Deep reinforcement learning

Types of Learning

We can categorize three types of learning procedures:

1. Supervised Learning:

$$\mathbf{y} = f(\mathbf{x})$$

Predict label y corresponding to observation x

2. Unsupervised Learning:

$$f(\mathbf{x})$$

Estimate the distribution of observation x

3. Reinforcement Learning (RL):

$$\mathbf{y} = f(\mathbf{x})$$

$$\mathbf{z}$$

Predict action y based on observation x, to maximize a future reward z

Reinforcement Learning (RL)

RL:

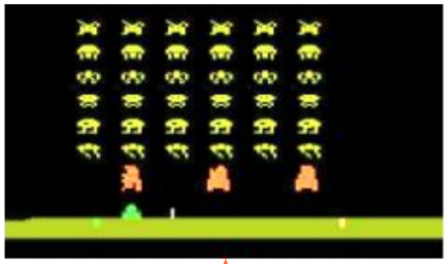
“a way of programming agents by reward and punishment without needing to specify how the task is to be achieved”

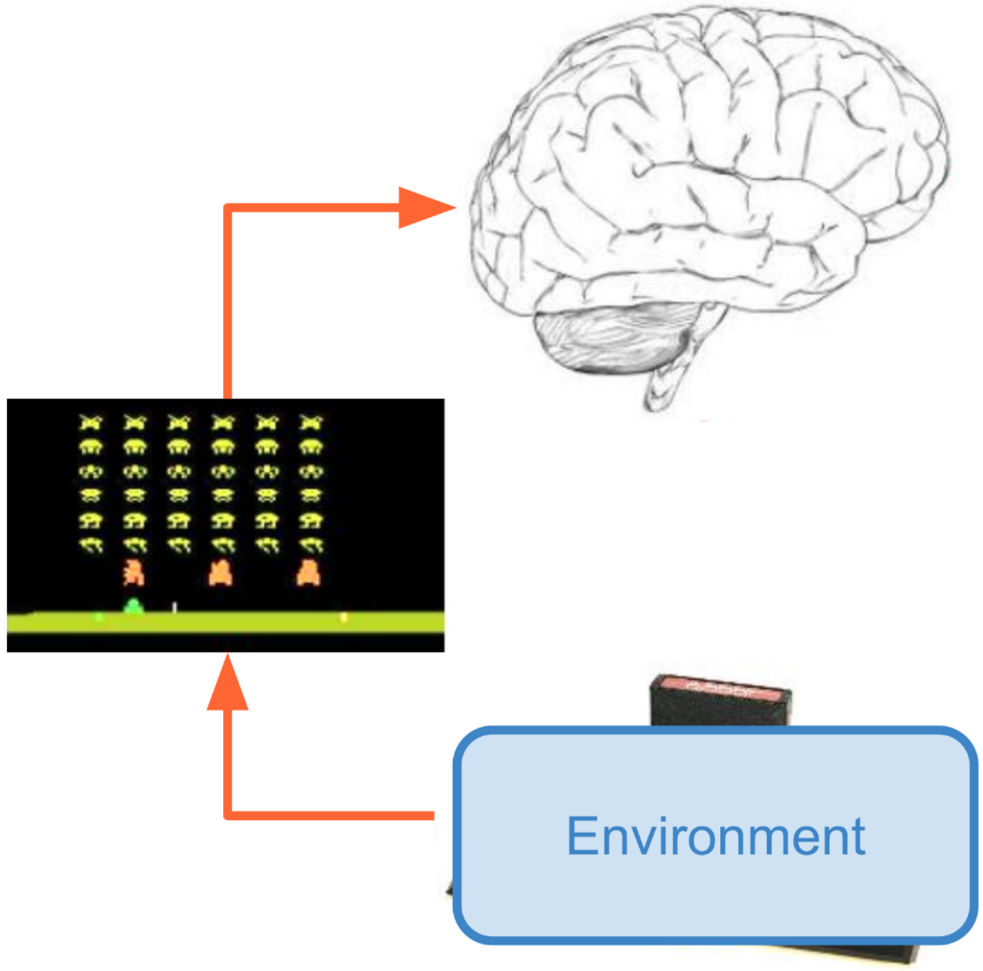
Atari Games

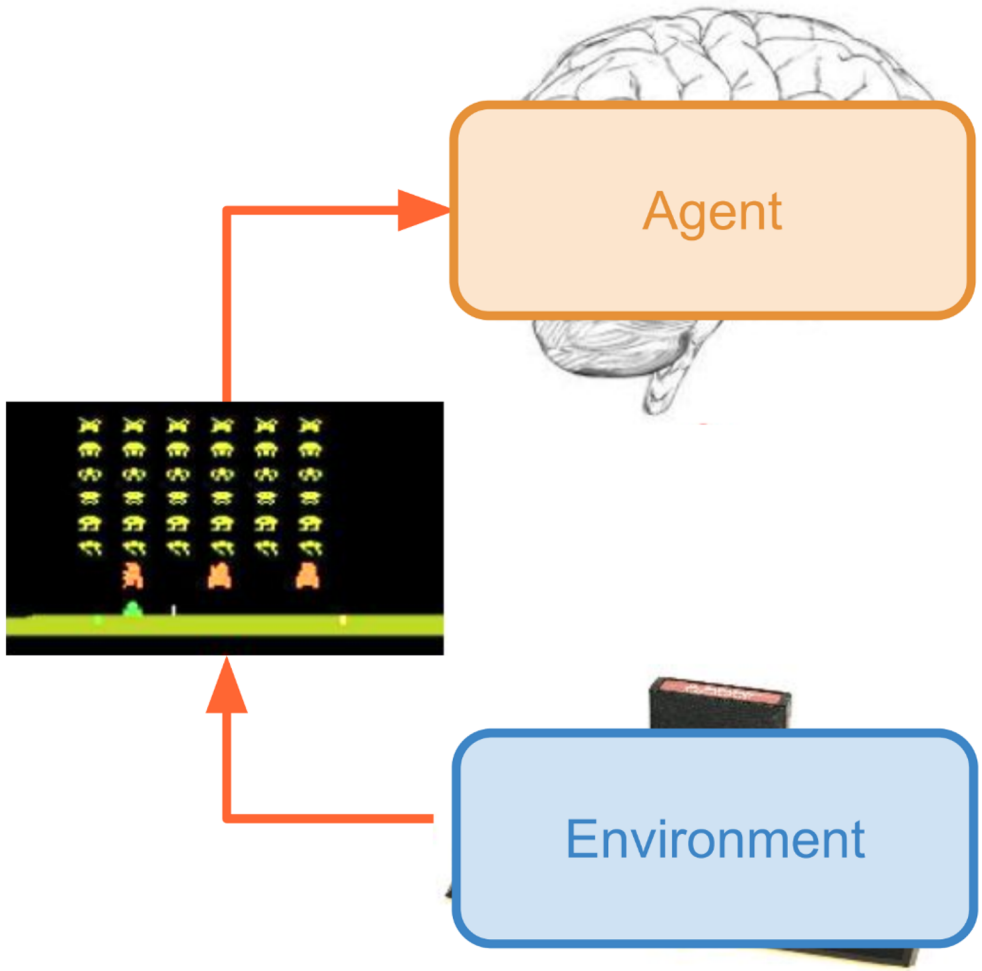

ATARI

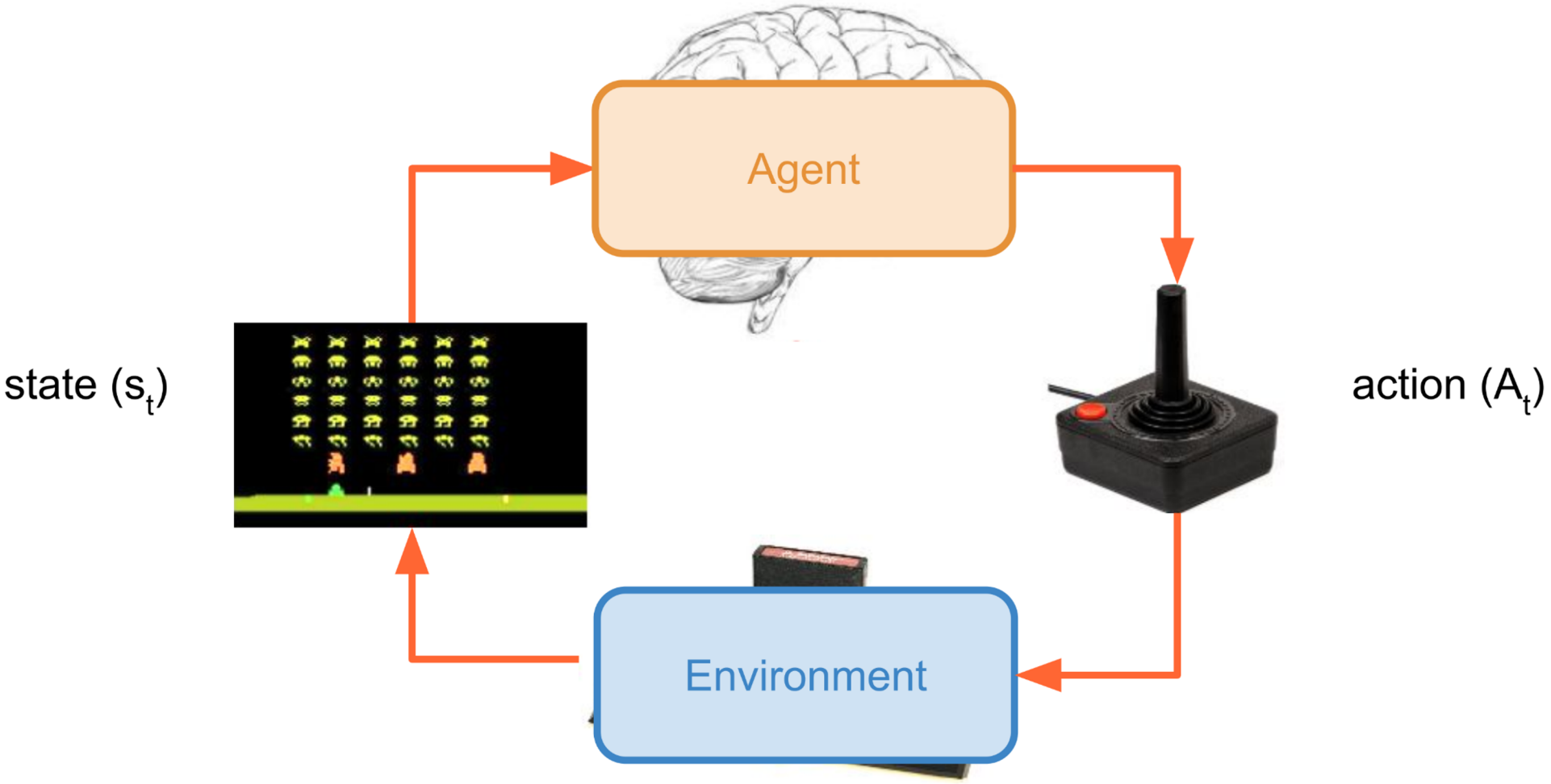

Environment

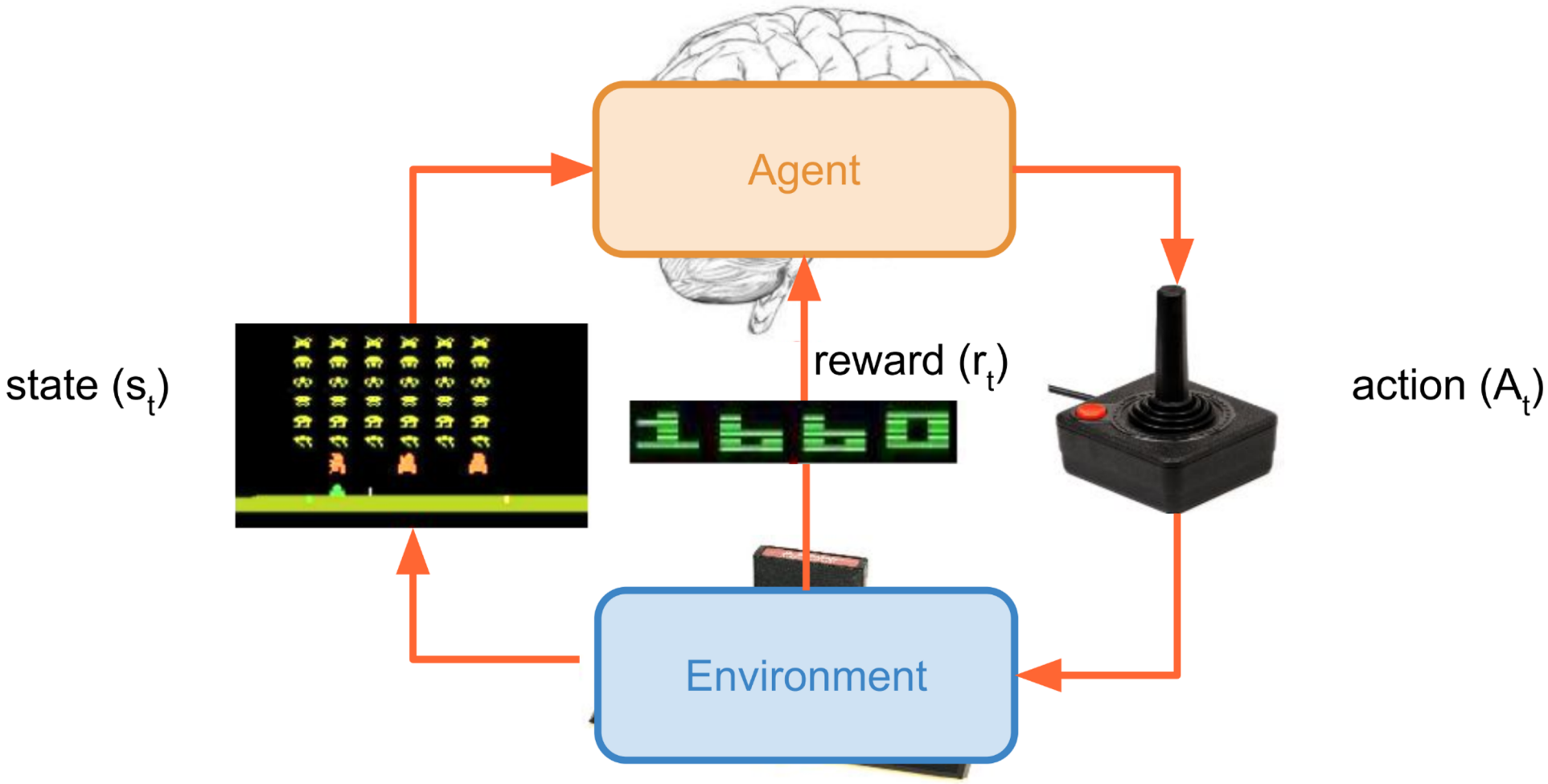
state (s_t)

