AMRITA
VISHWA VIDYAPEETHAM

21AIE311 - REINFORCEMENT LEARNING

MARKOV DECISION PROCESS

DR. AMUDHA J. AND MR. A. A. NIPPUN KUMAAR
DEPARTMENT OF CSE
AMRITA SCHOOL OF COMPUTING, BANGALORE

PREVIOUSLY

- RL - What ?? Why ??
- Key Features of RL
- History - Timeline
- Real World RL Examples
- Elements of RL
- An Example

LECTURE OVERVIEW

- Recap and Terminologies
- Markov Decision Process
- The Agent-Environment Interface
- Goal and Rewards
- Returns and Episodes
- Unified Notion

RECAP

- RL agent Learns from interaction and its goal oriented.
- RL agent makes sequence of decisions as actions and gets rewarded(Good/Bad) for its actions.
- This is inspired from nature, gazelle calf learning to walk and Preparation of Breakfast.
- Exploration vs Exploitation ??
- Elements of RL - Policy, Reward, Value and Model.
- Other Entities: Agent, Action, Environment
- Tic-Tac-Toe playing RL agent using value estimates and Temporal Difference method.

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[V(S_{t+1}) - V(S_t)\Big]$$

MARKOV DECISION PROCESS

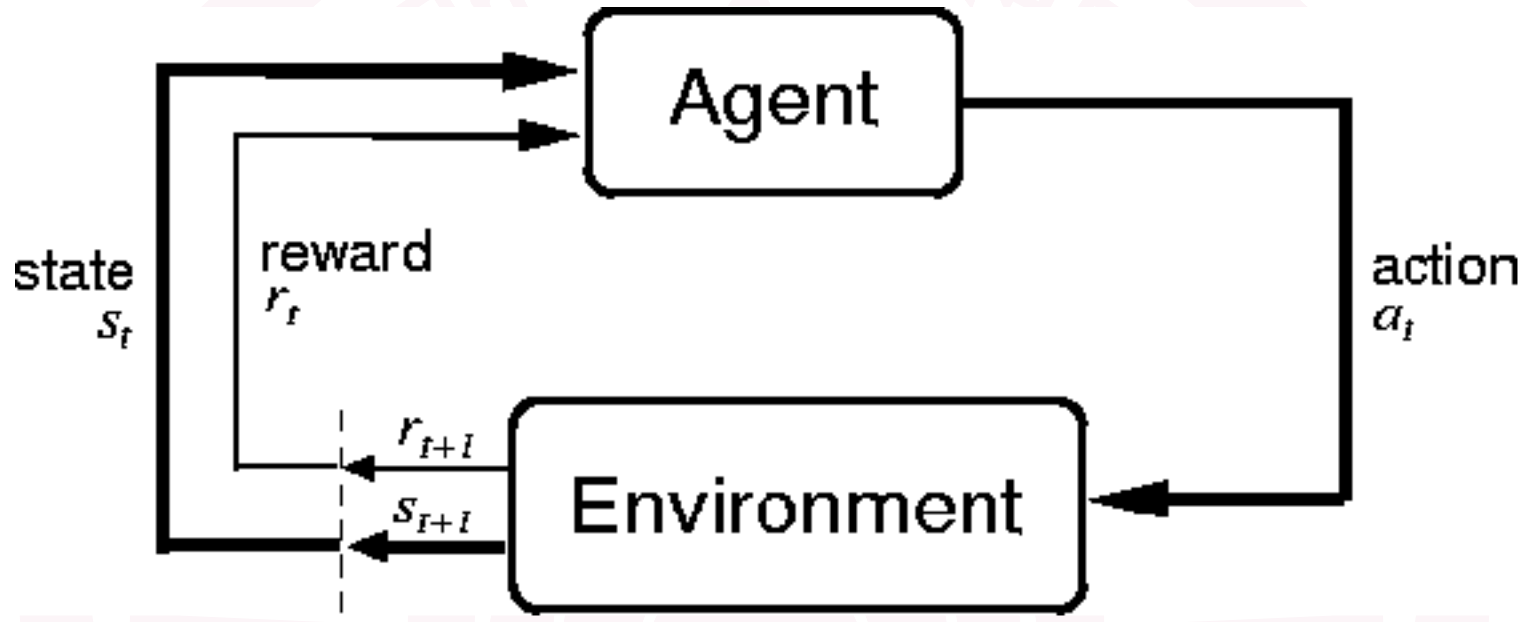
In mathematics, a Markov decision process (MDP) is a discrete-time stochastic control process. It provides a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker.

