube.com/watch?v=WXuK6gekU1Y

Healthcare

Application: Adaptive Treatment Plans

Problem: Tailor treatment dynamically based on patient response

How RL is used:

1.State: Patient vitals, current condition, previous treatments

2.Actions: Prescribe treatment A, B, or dosage level

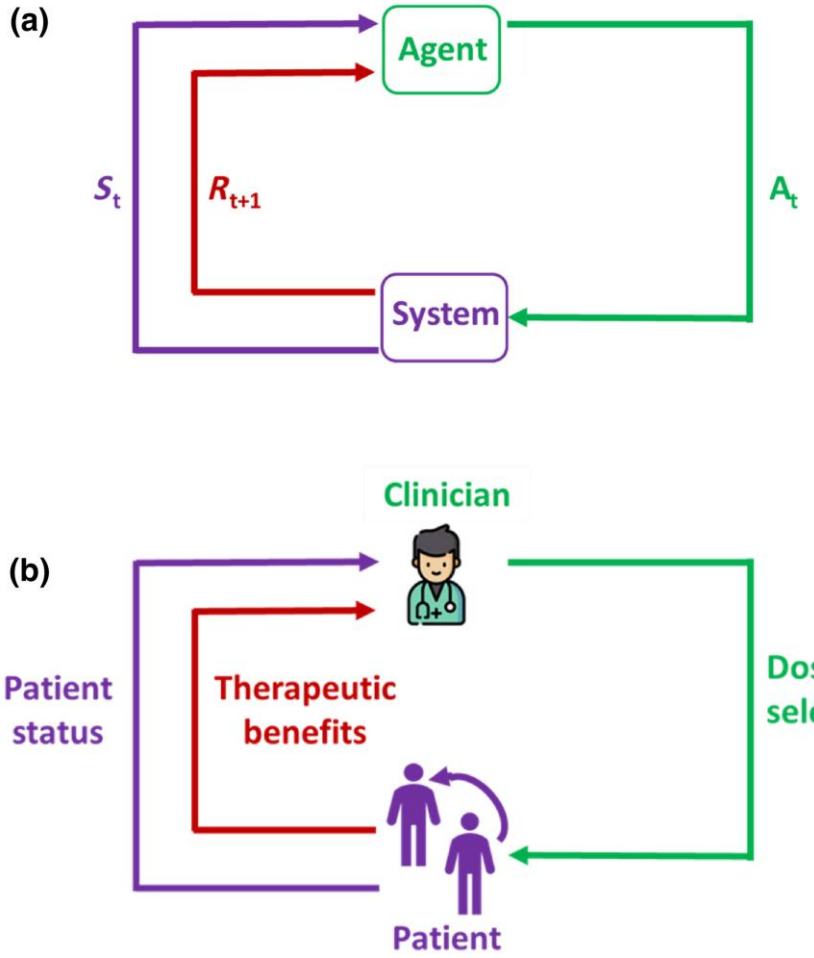
3.Reward:

1. +10 for improvement
2. –10 for adverse effect
3. 0 for neutral outcome

4.Goal: Maximize long-term patient outcome

Example:

- **RL for HIV therapy**: Learn optimal drug scheduling
- **Sepsis treatment** (Deep RL used to recommend ICU strategies)



Robotics

Application: Navigation & Surveillance

Problem: Teach a robot to move through complex environments or monitor spaces efficiently

How RL is used:

1.State: Robot's position, orientation, obstacles nearby (from sensors)

2.Actions: Move forward/backward, rotate, scan

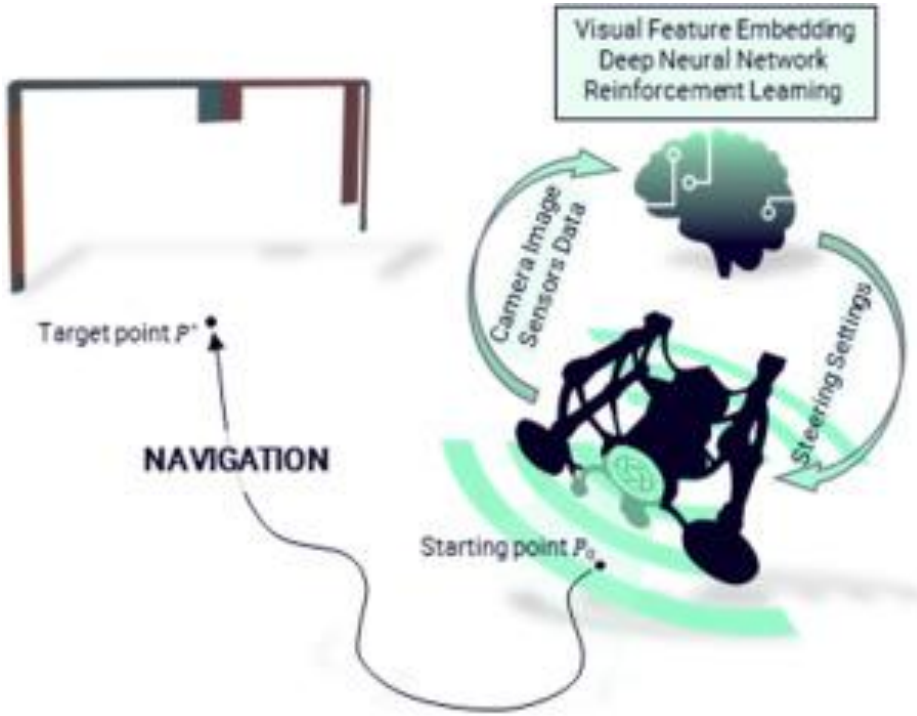
3.Reward:

1. +1 for reaching goal
2. –10 for bumping into walls
3. +0.1 per step towards goal

4.Goal: Learn a navigation strategy (policy) to reach goal with minimal collisions

Example:

- **Boston Dynamics robots** can be trained to adjust walking patterns using RL
- **Surveillance drones** use RL for path planning and persistent monitoring



SmartAd

Application: Personalized Ads

Problem: Show the right ad to the right user at the right time

How RL is used:

1.State: User profile, browsing behavior, time of day

2.Actions: Select one ad from many

3.Reward:

1. +1 if user clicks
2. 0 if ignored

4.Goal: Learn to display ads that maximize click-through rate (CTR)

Example:

- **YouTube** ad recommendation
- **Meta/Facebook** uses RL to optimize ad ranking and selection

Finance

Application: Portfolio Optimization

Problem: Decide which stocks to invest in, and how much

How RL is used:

1.State: Current portfolio weights, market indicators ;Current stock prices – Portfolio weights (how much is invested in each asset) – Technical indicators (like moving average, RSI) – Macroeconomic variables (e.g., interest rates)

