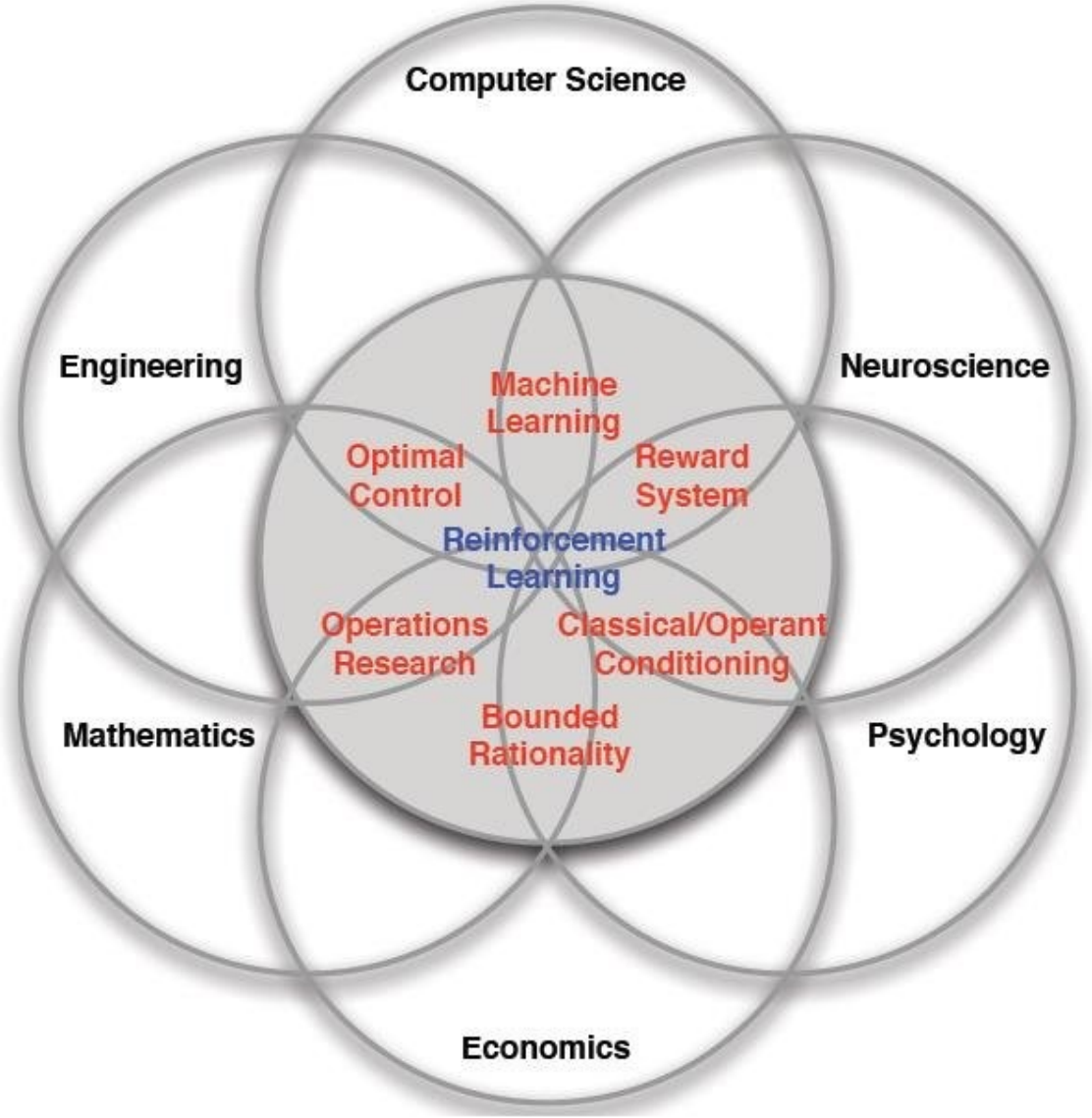
Reinforcement Learning Fundamentals

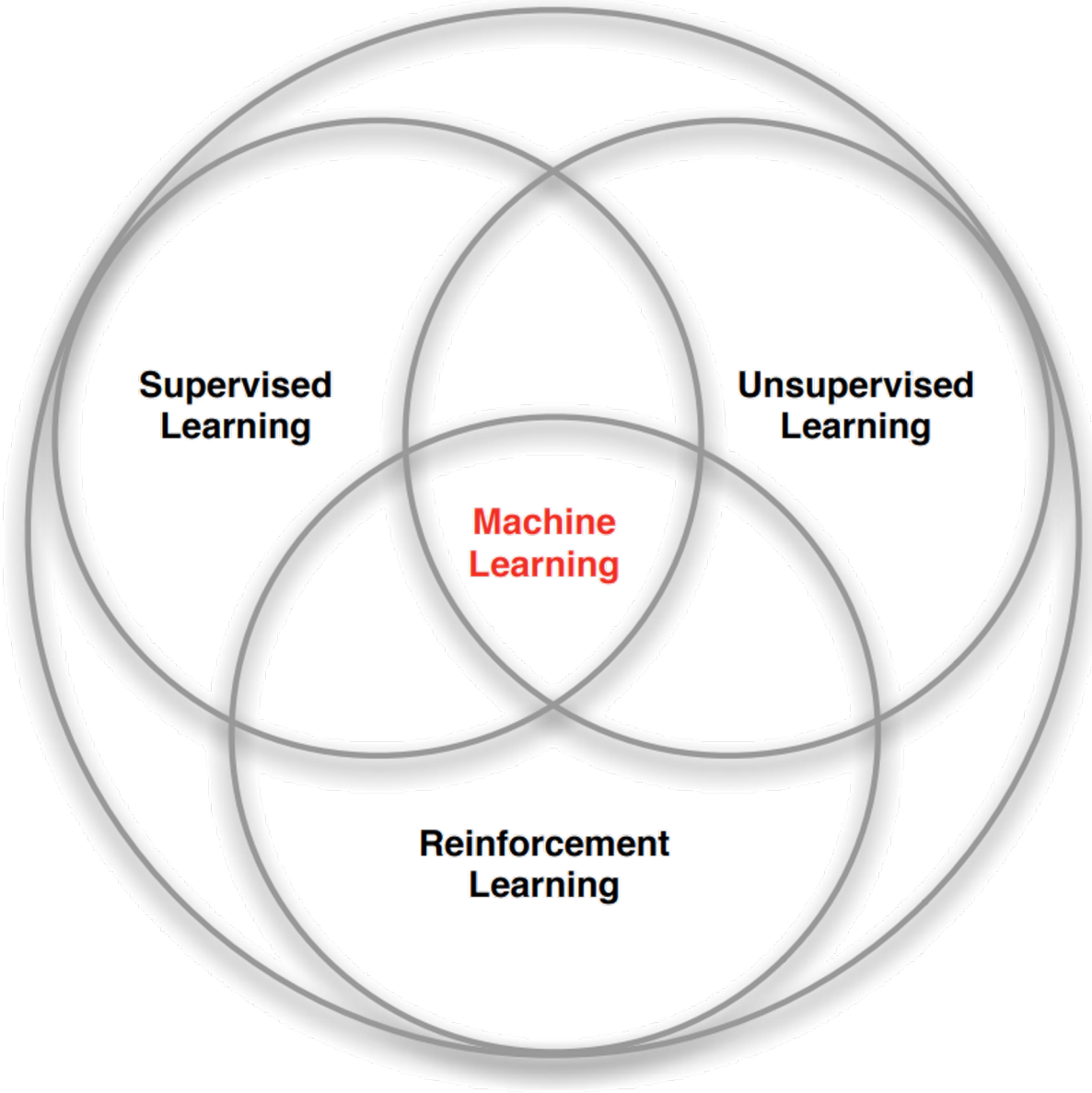
Lecture 1: Introduction

Dr Sandeep Manjanna
Assistant Professor, Plaksha University
sandeep.manjanna@plaksha.edu.in

Faces of Reinforcement Learning



Branches of Machine Learning



What is RL?

- By following someone's instructions?

  Comfort in Water: Get comfortable by splashing water on your face and gradually submerging.
  Floating on Back: Practice floating on your back. Keep ears, shoulders, and hips in the water. Use a gentle flutter kick.
  Basic Swimming Movements: Use a kickboard or hold onto the edge. Practice basic arm movements and kicking.
  Treading Water: Practice treading water. Move arms and legs in a circular motion. Keep your head above water.

  Or get exact control instructions: Move your right arm to 90^0 and simultaneously move the left arm to 270^0 and bend it at 30^0.

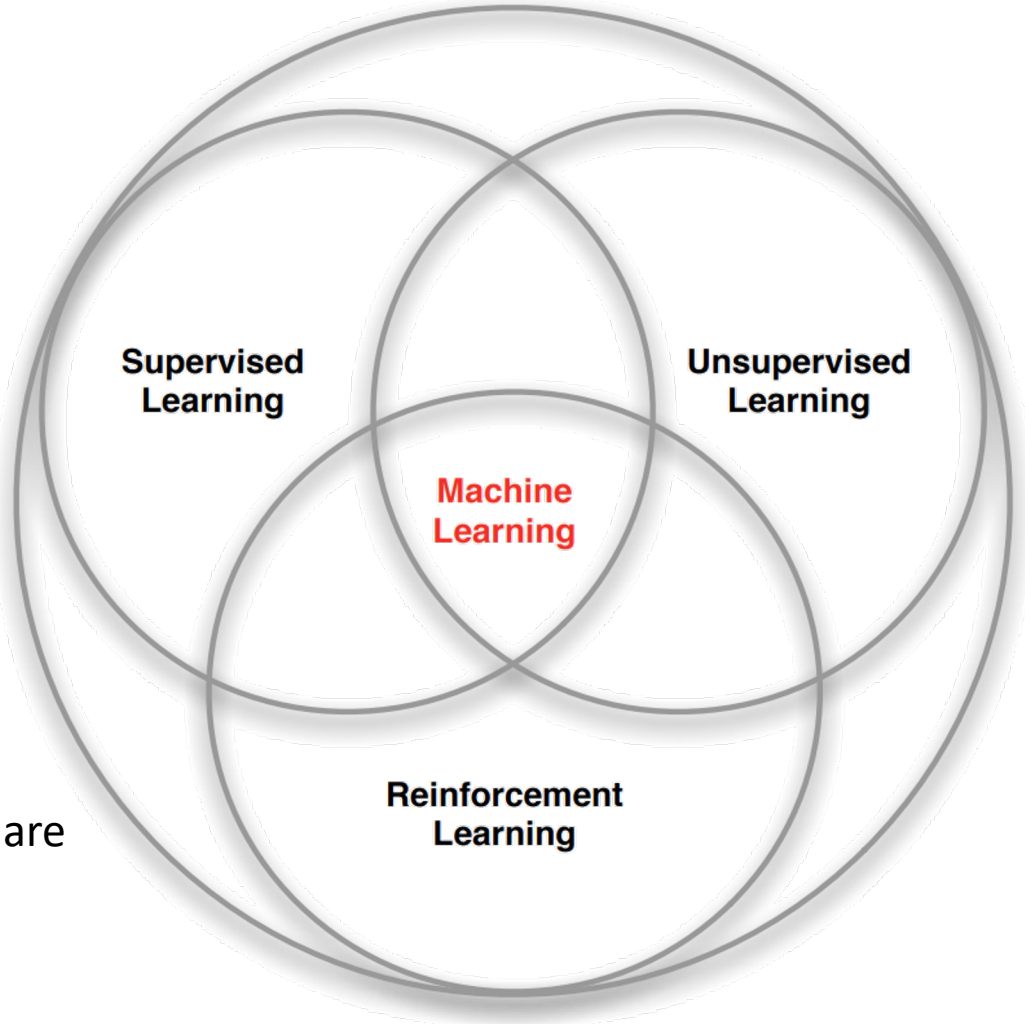
- By watching 100's of people swimming or videos of swimming?

- You need to get into the water and start swimming.
  - Positive feedback
  - Negative feedback

How is RL different?

What makes reinforcement learning different from other machine learning paradigms?