Wikipedia

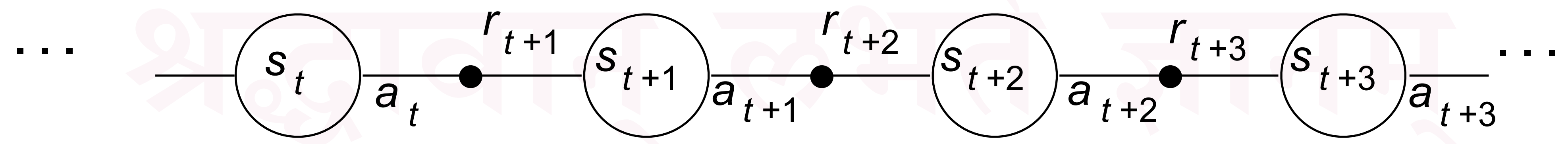
MARKOV DECISION PROCESS

- MDPs are a classical formalization of sequential decision making, where actions influence not just immediate rewards, but also subsequent situations, or states, and through those future rewards.
- Thus MDPs involve delayed reward and the need to tradeoff immediate and delayed reward.
- In MDPs we estimate the value of each action a in each state s, or we estimate the value of each state given optimal action selections.

THE AGENT-ENVIRONMENT INTERFACE



- Discrete time steps $t = 0,1,2,3,....$
- Agent observes State $S_t \in \mathcal{S}$
- Produces Action $A_t \in \mathcal{A}(s)$
- Gets resulting Reward $r_{t+1} \in \mathcal{R}$
- And resulting Next State s_{t+1}


$\dots$ s_t a_t r_{t+1} s_{t+1} a_{t+1} r_{t+2} s_{t+2} a_{t+2} r_{t+3} s_{t+3} a_{t+3} $\dots$

THE AGENT-ENVIRONMENT INTERFACE

The Agent Learns a Policy

- Policy at step t, π_t :
  - A mapping from states to action probabilities

    $\pi_t(s, a) =$ probablity that $a_t = a$when $s_t = s$

- Reinforcement learning methods specify how the agent changes its policy as a result of experience.
- Roughly, the agent's goal is to get as much reward as it can over the long run.

THE AGENT-ENVIRONMENT INTERFACE

- In a finite MDP, the sets of states, actions, and rewards ($\mathcal{S}, \mathcal{A}$, and $\mathcal{R}$) all have a finite number of elements.

- The random variables R_t and S_t have well defined discrete probability distributions dependent only on the preceding state and action.

- That is, for particular values of these random variables, $s' \in \mathcal{S}$ and $r \in \mathcal{R}$ there is a probability of those values occurring at time t, given particular values of the preceding state and action:

$$p(s', r \mid s, a) \doteq Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$$

THE AGENT-ENVIRONMENT INTERFACE

$$p(s', r \mid s, a) \doteq Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$$

- $\forall s', s \in \mathcal{S}, r \in \mathcal{R}$ and $a \in \mathcal{A}(s)$
- The function p defines the dynamics of the MDP.
- $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0,1]$ is an ordinary deterministic function of four arguments.

$$\sum_{s \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) = 1, \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$$

THE AGENT-ENVIRONMENT INTERFACE

- In a Markov decision process, the probabilities given by p completely characterize the environment's dynamics.
- That is, the probability of each possible value for St and Rt depends only on the immediately preceding state and action, St−1 and At−1, and, given them, not at all on earlier states and actions.
- The state must include information about all aspects of the past agent-environment interaction that make a difference for the future. If it does, then the state is said to have the Markov property.