2.Actions: Buy/Sell/Hold decision for each asset

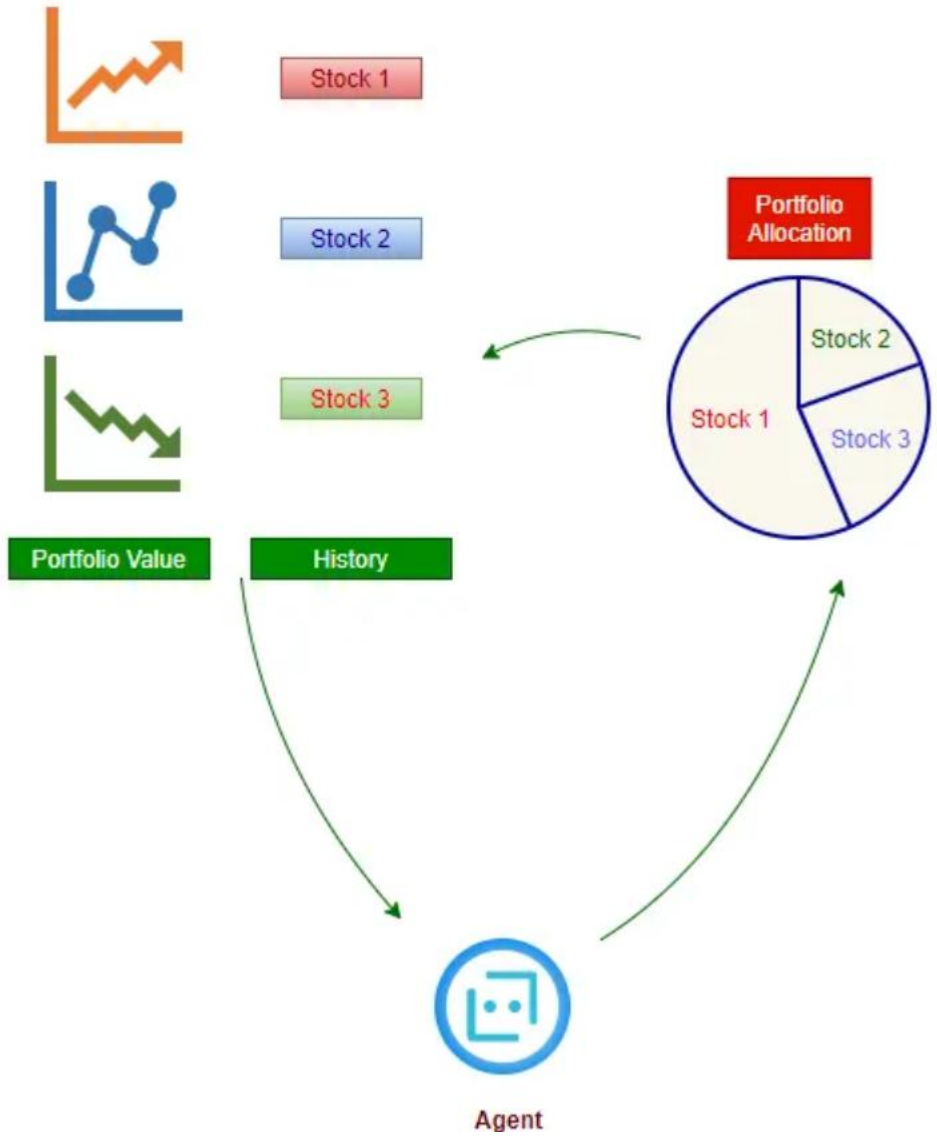
3.Reward:

1. for portfolio growth
1. – for loss or risk exposure

4.Goal: Maximize return over time, minimize risk

Example:

- **Deep RL in trading bots** that learn market timing
- **JP Morgan** and other banks use RL for investment automation



chatbots

Application: Dialogue Management in Voice Assistants

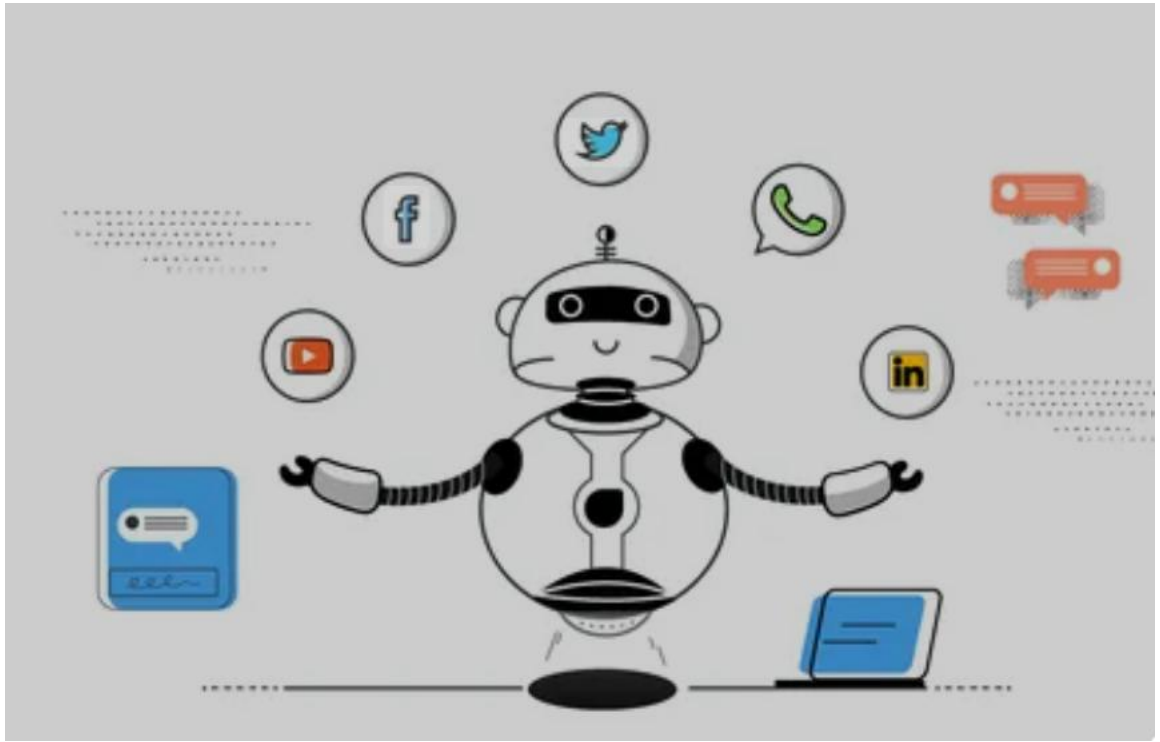
Problem: Engage in helpful, multi-turn conversations
How RL is used:

1.State: Dialogue history, user's query
2.Actions: Respond with an appropriate sentence or intent
3.Reward:

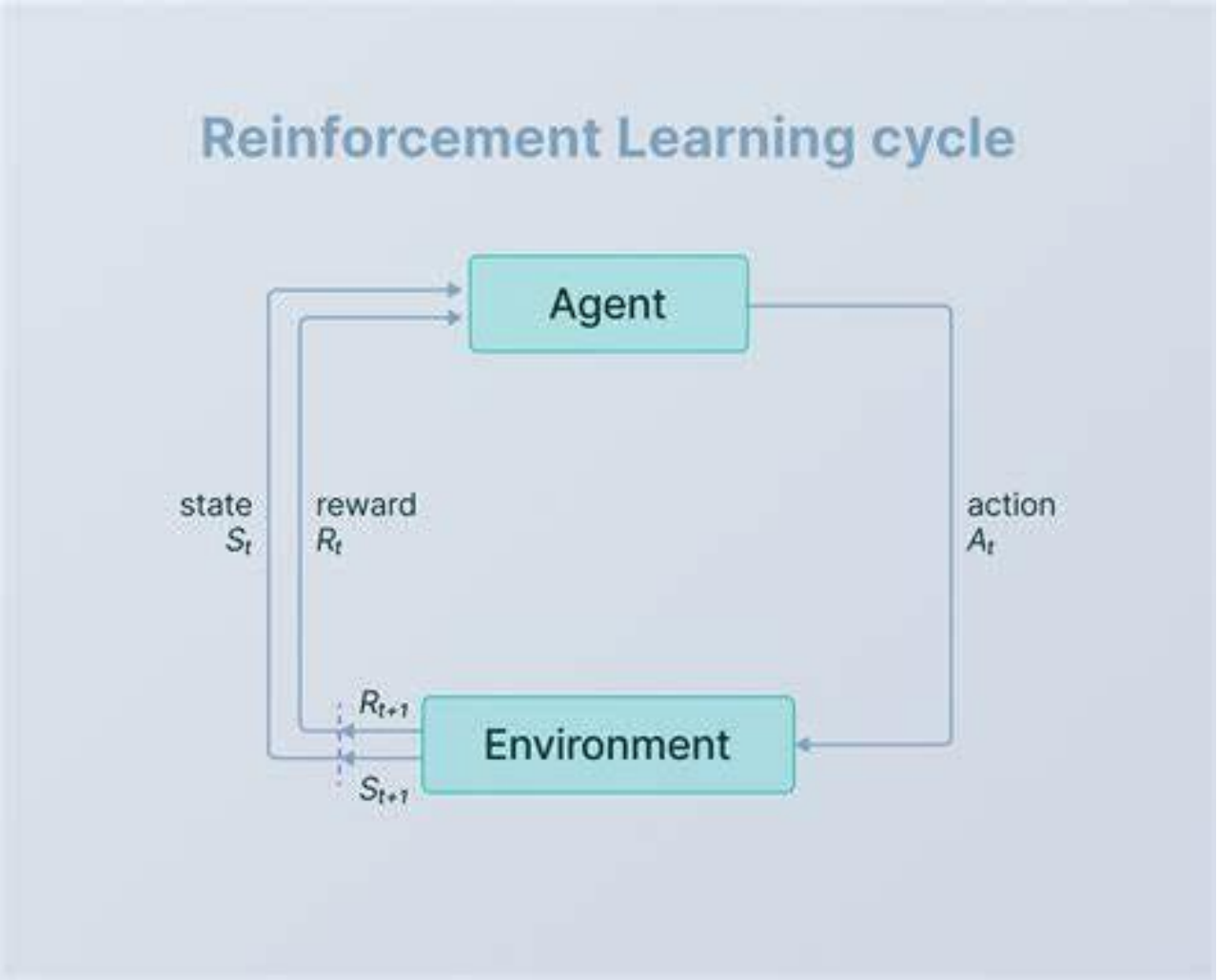
1. +1 if user is satisfied or completes task
2. –1 for unhelpful/off-topic response

4.Goal: Learn dialogue policy that keeps the user engaged and satisfied

Example:
•**Siri**, **Alexa**, **Google Assistant** optimize conversational strategies using RL
•**Conversational agents** like Replika use RL to improve engagement



Elements of Reinforcement Learning



Information state

An information state (a.k.a. Markov state) contains all useful information from the history.

Definition

A state S_t is Markov if and only if

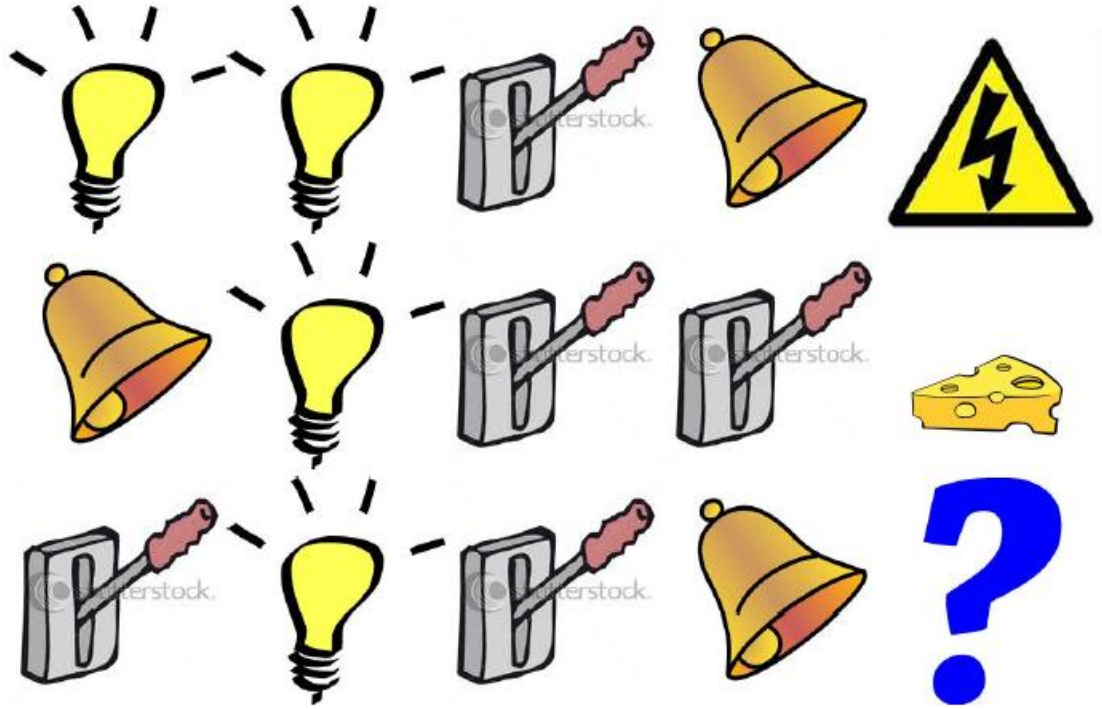
$$\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} \mid S_1, ..., S_t]$$

- "The future is independent of the past given the present"

$$H_{1:t} \rightarrow S_t \rightarrow H_{t+1:\infty}$$

- Once the state is known, the history may be thrown away
- i.e. The state is a sufficient statistic of the future
- The environment state S_t^e is Markov
- The history H_t is Markov

Rat Example- State Representation is critical



- What if agent state = last 3 items in sequence?
- What if agent state = counts for lights, bells and levers?
- What if agent state = complete sequence?