- There is no supervisor, only a reward signal
- Feedback can be delayed, not instantaneous
- Time really matters (sequential, non-i.i.d data)
- Agent's actions affect the subsequent data it receives



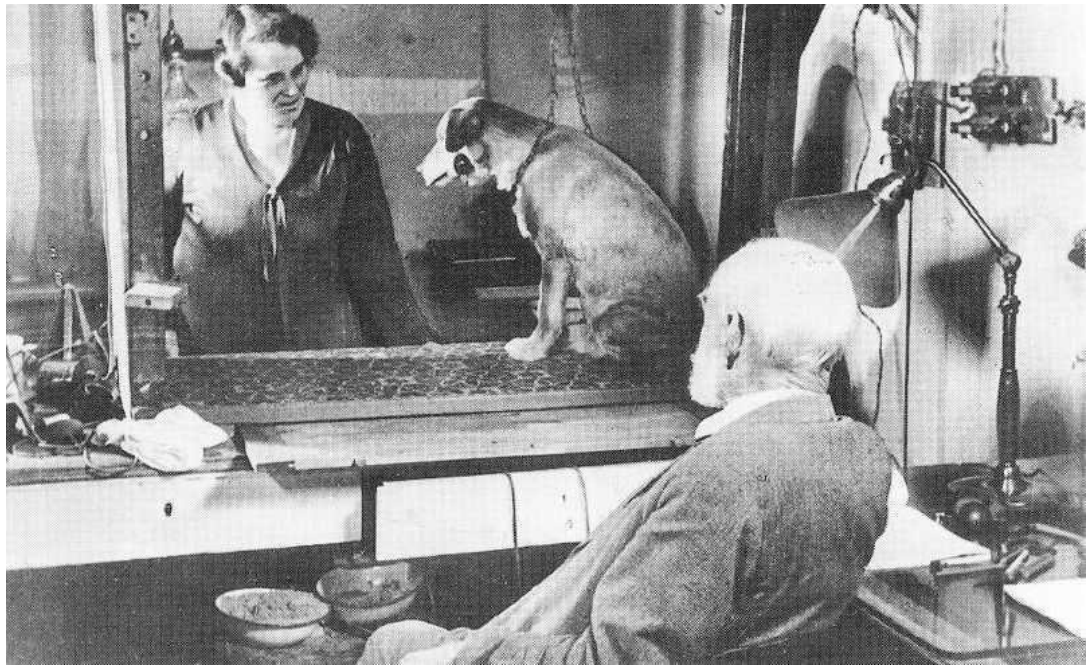
In machine learning, it is quite common to assume that the data are i.i.d,

- i.i.d is the acronym of "independent and identically distributed".
- meaning that the generative process does not have any memory of past samples to generate new samples.

Brief History

Ivan Pavlov's behavioral conditioning experiments:

Pavlov observed that dogs naturally salivated when presented with food, an unconditioned stimulus. However, through repeated pairings of a neutral stimulus, such as a bell, with the food, the dogs eventually began to associate the bell with the arrival of food.



Source: Wikimedia Commons

Sutton and Barto: co-founders of Modern field of Reinforcement Learning

What is RL?

- Learning about stimuli and actions based on rewards and punishments alone.
- No detailed supervision available
- Trial-and-error learning
- Delayed rewards
- Sequence of actions required to obtain reward
- Associative learning required
  - Need to associate actions to states
- Learn about policies not just actions
- Typically in a stochastic world

Reinforcement Learning Fundamentals

Lecture 2: RL Framework

Dr Sandeep Manjanna
Assistant Professor, Plaksha University
sandeep.manjanna@plaksha.edu.in

Some material in this lecture is taken from

1. Prof. Ravindran's course: "Reinforcement Learning."
2. Dr Silver's course: "Reinforcement Learning."

What is RL?

- Learning about stimuli and actions based on rewards and punishments alone.
- No detailed supervision available
- Trial-and-error learning
- Delayed rewards
- Sequence of actions required to obtain reward
- Associative learning required
  - Need to associate actions to states
- Learn about policies not just actions
- Typically in a stochastic world

Applications of RL

- Stanford's autonomous helicopter flight: extreme aerobatics under computer control (2008)

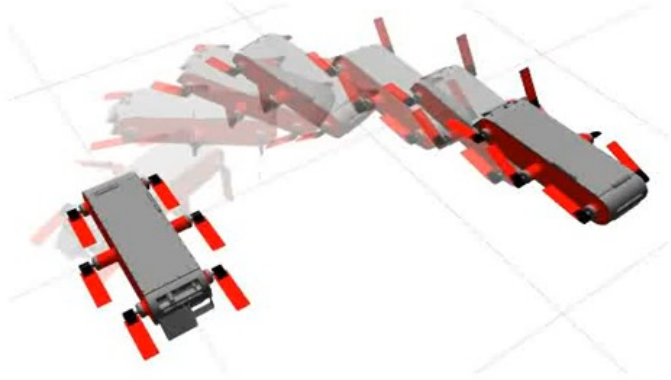
Source: http://heli.stanford.edu/

Applications of RL

- McGill's Adaptive Gait Control for Swimming Robots (2015)

Learning Legged Swimming Gaits from Experience

ICRA 2015 - Best Paper Award Nominee
http://www.cim.mcgill.ca/~dmeger/ICRA2015_GaitLearning/



David Meger, Juan Camilo Gamboa Higuera, Anqi Xu, Philippe Giguere and Gregory Dudek

McGill University and Universite Laval

Source: https://www.youtube.com/watch?v=VHmEdRXSi9M

Applications of RL

- Gerald Tesauro at IBM's Thomas J. Watson Research Center on TD Gammon (Backgammon game) (1992)

Applications of RL

- Ref: Playing Atari with Deep Reinforcement Learning (2013)

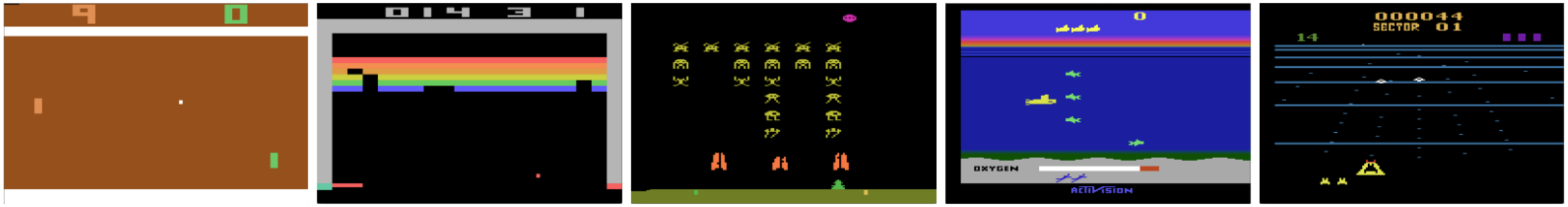
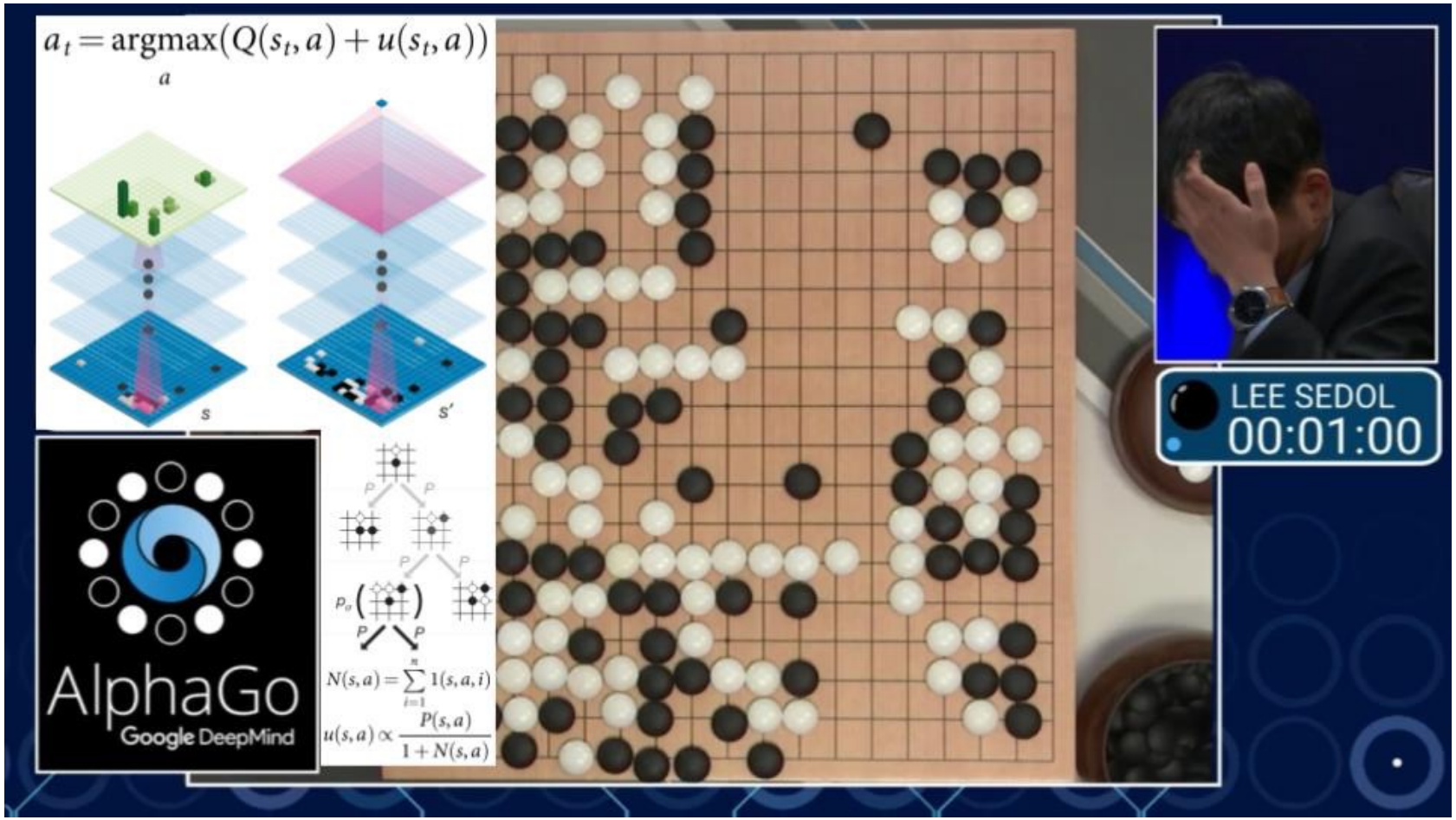


Figure 1: Screen shots from five Atari 2600 Games: (*Left-to-right*) Pong, Breakout, Space Invaders, Seaquest, Beam Rider

... first deep learning model to successfully learn control policies directly from **high-dimensional sensory input** using reinforcement learning.

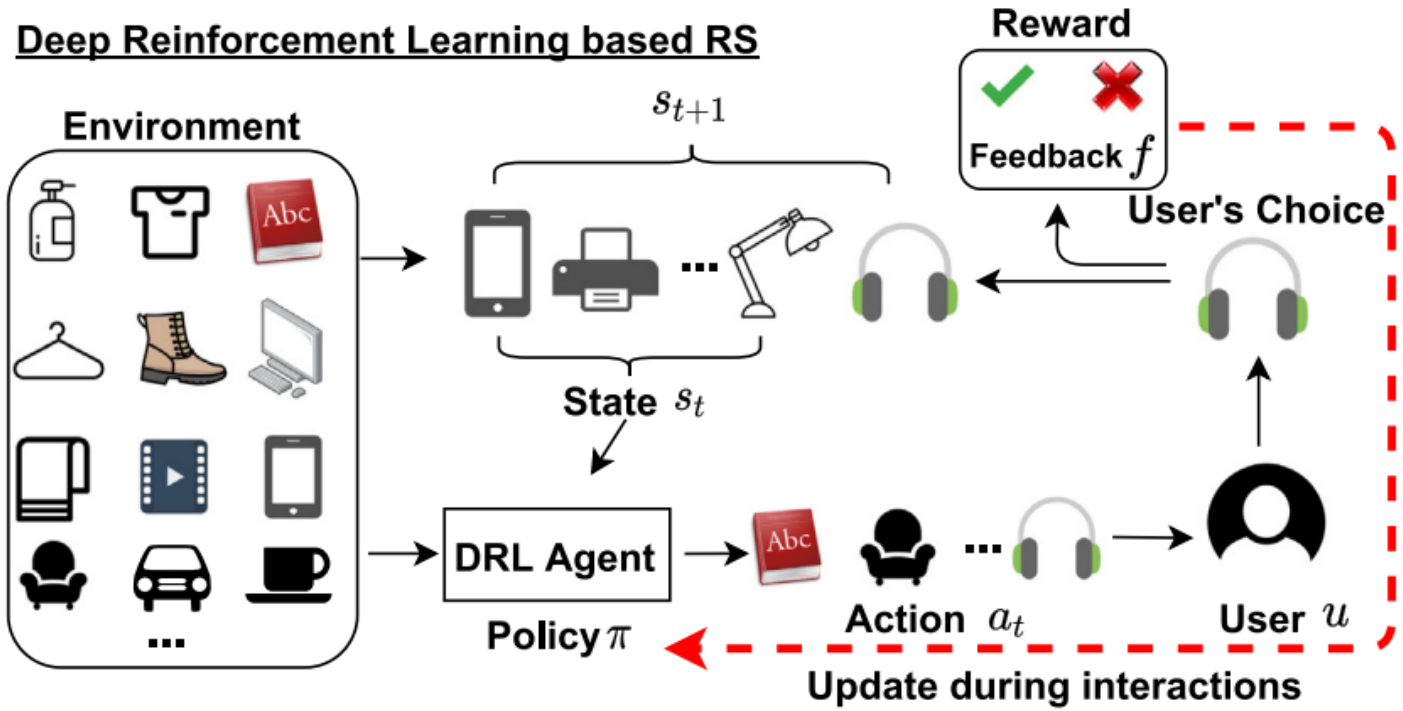
Applications of RL

- David Silver from DeepMind worked on TD algorithm for Go game (2015)