THE AGENT-ENVIRONMENT INTERFACE

- From the four-argument dynamics function, p, one can compute anything else one might want to know about the environment.
- State-transition probabilities $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0,1]$

$$p(s' \mid s, a) \doteq Pr\{S_t = s' \mid S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in \mathcal{R}} p(s', r \mid s, a)$$

THE AGENT-ENVIRONMENT INTERFACE

- Expected reward as two-argument function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$

$$r(s,a) \doteq \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r \mid s, a)$$

- Expected reward as three-argument function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$

$$r(s,a,s') \doteq \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a, S_t = s'] = \sum_{r \in \mathcal{R}} r \frac{p(s', r \mid s, a)}{p(s' \mid s, a)}$$

THE AGENT-ENVIRONMENT INTERFACE

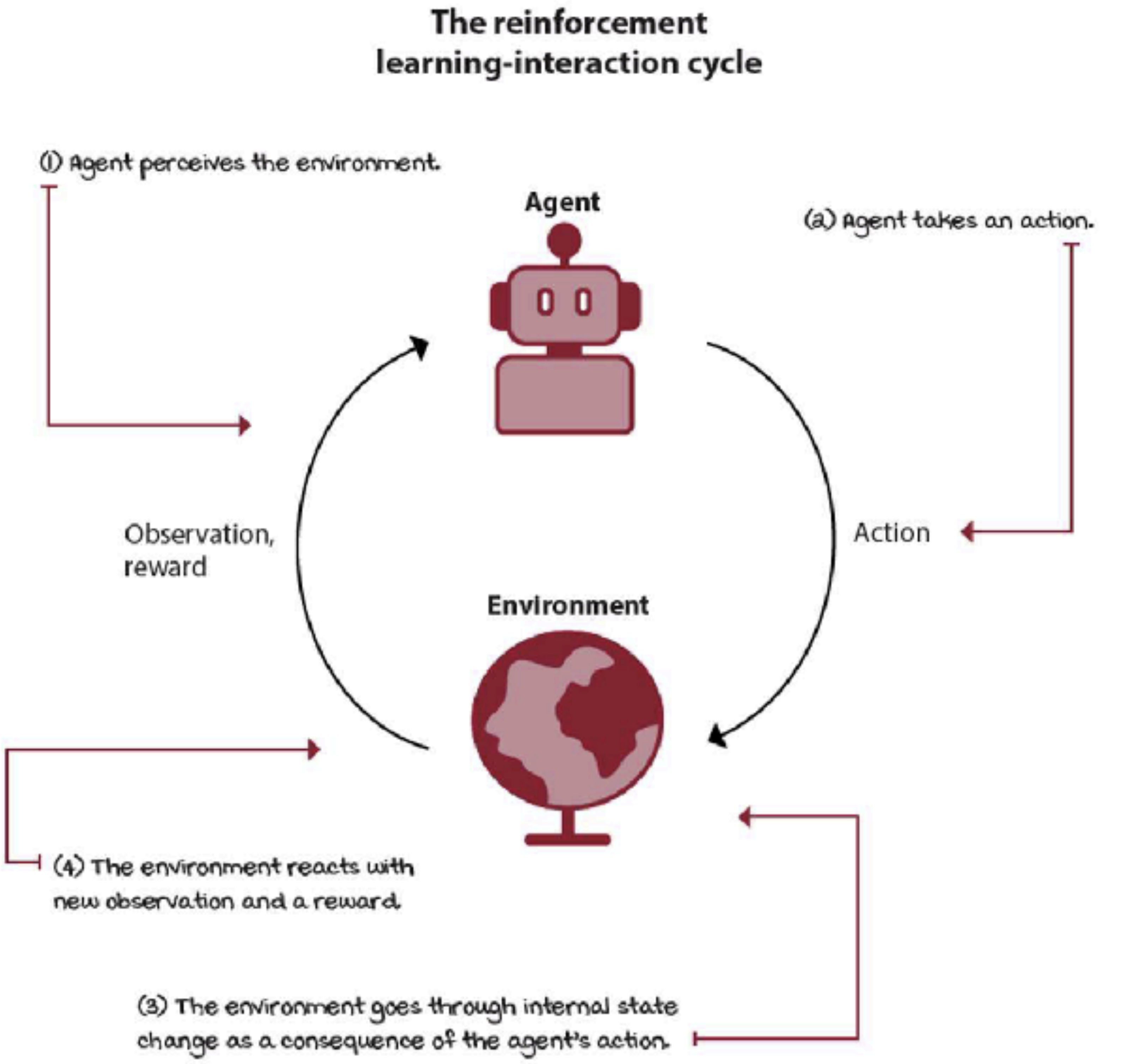
Getting the Degree of Abstraction Right

- Time steps need not refer to fixed intervals of real time.
- Actions can be low level (e.g., voltages to motors), or high level (e.g., accept a job offer), "mental" (e.g., shift in focus of attention), etc.
- States can low-level "sensations", or they can be abstract, symbolic, based on memory, or subjective (e.g., the state of being "surprised" or "lost").
- An RL agent is not like a whole animal or robot, which consist of many RL agents as well as other components.
