Reinforcement learning

1 – An overview of reinforcement learning

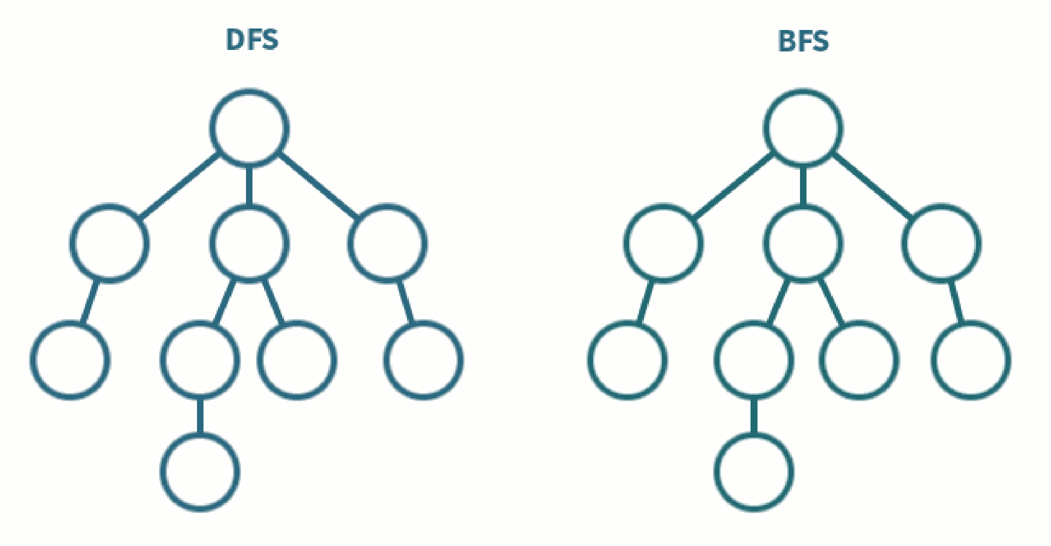
Artificial intelligence

ARTIFICIAL INTELLIGENCE

- A lot of different algorithms

ARTIFICIAL INTELLIGENCE

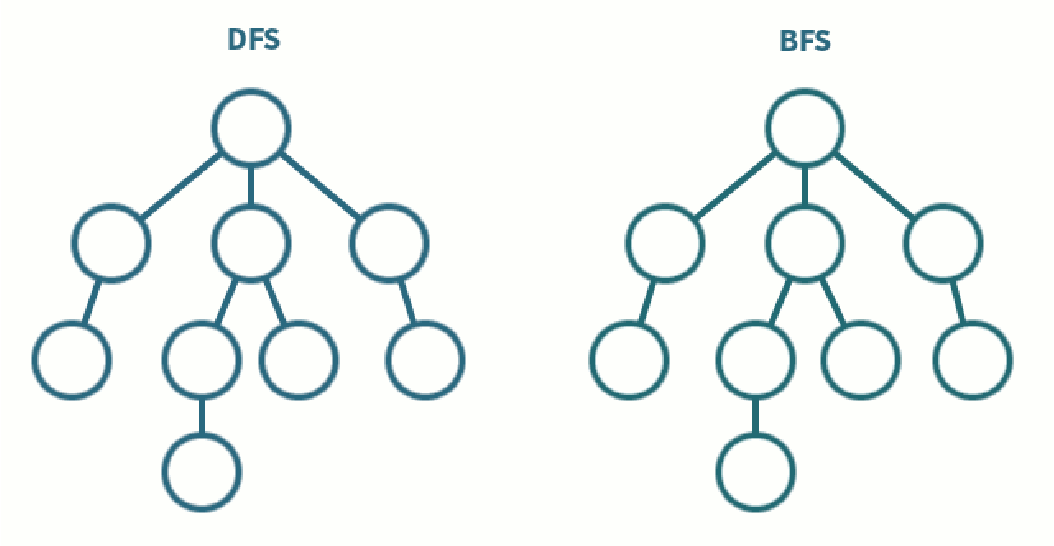
- A lot of different algorithms

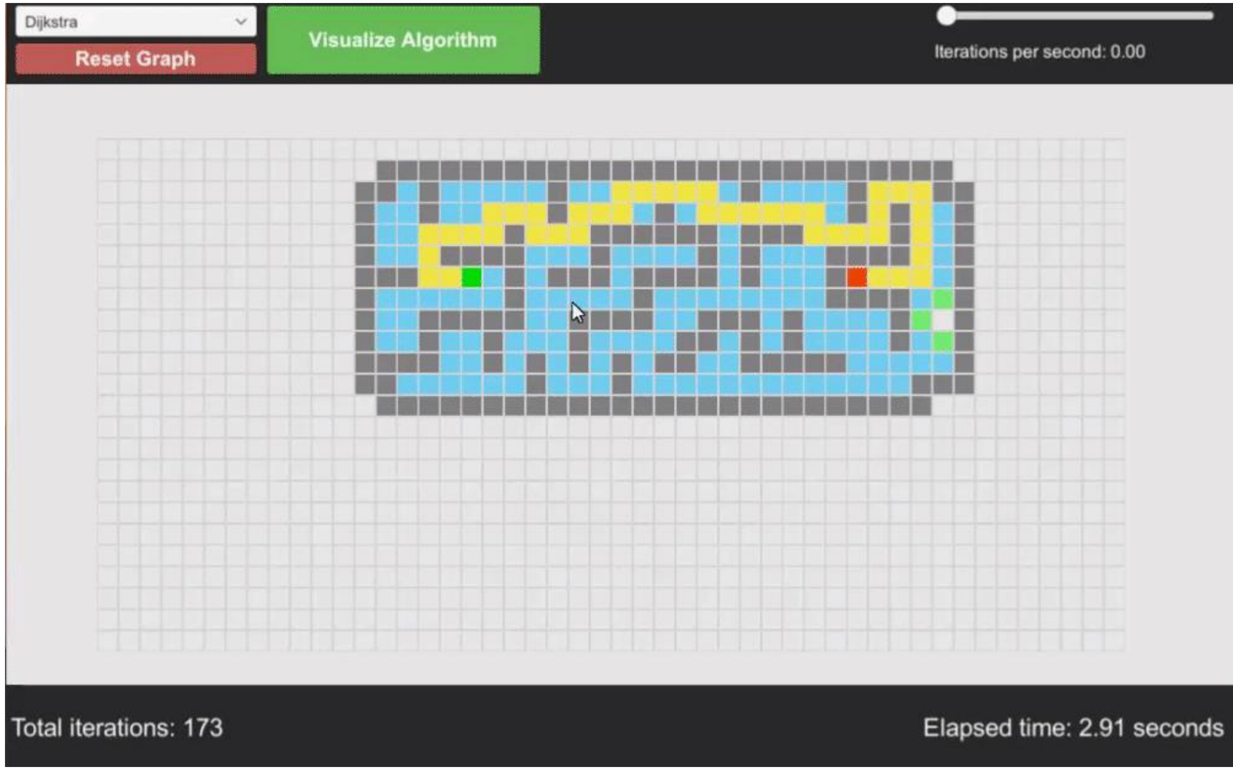


Source : https://gifer.com/en/NUPJ

ARTIFICIAL INTELLIGENCE

- A lot of different algorithms



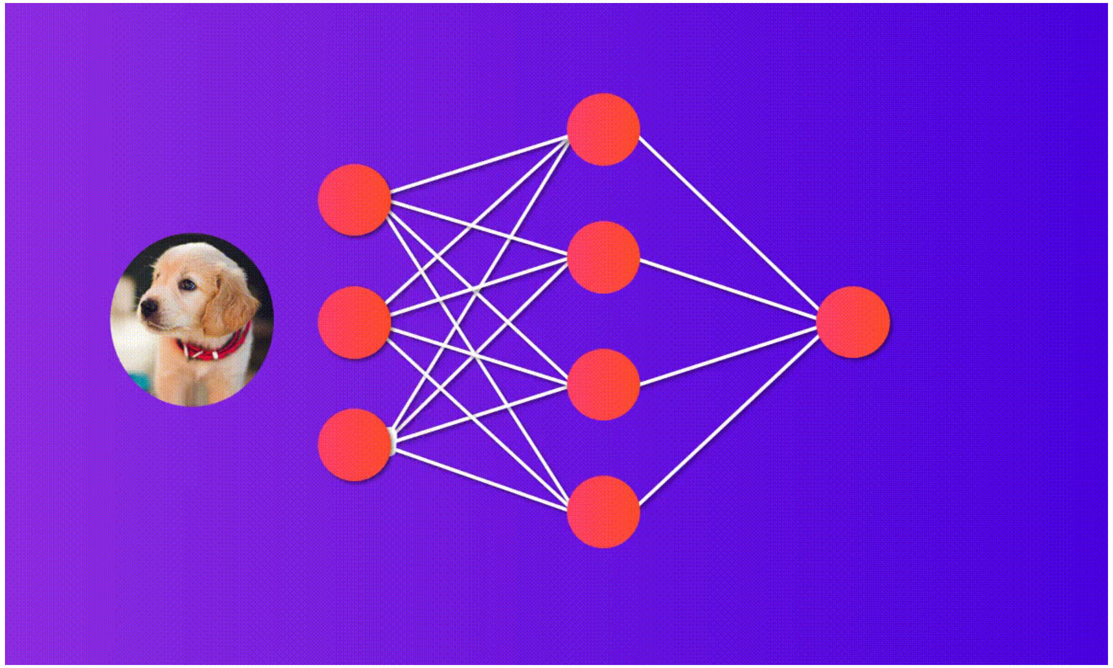


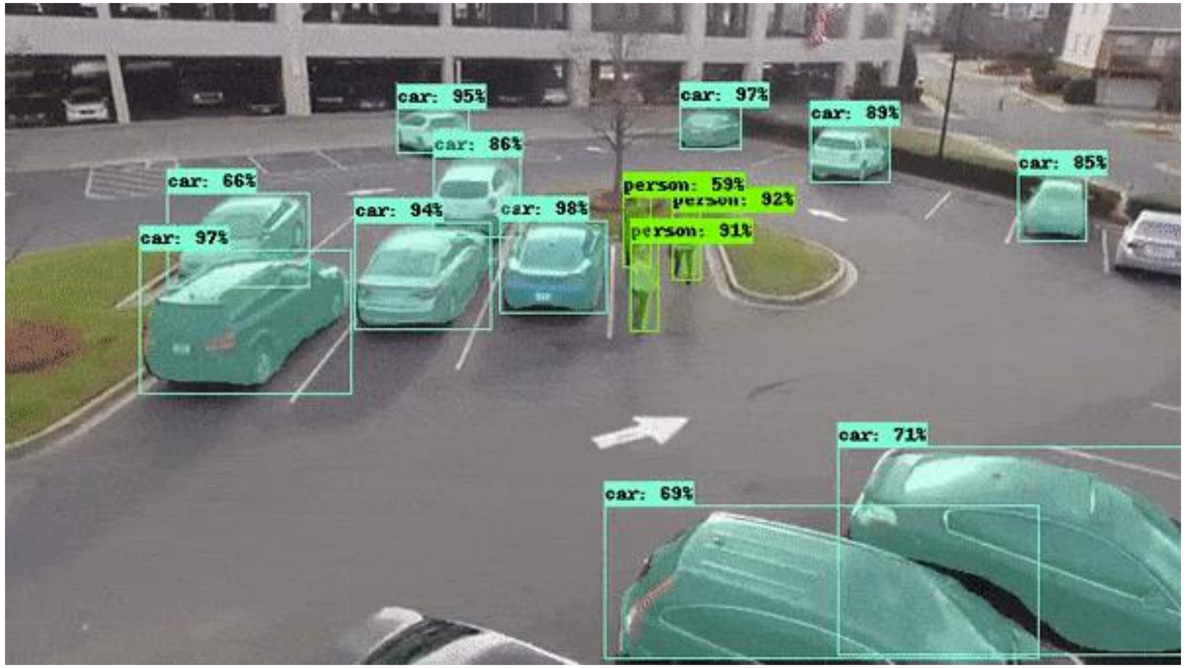