Applications of RL

- Manage investment portfolio
- RL for online learning (suggesting what to show on the stories / Ad)

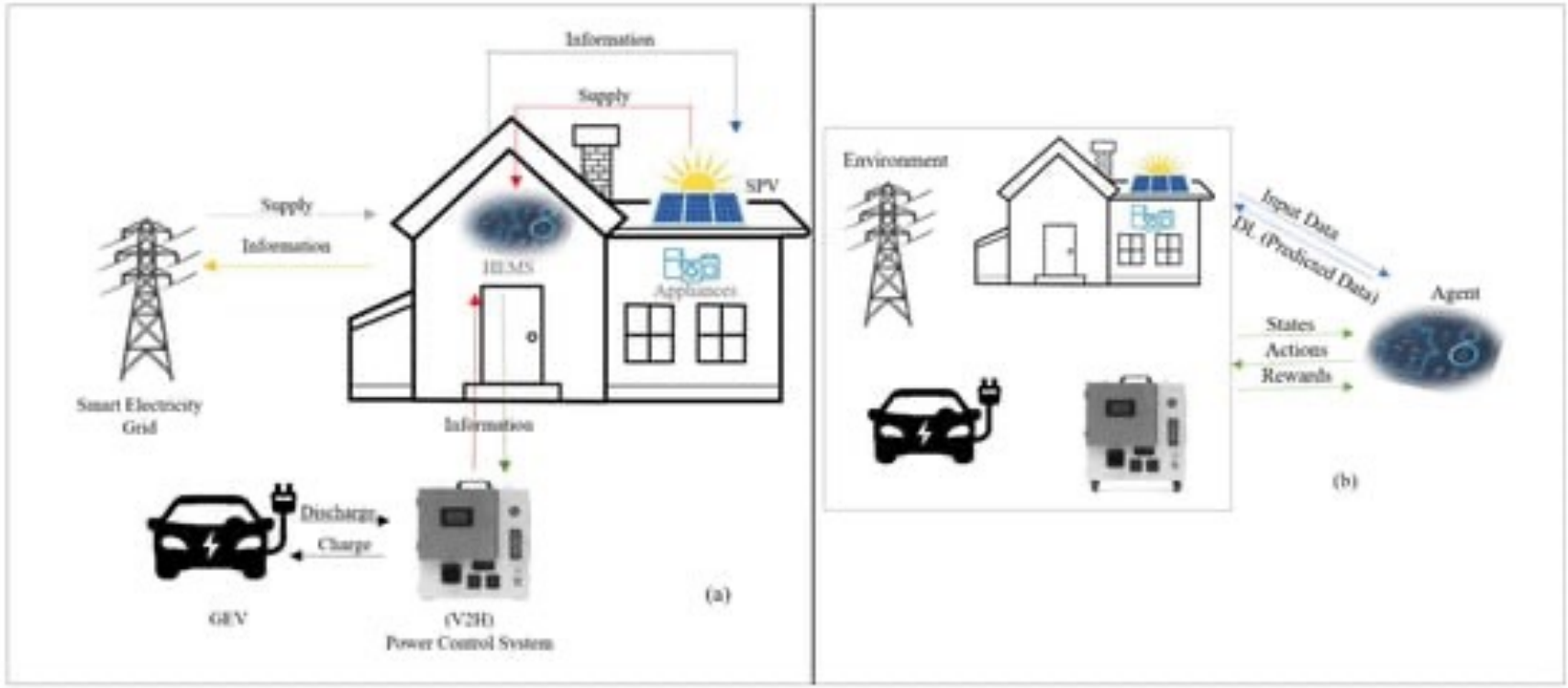


(b) Deep reinforcement learning-based recommender systems

Source: *Deep reinforcement learning in recommender systems: A survey and new perspectives* by Xiaocong Chen, et al.

Applications of RL

- Control a power station



Source: *A Reinforcement Learning Approach for Integrating an Intelligent Home Energy Management System with a Vehicle-to-Home Unit* by Ohoud Almughram, et al.

What are Rewards?

- A reward R_t is a scalar feedback signal
- Indicates how well agent is doing at step t
- The agent's goal is to maximize cumulative reward / total future reward

Reward hypothesis in RL:

All goals can be described by the maximization of expected cumulative reward.

- What if there are no intermediate reward?
- What if the task is time sensitive?
- Can the rewards be delayed?
- Is it better to sacrifice immediate reward to gain more long-term reward?

A financial investment may take months to mature.

Refueling a helicopter might prevent a crash in several hours.

What are Rewards?

- Fly stunt maneuvers in a helicopter
  - *+ve feedback for following desired trajectory*
  - *−ve feedback for crashing*
- Defeat the world champion at Backgammon
  - *+/−ve feedback for winning/losing a game*
- Manage an investment portfolio
  - *+ve feedback for each ₹ in bank*
- Control a power station
  - *+ve feedback for producing power*
  - *−ve feedback for exceeding safety thresholds*
- Make a humanoid robot walk
  - *+ve feedback for forward motion*
  - *−ve feedback for falling over*

Reinforcement Learning Fundamentals

Lecture 3: RL Framework

Dr Sandeep Manjanna
Assistant Professor, Plaksha University
sandeep.manjanna@plaksha.edu.in

Some material in this lecture is taken from

1. Prof. Ravindran's course: "Reinforcement Learning."
2. Dr Silver's course: "Reinforcement Learning."

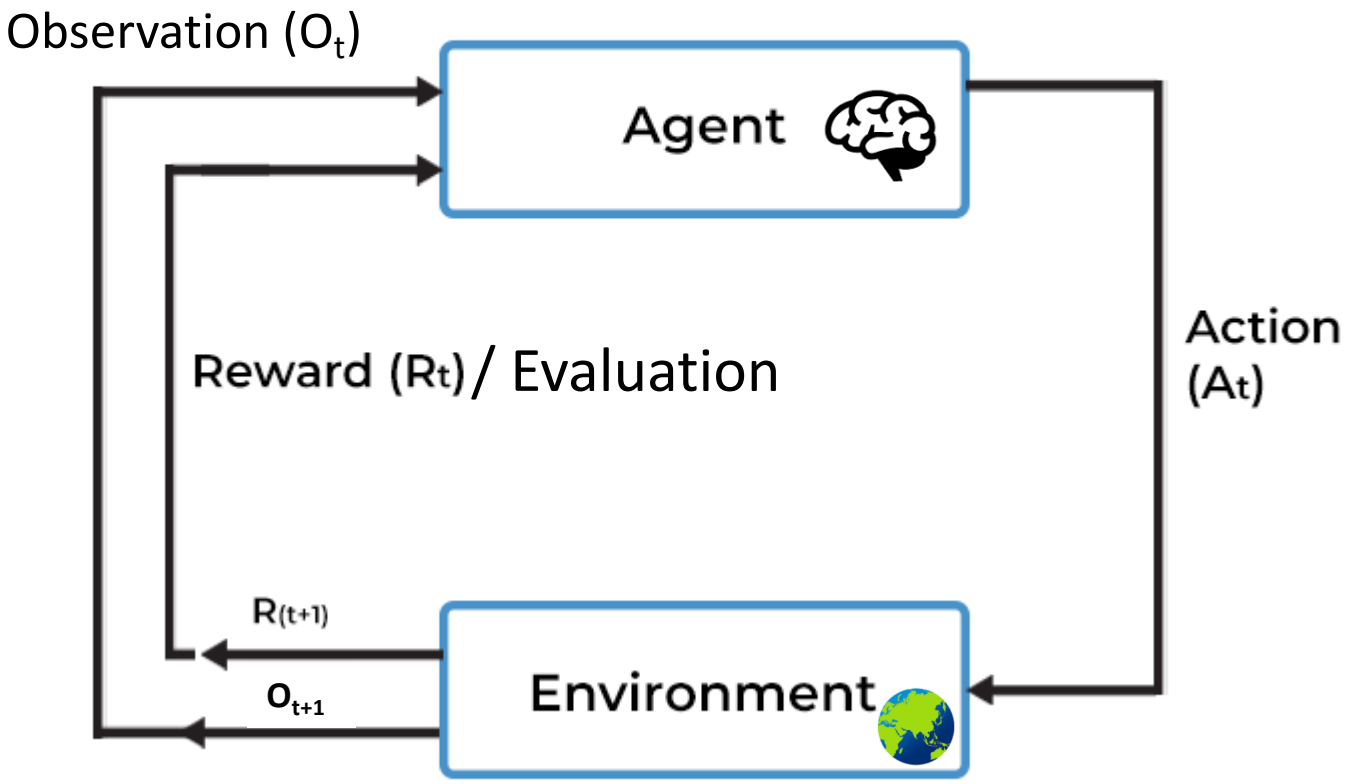
In today's class…

- RL Framework
- What are Rewards?
- What is a State?
- Special cases: Fully and Partially Observable environments
- Temporal Difference

RL Framework

Sequential Decision Making

- Goal: **Select actions or a sequence of actions to maximize the total future reward.**



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}

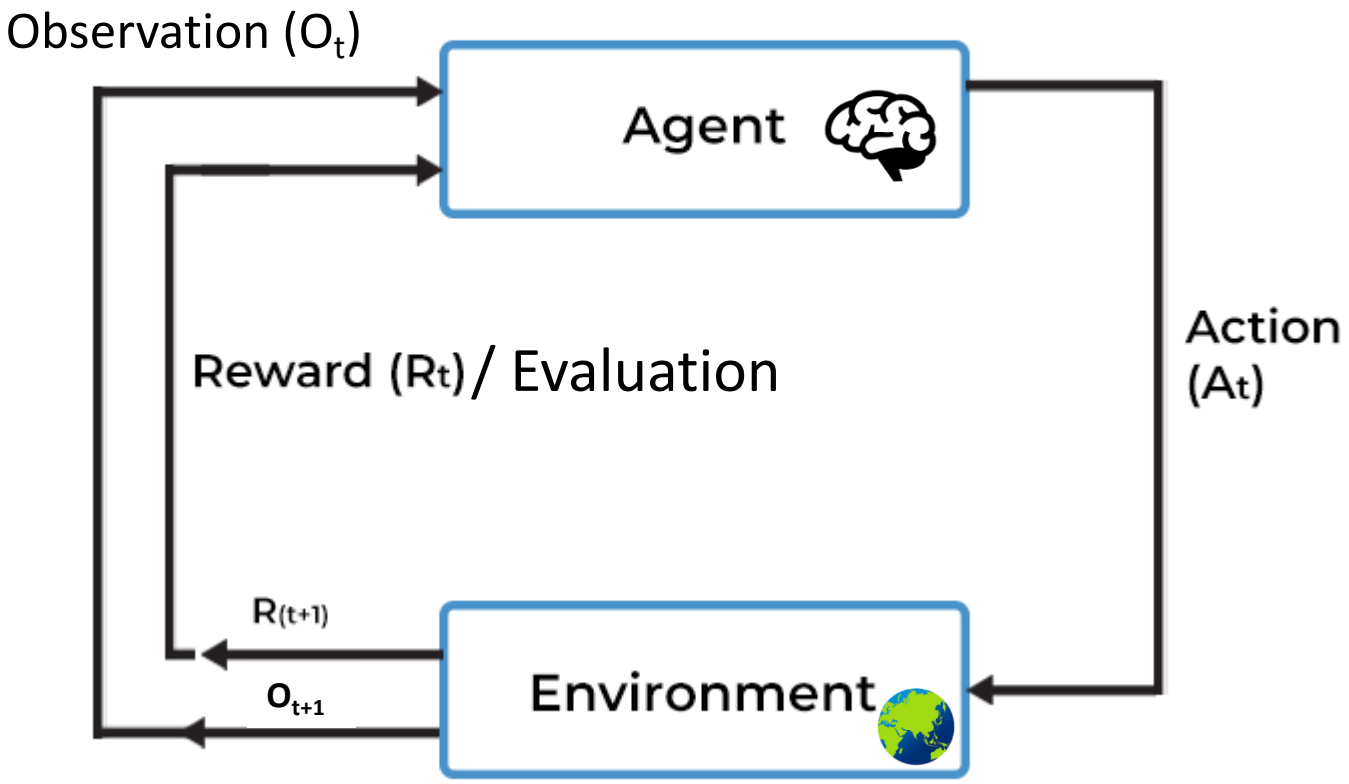
Really?

t increments at env. step

What is evaluation in supervised learning? How is it different in RL?