- The environment is not necessarily unknown to the agent, only incompletely controllable.
- Reward computation is in the agent's environment because the agent cannot change it arbitrarily.

THE AGENT-ENVIRONMENT INTERFACE

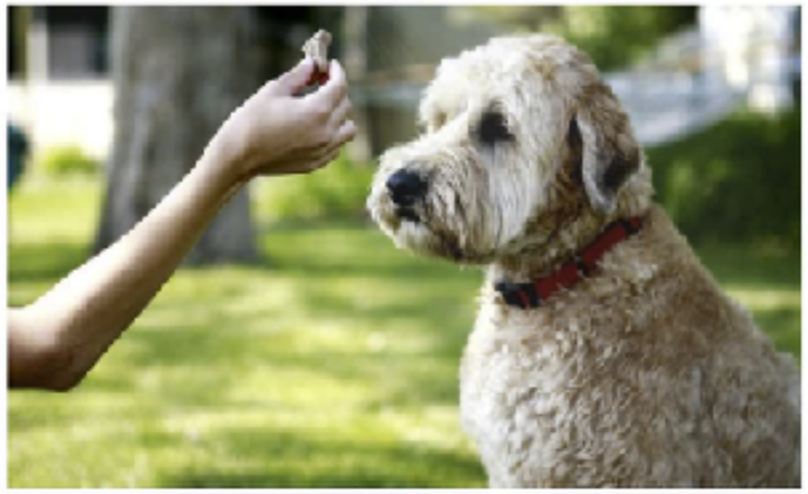
- MDP framework abstraction - goal-directed learning
- Three signals pass back and forth between the agent and environment
  - The choice made by the agent - Actions
  - The basis on which the choice is made - States
  - Agent's goal - Reward
- This framework may not be sufficient to represent all decision-learning problems usefully, but it has proved to be widely useful and applicable.
- Of course, the particular states and actions vary greatly from task to task, and how they are represented can strongly affect performance.
- In reinforcement learning, as in other kinds of learning, such representational choices are, at present, more art than science.

THE AGENT-ENVIRONMENT INTERFACE


The reinforcement learning-interaction cycle
(1) Agent perceives the environment.
Agent
(2) Agent takes an action.
Observation, reward
Action
Environment
(4) The environment reacts with new observation and a reward.
(3) The environment goes through internal state change as a consequence of the agent's action.

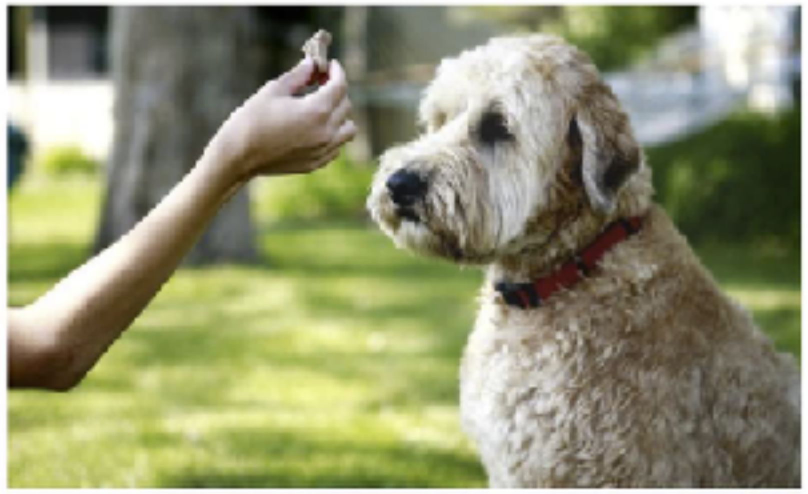
EXAMPLES OF PROBLEMS, AGENTS, AND ENVIRONMENTS