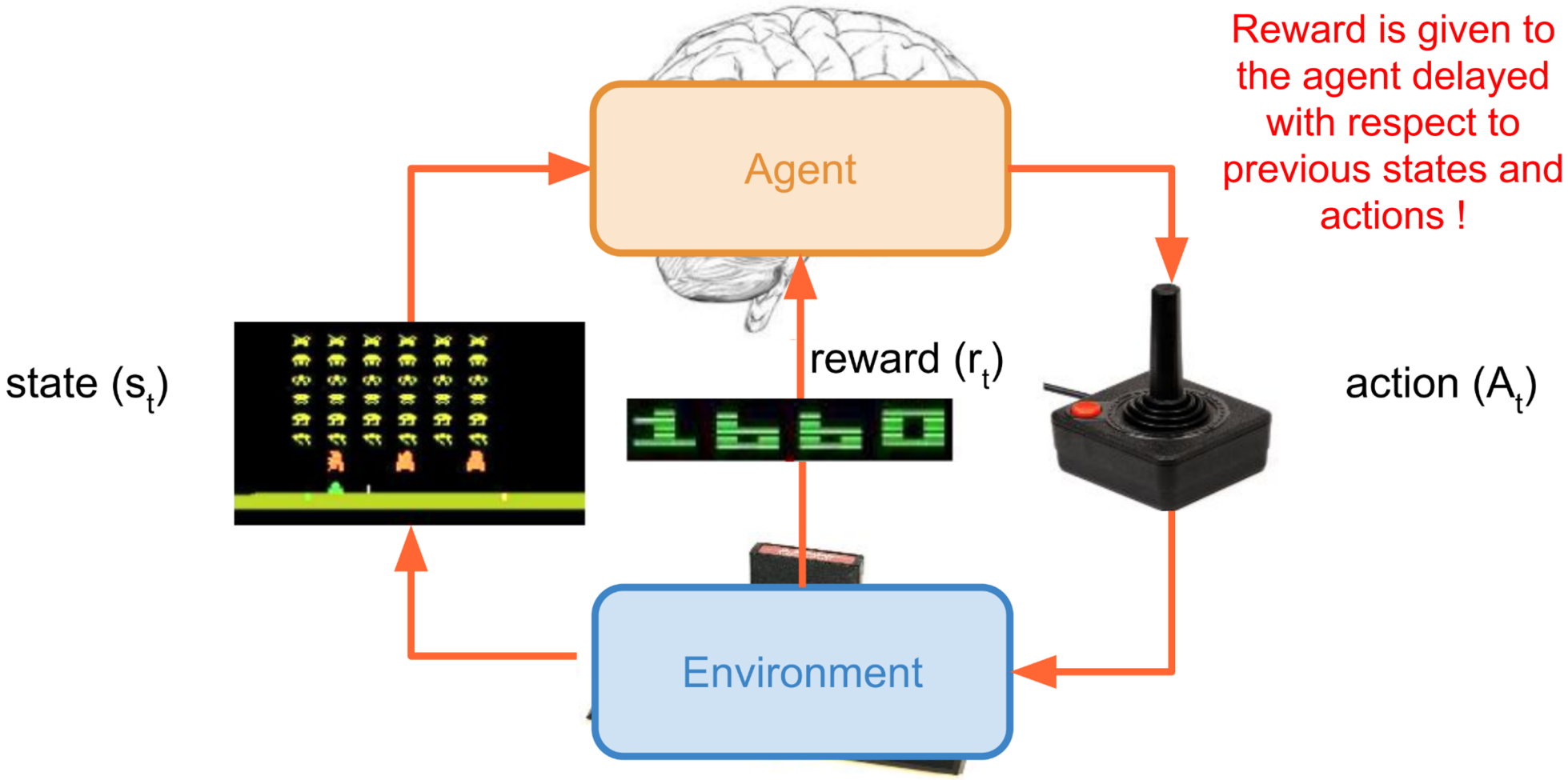


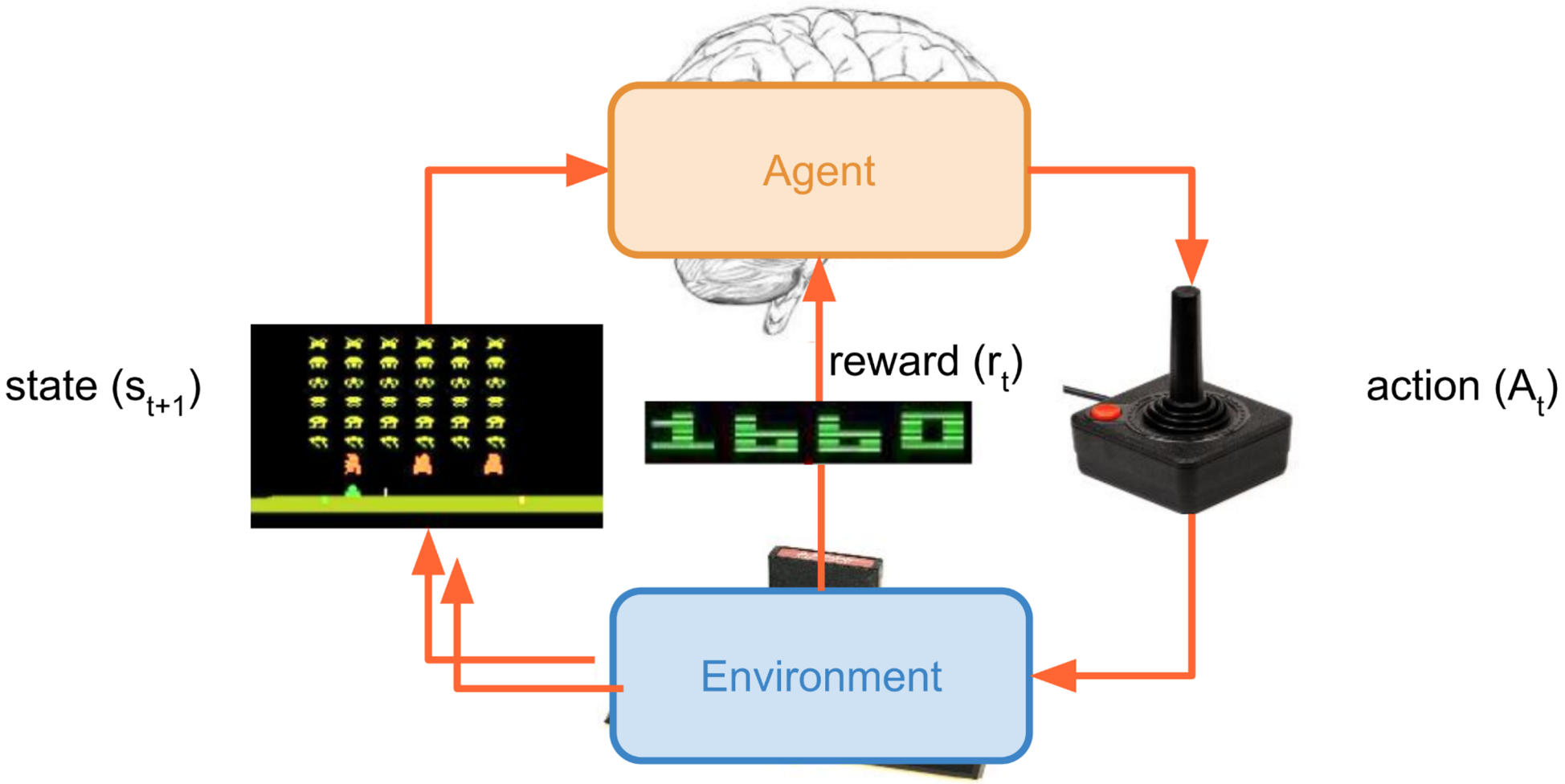

state (s_t)
Environment

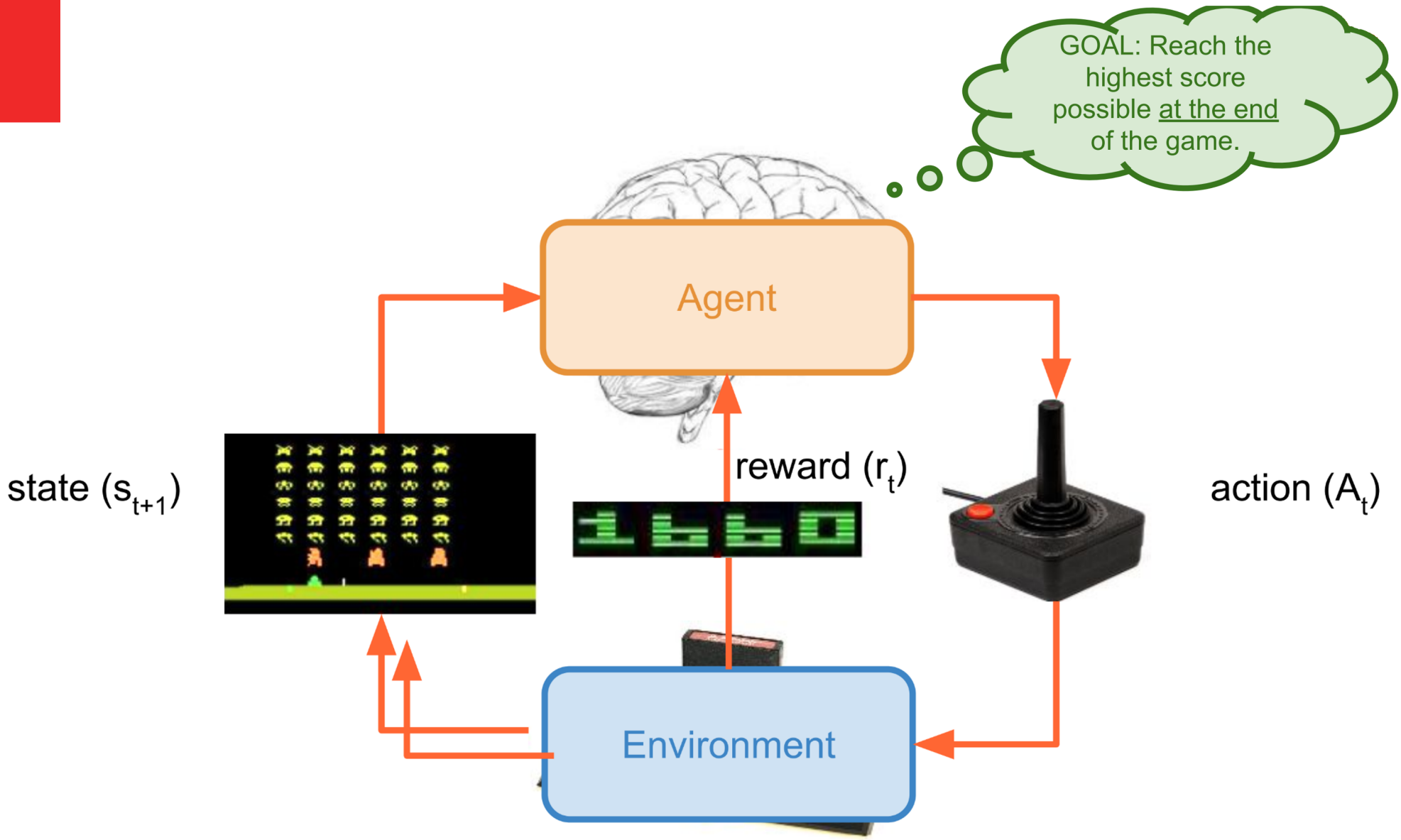

Agent
state (s_t)
Environment

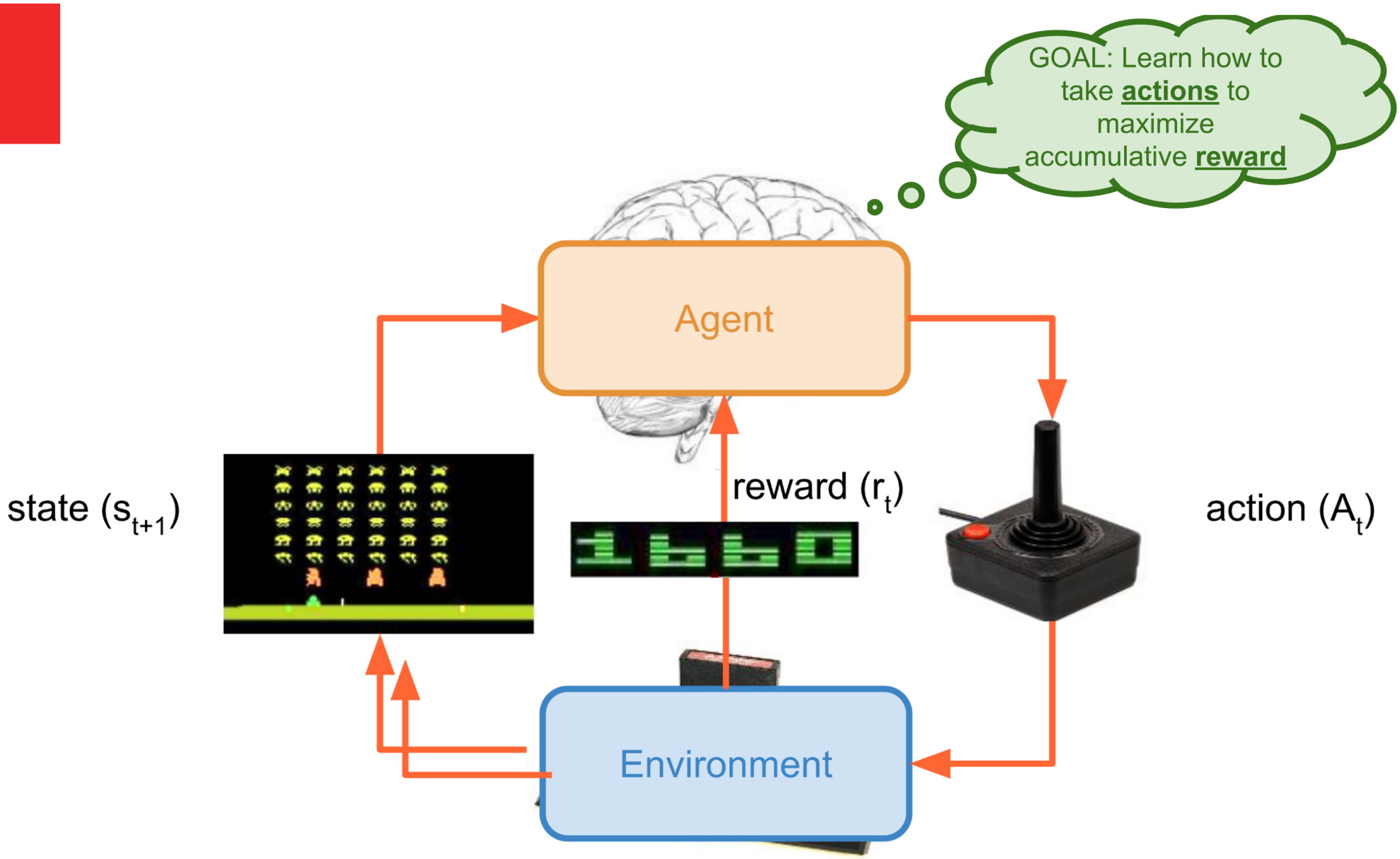

Agent
state (s_t)
action (A_t)
Environment

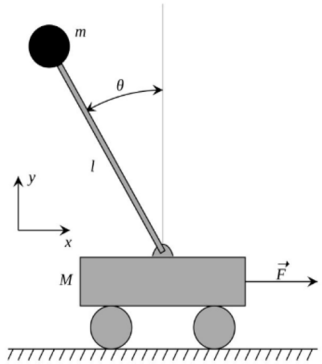

Agent
state (s_t)
reward (r_t)
action (A_t)
Environment


Agent
Environment
state (s_t)
reward (r_t)
action (A_t)
Reward is given to the agent delayed with respect to previous states and actions !


Agent
reward (r_t)
state (s_{t+1})
action (A_t)
Environment


GOAL: Reach the highest score possible at the end of the game.
Agent
state (s_{t+1})
reward (r_t)
action (A_t)
Environment