RL Framework

Sequential Decision Making



- Agent learns by interacting with the environment.
- Environment returns some rewards.

Really?

- Environment gives noisy delayed scalar evaluation.
- Environment also provides some observations.
- Environment can be stochastic.
- Goal is to maximize a measure of long-term performance. In this case it can be the reward.

What is State?

At time ***t=1***, the environment / the world returned some observation O_1 and reward R_1 and the agent took an action A_1. Then the world returned O_2 and R_2. Now the agent took the action A_2.

What is history at ***time = t*** ?

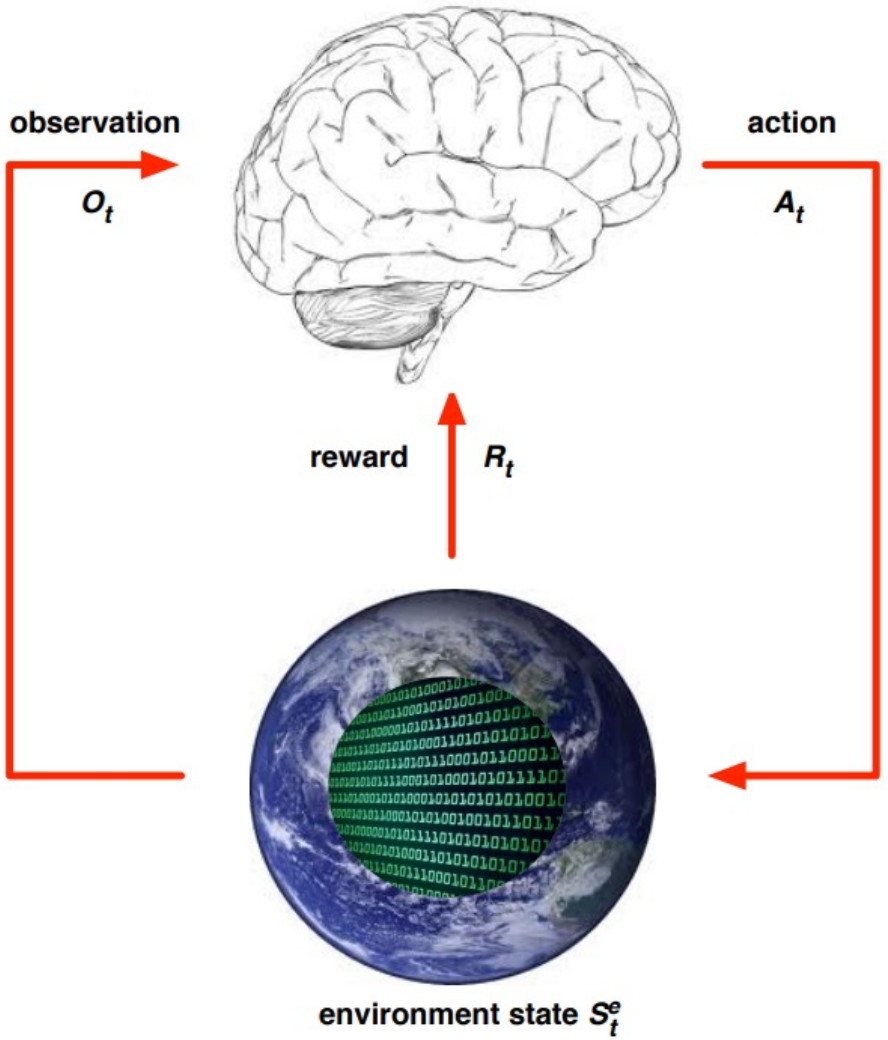
- The sequence of observations, actions, rewards from time = ***1*** to time = ***t*** is referred to as history.
- i.e., History is all the observable variables up to time ***t***.
- What happens next depends on the history.

State is the information used to determine what happens next. This is the **summary of the history**.

Formally, state if a function of history:

$$S_t = f(H_t)$$

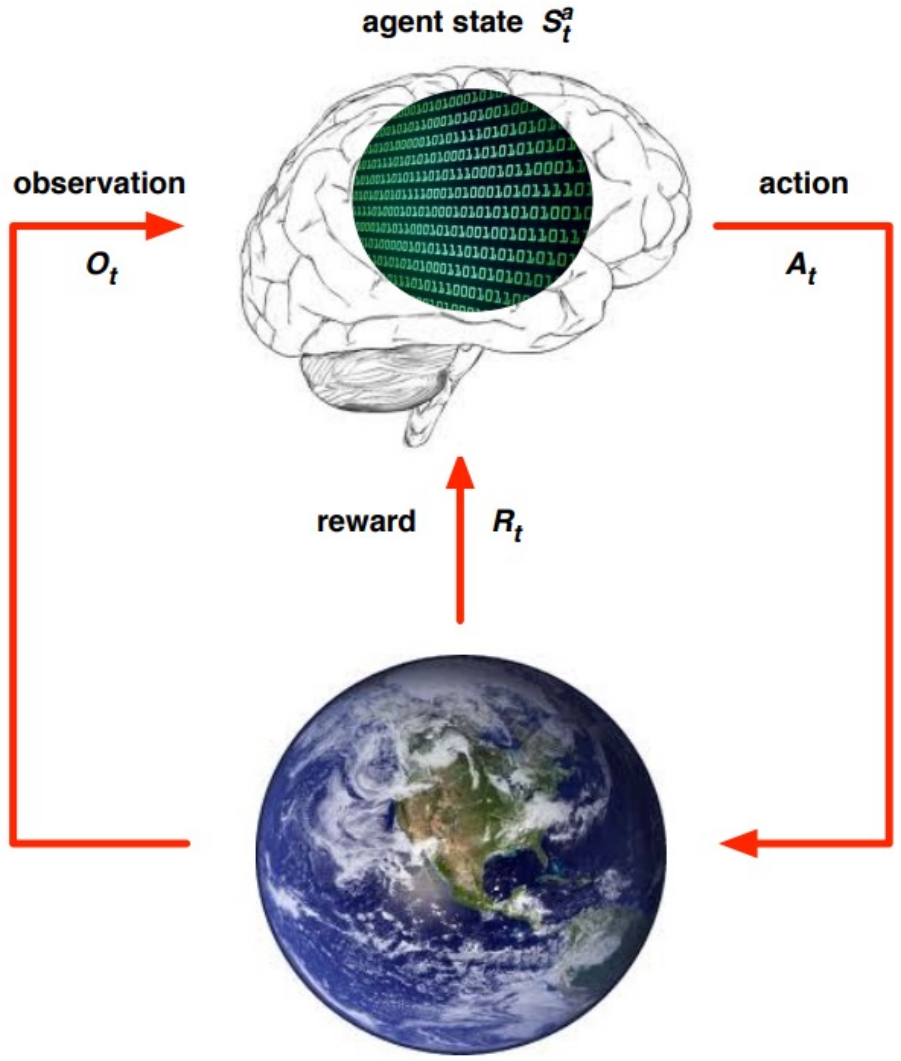
State: Environment State



- The environment state S_t^e is the environment's private representation
- i.e. whatever data the environment uses to pick the next observation/reward
- The environment state is not usually visible to the agent
- Even if S_t^e is visible, it may contain irrelevant information

Slide from David Silver's course: "Reinforcement Learning."

State: Agent State



- The agent state S_t^a is the agent's internal representation
- i.e. whatever information the agent uses to pick the next action
- i.e. it is the information used by reinforcement learning algorithms
- It can be any function of history:

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

Slide from David Silver's course: "Reinforcement Learning."

State: Information State

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

What is Markov Property?

How does it map to Helicopter Example?

Slide from David Silver's course: "Reinforcement Learning."

State: 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"

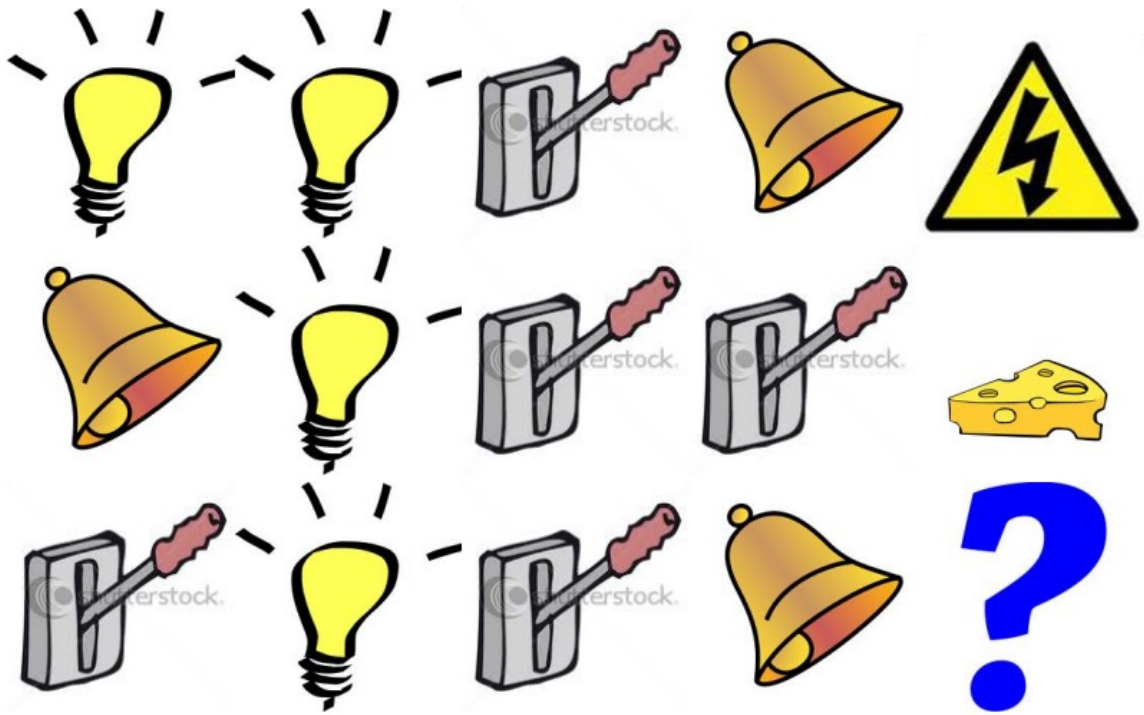
$$H_{1:t} \rightarrow S_t \rightarrow H_{t+1:\infty}$$

- 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

How does it map to Helicopter Example?

Slide from David Silver's course: "Reinforcement Learning."

How to pick a state?



- 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?

Slide from David Silver's course: "Reinforcement Learning."