- **Problem:** You're training your dog to sit.
- **Agent:** The part of your brain that makes decisions.
- **Environment:** Your dog, the treats, your dog's paws, the loud neighbor, and so on.
- **Actions:** Talk to your dog. Wait for the dog's reaction. Move your hand. Show treat. Give treat. Pet.
- **Observations:** Your dog is paying attention to you. Your dog is getting tired. Your dog is going away. Your dog sat on command.



EXAMPLES OF PROBLEMS, AGENTS, AND ENVIRONMENTS

- **Problem:** Your dog wants the treats you have.
- **Agent:** The part of your dog's brain that makes decisions.
- **Environment:** You, the treats, your dog's paws, the loud neighbor, and so on.
- **Actions:** Stare at owner. Bark. Jump at owner. Try to steal the treat. Run. Sit.
- **Observations:** Owner keeps talking loud at dog. Owner is showing the treat. Owner is hiding the treat. Owner gave the dog the treat.



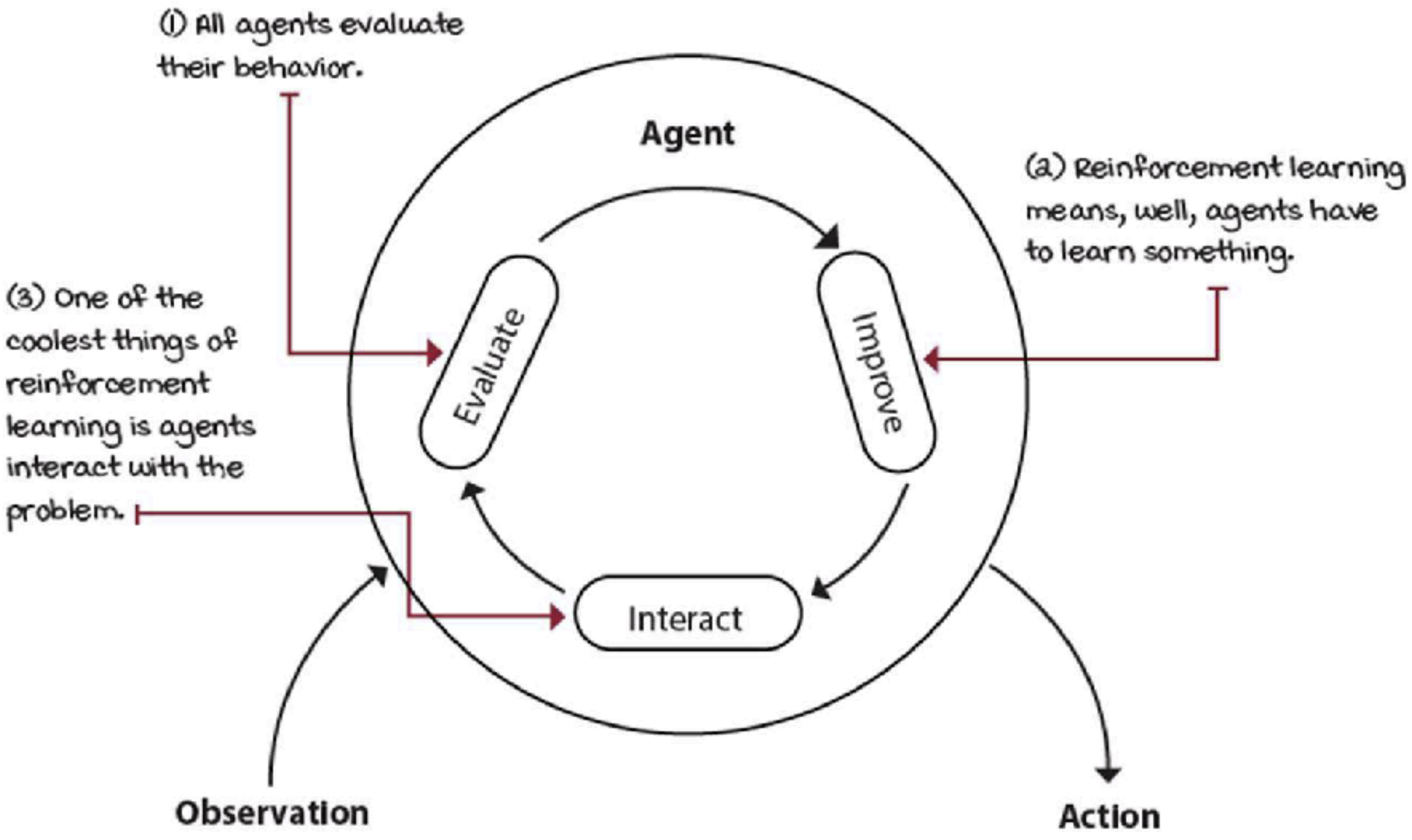
EXAMPLES OF PROBLEMS, AGENTS, AND ENVIRONMENTS