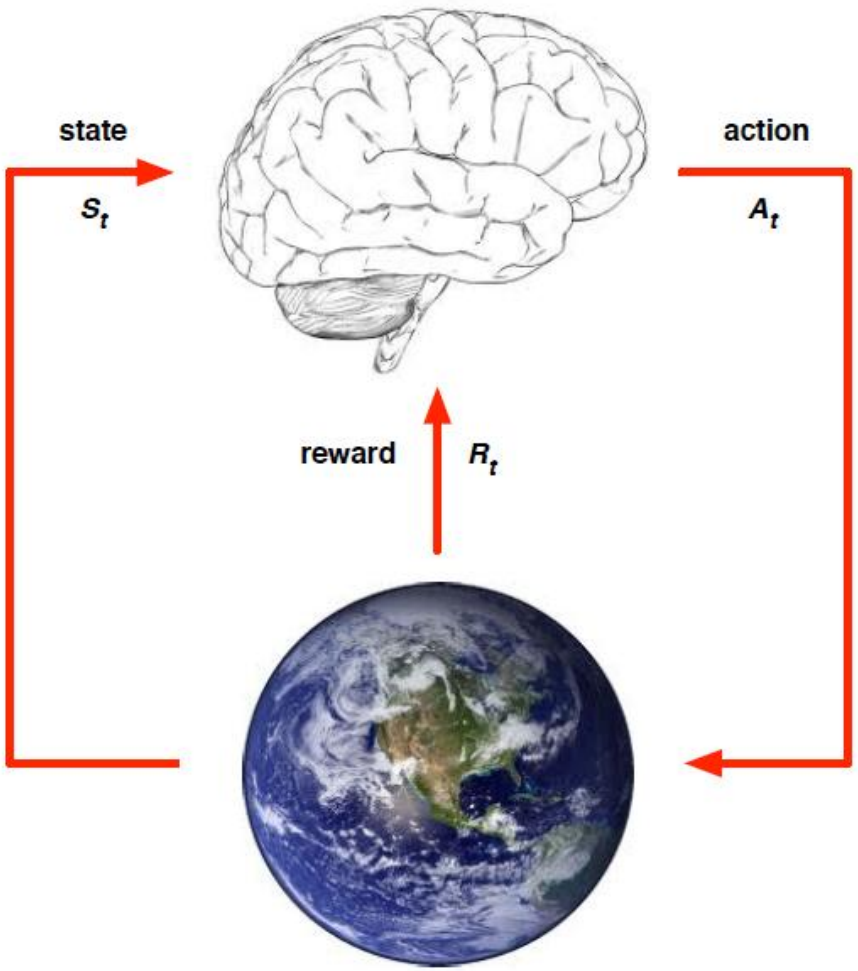
In RL, the Markov property says that the state should contain all the information necessary to predict the future given the action.

Choosing the state representation is critical:

Too small → agent cannot learn optimal behavior (information loss).

Too large → computationally infeasible.

Fully Observable Environment



Full observability: agent directly observes environment state

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

- Agent state = environment state = information state
- Formally, this is a Markov decision process (MDP)
- (Next lecture and the majority of this course)

Partially Observable Environment

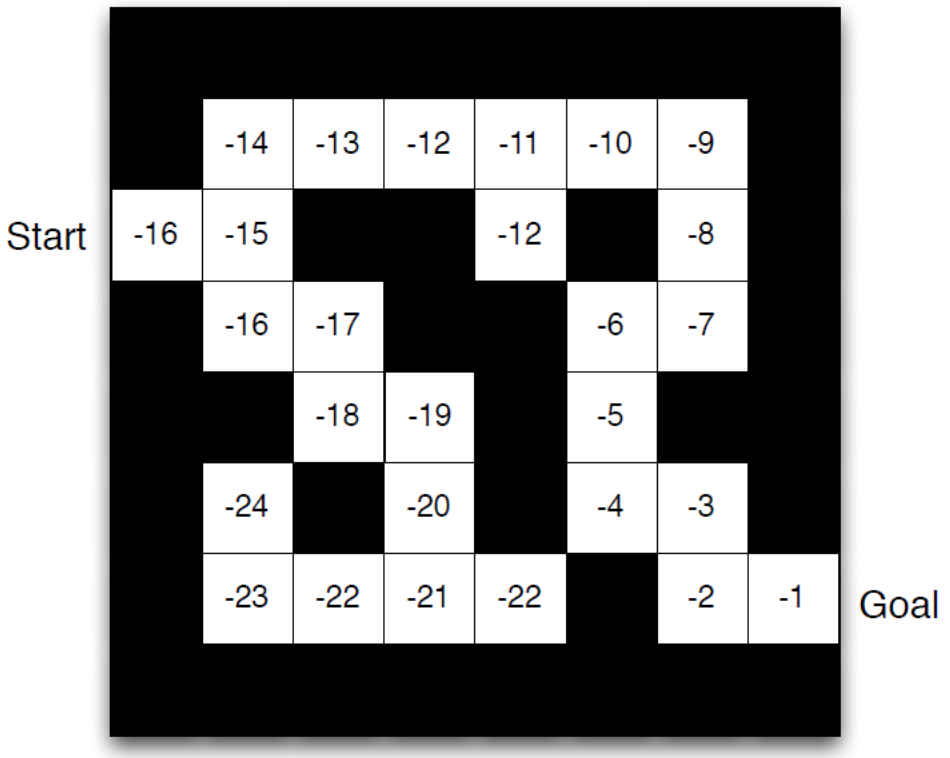
- Partial observability: agent indirectly observes environment:
  - A robot with camera vision isn't told its absolute location
  - A trading agent only observes current prices
  - A poker playing agent only observes public cards
- Now agent state $\neq$ environment state
- Formally this is a partially observable Markov decision process (POMDP)
- Agent must construct its own state representation S_t^a, e.g.
  - Complete history: $S_t^a = H_t$
  - Beliefs of environment state: $S_t^a = (\mathbb{P}[S_t^e = s^1], ..., \mathbb{P}[S_t^e = s^n])$
  - Recurrent neural network: $S_t^a = \sigma(S_{t-1}^a W_s + O_t W_o)$

Major Components of an RL agent

- An RL agent may include one or more of these components:
  - Policy: agent's behaviour function
  - Value function: how good is each state and/or action
  - Model: agent's representation of the environment

Maze example: Value function

$$V^{\pi}(s) = \mathbb{E}[G_t \mid S_t = s]$$



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

- **Meaning**:

  The expected **return** G_t when:

  1. The agent starts in state s at time t.
  2. Follows policy π thereafter.

  $\gamma \in [0, 1]$ is called the **discount factor**.