GOAL: Learn how to take actions to maximize accumulative reward
Agent
state (s_{t+1})
reward (r_t)
action (A_t)
Environment



Agent

state (s_t)

Angle
Angular speed
Position
Horizontal velocity

reward (r_t)

1 at each time step if the pole is upright

action (A_t)

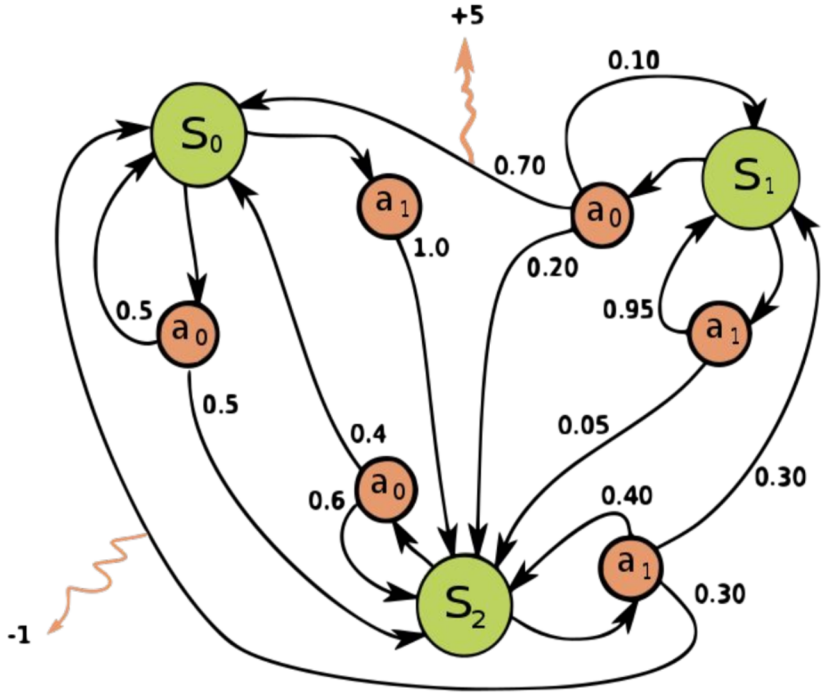
Horizontal force applied in the car

Environment

Markov Decision Process (MDP)

Markov Decision Processes provide a formalism for reinforcement learning problems.

Markov property:
Current state completely characterises the state of the world.

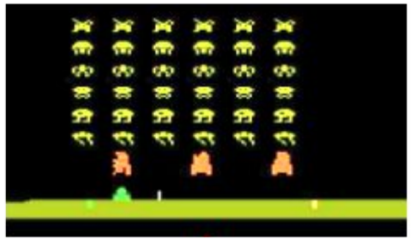


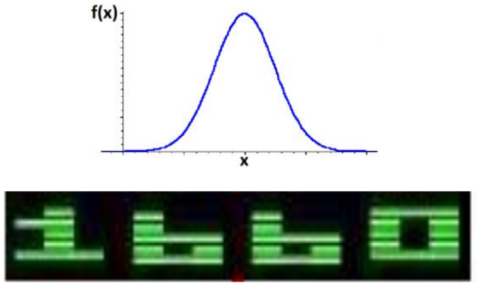
MDP

Defined by: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{P}, \gamma)$