- **Problem:** A trading agent investing in the stock market.
- **Agent:** The executing DRL code in memory and in the CPU.
- **Environment:** Your internet connection, the machine the code is running on, the stock prices, the geopolitical uncertainty, other investors, day traders, and so on.
- **Actions:** Sell n stocks of y company. Buy n stocks of y company. Hold.
- **Observations:** Market is going up. Market is going down. There are economic tensions between two powerful nations. There's danger of war in the continent. A global pandemic is wreaking havoc in the entire world.

EXAMPLES OF PROBLEMS, AGENTS, AND ENVIRONMENTS

- **Problem:** You're driving your car.
- **Agent:** The part of your brain that makes decisions.
- **Environment:** The make and model of your car, other cars, other drivers, the weather, the roads, the tires, and so on.
- **Actions:** Steer by x, accelerate by y. Break by z. Turn the headlights on. Defog windows. Play music.
- **Observations:** You're approaching your destination. There's a traffic jam on Main Street. The car next to you is driving recklessly. It's starting to rain. There's a police officer driving in front of you.

The three internal steps that every reinforcement learning agent goes through



(1) All agents evaluate their behavior.
Agent
(2) Reinforcement learning means, well, agents have to learn something.
(3) One of the coolest things of reinforcement learning is agents interact with the problem.
Evaluate
Improve
Interact
Observation
Action

TERMINOLOGIES

- The interactions between the agent and the environment go on for several cycles.
- Each cycle is called a **time step**. A time step is a unit of time, which can be a millisecond, a second, 1.2563 seconds, a minute, a day, or any other period of time.
- At each time step, the agent observes the environment, takes action, and receives a new **observation** and **reward**. Notice that, even though rewards can be negative values, they are still called rewards in the RL world. The set of the observation (or state), the action, the reward, and the new observation (or new state) is called an **experience tuple.**

TERMINOLOGIES

- The task the agent is trying to solve may or may not have a natural ending. Tasks that have a natural ending, such as a game, are called **episodic tasks**.
- Tasks that don't, such as learning forward motion, are called **continuing tasks**.
- The sequence of time steps from the beginning to the end of an episodic task is called an **episode**.
- Agents may take several time steps and episodes to learn to solve a task. The sum of rewards collected in a single episode is called a **return**. Agents are often designed to maximize the return. A **time step limit** is often added to continuing tasks, so they become episodic tasks, and agents can maximize the return

TERMINOLOGIES

- Every experience tuple has an opportunity for learning and improving performance. The agent may have one or more components to aid learning.
- The agent may be designed to learn mappings from observations/states to actions called **policies**.
- The agent may be designed to learn mappings from observations/states to new observations/states and/or rewards called **models**.
- The agent may be designed to learn mappings from observations (and possibly actions) to reward-to-go estimates (a slice of the return) called **value functions**.

TERMINOLOGIES

- The environment is represented by a set of variables related to the problem.
- The combination of all the possible values this set of variables can take is referred to as the **state space**. A state is a specific set of values the variables take at any given time.
- Agents may or may not have access to the actual environment's state; however, one way or another, agents can observe something from the environment. The set of variables the agent perceives at any given time is called an **observation**.
- The combination of all possible values these variables can take is the **observation space**.