Reinforcement Learning Fundamentals

Lecture 3: RL Framework

Dr Sandeep Manjanna
Assistant Professor, Plaksha University
sandeep.manjanna@plaksha.edu.in

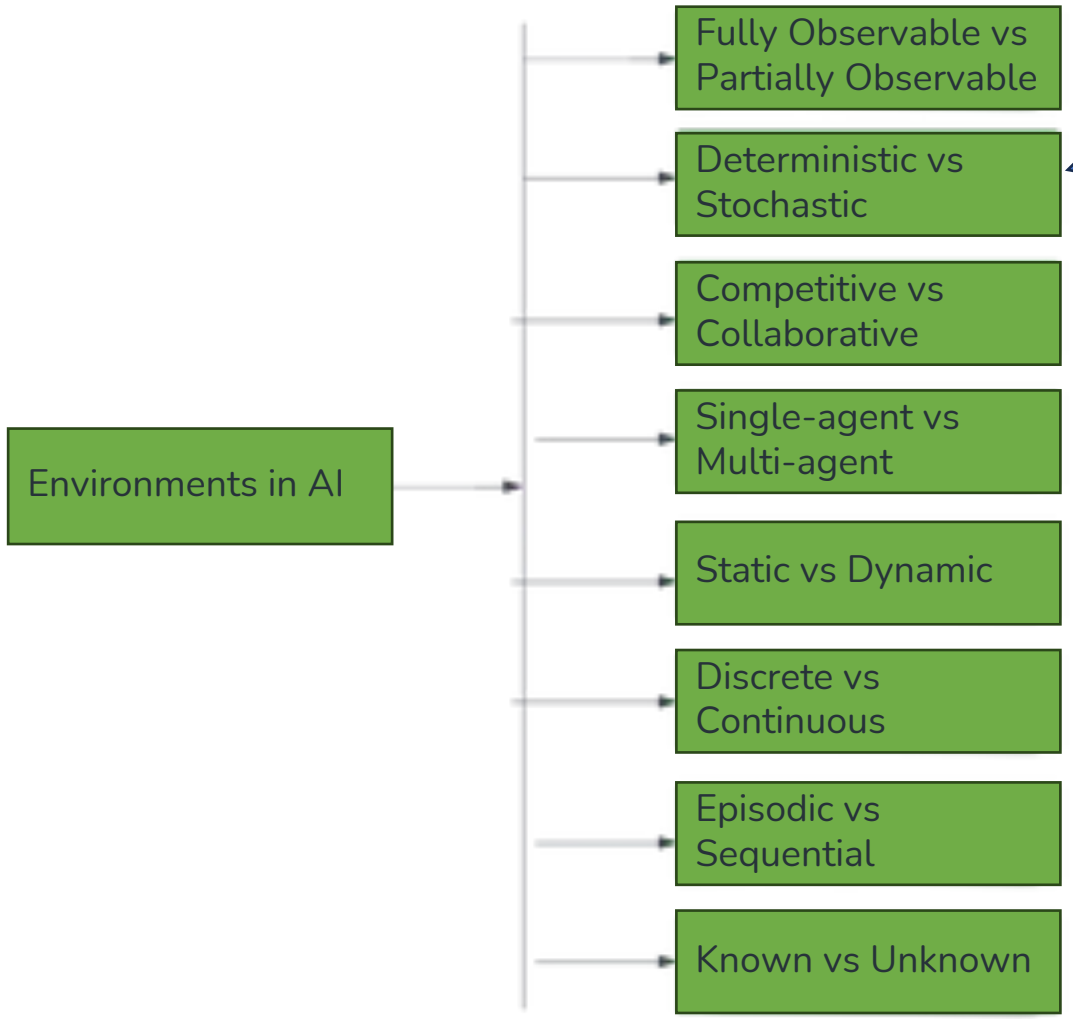
Some material in this lecture is taken from

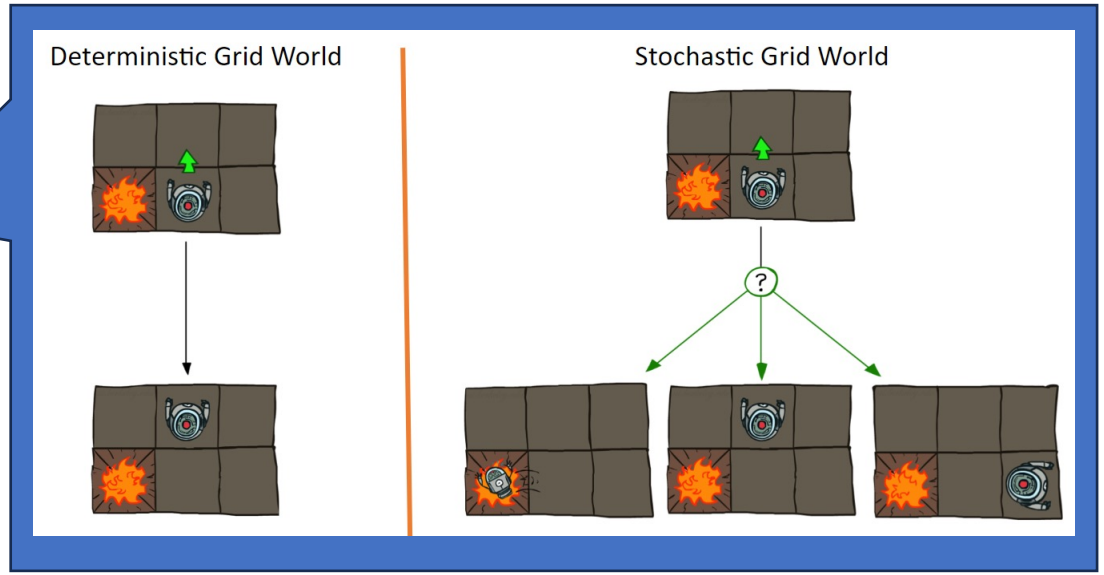
1. Prof. Ravindran's course: "Reinforcement Learning."
2. Dr Silver's course: "Reinforcement Learning."

In today's class…

- RL Framework
- What is a State?
- Special cases: Fully and Partially Observable environments
- Temporal Difference
- Components of an RL agent
  - Value function
  - Policy
  - Model
- Categories of RL

Types of Environments





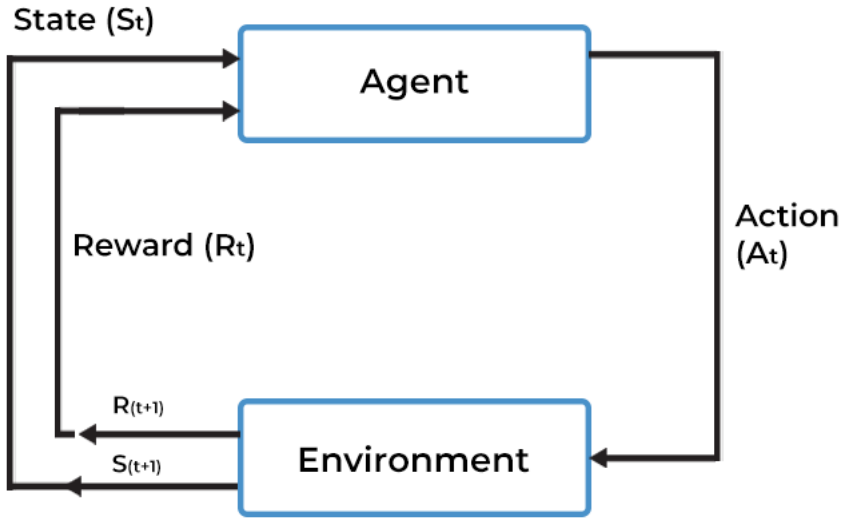
https://www.geeksforgeeks.org/types-of-environments-in-ai/

Fully Observable Environments

… extent to which the agent has access to information about the current state of the environment.

- A fully observable environment is one in which the **agent has complete information about the current state of the environment.**
- The agent has direct access to all environmental features that are necessary for making decisions.
- Example?

Board games like chess or checkers.



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)

Partially Observable Environments

- A partially observable environment is one in which the **agent does not have complete information about the current state of the environment.**

- The agent can only observe a subset of the environment, and some aspects of the environment may be hidden or uncertain.

- Examples?

  driving a car in traffic.

  A trading agent only observes current prices.

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: $S_t^a = (\mathbb{P}[S_t^e = s^1], \ldots, \mathbb{P}[S_t^e = s^n])$
- Recurrent neural network: $S_t^a = \sigma(S_{t-1}^a W_s + O_t W_o)$

Inside an RL Agent

An RL agent may include one or more of these components:

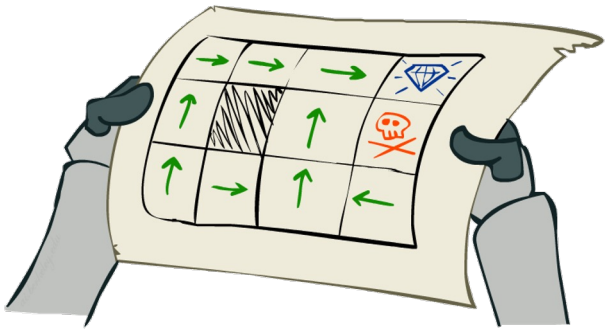
- **Policy**: agent's behavior function
- **Value function**: how good is each state and/or action
- **Model**: agent's representation of the environment

Inside an RL Agent

Policy

- A policy is the agent's behaviour
- It is a map from state to action, e.g.
- Deterministic policy: $a = \pi(s)$
- Stochastic policy: $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$

- In deterministic single-agent search problems, we wanted an optimal plan, or sequence of actions, from start to a goal

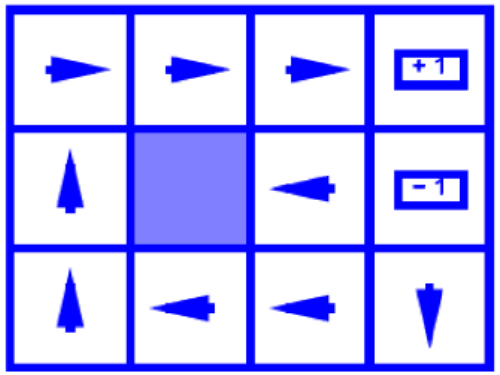


- For MDPs, we want an optimal policy $\pi^*: S \rightarrow A$
  - A policy π gives an action for each state
  - An optimal policy is one that maximizes expected utility / value if followed
  - An explicit policy defines a reflex agent

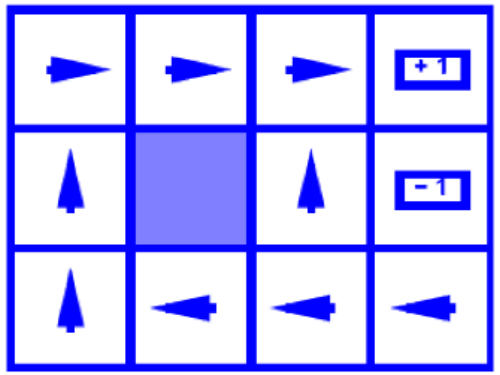
Inside an RL Agent

Policy

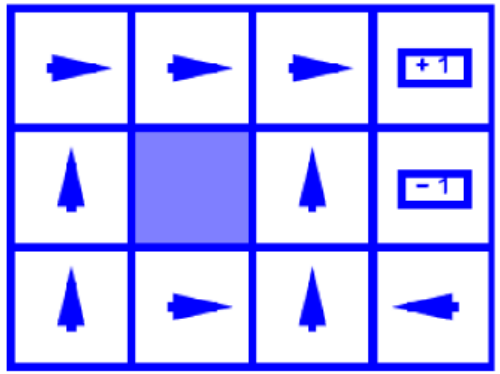
Optimal Policies



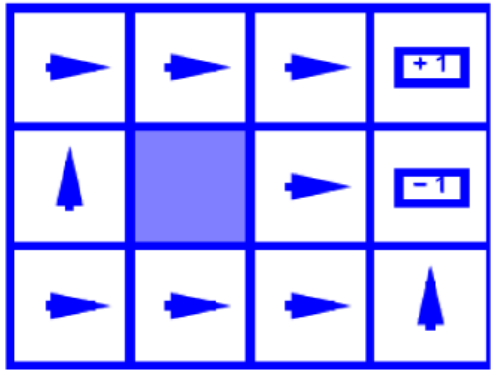
R(s) = -0.01



R(s) = -0.03



R(s) = -0.4



R(s) = -2.0

Inside an RL Agent

Value function

- Value function is a prediction of future reward
- Used to evaluate the goodness/badness of states
- And therefore to select between actions, e.g.

$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s \right]$$

Why Discounting?