- $\mathcal{S}$: set of possible states
- $\mathcal{A}$: set of possible actions
- $\mathcal{R}$: distribution of reward given (state, action) pair
- $\mathbb{P}$: transition probability distribution over next state given (state, action) pair
- γ : discount factor

MDP

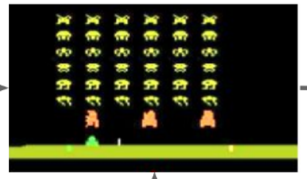




S A R P ɣ

Environment samples initial state $s_0 \sim p(s_0)$

state (s_t)

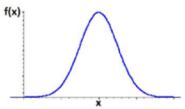


Agent selects action a_t

action (a_t)

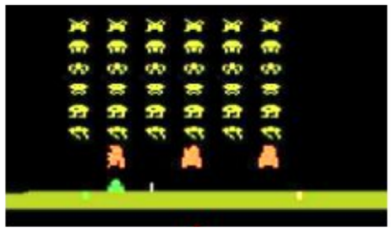
Environment samples reward $r_t \sim R(. \mid s_t, a_t)$

reward (r_t)



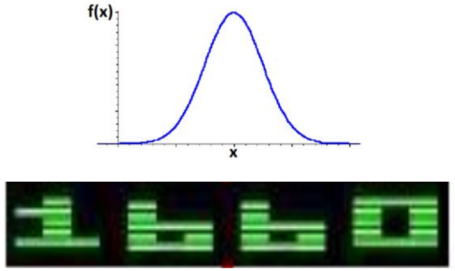
Environment samples next state $s_{t+1} \sim P(. \mid s_t, a_t)$