  R_{t+1} = reward we get **right after** being in state s

- **Return** G_t:

  The total (possibly discounted) sum of future rewards:

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

- The expectation $\mathbb{E}[\cdot]$ is taken over all possible sequences of states, actions, and rewards that could happen following policy π.

In short: $V^{\pi}(s)$ tells you *how good it is to be in state* s under policy π.

Why discount factor

- Mathematical stability
- Preference for sooner rewards
- Uncertainty about the future

2. Action value function

$$Q^{\pi}(s, a) = \mathbb{E}[G_t \mid S_t = s, A_t = a]$$

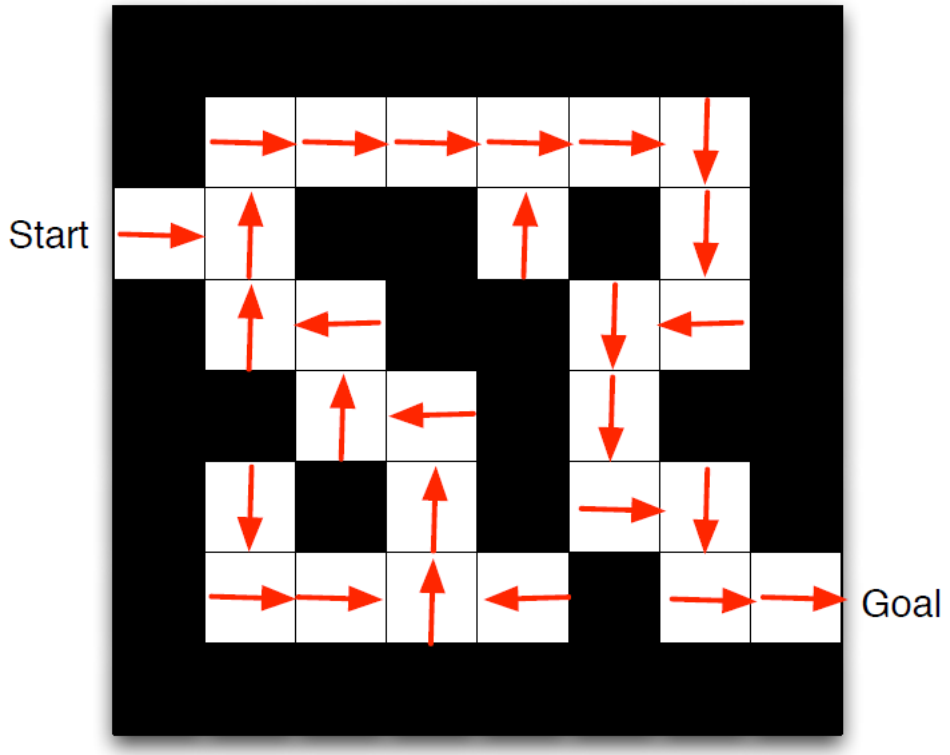
- **Meaning**:

  The expected return G_t when:

  1. The agent starts in state s at time t.
  2. Takes action a **first**.
  3. Then follows policy π thereafter.
- This is more fine-grained than $V^{\pi}(s)$ because it distinguishes between different actions available in the same state.

In short: $Q^{\pi}(s, a)$ tells you *how good it is to take action a in state s* under policy π.

Maze example: Policy



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

- **Definition**: A policy defines the **agent's behavior** at any given time.
- It maps **states** to **actions**:
  - **Deterministic policy**:

$$a = \pi(s)$$

    Always picks the same action for a given state.
  - **Stochastic policy**:

$$\pi(a|s) = \mathbb{P}[A_t = a \mid S_t = s]$$

    Gives a probability distribution over possible actions.

Key point:

The policy is the *decision-making rule* — it's "what the agent does."

Model Based

- A model predicts what the environment will do next
- $\mathcal{P}$ predicts the next state
- $\mathcal{R}$ predicts the next (immediate) reward, e.g.

$$\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$
$$\mathcal{R}^a_s = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$$

What is a "model" in RL?

•In reinforcement learning, a **model** is the agent's internal representation of the **environment's dynamics**.

•If the agent has a model, it can **simulate** what will happen next without actually interacting with the real environment.

•This is the essence of **model-based RL**:

- Learn or have access to a model.
- Use the model to plan actions and improve the policy.

Two components of the model

(a) Transition model $\mathcal{P}$

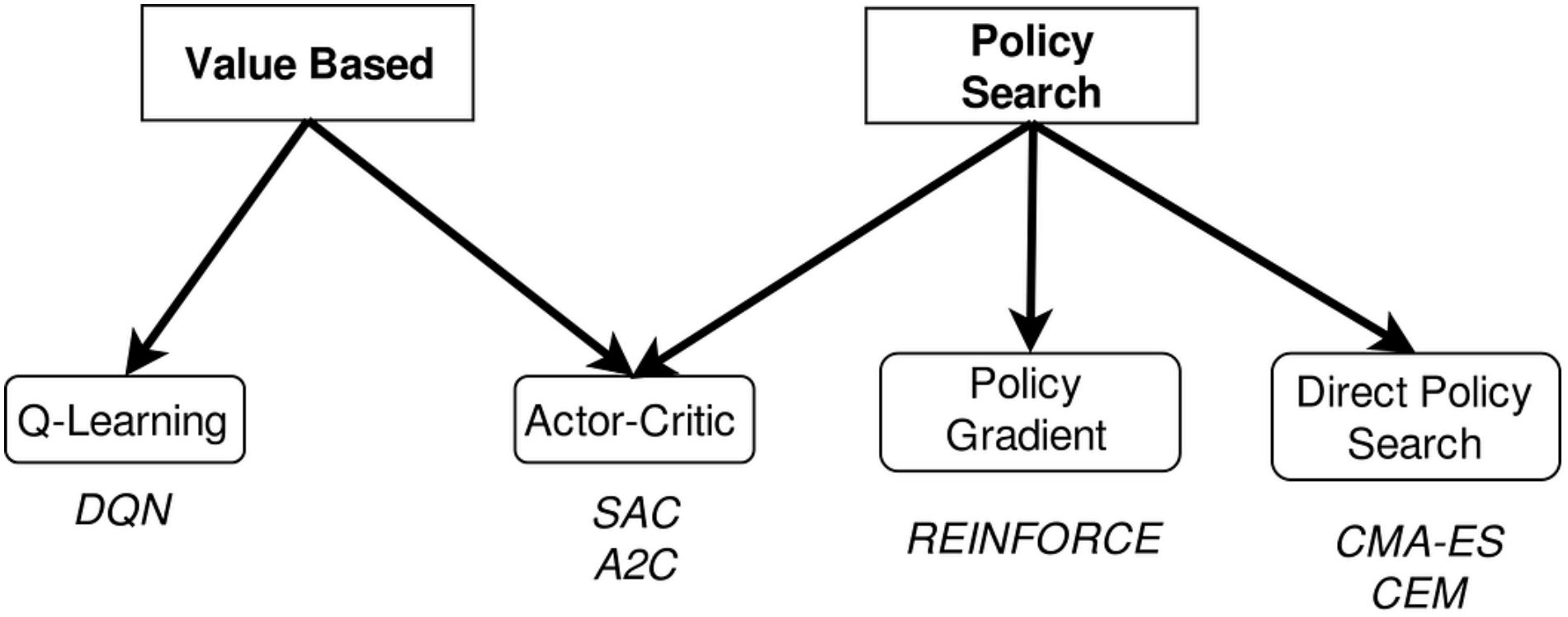
$$\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$

- **Meaning**: The probability that the next state will be s', given:
  - Current state $S_t = s$
  - Current action $A_t = a$
- This captures **how the environment changes** in response to an action.