- It's reasonable to maximize the sum of rewards
- It's also reasonable to prefer rewards now to rewards later
- One solution: values of rewards decay exponentially

1

Worth Now

γ

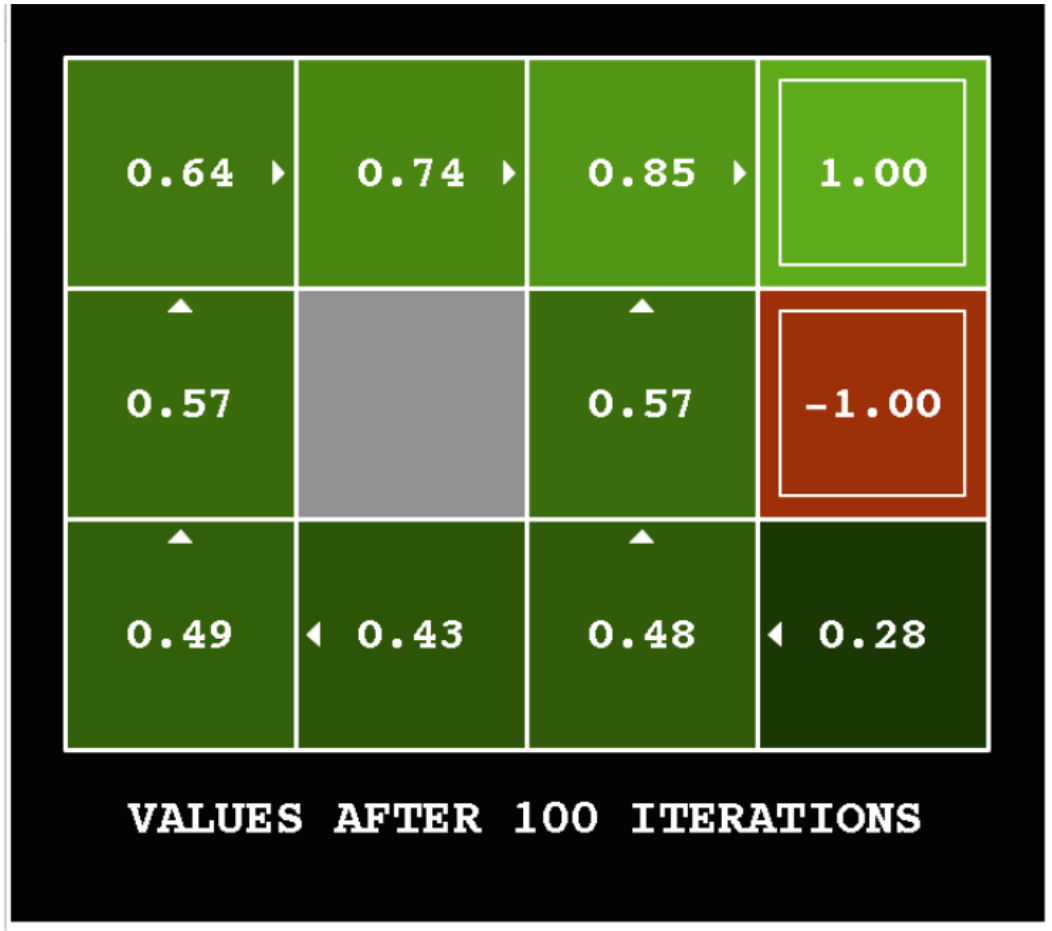
Worth Next Step

γ^2

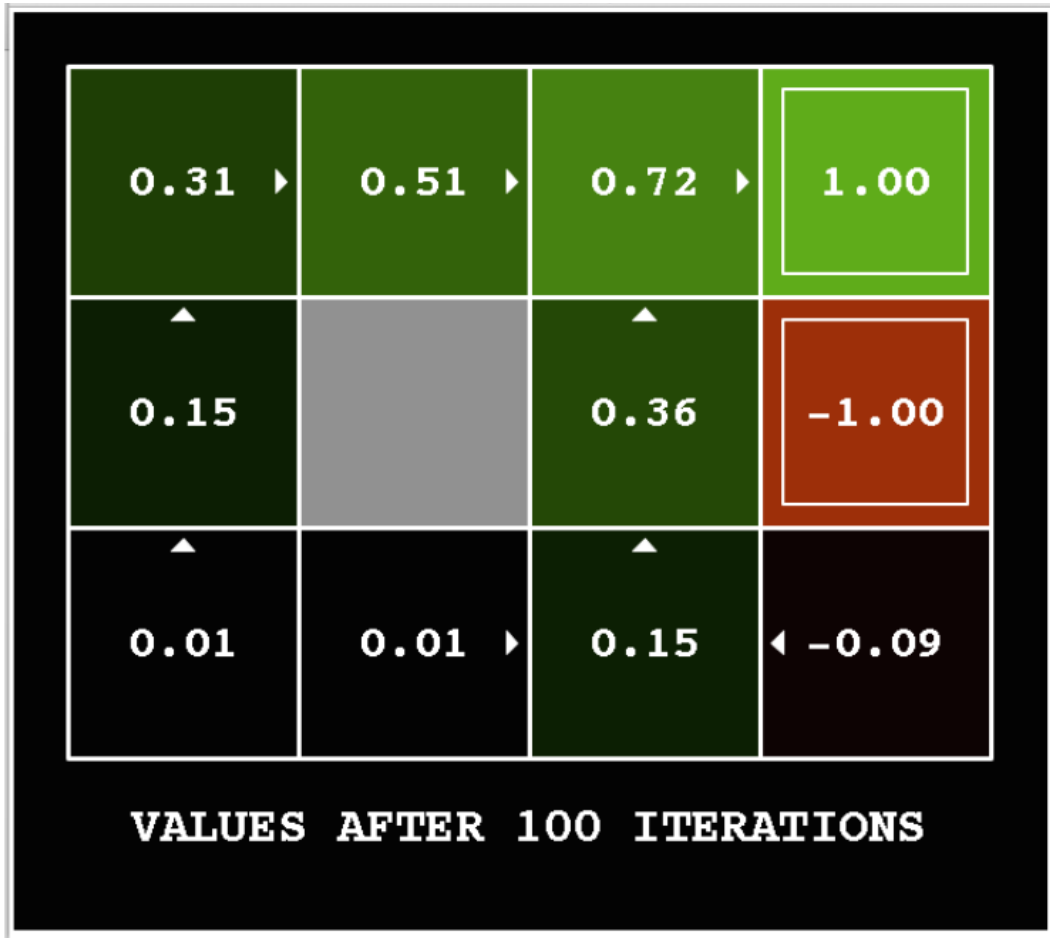
Worth In Two Steps

Inside an RL Agent
Value function

Living reward = 0



Living reward = -0.1



Dan Klein's and Pieter Abbeel's course: CS188 Intro to AI at UC Berkeley.