Action Policy π



S

A



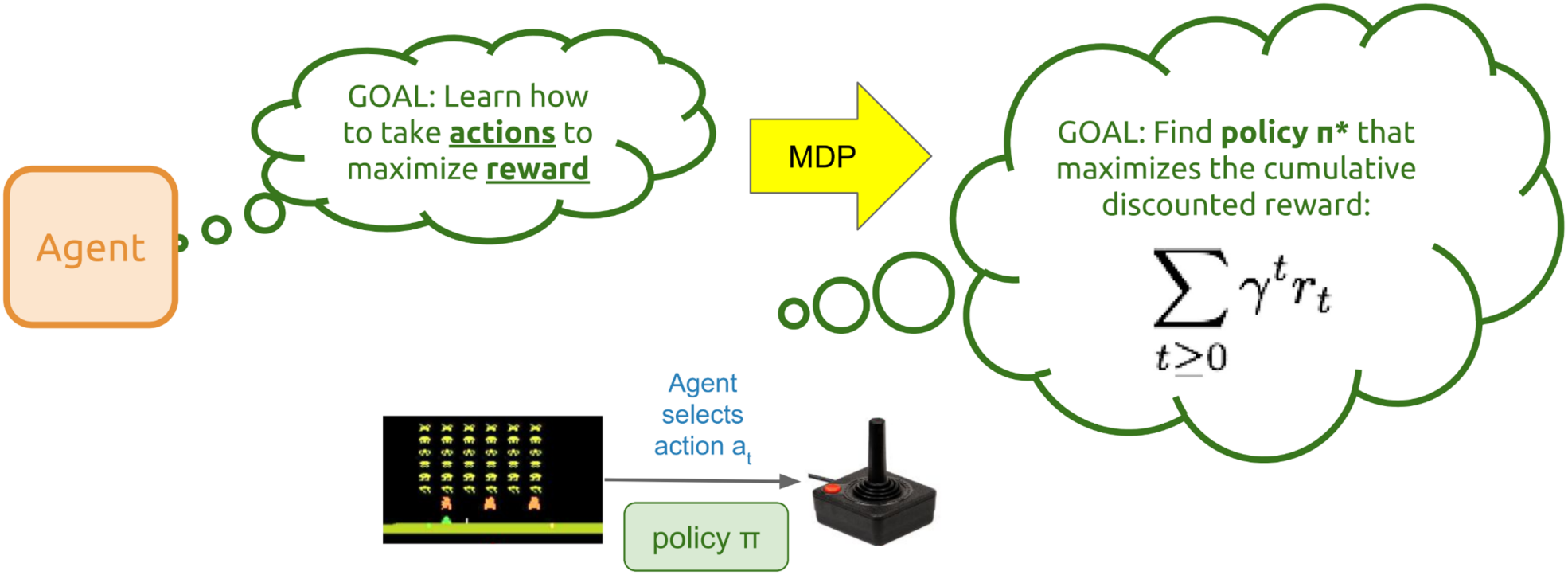
R

P

ɣ

A **Policy π** is a function S → A that specifies which action to take in each state.

Optimal policy

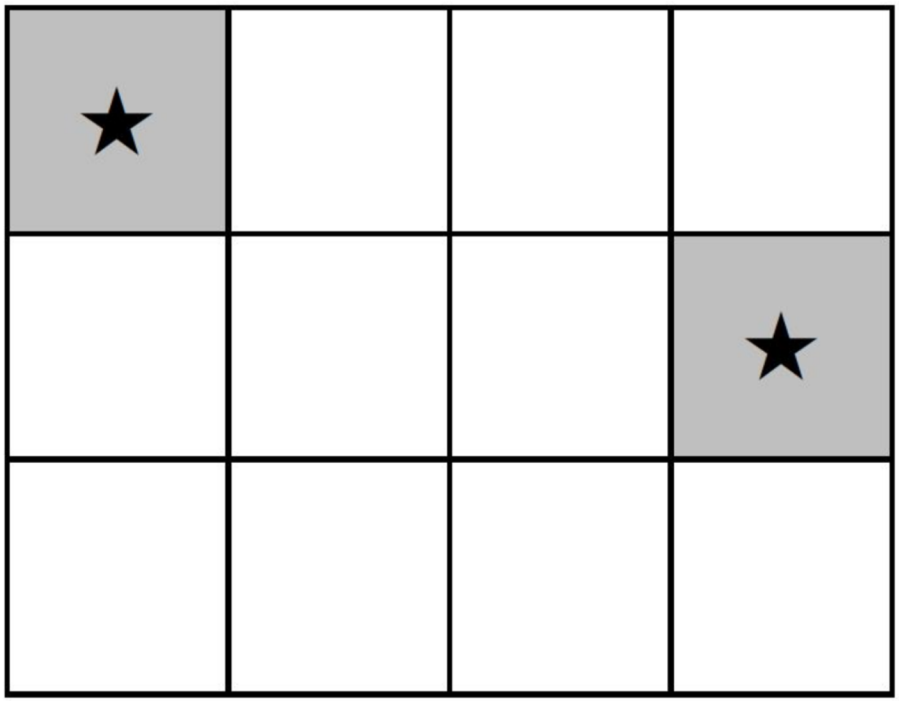


A **Policy π** is a function S → A that specifies which action to take in each state.