Source : https://gifer.com/en/NUPJ

Source : https://unitylist.com/p/p7z/Unity-Path-Finding

ARTIFICIAL INTELLIGENCE

- A lot of different algorithms

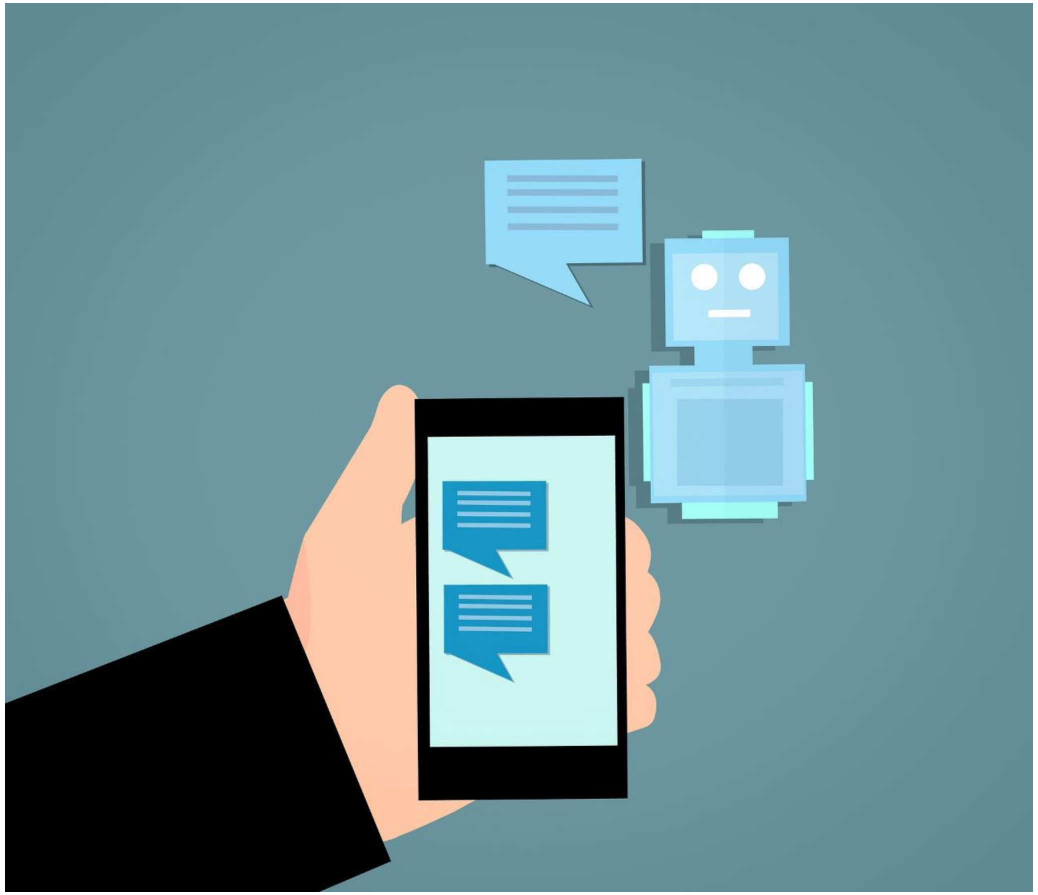




Source: https://towardsdatascience.com/how-to-build-your-own-neural-network-from-scratch-in-python-68998a08e4f6?gi=d3f0903bbda2

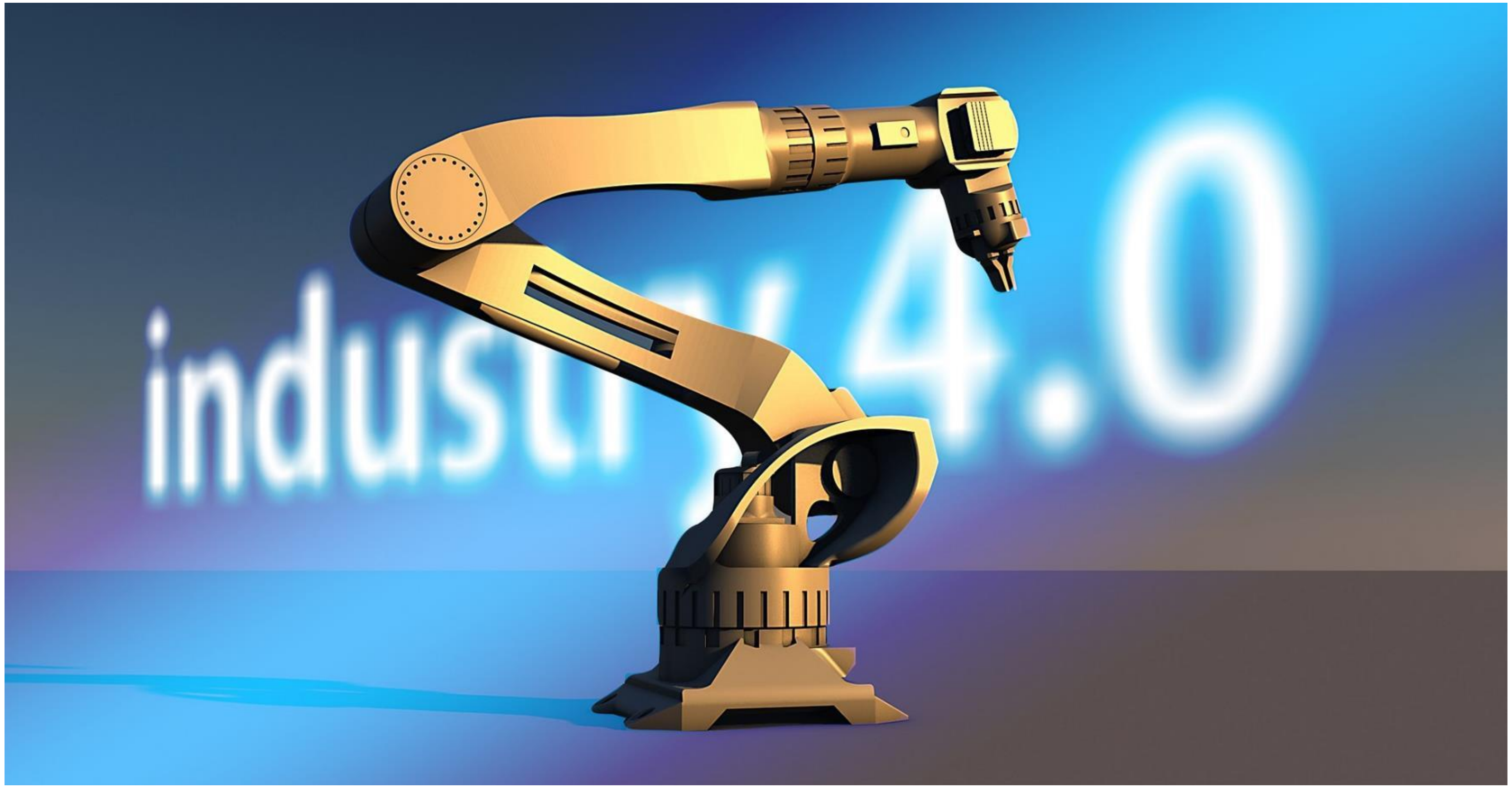
Source: https://medium.com/bitgrit-data-science-publication/5-computer-vision-trends-for-2021-96fd18d5596c