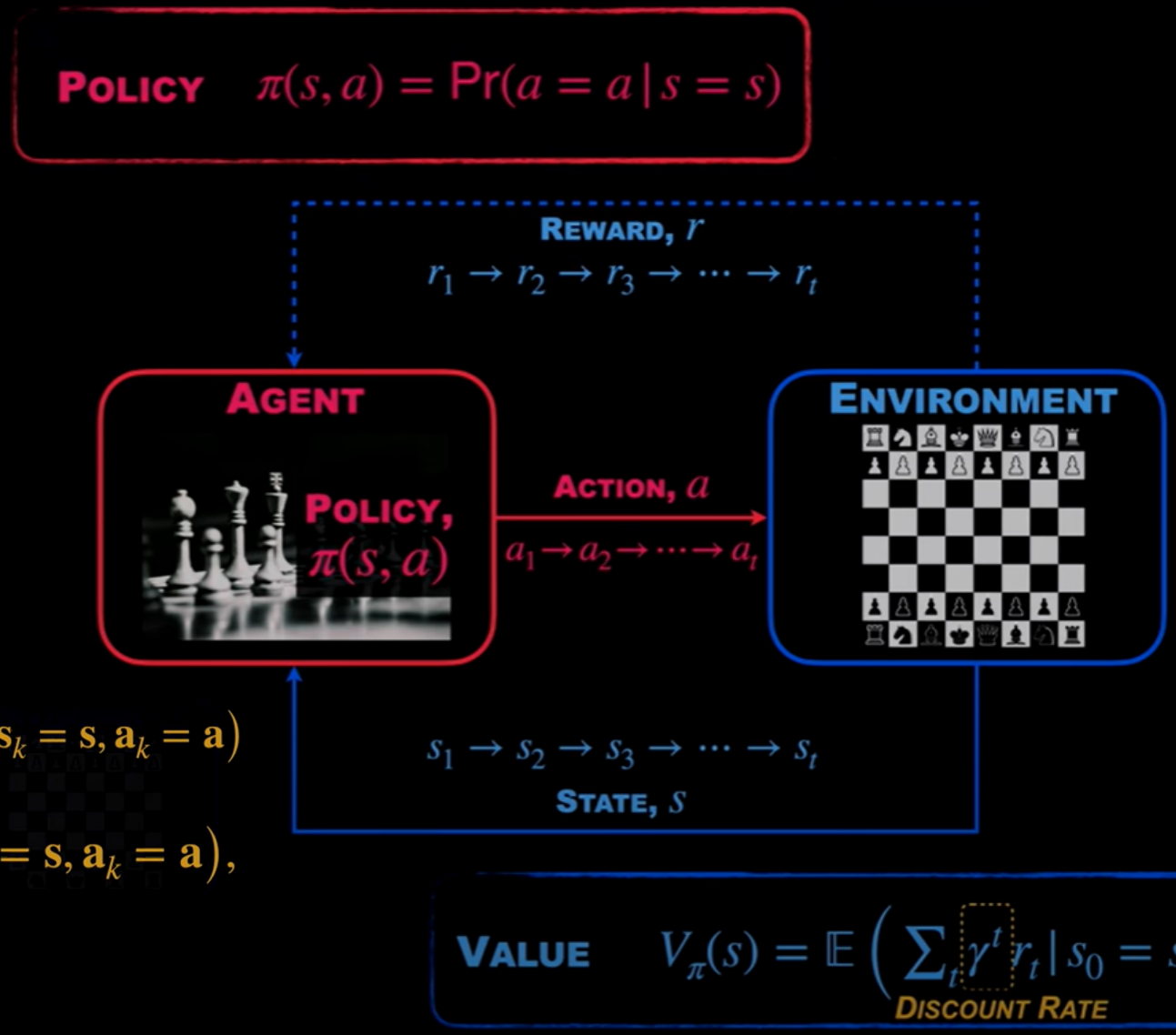
Inside an RL Agent
Model

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

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

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

Inside an RL Agent

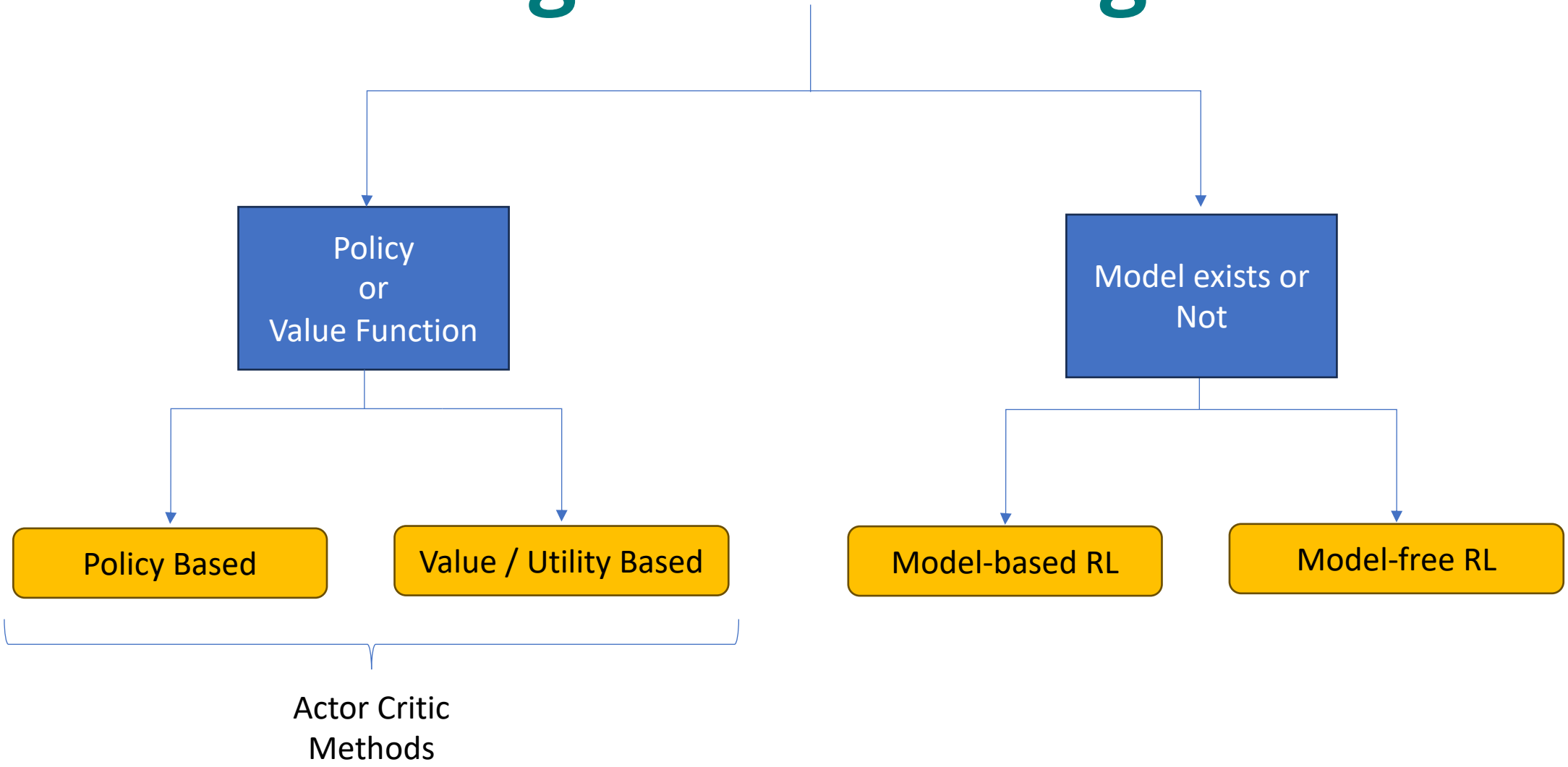


$$R(\mathbf{s}', \mathbf{s}, \mathbf{a}) = \mathbf{Pr}\left(\mathbf{r}_{k+1} \mid \mathbf{s}_{k+1} = \mathbf{s}', \mathbf{s}_k = \mathbf{s}, \mathbf{a}_k = \mathbf{a}\right)$$

$$P(\mathbf{s}', \mathbf{s}, \mathbf{a}) = \mathbf{Pr}\left(\mathbf{s}_{k+1} = \mathbf{s}' \mid \mathbf{s}_k = \mathbf{s}, \mathbf{a}_k = \mathbf{a}\right),$$

Dr Steve Brunton's course : Reinforcement Learning.

Categories of RL Agent



Reinforcement Learning

Model-based RL

Markov Decision Process $P(s', s, a)$

Policy Iteration $\pi_\theta(s, a)$

Value Iteration $V(s)$

Actor Critic

Dynamic programming & Bellman optimality

Nonlinear Dynamics

$$\frac{d}{dt}x = f(x(t), u(t), t)\, dt$$

Optimal Control & HJB

Deep MPC

Model-free RL

Gradient free

Off Policy

DQN

$Q(s, a)$

Q Learning

On Policy

TD(0)

⋮

TD(∞) ≡ MC

TD-λ

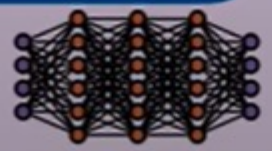
SARSA

Gradient based

Deep Policy Network

$$\theta^{new} = \theta^{old} + \alpha \nabla_\theta R_{\Sigma,\theta}$$

Policy Gradient Optimization



Deep RL

Dr Steve Brunton's course : Reinforcement Learning.

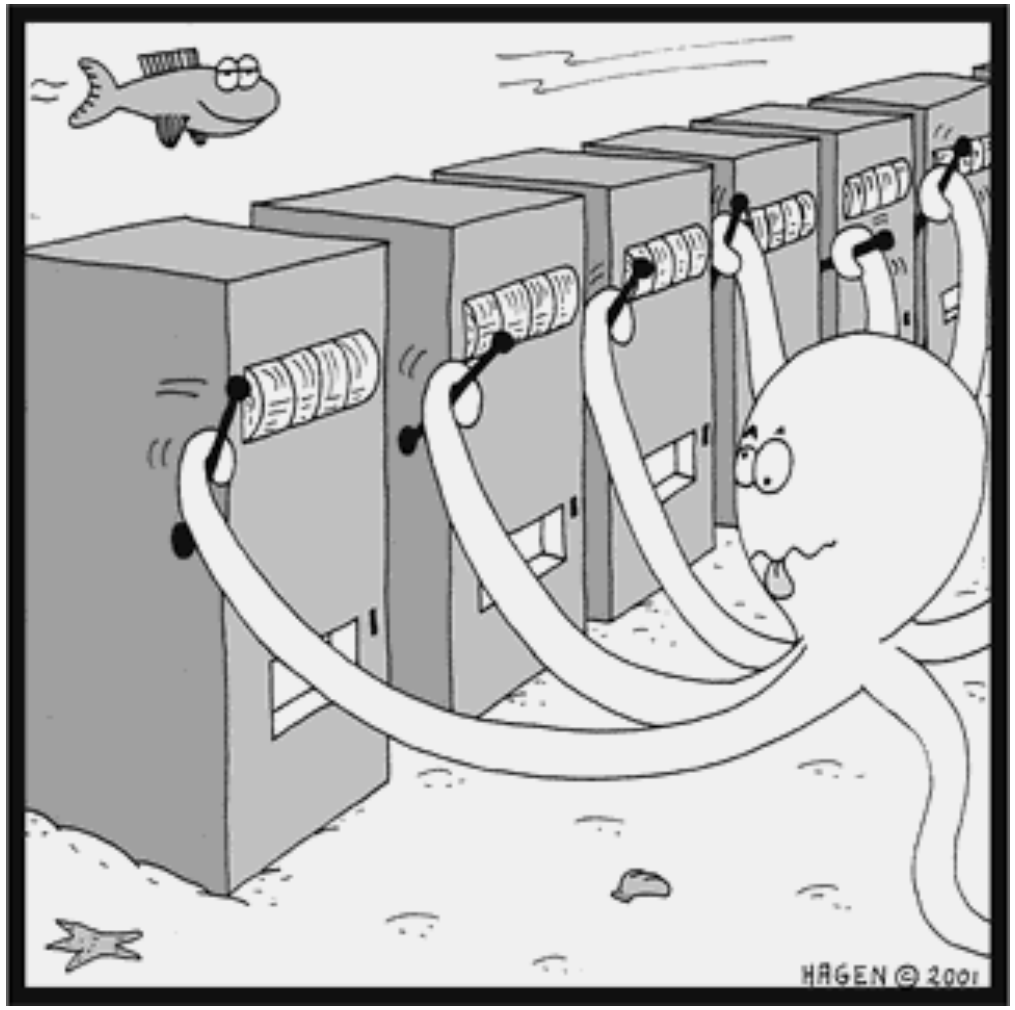
Reinforcement Learning Fundamentals

Lecture 5: Multi-armed Bandit

Dr Sandeep Manjanna
Assistant Professor, Plaksha University
sandeep.manjanna@plaksha.edu.in

In today's class…

- Immediate RL Problems
- Exploration vs. Exploitation
- Multi-armed Bandit
- ϵ-greedy algorithms
- Performance metrics



Immediate RL Problems

- Every time instant **t**, pick an action $\mathbf{a_t}$ and get a reward $\mathbf{R_t}$.
- There is no state!
- Example:
  - Testing a drug for effectiveness
  - Tossing a coin
- 3 Actions: Choose one of the 3 coins

Coin 1

$\mathbb{P}\{\text{heads}\} = p_1$

Coin 2

$\mathbb{P}\{\text{heads}\} = p_2$

Coin 3

$\mathbb{P}\{\text{heads}\} = p_3$

Given 10 trials at tossing, maximize the total number of heads.

If, the probabilities p1, p2, and p3 are given, then?

How many heads in 10 tosses?

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
- It is usually important to explore as well as exploit

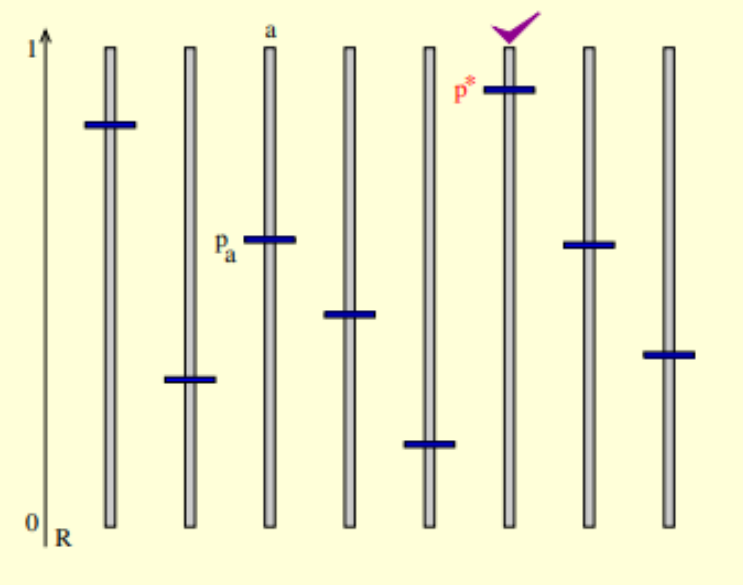
Exploration vs. Exploitation

Examples?
What is exploration and exploitation in these examples?

- Restaurant Selection
- Online Advertising: Template optimization
- Clinical trials
- Packet routing in communication networks
- Game playing and reinforcement learning
- Oil Drilling or Mining

Multi-armed Bandit

- A hypothetical experiment where a person must **choose between multiple actions** (i.e., slot machines, the "one-armed bandits"), **each with an unknown payout**.



- The goal is to determine the best or **most profitable outcome** through a series of choices.

- At the beginning of the experiment, when odds and payouts are unknown, the gambler must determine **which machine to pull, in which order and how many times**.

Algorithm to solve Multi-armed Bandit?

- Here is what an algorithm does—

  For $t = 0, 1, 2, \ldots, T-1$:

  - Given the history $h_t = (a_0, r_0, a_1, r_1, a_2, r_2, \ldots, a_{t-1}, r_{t-1})$,
  - Pick an arm a_t to sample (or "pull"), and
  - Obtain a reward r_t drawn from the distribution corresponding to arm a_t.
- T is the total sampling budget, or the horizon.
- Formally: a deterministic algorithm is a mapping

  from the set of all histories

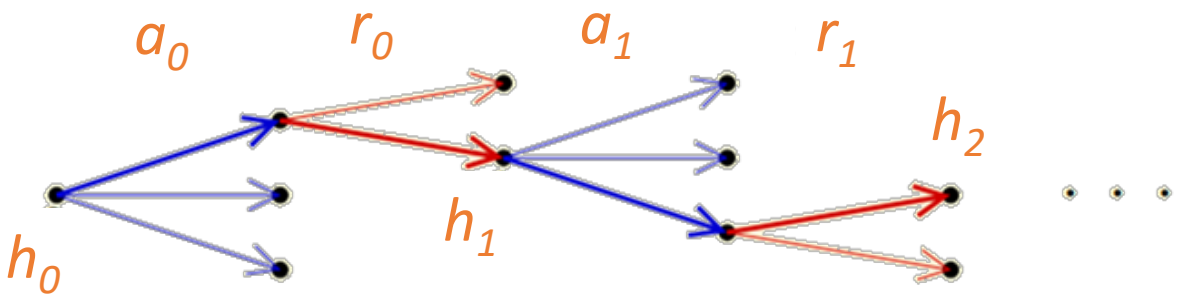
  to the set of all arms.
- Formally: a **stochastic** algorithm is a mapping

  from the set of all histories

  to the set of all probability distributions over arms.
- The algorithm picks the arm to pull; the bandit instance returns the reward.

Slide from Prof. Shivaram Kalyanakrishnan's course: "Foundations of Intelligent and Learning Agents."