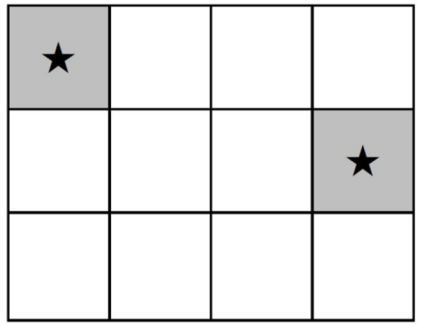
Policy example

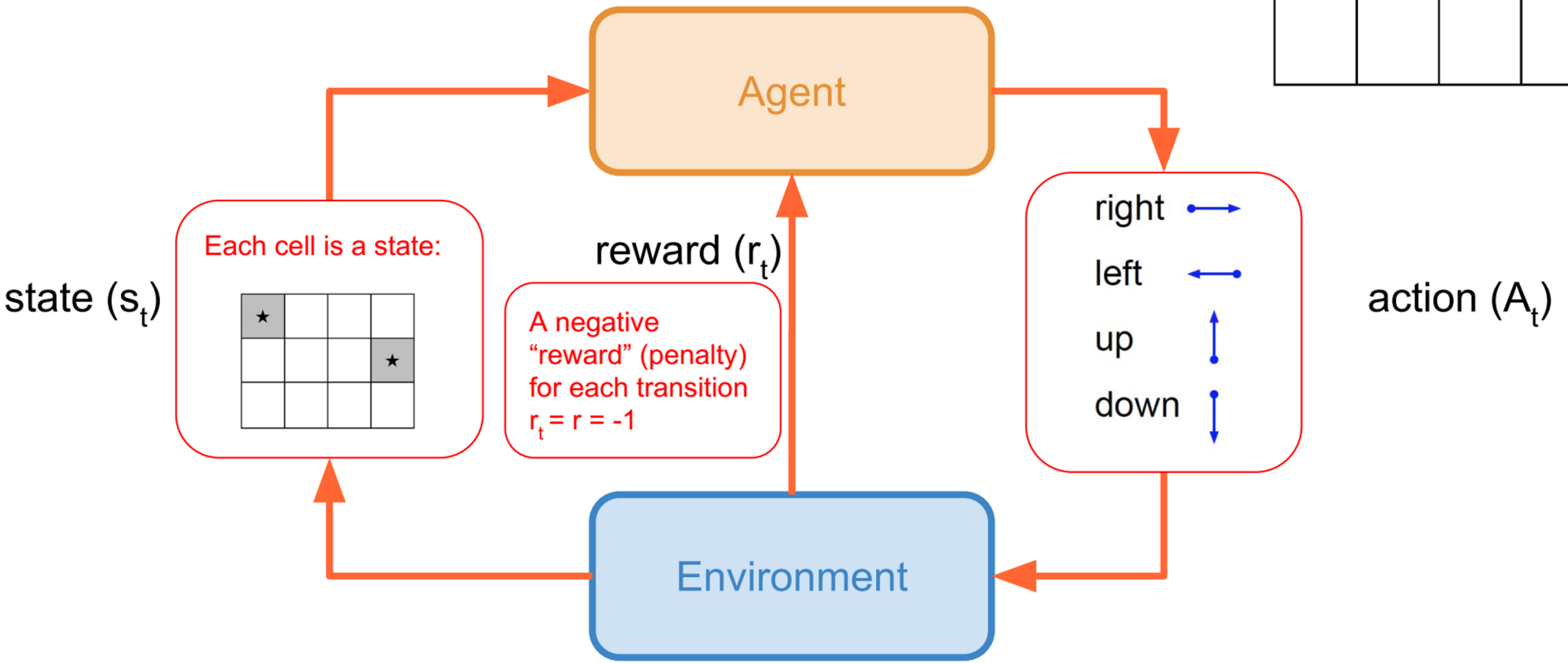
Multiple problems that can be formulated with a RL architecture.

Grid World (a simple MDP)
Objective: reach one of the terminal states (greyed out) in least number of actions.



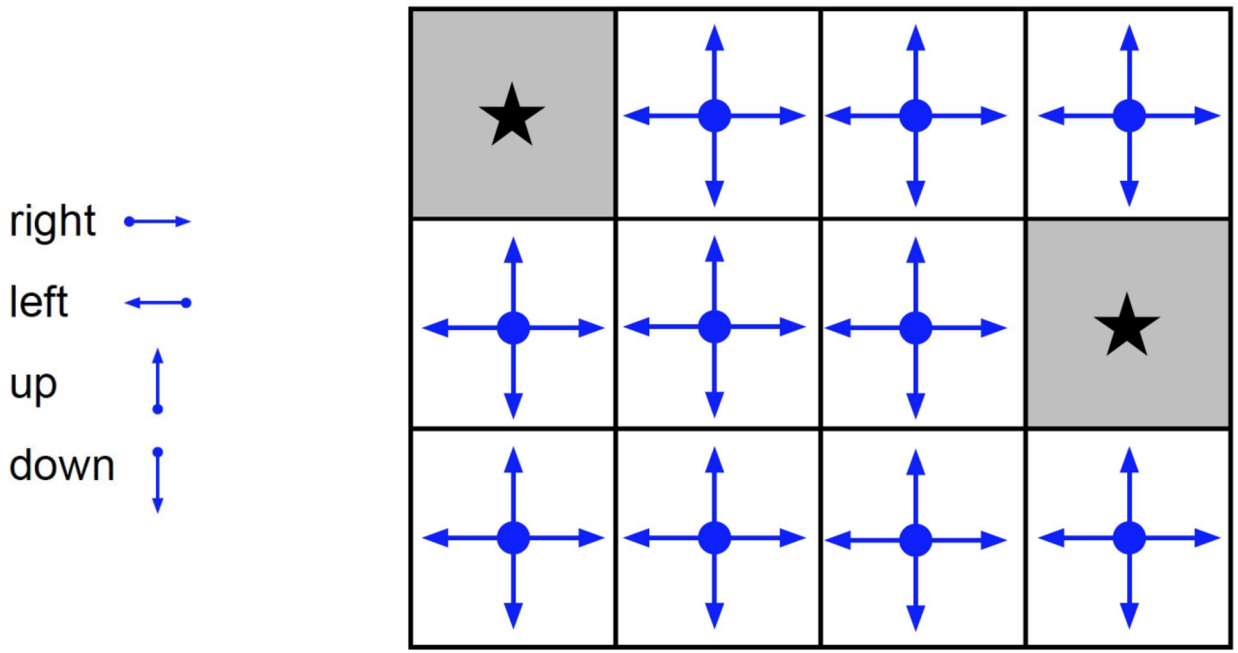
Policy example





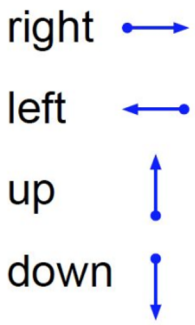
Policy example

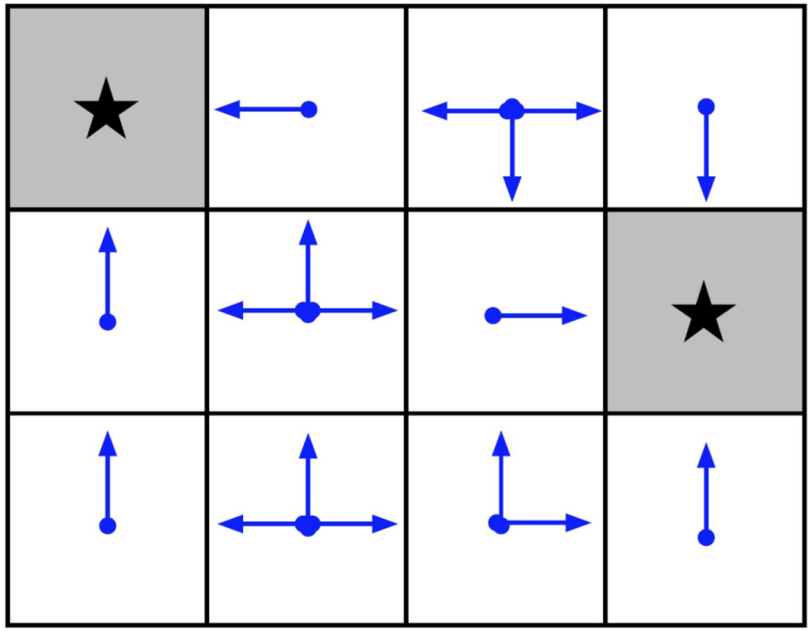
Example: Actions resulting from applying a **random policy** on this Grid World problem.



Policy example

Optimal policy





The optimal policy

The **optimal policy π*** will maximize the expected sum of rewards:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | \pi\right] \quad \text{with} \quad s_0 \sim p(s_0), a_t \sim \pi(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t, a_t)$$

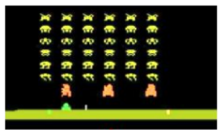
- $\mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | \pi\right]$: expected cumulative discounted reward
- $s_0 \sim p(s_0)$: initial state
- $a_t \sim \pi(\cdot|s_t)$: selected action at t
- $s_{t+1} \sim p(\cdot|s_t, a_t)$: sampled state for t+1

The optimal policy

How to estimate how good state *s* is for a given policy π ?