(b) Reward model $\mathcal{R}$

$$\mathcal{R}^a_s = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$$

- **Meaning**: The expected immediate reward when:
  - Current state $S_t = s$
  - Current action $A_t = a$
- This tells the agent **what reward it will likely get right after** taking the action.


Value Based
Policy Search
Q-Learning
Actor-Critic
Policy Gradient
Direct Policy Search
DQN
SAC
A2C
REINFORCE
CMA-ES
CEM

Categorisation of RL

Value Based

- No Policy (Implicit)
- Value Function

Policy Based

- Policy
- No Value Function

Actor Critic

- Policy
- Value Function

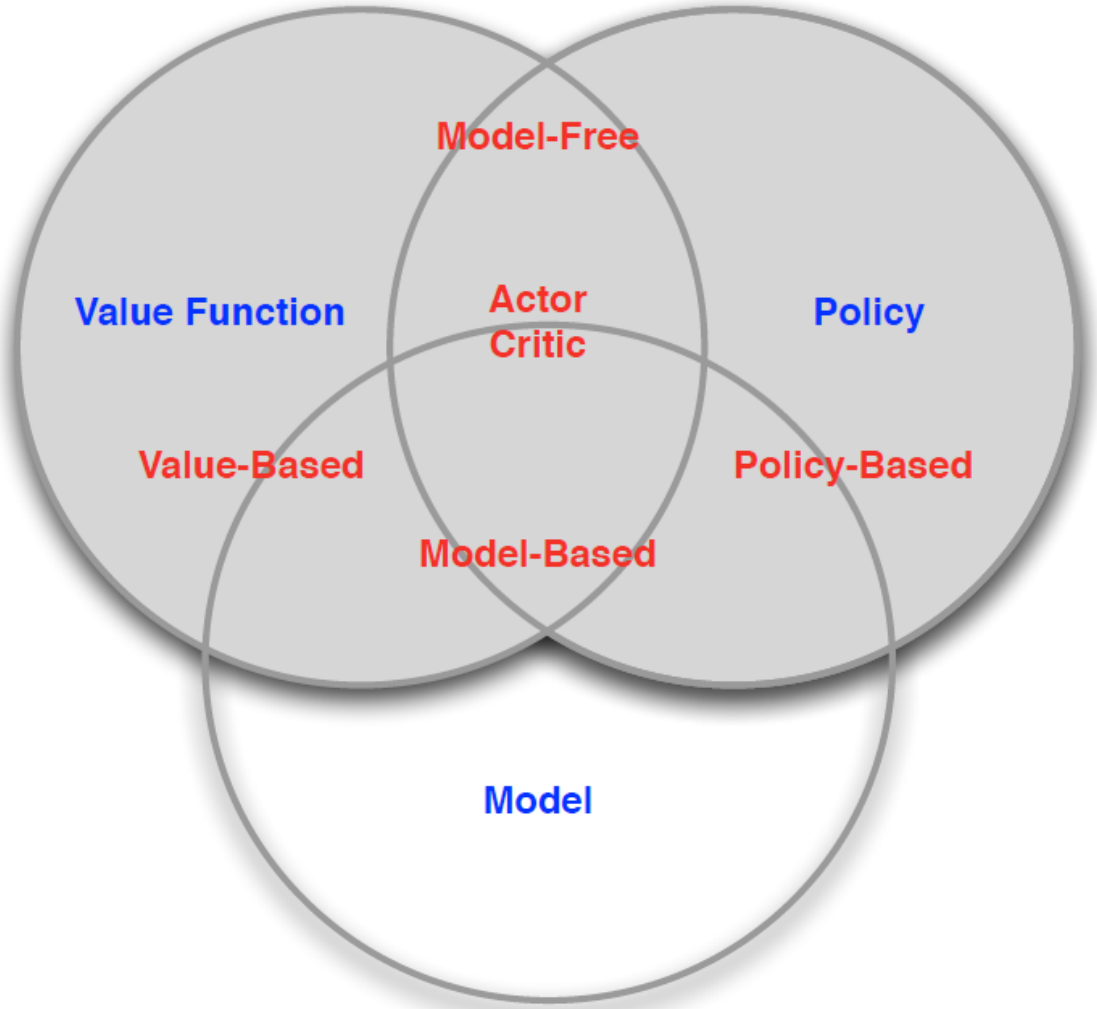
Model Free

- Policy and/or Value Function
- No Model

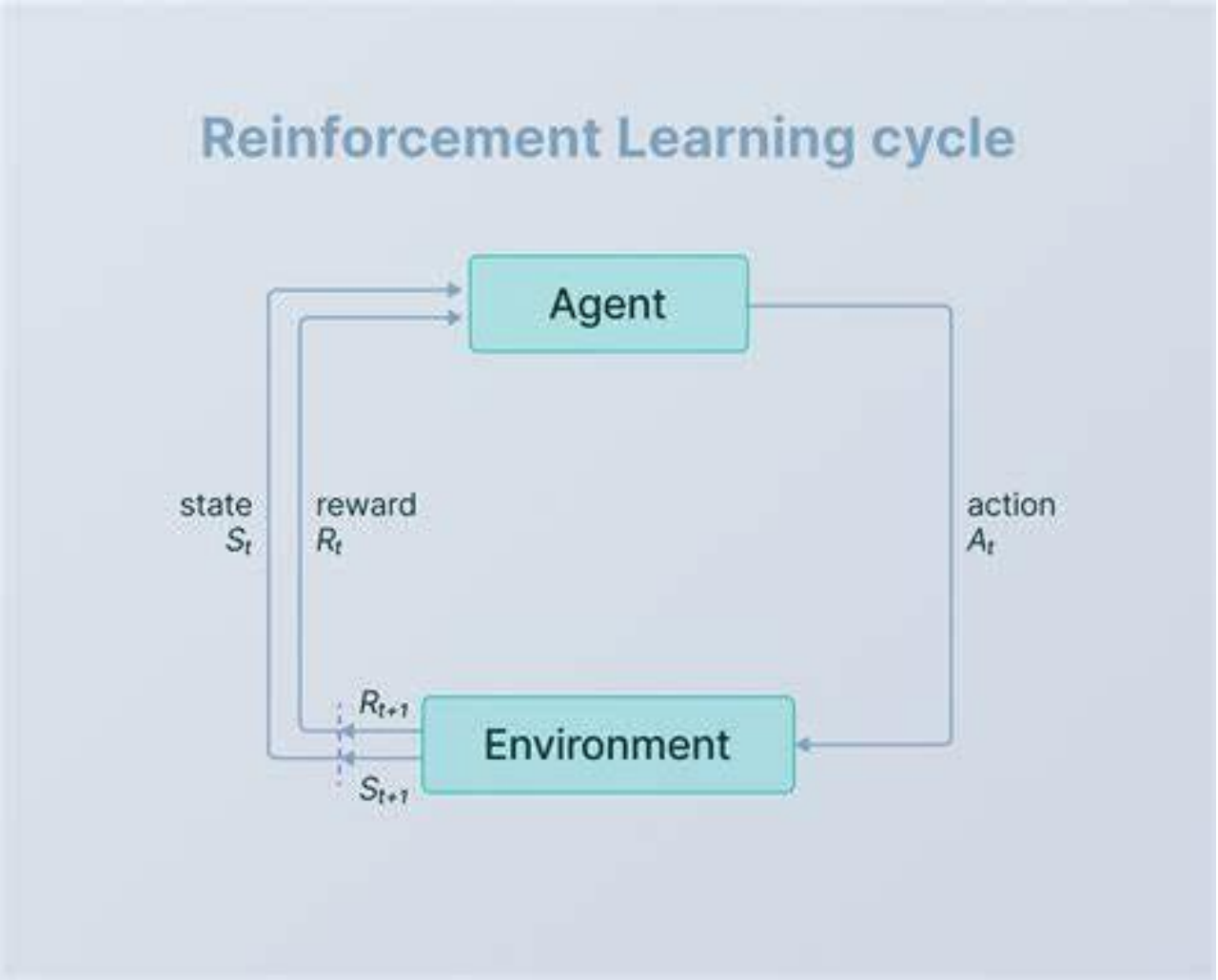
Model Based

- Policy and/or Value Function
- Model

RL Agent Taxonomy



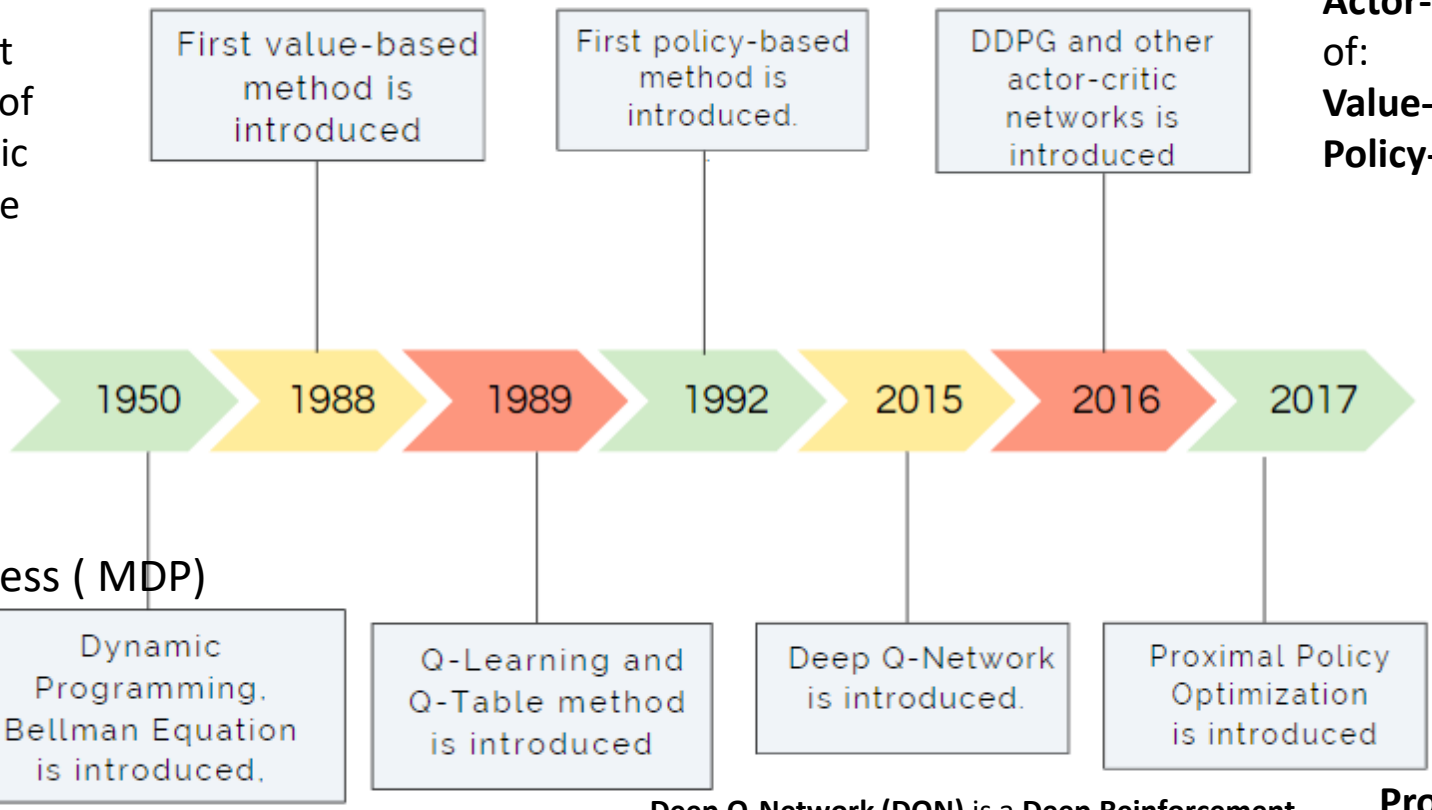
Evolution of Reinforcement Learning



•Policy-based methods directly optimize the policy. -finding the optimal policy that maximizes cumulative rewards over time.

Estimating value functions (such as state-values or action-values)

1988: Temoral difference Learning Method first value-based method that combined the strengths of Monte Carlo and Dynamic Programming,-first online learning



Actor-Critic method is a hybrid of:
Value-based methods and Policy-based methods

Markov Decision process (MDP)

Q-learning learns the **quality (Q-value)** of state–action pairs:how good it is to take a certain action in a certain state.

Deep Q-Network (DQN) is a **Deep Reinforcement Learning algorithm** that combines: **Q-learning** (to learn action values), and
•**Deep Neural Networks** (to approximate the Q-function).