ARTIFICIAL INTELLIGENCE



ARTIFICIAL INTELLIGENCE

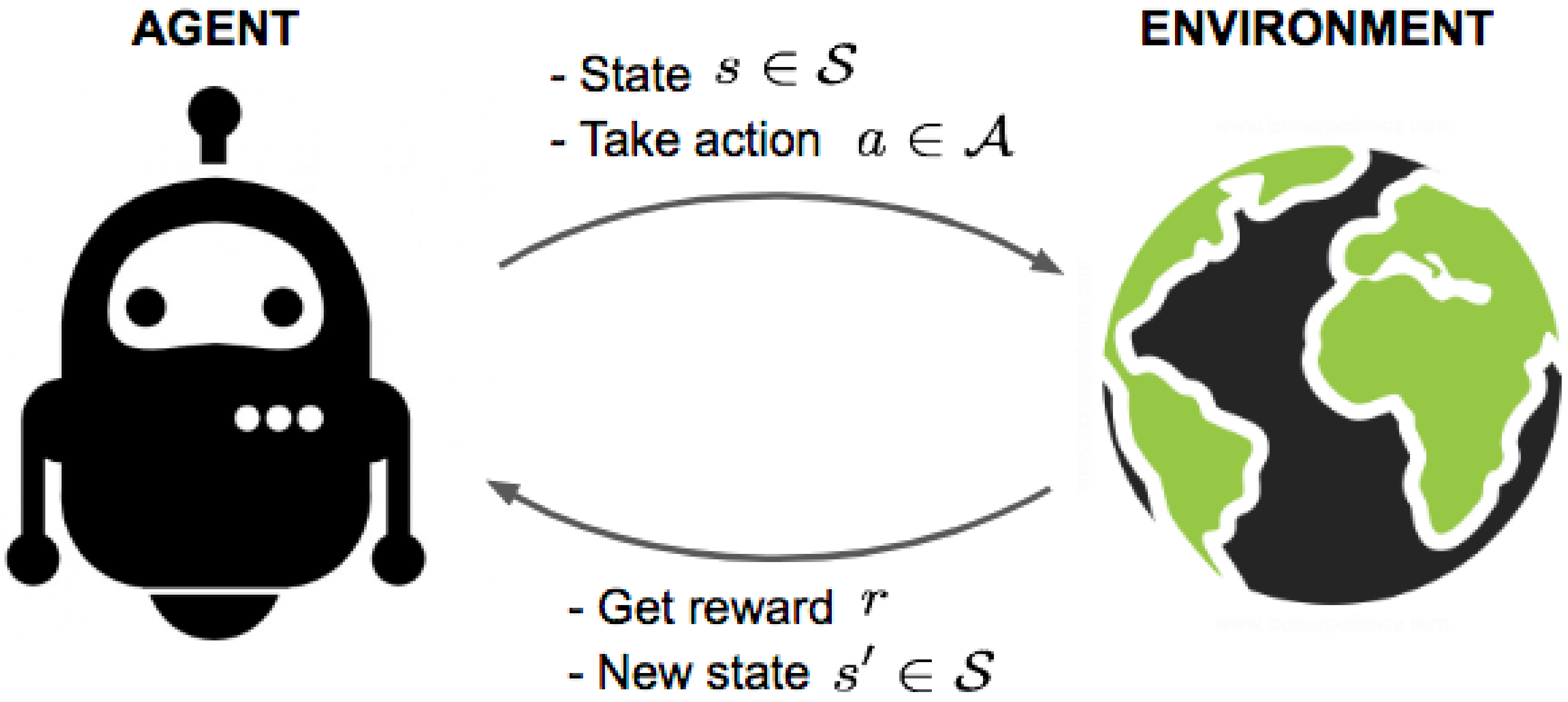
ARTIFICIAL INTELLIGENCE



ARTIFICIAL INTELLIGENCE

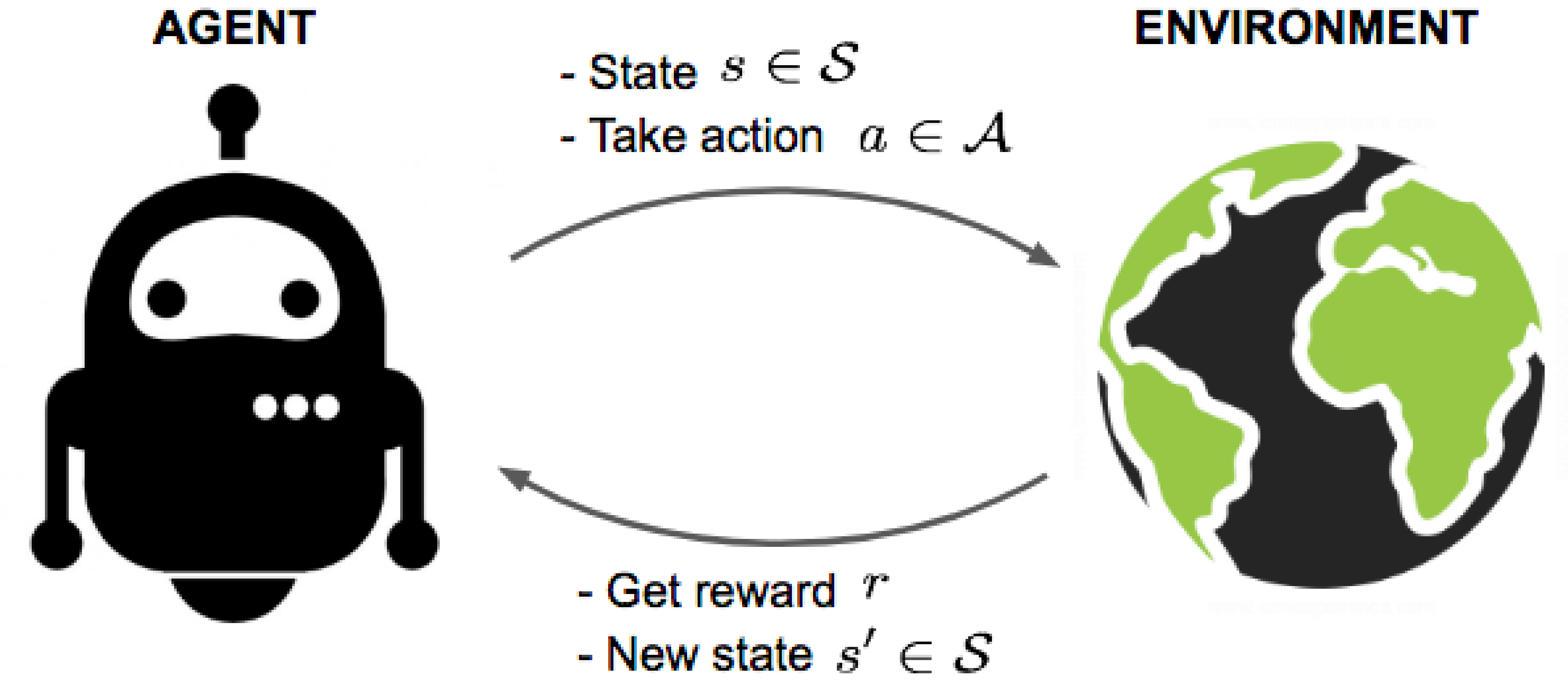
Reinforcement learning

REINFORCEMENT LEARNING



Source: https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html

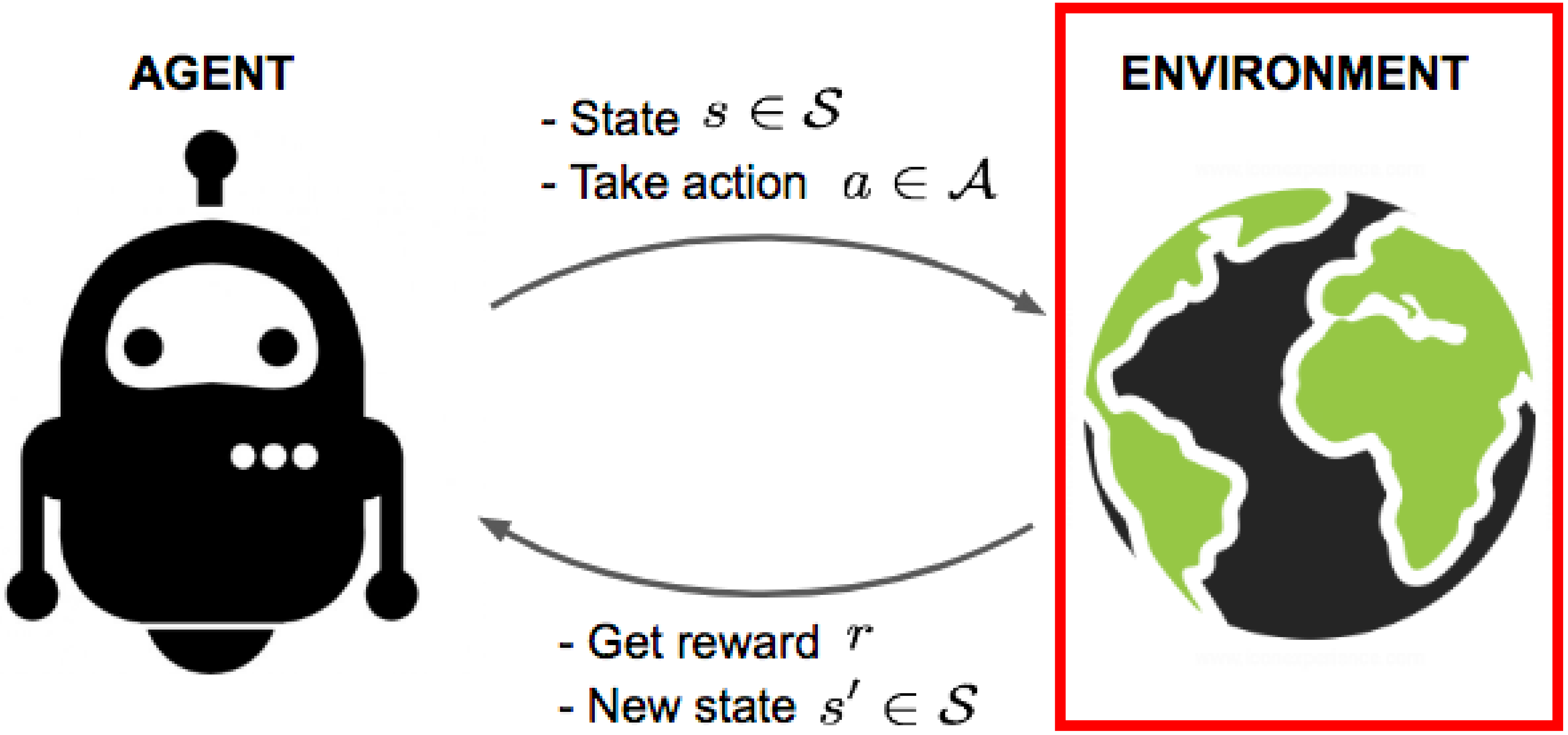
REINFORCEMENT LEARNING



Building blocks:

Source: https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html

REINFORCEMENT LEARNING

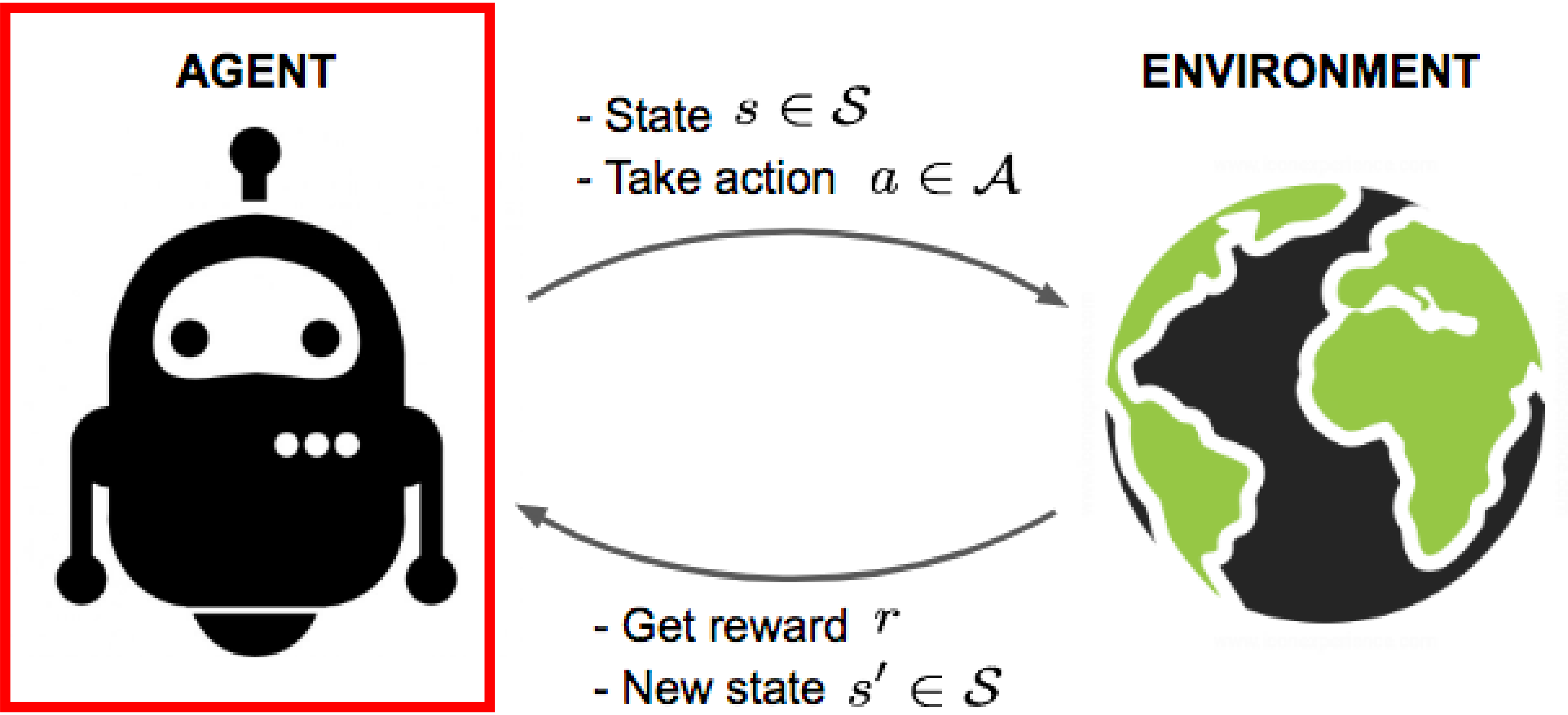


Building blocks:

- Environment

Source: https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html

REINFORCEMENT LEARNING

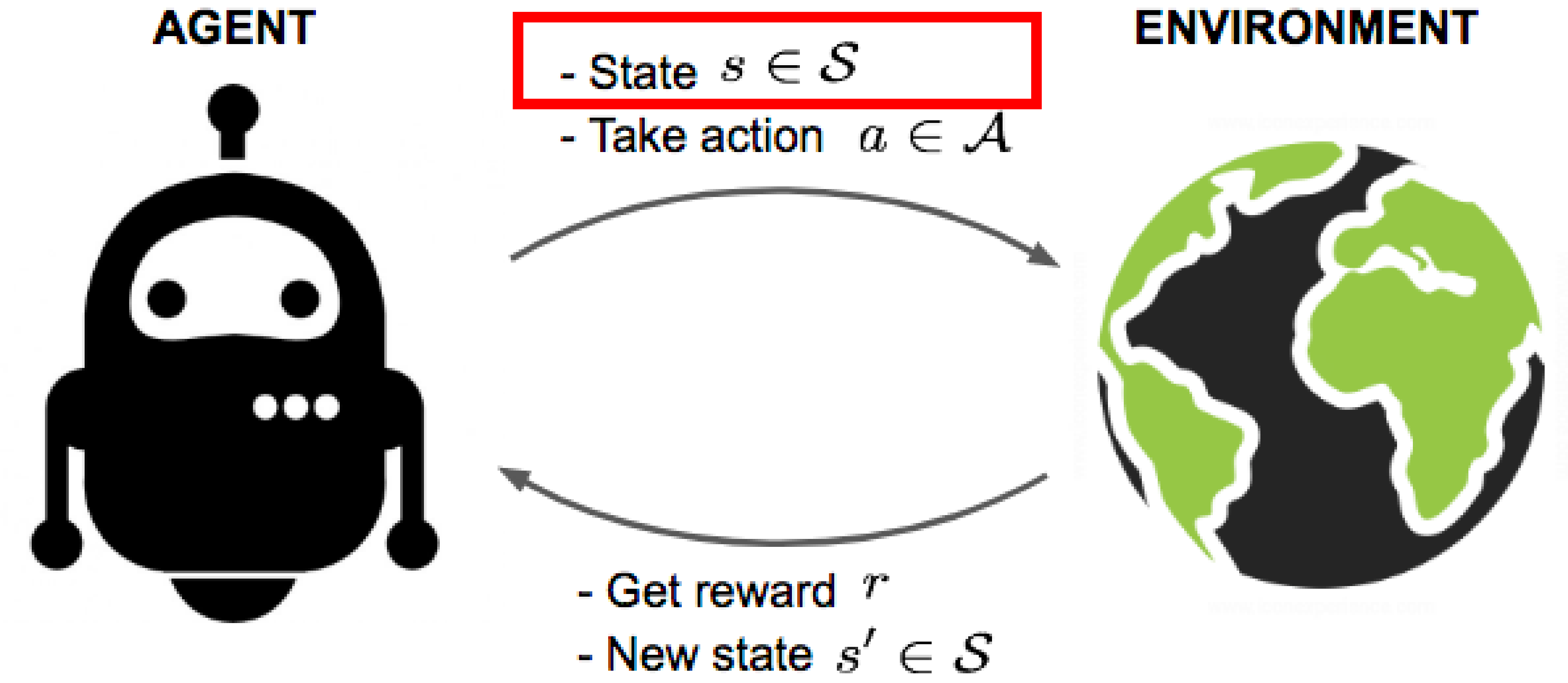


Building blocks:

- Environment
- Agent

Source: https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html

REINFORCEMENT LEARNING

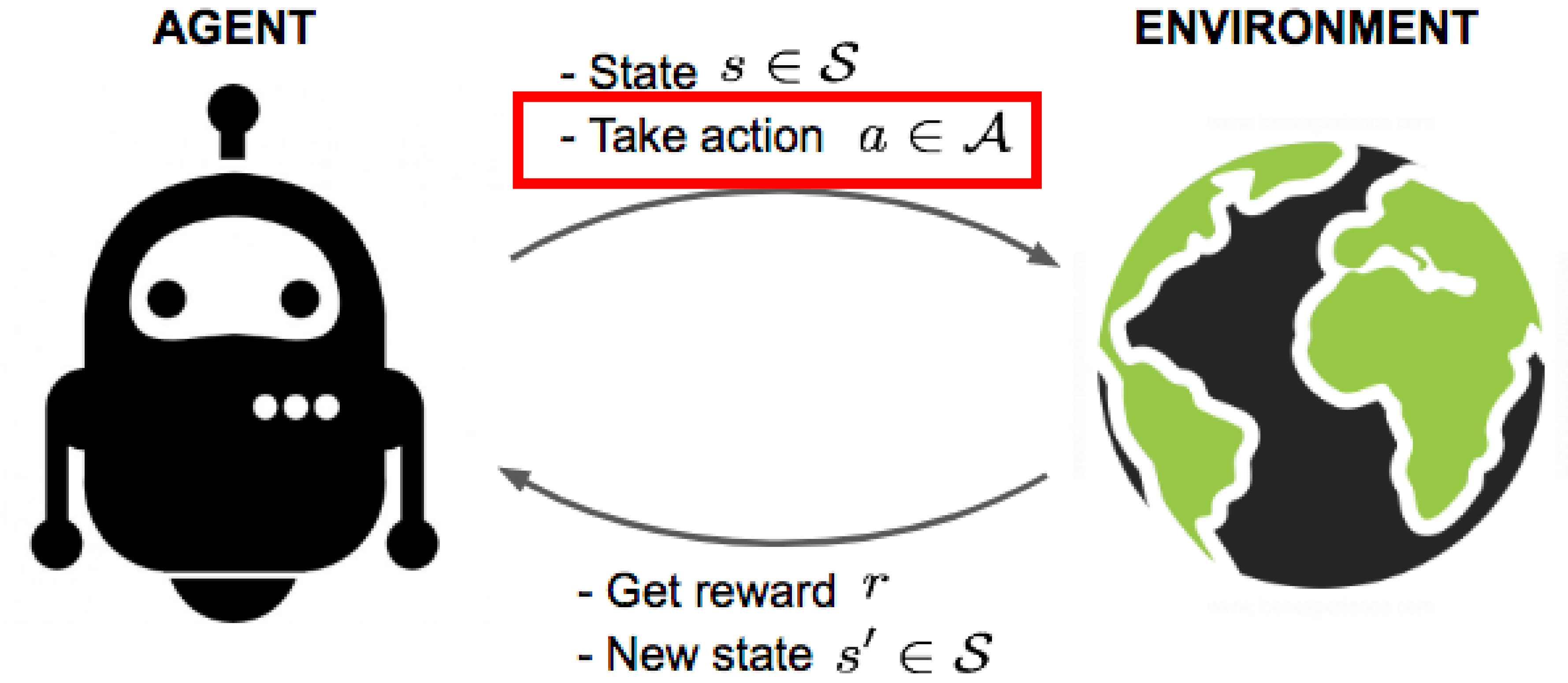


Building blocks:

- Environment
- Agent
- State (s)

Source: https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html

REINFORCEMENT LEARNING

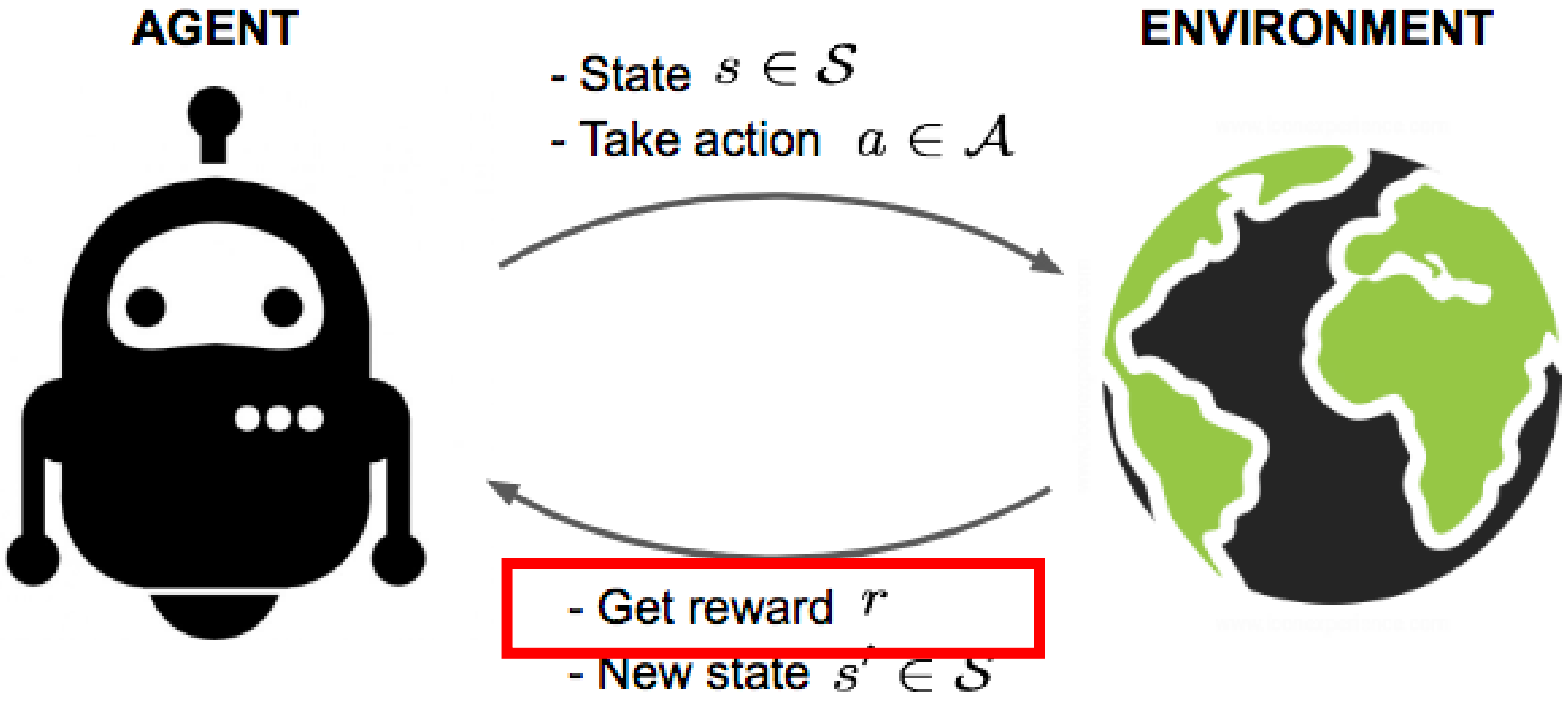


Building blocks:

- Environment
- Agent
- State (s)
- Action (a)

Source: https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html

REINFORCEMENT LEARNING

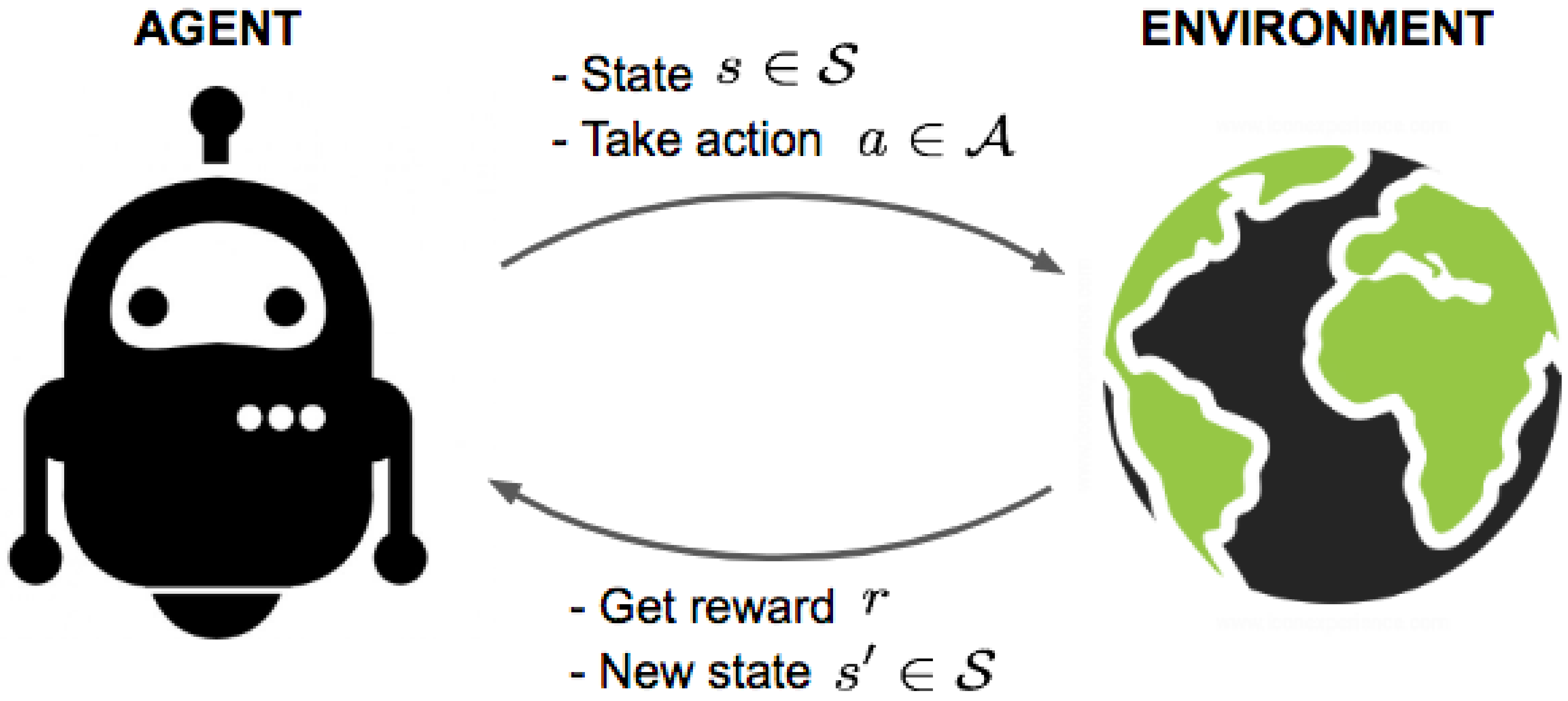


Building blocks:

- Environment
- Agent
- State (s)
- Action (a)
- Reward (r)

Source: https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html

REINFORCEMENT LEARNING



Building blocks:

- Environment
- Agent
- State (s)
- Action (a)
- Reward (r)

Source: https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html

Reinforcement learning

2 – Basics of reinforcement learning

What is reinforcement learning?