TERMINOLOGIES

- At every state, the environment makes available a set of actions the agent can choose from. Often the set of actions is the same for all states, but this isn't required. The set of all actions in all states is referred to as the **action space**.
- The agent attempts to influence the environment through these actions. The environment may change states as a response to the agent's action. The function that is responsible for this transition is called the **transition function**.
- After a transition, the environment emits a new observation. The environment may also provide a reward signal as a response. The function responsible for this mapping is called the **reward function**. The set of transition and reward function is referred to as the **model** of the environment.

GOALS AND REWARDS

- In reinforcement learning, the purpose or goal of the agent is formalized in terms of a special signal, called the reward, passing from the environment to the agent.
- At each time step, the reward is a simple number, $R_t \in \mathbb{R}$
- Informally, the agent's goal is to maximize the total amount of reward it receives.
- This means maximizing not immediate reward, but cumulative reward in the long run.

GOALS AND REWARDS

- Reward Hypothesis
  - That all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward).
- Reward to formalize the idea of Goal.
- Robot that learns walking
  - Reward promotional to forward motion
- Escape from Maze
  - Reward -1 for every step - Encourages the agent to act fast
- Empty Soda Can collecting robot
  - Reward 1 for each can collected else 0
  - Negative Reward if the robot bumps into things

GOALS AND REWARDS

- The agent always learns to maximize its reward.
- Rewards should be designed to achieve the goal.
- The reward signal is not the place to impart to the agent prior knowledge about how to achieve what we want it to do.
  - In Chess reward should be given to only for actual winning not for taking opponent pieces.
  - Else it might find a way to take the opponent's pieces even at the cost of losing the game.
- The reward signal is your way of communicating to the robot what you want it to achieve, not how you want it achieved.

GOALS AND REWARDS

- Is a scalar reward signal an adequate notion of a goal?
    - maybe not, but it is surprisingly flexible.
- A goal should specify what we want to achieve, not how we want to achieve it.
- A goal must be outside the agent's direct control
    - thus outside the agent.
- The agent must be able to measure success:
    - explicitly;
    - frequently during its lifespan.

RETURNS AND EPISODES

- Suppose the sequence of rewards after step t is

$$r_{t+1}, r_{t+2}, r_{t+3}, \dots$$

- What do we want to maximize?
- Episodic tasks: Interaction breaks naturally into episodes, e.g., plays of a game, trips through a maze.
- In general, we want to maximize the expected return $E\{R_t\}$ for each step t

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

- where T is a final time step at which a terminal state is reached, ending an episode.

RETURNS AND EPISODES

- Continuing tasks: Interaction does not have natural episodes. Here terminal state time T is infinite.
- Discounter Return: The agent tries to select actions so that the sum of the discounted rewards it receives over the future is maximized.

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- Where γ is a parameter $0 \leq \gamma \leq 1$, called as discount rate.
- It determines the The discount rate determines the present value of future rewards: a reward received k time steps in the future is worth only γ^{k-1} times what it would be worth if it were received immediately.

$$\text{shortsighted } 0 \leftarrow \gamma \rightarrow 1 \text{ farsighted}$$

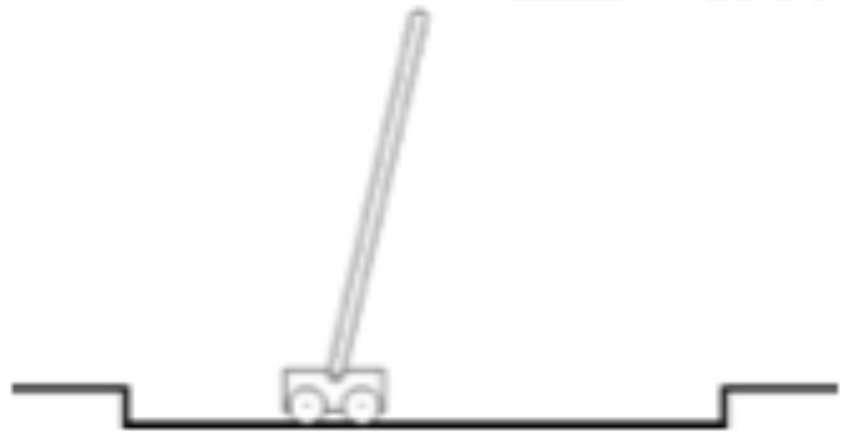
RETURNS AND EPISODES

- Returns at successive time steps are related to each other in a way that is important for the theory and algorithms of reinforcement learning

$$\begin{aligned} G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \dots \\ &\doteq R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \dots) \\ &\doteq R_{t+1} + \gamma G_{t+1} \end{aligned}$$

- Note that this works for all time steps $t < T$, even if termination occurs at $t + 1$, if we define $G_T = 0$.