Proximal Policy Optimization (PPO) is a **policy gradient algorithm** in reinforcement learning

Courtesy:Medium.com

Evolution of Reinforcement Learning

Pre-1980 – Foundations

1950s–1960s: Development of **dynamic programming**, Bellman equations, and Markov Decision Processes — the mathematical backbone of RL

1982: Stevo Bozinovski presents delayed reinforcement learning in the Crossbar Adaptive Array, an early precursor to later RL algorithms

Late 1980s – Early Algorithms

•**1989**: Chris Watkins introduces **Q-learning**, a model-free, off-policy algorithm that became foundational in RL **Early 1990s**: Richard Sutton and Andrew Barto advance **temporal-difference learning** and **actor-critic methods**, formalizing RL theory **1992–1993**: Gerald Tesauro's **TD-Gammon**—a neural-net trained via TD learning—demonstrates superhuman backgammon play using self-play

Evolution of Reinforcement Learning

Pre-2010 – Foundations of Deep RL

1996: Kaelbling, Littman & Moore publish a major survey, solidifying practical RL approaches and exploration-exploitation theory

Mid-2010s – Deep RL Breakthrough

2013: Mnih et al. publish “Playing Atari with Deep Reinforcement Learning,” introducing **Deep Q-Network (DQN)** that learns directly from raw pixel input

2015: Distributed DQN scales via experience replay and parallel learners, achieving state-of-the-art results across Atari games

2015: Development of **Double DQN** and improvements in stability and learning efficiency

Evolution of Reinforcement Learning

Late 2010s – AlphaGo Era
2015–2016: DeepMind's **AlphaGo** defeats professional human Go players; policy-network-guided Monte Carlo Tree Search combines deep learning and planning
2017: **AlphaGo Zero** and **AlphaZero** emerge—learning via self-play **without human data** and generalizing across board games

2019 Onwards – Model-based and General RL(Deep Mind)
2019: **MuZero** combines model-based planning and model-free learning, achieving high performance without knowing game rules explicitly
2020s: RL is adopted in real-world domains, from robotics and energy optimization to alignment with large language models (e.g., **RL from Human Feedback**

Evolution of Reinforcement Learning

2025 GRPO (DeepSeek) GRPO introduced critic-free RL for training LLMs in reasoning. It compared output groups to optimize generation without value networks. Opened the path for RL in large-scale language models.

TIC-GRPO, AGPO, Dr. GRPO Variants like TIC-GRPO added unbiased learning with trajectory correction. AGPO improved sample efficiency and reduced reward hacking. Dr. GRPO reduced bias and improved stability.

Turing Award & Industry Boom Sutton & Barto received the 2025 Turing Award for foundational RL work. The RL market grew to $120B+, with use in robotics, supply chains, and language models. RL is now central to AI progress.

Evolution of Reinforcement Learning

2025 – Recognition and Perspective

2025: **Andrew Barto and Richard Sutton** receive the **A.M. Turing Award** for their pioneering contributions in RL theory and practical methods, reinforcing the importance of RL's conceptual roots

RL Timeline 1950–2025 With Contributors

	Year	Milestone	Person(s)	Institution/Company	Explanation
1	1950	Dynamic Programming & MDPs	Richard Bellman	RAND Corporation	Introduced Bellman Equations and MDPs, forming the mathematical foundation of RL.
2	1988	Temporal Difference (TD) Learning	Richard Sutton	University of Massachusetts	Combined Monte Carlo and Dynamic Programming for online learning.
3	1989	Q-Learning & Actor-Critic	Chris Watkins	Royal Holloway, University of London	Introduced Q-learning; enabled off-policy value updates.
4	1992	REINFORCE Algorithm	Ronald J. Williams	Northeastern University	First policy-gradient method using stochastic policies.

5	2015	DQN, TRPO	Volodymyr Mnih et al.	DeepMind	DQN learned from raw pixels; TRPO stabilized policy gradients.
6	2016	DDPG	Tim Lillicrap et al.	DeepMind	Enabled continuous action space control with actor-critic method.
7	2017	PPO	John Schulman et al.	OpenAI	Simplified and stabilized policy gradient methods.
8	2018	SAC	Tuomas Haarnoja	UC Berkeley	Introduced entropy-maximizing off-policy actor-critic method.
9	2019	MuZero	Julian Schrittwieser et al.	DeepMind	Learned planning without explicit environment models.

10	2021	DSAC	Various	Multiple institutions	Used value distributions for better uncertainty handling.
11	2022	Decision Transformer	Chen et al.	Stanford University	Used transformer models for offline RL via sequence modeling.
12	2023	AlphaDev	DeepMind Team	DeepMind	Discovered new efficient algorithms for sorting and hashing.
13	2024	AutoRL, Sim2Real	Various	Meta AI, Google DeepMind	Automated tuning and transfer learning for robotics.

4	2025	GRPO & Variants	DeepSeek Team	DeepSeek	Introduced critic-free RL for training LLMs with reasoning abilities.

Lecture Plan

1. **Introduction to Reinforcement Learning** – Goals, examples, elements (policy, reward, value function, model), Tic-Tac-Toe example, limitations, scope.
2. **Multi-armed Bandits** – n-armed bandit problem, action-value methods, incremental implementation, nonstationary problems, gradient bandits, associative search.
3. **Agent–Environment Interaction** – Agent, environment, state, action, goals, rewards, pole-balancing, Markov property, **Markov decision processes**.
4. **Value Functions & Bellman Equations** – Optimal value functions, action-value functions, Bellman equations.
5. **Dynamic Programming** – Policy evaluation, policy improvement, policy iteration, value iteration, asynchronous dynamic programming.
6. **Monte Carlo Methods** – Monte Carlo prediction, Monte Carlo control.
7. **Temporal-Difference Learning** – SARSA, Q-learning, applications in RL problems.

Course outcomes

Course Outcomes

After completing this course, the students will be able to

CO1: Define the key features of reinforcement learning that distinguishes it from AI and non-interactive machine learning

CO2: Decide if an application problem should be formulated as a RL problem; if yes be able to define it formally (in terms of the state space, action space, dynamics and reward model), state what algorithm (from class) is best suited for addressing it

CO3: Implement in code common RL algorithms

CO4: Describe (list and define) multiple criteria for analysing RL algorithms and evaluate algorithms on these metrics: e.g., regret, sample complexity, computational complexity, empirical performance, convergence, etc.

Evaluation Pattern

Assessment	Internal/External	Weightage (%)
Mid Term	Internal	20%
2 Online Quiz (20 minutes)	Internal	10%
Lab		
1. 5 Lab Assignment Submissions	Internal	5%
2. Lab Evaluation and viva (1 viva conducted during lab hours)	Internal	10%
3. Tutorial Hour	Internal	5%
4. Project -	Internal	20%
End Semester	External	30%

Textbook(s)

Reinforcement Learning', Richard.S.Sutton and Andrew G.Barto, Second edition, MIT Press, 2018

Reference(s)

Powell, Warren B.. Reinforcement Learning and Stochastic Optimization: A Unified Framework for Sequential Decisions. United States, Wiley, 2022.

Namah Shivaya