REINFORCEMENT LEARNING

- Reinforcement learning (RL) is an area of machine learning concerned with how intelligent agents tought to take actions in an environment in order to maximize the notion of cumulative reward. (Wikipedia)

REINFORCEMENT LEARNING

- Reinforcement learning (RL) is an area of machine learning concerned with how intelligent agents tought to take actions in an environment in order to maximize the notion of cumulative reward. (Wikipedia)
- But what does this mean?

REINFORCEMENT LEARNING

- Reinforcement learning (RL) is an area of machine learning concerned with how intelligent **agents** tought to take **actions** in an **environment** in order to maximize the notion of cumulative **reward**. (Wikipedia)

REINFORCEMENT LEARNING

- The key building blocks of reinforcement learning are the following:
  - agent
  - environment

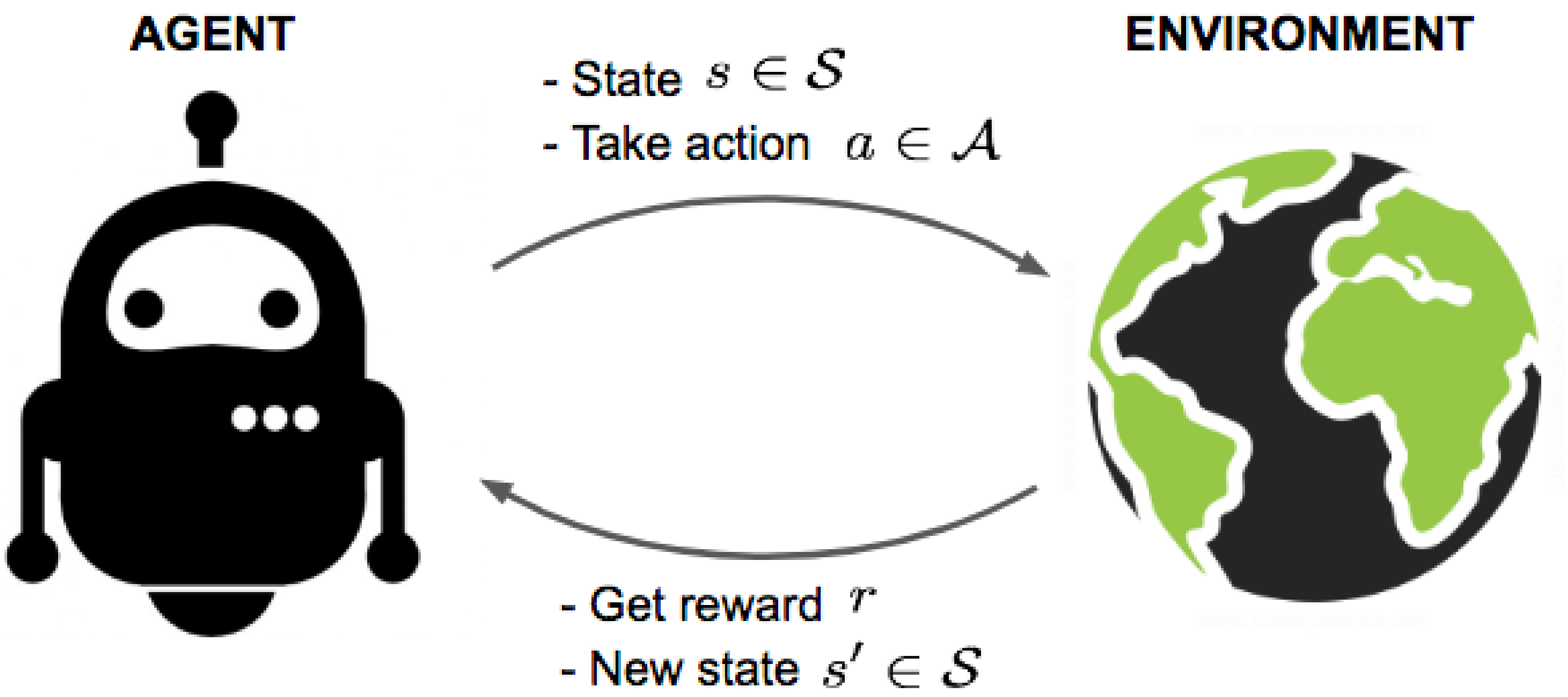
REINFORCEMENT LEARNING

- The key building blocks of reinforcement learning are the following:
  - agent
  - environment
- These components <u>**interact**</u> with each other

REINFORCEMENT LEARNING

- The key building blocks of reinforcement learning are the following:
  - agent
  - environment
- These components **interact** with each other
- The goal of the agent is to obtain **as much reward as possible**

REINFORCEMENT LEARNING



Source: https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html

REINFORCEMENT LEARNING

- We can formularize this workflow as the following.

REINFORCEMENT LEARNING

- We can formularize this workflow as the following. For each timestep t:

REINFORCEMENT LEARNING

- We can formularize this workflow as the following. For each timestep t:
  - state (s_t) – the state that the environment is

REINFORCEMENT LEARNING

- We can formularize this workflow as the following. For each timestep t:
  - state (s_t) – the state that the environment is
  - action (a_t) – the action of our agent

REINFORCEMENT LEARNING

- We can formularize this workflow as the following. For each timestep t:
  - state (s_t) – the state that the environment is
  - action (a_t) – the action of our agent
  - reward (r_{t+1}) – the reward of taking the action a_t in state s_t

REINFORCEMENT LEARNING

- We can formularize this workflow as the following. For each timestep t:
  - state (s_t) – the state that the environment is
  - action (a_t) – the action of our agent
  - reward (r_{t+1}) – the reward of taking the action a_t in state s_t
  - next state (s_{t+1}) – the state that we reach from s_t after taking a_t

REINFORCEMENT LEARNING

- We can formularize this workflow as the following. For each timestep t:
  - state (s_t) – the state that the environment is
  - action (a_t) – the action of our agent
  - reward (r_{t+1}) – the reward of taking the action a_t in state s_t
  - next state (s_{t+1}) – the state that we reach from s_t after taking a_t
- One tiny little remark:
  - We may not always have access to the whole state
  - In those cases we use an observation o_t of the state s_t

REINFORCEMENT LEARNING

- Why do we care about rewards? And why do we care about cumulative rewards?

REINFORCEMENT LEARNING

- Why do we care about rewards? And why do we care about cumulative 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)."* – Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

MARKOV DECISION PROCESS (MDP)

- Markov property
  - The current state (s_t) only depends on the previous state (s_{t-1})

MARKOV DECISION PROCESS (MDP)

- Markov property
  - The current state (s_t) only depends on the previous state (s_{t-1})
  - Sidenote: this may not always be 100% accurate for the given problem but it works well in practice

MARKOV DECISION PROCESS (MDP)

- Markov property
  - The current state (s_t) only depends on the previous state (s_{t-1})
  - Sidenote: this may not always be 100% accurate for the given problem but it works well in practice
- Markov Decision Processes (MDPs)
  - (S, A, R, P, γ)

MARKOV DECISION PROCESS (MDP)

- Markov property
  - The current state (s_t) only depends on the previous state (s_{t-1})
  - Sidenote: this may not always be 100% accurate for the given problem but it works well in practice
- Markov Decision Processes (MDPs)
  - (S, A, R, P, γ)
  - S – states
A – actions
R – reward function
P – state transition probabilities
γ – discount factor $\gamma \in [0, 1]$

STATES & ACTIONS

STATES & ACTIONS

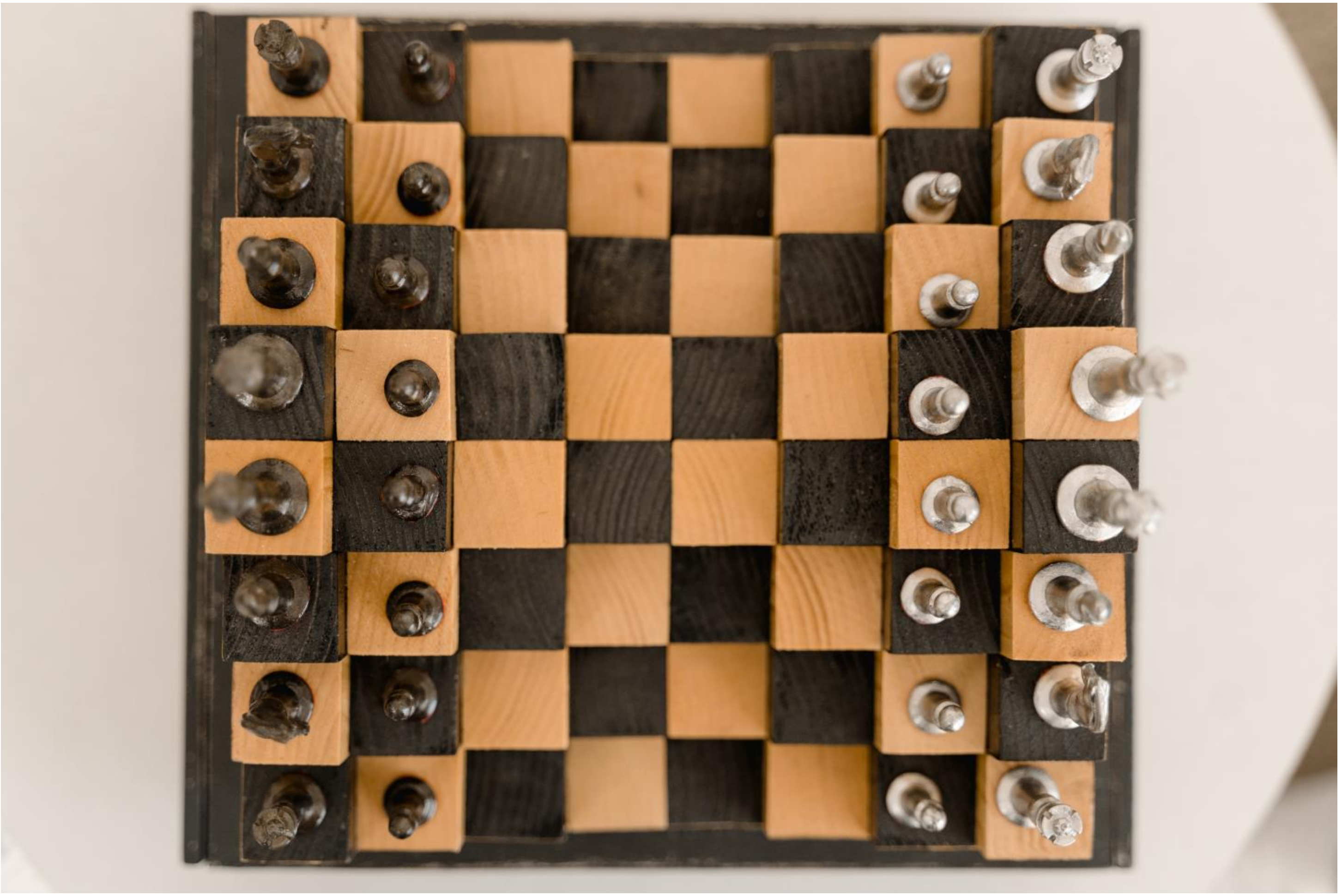
- **State spaces**
  - States describe the state of the observed world
  - There are cases when we can only **partially** observe the world (e.g. pixel data)
  - Can be *finite* or *infinite* and *discrete* or *continuous*
  - Example: the current state of the chess board