RETURNS AND EPISODES



Avoid **failure**: the pole falling beyond a critical angle or the cart hitting end of track.

As an **episodic task** where episode ends upon failure:

$$\begin{aligned} \text{reward} &= +1 \text{ for each step before failure} \\ \Rightarrow \quad \text{return} &= \text{number of steps before failure} \end{aligned}$$

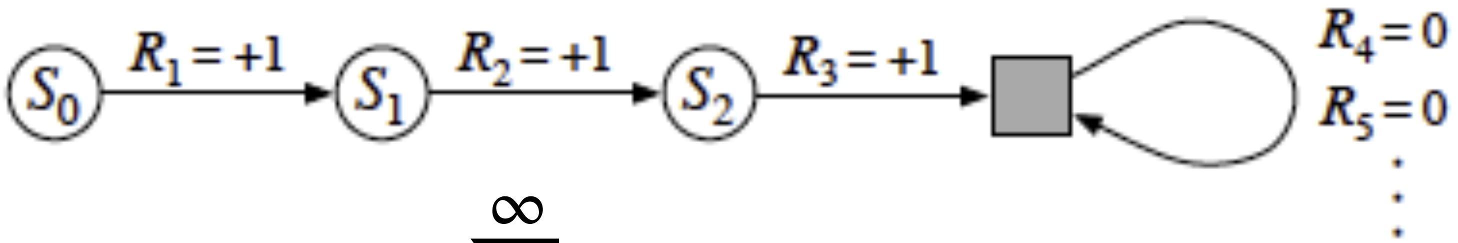
As a **continuing task** with discounted return:

$$\begin{aligned} \text{reward} &= -1 \text{ upon failure; } 0 \text{ otherwise} \\ \Rightarrow \quad \text{return} &= -\gamma^{k}, \text{ for } k \text{ steps before failure} \end{aligned}$$

In either case, return is maximized by avoiding failure for as long as possible.

UNIFIED NOTION

- In episodic tasks, we number the time steps of each episode starting from zero.
- We usually do not have distinguish between episodes, so we write s_t instead of $s_{t,j}$ for the state at step t of episode j.
- Think of each episode as ending in an absorbing state that always produces reward of zero:



$$G_t \doteq \sum_{k=0}^{\infty} \gamma^{k-t-1} R_k$$

- Including the possibility that $T = \infty$ or $\gamma = 1$ (but not both)

THE MARKOV PROPERTY

- By "the state" at step t, we mean whatever information is available to the agent at step t about its environment.
- The state can include immediate "sensations," highly processed sensations, and structures built up over time from sequences of sensations.
- Ideally, a state should summarize past sensations so as to retain all "essential" information, i.e., it should have the Markov Property:

$$Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t, r_t, s_{t-1}, a_{t-1}, \dots, r_1, s_0, a_0\} = Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t\}$$

for all s', r, and histories st, at, st-1, at-1, …, r1, s0, a0.

MARKOV DECISION PROCESSES

- If a reinforcement learning task has the Markov Property, it is basically a Markov Decision Process (MDP).
- If state and action sets are finite, it is a finite MDP.
- To define a finite MDP, you need to give:
  - state and action sets
  - one-step "dynamics" defined by transition probabilities:

$$P_{ss'}^{a} = \Pr\left\{s_{t+1} = s' \mid s_t = s, a_t = a\right\} \text{ for all } s, s' \in S,\ a \in A(s).$$

  - reward probabilities:

$$R_{ss'}^{a} = E\left\{r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'\right\} \text{ for all } s, s' \in S,\ a \in A(s).$$

THANK YOU !!!