Multi-armed Bandit Tree



For a complete horizon T,

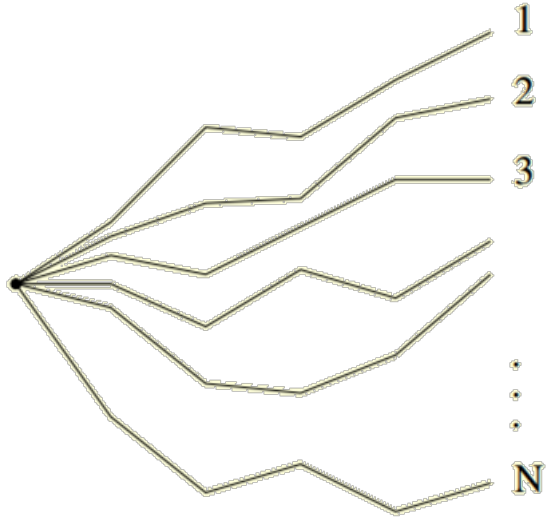
$h_T = (a_0, r_0, a_1, r_1, a_2, r_2, \ldots, a_{T-1}, r_{T-1})$,

$$P(h_T) = \prod_{t=0}^{t=T-1} P(a_t \mid h_t)\, P(r_t \mid a_t)$$

Decided by the Algorithm

Decided by the bandit instance

- An algorithm, bandit instance pair can generate many possible T-length histories.



How many histories possible if the algorithm is deterministic and rewards 0–1?

Slide from Prof. Shivaram Kalyanakrishnan's course: "Foundations of Intelligent and Learning Agents."

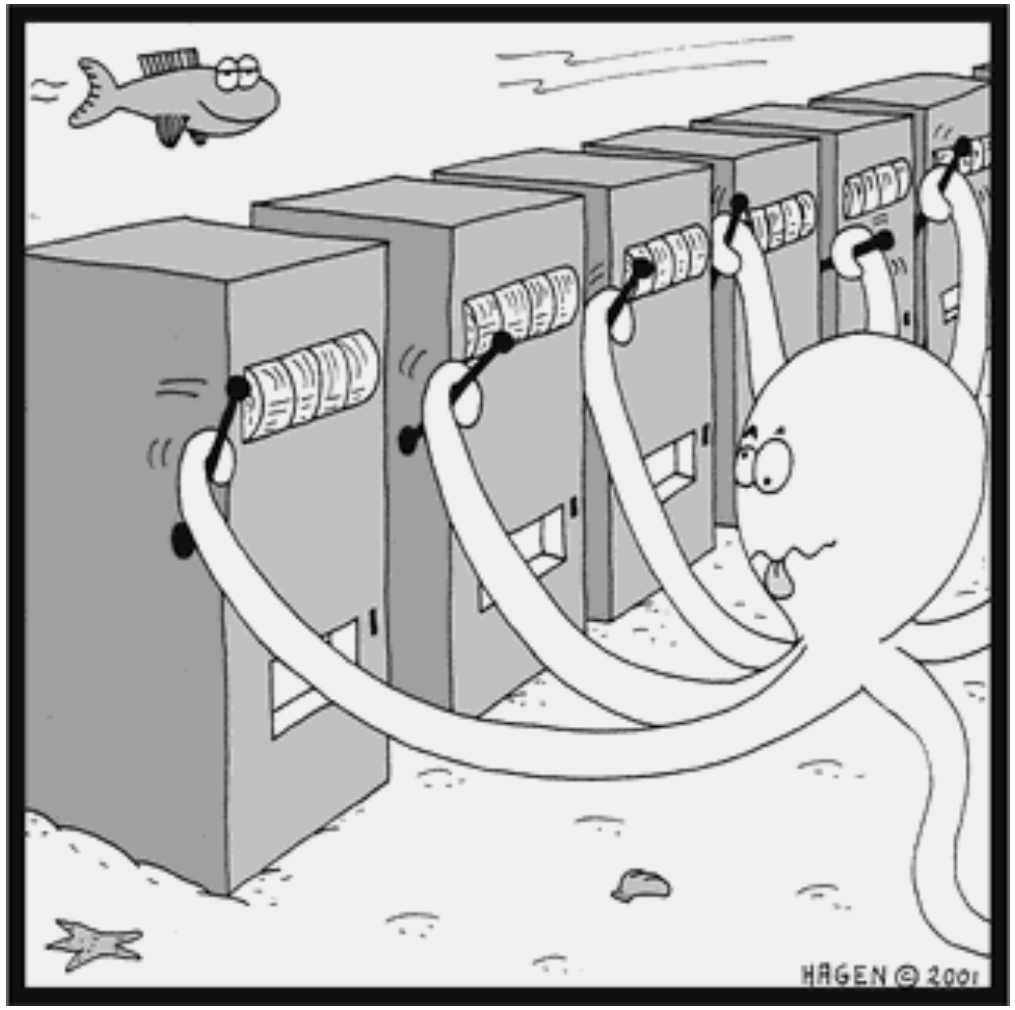
Reinforcement Learning Fundamentals

Lecture 6: Multi-armed Bandit

Dr Sandeep Manjanna
Assistant Professor, Plaksha University
sandeep.manjanna@plaksha.edu.in

In today's class…

- Value function-based methods
  - ϵ-greedy algorithms
- Performance metrics
  - Correctness
  - Convergence
  - Sample Efficiency



Value function-based Methods

- Consider that the estimated value of a given action ***a*** at timestep ***t*** is given by $Q_t(a)$

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}}$$

- Over time as the denominator goes to infinity, $Q_t(a)$ converges to $q_*(a)$ *(the true expected value of action a)*.

- We want to choose an action that gives the maximum estimated reward. What would be a greedy strategy to do this?

$$A_t \doteq \arg\max_a Q_t(a)$$

- But, what about exploration?

ϵ-greedy Algorithms

- Parameter $\epsilon \in [0, 1]$ controls the amount of exploration.

$$A_t \doteq \arg\max_a Q_t(a)$$

- ϵG1
  - If $t \leq \epsilon T$, sample an arm uniformly at random.
  - At $t = \lfloor \epsilon T \rfloor$, identify a^{best}, an arm with the highest empirical mean.
  - If $t > \epsilon T$, sample a^{best}.
- ϵG2
  - If $t \leq \epsilon T$, sample an arm uniformly at random.
  - If $t > \epsilon T$, sample an arm with the highest empirical mean.
- ϵG3
  - With probability ϵ, sample an arm uniformly at random; with probability $1 - \epsilon$, sample an arm with the highest empirical mean.

Usually unless mentioned, ϵ-greedy method refers to ϵG3

Slide from Prof. Shivaram Kalyanakrishnan's course: "Foundations of Intelligent and Learning Agents."

ϵ-greedy Algorithms

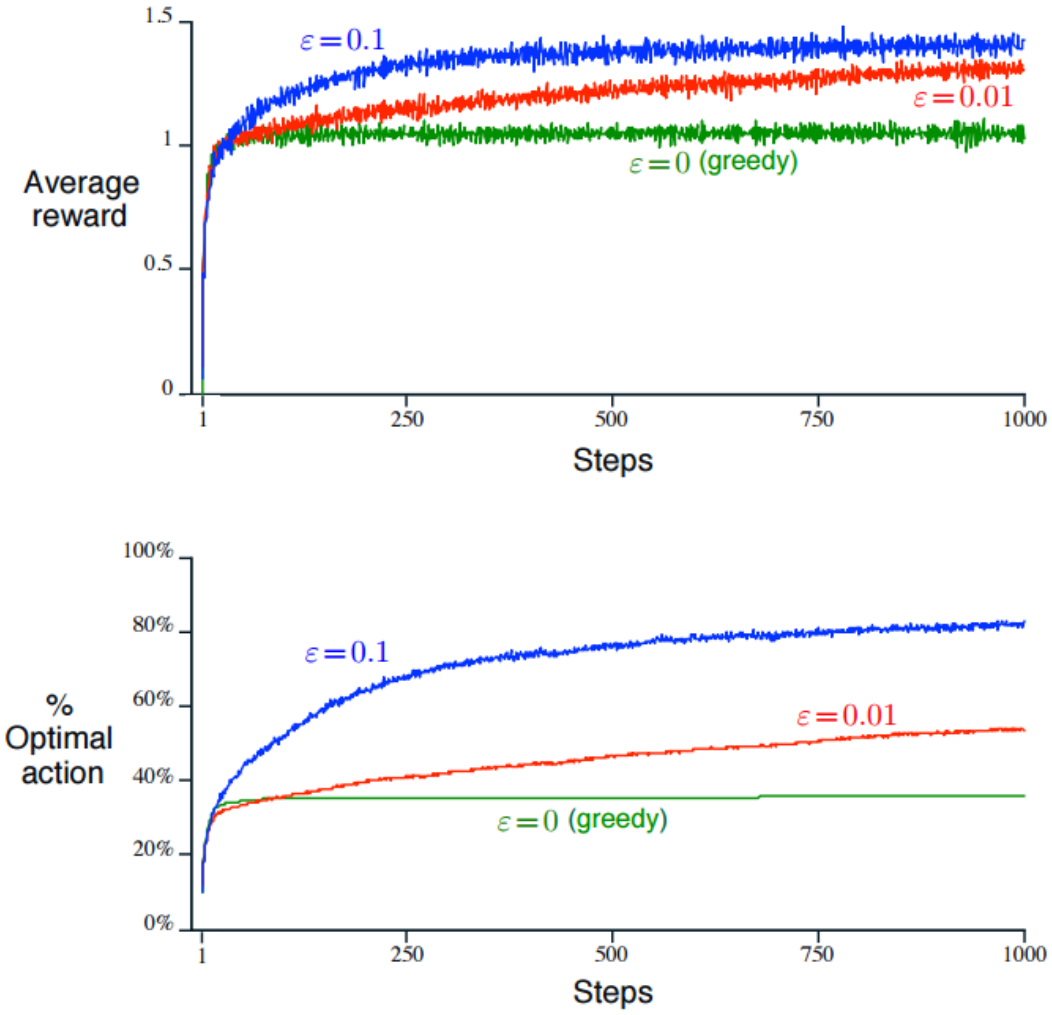


Figure 2.2: Average performance of ε-greedy action-value methods on the 10-armed testbed. These data are averages over 2000 runs with different bandit problems. All methods used sample averages as their action-value estimates.

From the textbook: "Reinforcement Learning an Introduction", by Sutton and Barto

ϵ-greedy Algorithms

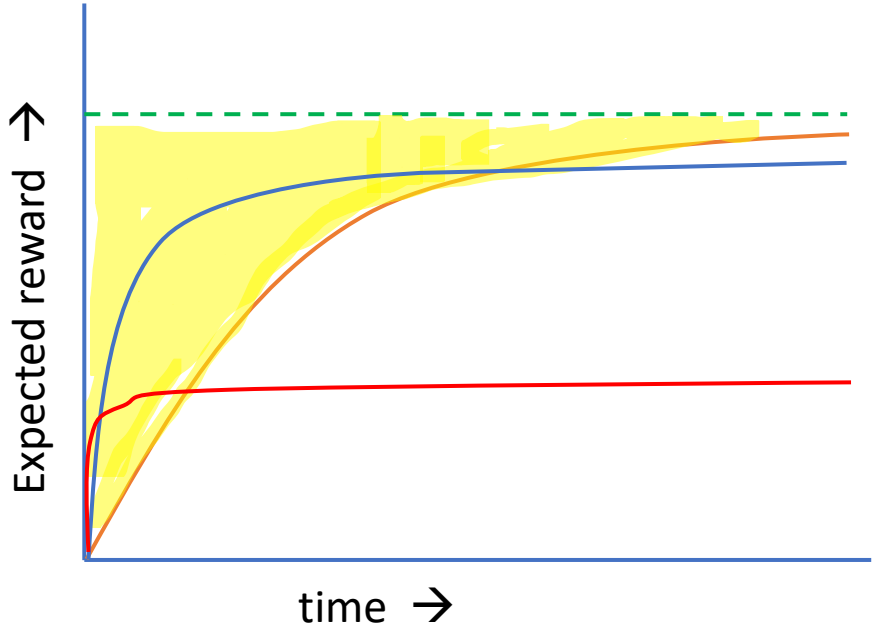
- Is ϵG2 better than ϵG1 ?
- What is the probability of picking an arm in each of these algorithms ?
- How can ϵ-greedy algorithms be written in the form of P(arm | history) ?

Performance Metrics

- **Asymptotic Correctness**
  - Gives a guarantee that eventually the algorithm will be selecting an arm that has the highest pay-off.
    - As T tends to infinity,
  - Will ϵ-greedy algorithm give asymptotic correctness?
    - Keep decreasing the ϵ value with time (cooling).

Performance Metrics

- **Regret Optimality**
  - "disappointed over (something that one has done or failed to do)" – *definition of regret from Oxford Dictionary*



  - Regret optimality refers to increasing the total reward one gets over the process of learning.