STATES & ACTIONS

- **State spaces**
  - States describe the state of the observed world
  - There are cases when we can only **<u>partially</u>** observe the world (e.g. pixel data)
  - Can be *finite* or *infinite* and *discrete* or *continuous*
  - Example: the current state of the chess board
- **Action spaces**
  - Actions denote actions that the agent can take in a given state
  - Can be *finite* or *infinite* and *discrete* or *continuous*
  - Example: moving a chest piece to a given field on the board

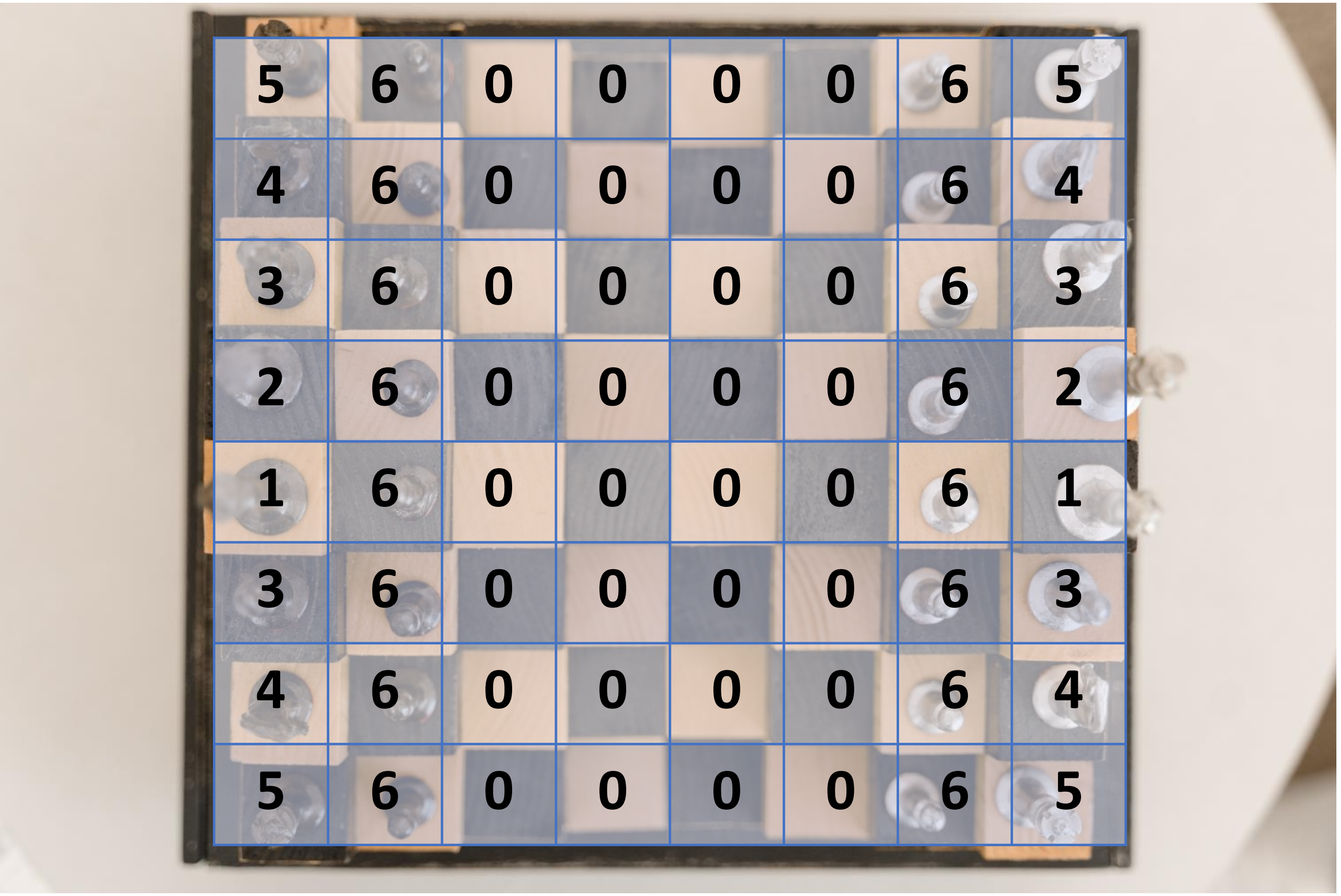
STATES & ACTIONS

State



STATES & ACTIONS

Observation



STATES & ACTIONS

Observation

5	6	0	0	0	0	6	5
4	6	0	0	0	0	6	4
3	6	0	0	0	0	6	3
2	6	0	0	0	0	6	2
1	6	0	0	0	0	6	1
3	6	0	0	0	0	6	3
4	6	0	0	0	0	6	4
5	6	0	0	0	0	6	5

STATES & ACTIONS

Action (9, 25)

STATES & ACTIONS

from

Action (9, 25)

to

STATES & ACTIONS

2nd row 2nd field

from

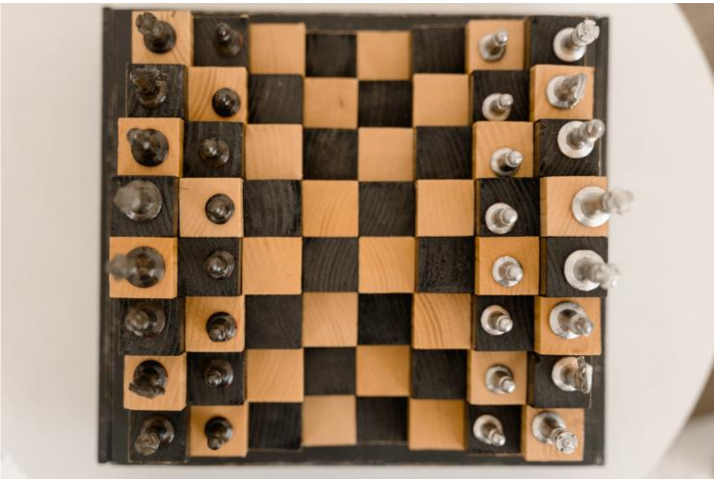
Action (9, 25)

to

4th row 2nd field

Note: indexing starts at 0!

STATES & ACTIONS

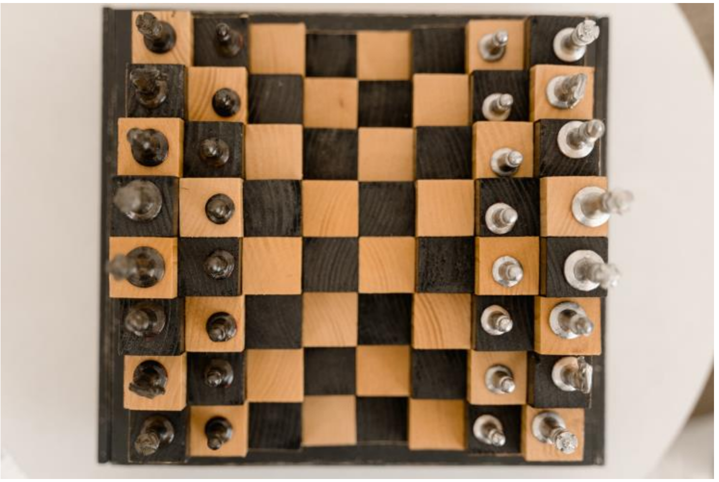


state

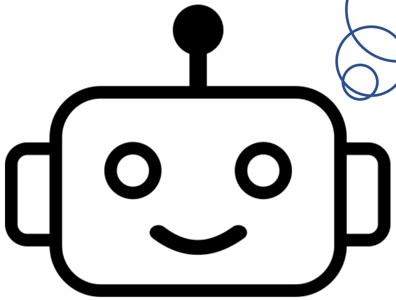
STATES & ACTIONS

5	6	0	0	0	0	6	5
4	6	0	0	0	0	6	4
3	6	0	0	0	0	6	3
2	6	0	0	0	0	6	2
1	6	0	0	0	0	6	1
3	6	0	0	0	0	6	3
4	6	0	0	0	0	6	4
5	6	0	0	0	0	6	5

observation



state



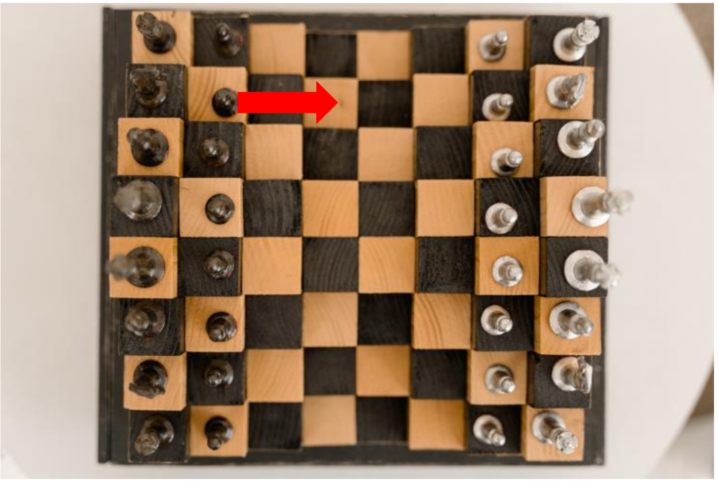
agent

STATES & ACTIONS

5	6	0	0	0	0	6	5
4	6	0	0	0	0	6	4
3	6	0	0	0	0	6	3
2	6	0	0	0	0	6	2
1	6	0	0	0	0	6	1
3	6	0	0	0	0	6	3
4	6	0	0	0	0	6	4
5	6	0	0	0	0	6	5

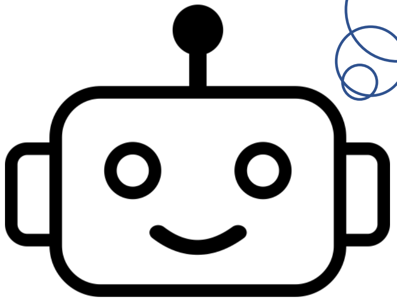
observation

reward (+1)



(9, 25)

action



agent

state

ONE LAST THING…

- **Policy**
    - describes the strategy that our agent executes
    - is a function that maps each s_t state to an action a_t
    - it can be deterministic or stochastic

Today’s RL problem

BANDITS PROBLEM

- We have n bandits (we will use n=3)

BANDITS PROBLEM

- We have n bandits (we will use n=3)
- They each have a fixed ratio for hitting the jackpot

BANDITS PROBLEM

- We have n bandits (we will use n=3)
- They each have a fixed ratio for hitting the jackpot
- Our goal is to find the bandit that yields the best results (e.g. big rewards or frequent rewards)

BANDITS PROBLEM

Which one should we choose?

Reinforcement learning

3 – Types of RL algorithms. Q-learning.

