Reinforcement Learning

Juan Leon Alcazar

Reinforcement Learning

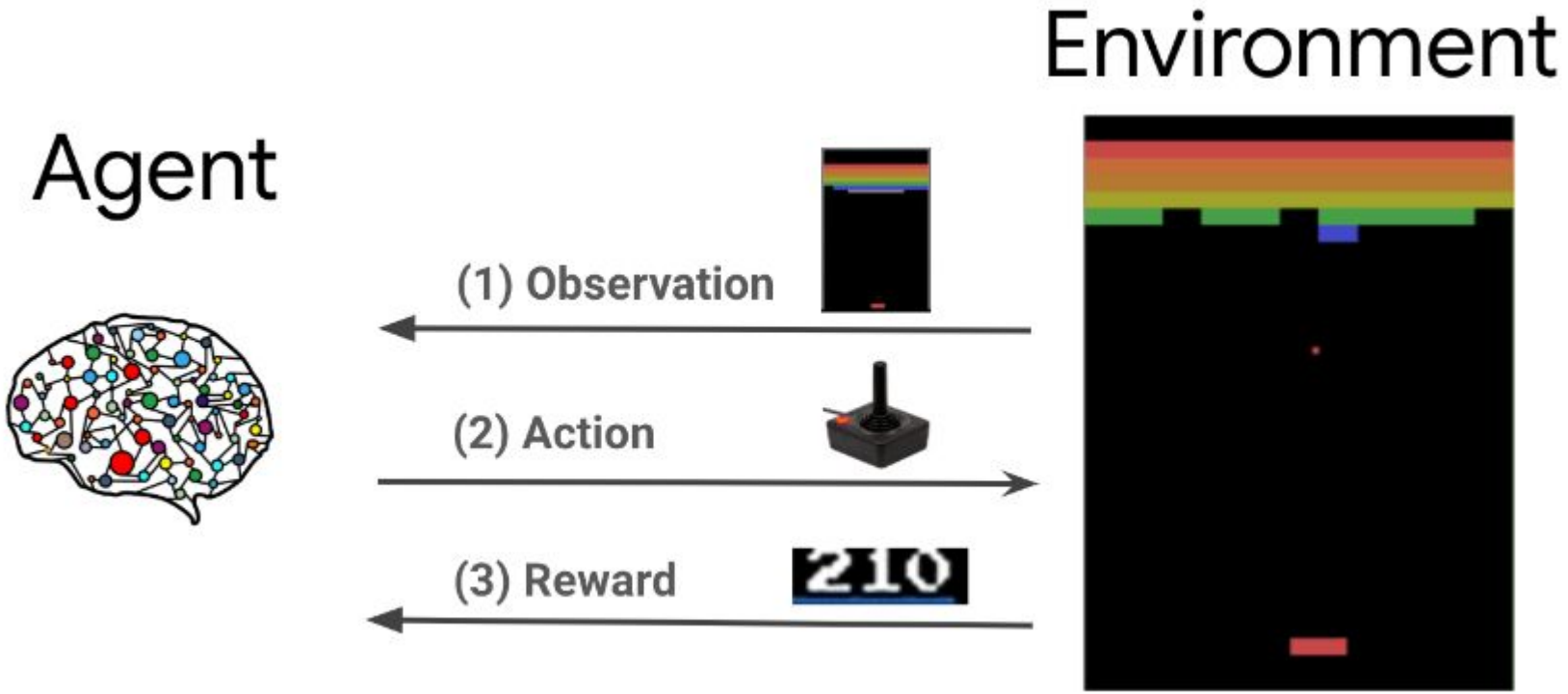
Reinforcement learning allows an agent to learn how to behave in an environment by “trial and error”. The agent starts by taking random actions and observing the results of those actions.

The agent receives feedback that can be either positive or negative.

- Positive feedback **encourages** the agent to take a particular action.
- Negative feedback **discourages** the agent from taking a particular action.

Breaking Away from Labeled Data

- There is no labeled data, so the agent learn by interacting with the environment.
- Decision making is sequential (one decision per time-step), but **the goal is typically long-term**.
- The agent interacts with the environment and explores it by itself.
- The agent learns by **trial and error**.



https://www.tensorflow.org/agents/tutorials/0_intro_rl

Policy Based RL

The agent and environment continuously interact with each other. At each time step, the agent takes an action on the environment based on its policy π.

Here π depends on the observation S and generates an action a.

$$a = \pi(s)$$

After an action the agent receives a reward *r* and the next observation *s* from the environment.

The goal is to **improve the policy π(s)** so as to maximize the sum of rewards (return).

Types of Policies in Reinforcement Learning

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.

For deterministic policy: $a = \pi(s)$

For stochastic policy: $\pi(a \mid s) = P[A_t = a \mid S_t = s]$

Q Learning

Q-learning lets the agent use the environment's rewards to learn the best action to take in a given state.

The “Q” stands for quality. Quality represents how valuable the action is in maximizing future rewards. Policy-based methods train the policy directly to learn which action to take in a given state.

The agent will use a **Q-table** to take the best possible action based on the expected reward for each state in the environment.

The Expected Return.

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T,$$

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \\ &= \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \end{aligned}$$

The goal of the agent is to maximize the cumulative rewards.

We need to calculate this cumulative reward over a very long interval [t,..., T]

Therefore we must include a scaling factor γ

We can call this the **Expected Return**.

Action Value function

$$q_\pi(s,a) = E_\pi[G_t \mid S_t = s, A_t = a]$$
$$= E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right].$$

Determines the goodness of the action taken by the agent from a given *s* state for policy π.

This function gives the value which is the expected return starting from state *s* at time step *t*, with action *a*, and following policy π afterward.

Bellman Equation

$$q_*(s,a) = E\left[R_{t+1} + \gamma \max_{a'} q_*(s',a')\right]$$

https://www.analyticsvidhya.com/blog/2021/02/understanding-the-bellman-optimality-equation-in-reinforcement-learning/

At time t, for any state-action pair (s, a), the expected return from starting state s, taking action a, and with the optimal policy π will be equal to the **expected reward Rt+1**, in addition with **the maximum of "expected discounted return"** that is achievable of any (s′, a′). (s′, a′) is a potential next state-action pair

Bellman Equation - Intuitions

Immediate gratification vs. long-term benefits: Getting a small reward now ise nice, but ultimately, you want to find the biggest possible reward.

Discounting for uncertainty: Since the future is uncertain, rewards further down the line are less guaranteed. The discount factor (γ) helps account for this by decreasing the importance of rewards received further in the future

Building value iteratively: We don't know the value of every state beforehand. The Bellman equation allows us to start with an initial guess for the value of each state and then iteratively update them.

Q* Values

q∗(s′, a′) denotes the expected return after choosing an action *a* in state *s* which is then maximized to gain the optimal Q-value.

The entire Q-learning algorithm is summarized by performing actions updating the reward and updating the Q-Table.