With the **value function at state s, $V^{\pi}(s)$,** the expected cumulative reward from following policy π from state s.

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, \pi\right]$$



"Expected cumulative reward…""

"...from following policy π from state s."

Q-value function

How to estimate how good a state-action pair (s,a) is for a given policy π ?

With the **Q-value function at state *s* and action *a*, $Q^{\pi}(s,a)$,** the expected cumulative reward from taking action *a* in state *s*, and then following policy π.

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi\right]$$

"Expected cumulative reward…""

"...from taking action a in state s and then following policy π."

Optimal Q-value function

(From the previous page)
Q-value function

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{t\geq 0}\gamma^{t} r_t | s_0 = s, a_0 = a, \pi\right]$$

The **optimal Q-value function at state s and action, $Q^*(s,a)$,** is the **maximum expected cumulative reward** achievable from a given (state, action) pair:

$$Q^{*}(s,a) = \max_{\pi} \mathbb{E}\left[\sum_{t\geq 0}\gamma^{t} r_t | s_0 = s, a_0 = a, \pi\right]$$

choose the **policy** that maximizes the expected cumulative reward

Optimal Q-value function

Q^*(s,a) satisfies the following **Bellman equation:**

$$\underbrace{Q^*(s,a)}_{\text{Maximum expected cumulative reward for considered pair (s,a)}} = \underbrace{\mathbb{E}_{s' \sim \mathcal{E}}}_{\text{Expectation across possible future states s' (randomness)}} \Big[\underbrace{r}_{\text{reward for considered pair (s,a)}} + \underbrace{\gamma}_{\text{discount factor}} \max_{a'} \underbrace{Q^*(s',a')}_{\text{Maximum expected cumulative reward for future pair (s',a')}} \Big| s, a \Big]$$

FUTURE REWARD

Searching for the optimal policy

(From the previous page)
Bellman Equation

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q^*(s',a') | s, a \right]$$

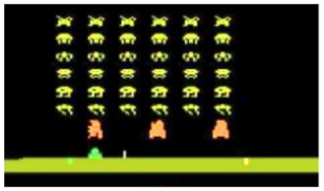
Value iteration algorithm: Estimate the Bellman equation with an iterative update.

$$\underbrace{Q_{i+1}(s,a)}_{\text{Updated Q-value function}} = \mathbb{E} \left[r + \gamma \max_{a'} \underbrace{Q_i(s',a')}_{\text{Current Q-value for future pair (s',a')}} | s, a \right]$$

The iterative estimation $Q_i(s,a)$ will converge to the optimal $Q^*(s,a)$ as $i \rightarrow \infty$.

Deep Q-learning

Exploring all positive states and action is **not scalable**
Eg. If video game, it would require generating all possible pixels and actions.

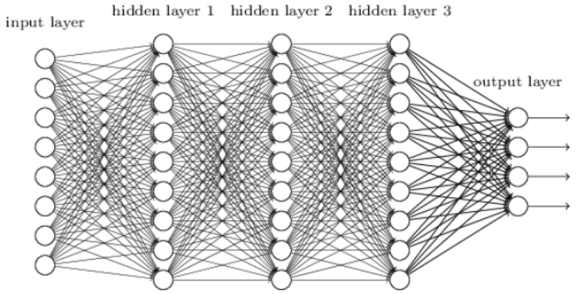


$$Q_{i+1}(s,a) = \mathbb{E}\left[r + \gamma \max_{a'} Q_i(s',a') | s,a\right]$$

Solution: Use a **deep neural network** as an function approximator of Q*(s,a).

$$Q(s,a,\Theta) \approx Q^*(s,a)$$

Neural Network parameters

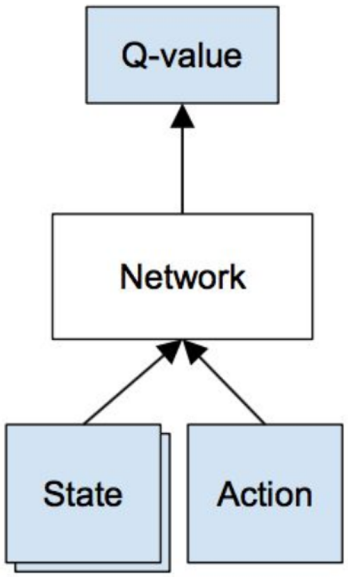


Deep Q-learning

The function to approximate is a Q-function that satisfies the Bellman equation:

$Q(s,a,\Theta) \approx Q^*(s,a)$

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q^*(s',a') | s,a \right]$$



Deep Q-learning

The function to approximate is a Q-function that satisfies the Bellman equation:

Q(s,a,Θ) ≈ Q*(s,a)

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q^*(s',a') | s,a \right]$$

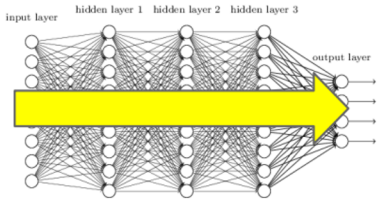
Forward Pass

Sample a (s,a) pair

Predicted Q-value with Θ_i

Loss function:

$$L_i(\theta_i) = \mathbb{E}_{s,a \sim \rho(\cdot)} \left[(y_i - Q(s,a;\theta_i))^2 \right]$$



$$y_i = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) | s,a \right]$$

Sample a future state s'

Predict Q-value with Θ_{i-1}

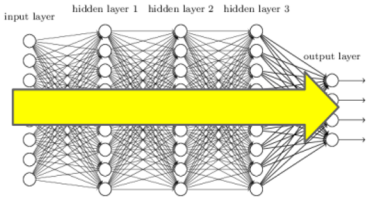
Deep Q-learning

Train the DNN to approximate a Q-value function that satisfies the Bellman equation

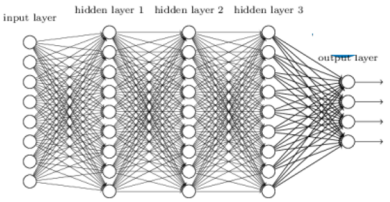
$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q^*(s', a') | s, a \right]$$

$$L_i(\theta_i) = \mathbb{E}_{s,a \sim \rho(\cdot)} \left[(y_i - Q(s, a; \theta_i))^2 \right]$$

$$y_i = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) | s, a \right]$$



Deep Q-learning



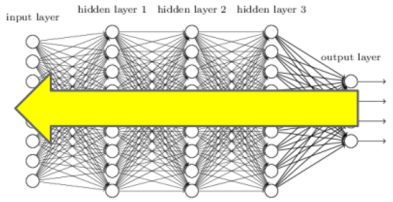
Forward Pass

Loss function: $$L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot)}\left[(y_i - Q(s,a;\theta_i))^2\right]$$

$$y_i = \mathbb{E}_{s'\sim\mathcal{E}}\left[r + \gamma \max_{a'} Q(s',a';\theta_{i-1})|s,a\right]$$

Backward Pass

Gradient update (with respect to Q-function parameters Θ):



$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot);s'\sim\mathcal{E}}\left[r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) - Q(s,a;\theta_i))\nabla_{\theta_i} Q(s,a;\theta_i)\right]$$