TYPES OF RL ALGORITHMS

- Value-based

TYPES OF RL ALGORITHMS

- Value-based

- Policy-based

TYPES OF RL ALGORITHMS

- Value-based

- Policy-based

- Model-based

TYPES OF RL ALGORITHMS

- Value-based
  - Our goal is to learn a state-value $V(s)$ or action-value $Q(s, a)$ function
  - This way, we can tell which state is better
- Policy-based
  -
- Model-based
  -

TYPES OF RL ALGORITHMS

- Value-based
  - Our goal is to learn a state-value $V(s)$ or action-value $Q(s, a)$ function
  - This way, we can tell which state is better
- Policy-based
  - Our goal is to directly learn a policy $\boldsymbol{\pi}$ *that maps any state* **s** *to an action* ***a*** such that the obtained cumulative reward is maximal
- Model-based

TYPES OF RL ALGORITHMS

- Value-based
  - Our goal is to learn a state-value $V(s)$ or action-value $Q(s,a)$ function
  - This way, we can tell which state is better
- Policy-based
  - Our goal is to directly learn a policy $\boldsymbol{\pi}$ *that maps any state* **s** *to an action* ***a*** such that the obtained cumulative reward is maximal
- Model-based
  - Learn a model of the world and then use this learnt model for planning

THE BASICS OF CLASSIC RL

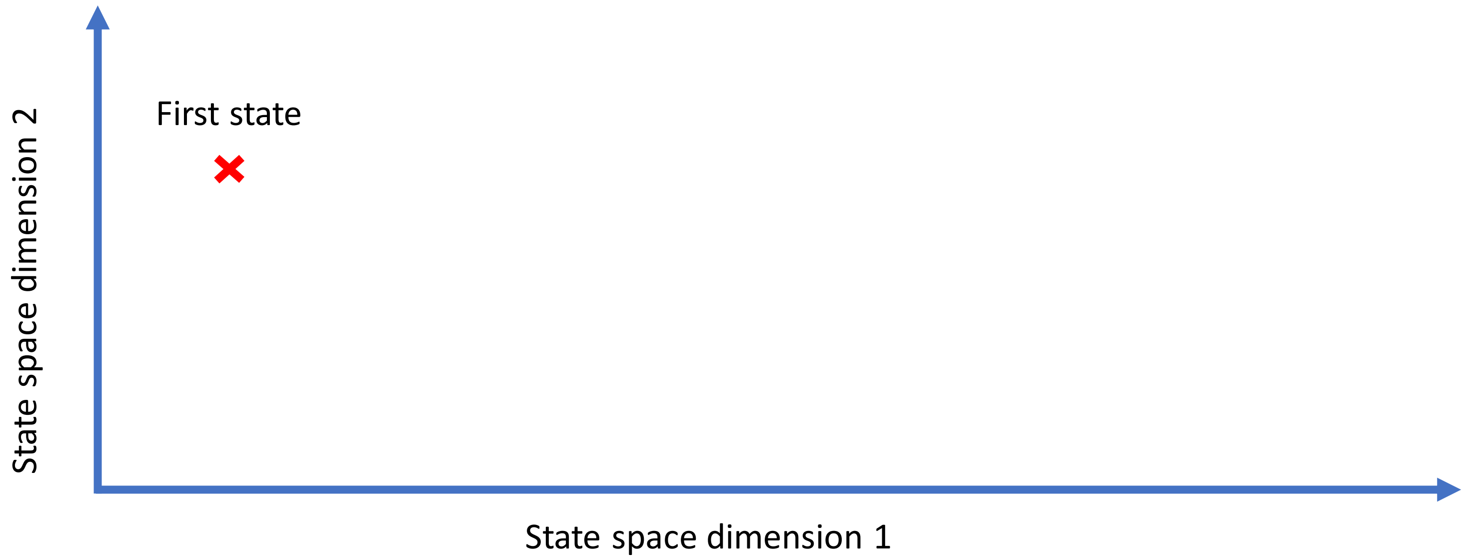
- We want to maximize the **cumulative reward**
  (remember the reward hypothesis from the previous lecture)

THE BASICS OF CLASSIC RL

- We want to maximize the **cumulative reward** (remember the reward hypothesis from the previous lecture)
- When our agent is making decisions, we refer to it as a **trajectory**

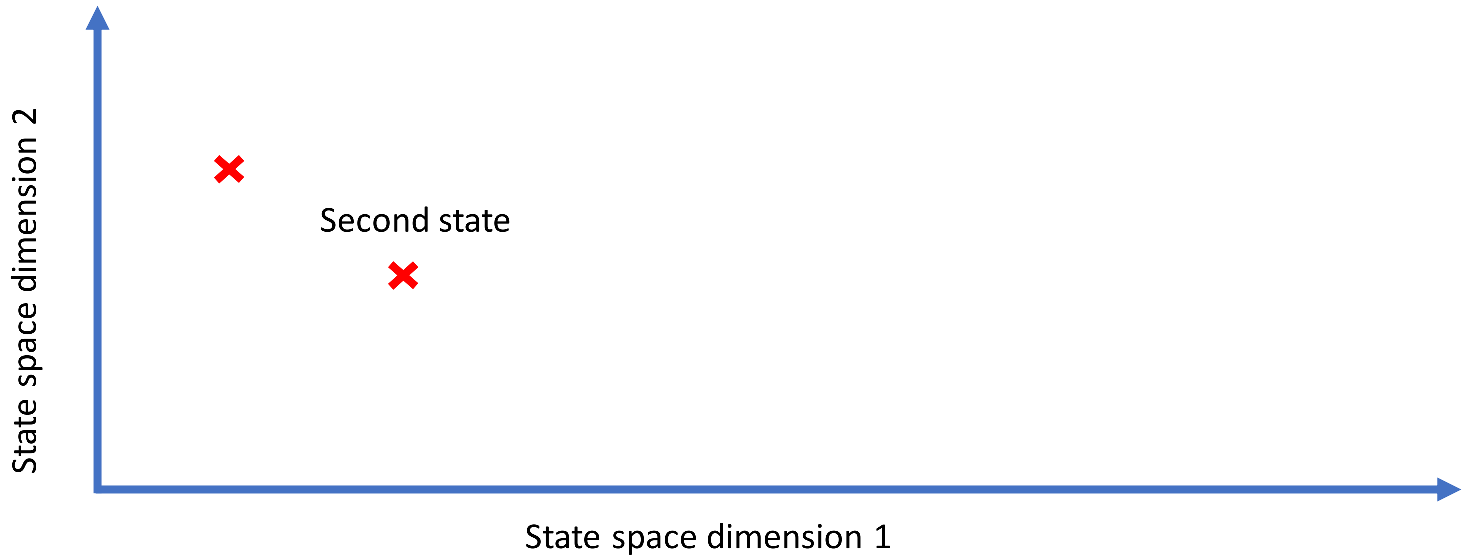
THE BASICS OF CLASSIC RL

- We want to maximize the **cumulative reward** (remember the reward hypothesis from the previous lecture)
- When our agent is making decisions, we refer to it as a **trajectory**



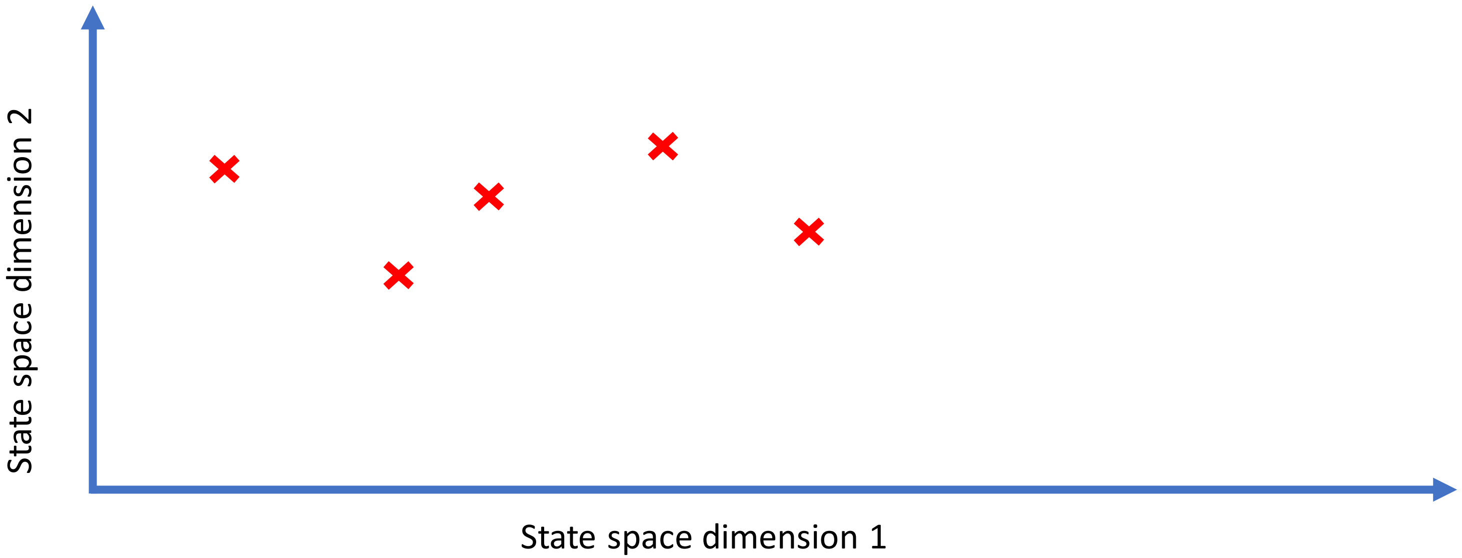
THE BASICS OF CLASSIC RL

- We want to maximize the **cumulative reward** (remember the reward hypothesis from the previous lecture)
- When our agent is making decisions, we refer to it as a **trajectory**



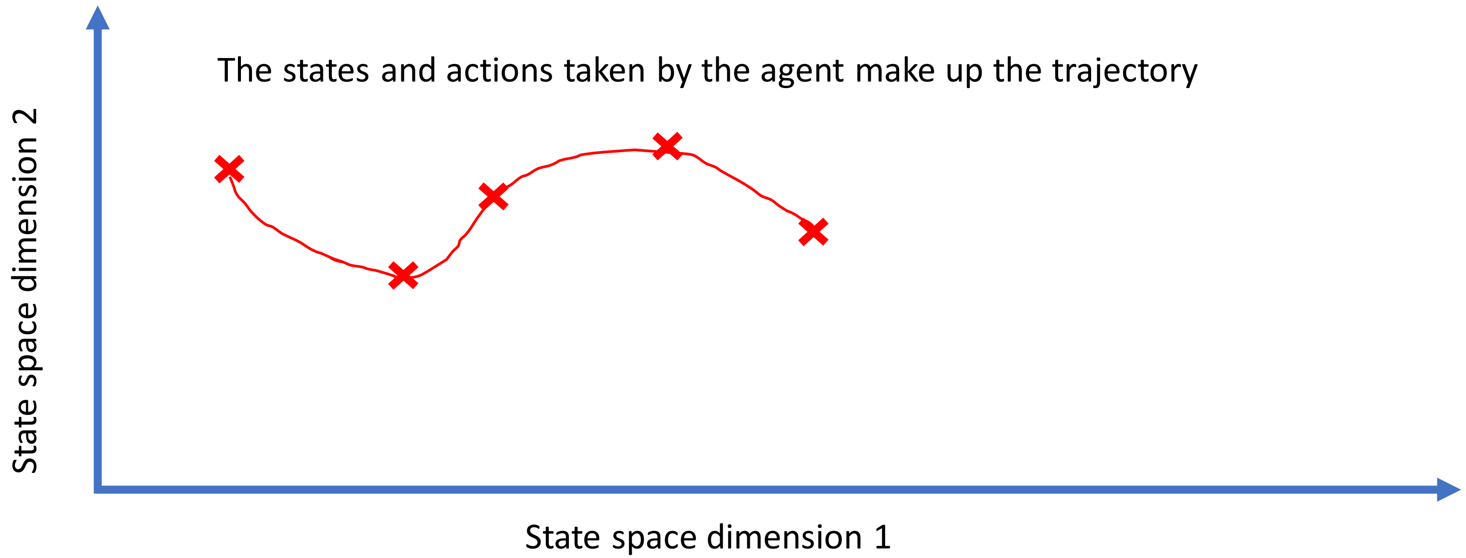
THE BASICS OF CLASSIC RL

- We want to maximize the **<u>cumulative reward</u>** (remember the reward hypothesis from the previous lecture)
- When our agent is making decisions, we refer to it as a **trajectory**



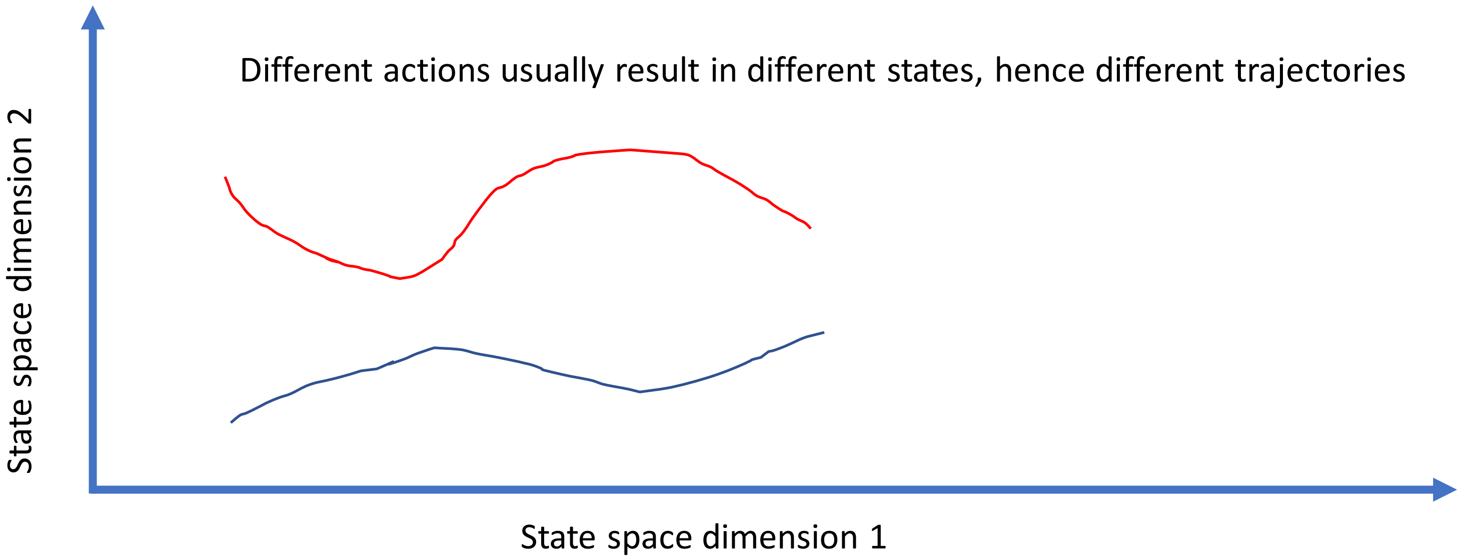
THE BASICS OF CLASSIC RL

- We want to maximize the **cumulative reward** (remember the reward hypothesis from the previous lecture)
- When our agent is making decisions, we refer to it as a **trajectory**



THE BASICS OF CLASSIC RL

- We want to maximize the **cumulative reward** (remember the reward hypothesis from the previous lecture)
- When our agent is making decisions, we refer to it as a **trajectory**



THE BASICS OF CLASSIC RL

- We can define the cumulative reward for any trajectory τ:
  - $R(\tau) = r_{t+1} + r_{t+2} + \cdots$

THE BASICS OF CLASSIC RL

- We can define the cumulative reward for any trajectory τ:
  - $R(\tau) = r_{t+1} + r_{t+2} + \cdots$
- What's the problem with this formula?

THE BASICS OF CLASSIC RL

- We can define the cumulative reward for any timestep ***t*** as:
  - $R(\tau) = r_{t+1} + r_{t+2} + \cdots$
- What's the problem with this formula?
  - Question:
    Which one is better: getting 5 dollars now or 100 dollars a year later?

THE BASICS OF CLASSIC RL

- We can define the cumulative reward for any timestep ***t*** as:
  - $R(\tau) = r_{t+1} + r_{t+2} + \cdots$
- What's the problem with this formula?
  - Question:
    Which one is better: getting 5 dollars now or 100 dollars a year later?
  - Usually, we prefer getting rewards as soon as possible, meaning that the further the reward is in the future, the less we are concerned about it

THE BASICS OF CLASSIC RL

- We can thus modify the formula and change it to:
  - $R(\tau) = r_{t+1} + \gamma * r_{t+2} + \gamma^2 * r_{t+2} + \gamma^3 * r_{t+3} + \ \dots$

THE BASICS OF CLASSIC RL

- We can thus modify the formula and change it to:
  - $R(\tau) = r_{t+1} + \gamma * r_{t+2} + \gamma^2 * r_{t+2} + \gamma^3 * r_{t+3} + \ \ldots$
- γ is between 0 and 1
  - γ is called the discount factor
  - γ =0 means that we only care about immediate rewards
  - γ =1 means that we treat immediate and future rewards the same way

EXPLORATION VS EXPLOITATION

- State spaces are usually really large

EXPLORATION VS EXPLOITATION

- State spaces are usually really large
- There can also be many actions

EXPLORATION VS EXPLOITATION

- State spaces are usually really large
- There can also be many actions
- How do we know which one to try and why?

EXPLORATION VS EXPLOITATION

- State spaces are usually really large
- There can also be many actions
- How do we know which one to try and why?
- More importantly: how do we know if the action(s) we took are the optimal ones?

EXPLORATION VS EXPLOITATION

- State spaces are usually really large
- There can also be many actions
- How do we know which one to try and why?
- More importantly: how do we know if the action(s) we took are the optimal ones?
- Answer: exploration – exploitation tradeoff

EXPLORATION VS EXPLOITATION

- Answer: exploration – exploitation tradeoff
  - Exploration: we explore the environment
  - Exploitation: we use take the action that proved to be the best so far (hence the name exploitation)

EXPLORATION VS EXPLOITATION

- Answer: exploration – exploitation tradeoff
  - Exploration: we explore the environment
  - Exploitation: we use take the action that proved to be the best so far (hence the name exploitation)
- We will use different stragies, like the epsilon-greedy strategy, where we only exploit with 1-ε probability. (ε ∈ [0,1] and decreases continuously)

EXPLORATION VS EXPLOITATION

Example:

- You want to buy a new phone. Will you buy a newer model from the same brand that you currently have (exploitation) or will you look at other brands as well (exploration)?
- In the first case (exploitation), you are more likely to get a reliable model due to the same manufacturer.
- In the second case (exploration), you may get a less reliable phone or you may dislike the new ecosystem but you could also end up liking the new version more.

VALUE-BASED METHODS

- We are **not** looking for a policy

VALUE-BASED METHODS

- We are **not** looking for a policy
- Instead, we are looking for a $V(s)$ value function that tells us how „good" each state is
- But how do we define $V(s)$?

VALUE-BASED METHODS

- We are **not** looking for a policy
- Instead, we are looking for a $V(s)$ value function that tells us how „good” each state is
- But how do we define $V(s)$?
- We will say that the value of a state ***s*** is the sum of rewards that we get, **starting from that state**

VALUE-BASED METHODS

- We are **not** looking for a policy
- Instead, we are looking for a $V(s)$ value function that tells us how „good" each state is
- But how do we define $V(s)$?
- We will say that the value of a state ***s*** is the sum of rewards that we get, **starting from that state**
- We will want to get to states with the **highest** $V(s)$ **values**

VALUE-BASED METHODS

- There are also times, when we care more about the value of a given action ***a*** in a state ***s***
- In those cases, we can use Q-learning

VALUE-BASED METHODS

- There are also times, when we care more about the value of a given action ***a*** in a state ***s***
- In those cases, we can use Q-learning
- We define a $Q(s, a)$ function as the sum of rewards that we get starting from ***s*** and taking action ***a***
- In a given ***s*** state, we will want to use the action ***a*** that has the highest $Q(s, a)$ value

BELLMAN EQUATION

- The formula for *V(s)* is:
  $V(s) = E[r_{t+1} + \gamma * r_{t+2} + \gamma^2 * r_{t+3} + \gamma^3 * r_{t+4} + \ldots | \mathrm{st} = \mathrm{s}]$

BELLMAN EQUATION

- The formula for $V(s)$ is:
  $V(s) = \mathrm{E}[r_{t+1} + \gamma * r_{t+2} + \gamma^2 * r_{t+3} + \gamma^3 * r_{t+4} + \ \ldots \mid \mathrm{st} = \mathrm{s}]$
- What do we see here?

BELLMAN EQUATION

- The formula for $V(s)$ is:
  $V(s) = E[r_{t+1} + \gamma * \underbrace{r_{t+2} + \gamma^2 * r_{t+3} + \gamma^3 * r_{t+4} + \ \ldots}_{V(s_{t+1})} \mid st = s]$

BELLMAN EQUATION

- The formula for $V(s)$ is:
  $$V(s) = E[r_{t+1} + \gamma * \underbrace{r_{t+2} + \gamma^2 * \overbrace{r_{t+3} + \gamma^3 * r_{t+4} + \ldots}^{V(s_{t+2})}}_{V(s_{t+1})} \mid st = s]$$

BELLMAN EQUATION

- The formula for $V(s)$ is:
  $$V(s) = E[r_{t+1} + \underbrace{\gamma * r_{t+2} + \overbrace{\gamma^2 * r_{t+3} + \gamma^3 * r_{t+4} + \ldots}^{V(s_{t+2})}}_{V(s_{t+1})} \mid st = s]$$

- And so on

BELLMAN EQUATION

- So we get:
  $V(s) = E[r_{t+1} + \gamma * V(s_{t+1}) | \mathrm{st} = \mathrm{s}]$
- Which is called the **Bellman equation**

BELLMAN EQUATION

- So we get:
  $V(s) = E[r_{t+1} + \gamma * V(s_{t+1}) | \mathrm{st} = \mathrm{s}]$
- Which is called the **Bellman equation**
- This means that the value of a state is the immediate reward r_{t+1} plus the discounted value of the next state, which is $\gamma * V(s_{t+1})$

HOW DO WE GET THESE VALUES?

- The goal: we want to know the $V(s)$ or $Q(s,a)$ value of the different states or states and actions

HOW DO WE GET THESE VALUES?

- The goal: we want to know the $V(s)$ or $Q(s, a)$ value of the different states or states and actions
- We can start from scratch and update these values periodically:
  - If we update at each step, it is called TD-learning (Temporal Difference learning)
  - If we update at the end of each episode, it is called the Monte Carlo method

TEMPORAL DIFFERENCE LEARNING

- The main idea behind Temporal Difference learning or TD-learning is to update our estimate of the value function **at each step**
- That is:
- $V(st) = V(st_{)} + a * [r_{t+1} + \gamma * V(s_{t+1}) - V(st_{)}]$

TEMPORAL DIFFERENCE LEARNING

- The main idea behind Temporal Difference learning or TD-learning is to update our estimate of the value function **<u>at each step</u>**
- That is:
- $V(st) = V(st) + a \ * [\overbrace{r_{t+1} + \gamma * V(s_{t+1})}^{\text{New estimate}} - V(st)]$

Learning rate

Old estimate

TEMPORAL DIFFERENCE LEARNING

- The idea is the same for Q-learning as well but now we have (s, a) pairs
- The formula is:
- $Q(st, at_{)} = Q(st_{,} at_{)} + a * [r_{t+1} + \gamma * \max_{a}(Q(s_{t+1,} a)) - Q(st_{,} at_{)}]$

TEMPORAL DIFFERENCE LEARNING

- The idea is the same for Q-learning as well but now we have (s, a) pairs
- The formula is:
- $Q(st, at) = Q(st, at) + \underset{\text{Learning rate}}{a} * [\overbrace{r_{t+1} + \gamma * \max_a(Q(s_{t+1}, a))}^{\text{New estimate}} - \underset{\text{Old estimate}}{Q(st, at)}]$