Deep dive into Robotics I

AI: Algorithms to Actuation

Who is Kv?

Why should you hear me?

Python Programming

ROBOPACK

Powerful multi-purpose programming language

Simple and easy-to-use syntax

Most popular first-choice language for beginners!

By definition!

Python is an **interpreted**, **object-oriented**, **high-level** programming language. As it is general-purpose, it has a wide range of applications from web development, building desktop GUI to scientific and mathematical computing.

Applications : Web Applications

Hello, World! – a breakdown!

ROBOPACK

print(“Hello, World!”)

Here, `print()` is a function tells the computer to perform an action. What is that action? To output the value residing in it!

Hello, World! – a breakdown!

print(“Hello, World!”)

Inside the parentheses of the `print()` function is a sequence of characters — `Hello, World!` — that is enclosed in quotation marks `'` or ".

Any characters that are inside of quotation marks are called a ***string***.

Data types

Every value in Python has a **datatype**. Since everything is an object in Python programming, data types are actually **classes** and **variables** are instance (object) of these classes.

List

Ordered sequence of items.

```
1 x = [6, 99, 77, 'Apple']
2 print(x, "is of type", type(x))
```

```
[6, 99, 77, 'Apple'] is of type <class 'list'>
```

Dictionary

Unordered collection of **key-value** pairs. Optimized for **retrieving data**.

```python
d = {1: 'Apple', 2: 'Cat', 3: 'Food'}
print(d, type(d))

d[3]
```

```
{1: 'Apple', 2: 'Cat', 3: 'Food'} <class 'dict'>
'Food'
```

Variables

To *temporarily* store data in the computer's memory, for example, someone's name, age, email, address, price of a product, and so on.

Arithmetic Operators

Perform **mathematical operations** like addition, subtraction, multiplication etc.

Symbol	Task Performed	Meaning	Example
+	Addition	add two operands or unary plus	**x** + **y** or **+2**
-	Subtraction	substract right operand from the left or unary minus	**x** - **y** or **-2**
*	Multiplication	Multiply two operands	**x * y**
/	Division	Divide left operand by the right one (always results into float)	**x / y**
%	Modulus (remainder)	remainder of the division of left operand by the right	**x** % **y** (remainder of **x/y**)
//	Integer/Floor division	division that results into whole number adjusted to the left in the number line	**x // y**
**	Exponentiation (power)	left operand raised to the power of right	**x \\| y** (**x** to the power **y**)

Comparison Operators

Used to **compare values**. It either returns True or False according to the condition.

Symbol	Task Performed	Meaning	Example
>	greater than	True if left operand is greater than the right	x > y
<	less than	True if left operand is less than the right	x < y
==	equal to	True if both operands are equal	x == y
!=	not equal to	True if both operands are not equal	x != y
>=	greater than or equal to	True if left operand is greater than or equal to the right	x >= y
<=	less than or equal to	True if left operand is less than or equal to the right	x <= y

Logical Operators

Symbol	Meaning	Example
`and`	True if both the operands are true	x and y
`or`	True if either of the operand is true	x or y
`not`	True if operand are false (complements the operand)	not x

Functions

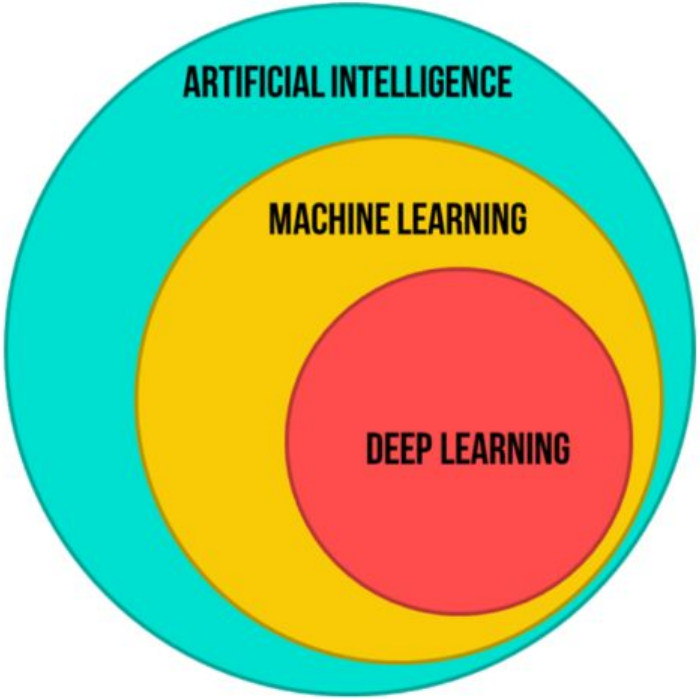
block of **organized**, **reusable** (DRY- Don’t Repeat Yourself) code with a name that is used to perform a single, specific task.

```python
def function_name(parameter1, parameter2):
    """docstring"""
    # function body
    # write some action
  return value
```

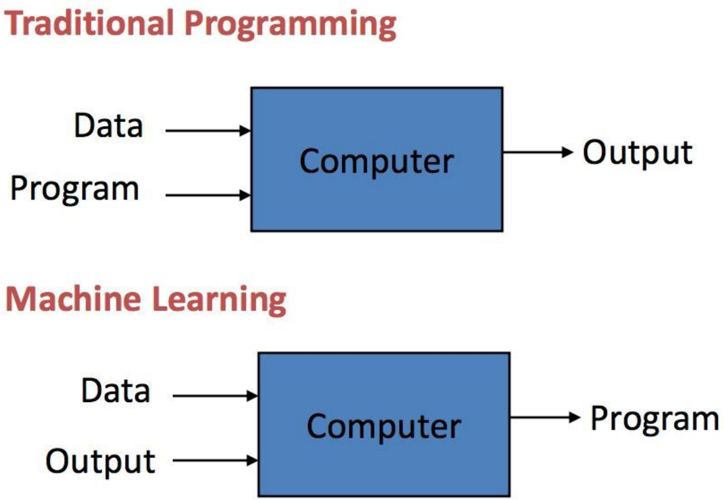
```python
# Example 1:
def greet():
    print("Welcome to Python for Data Science")

# call function using its name
greet()
```

```
Welcome to Python for Data Science
```

Difference between AI/ML/DL



Machine Learning



branch of artificial intelligence (AI) and computer science that focuses on the using **data** and **algorithms** to enable AI to *imitate the way that humans learn*, gradually improving its accuracy.

Supervised Learning

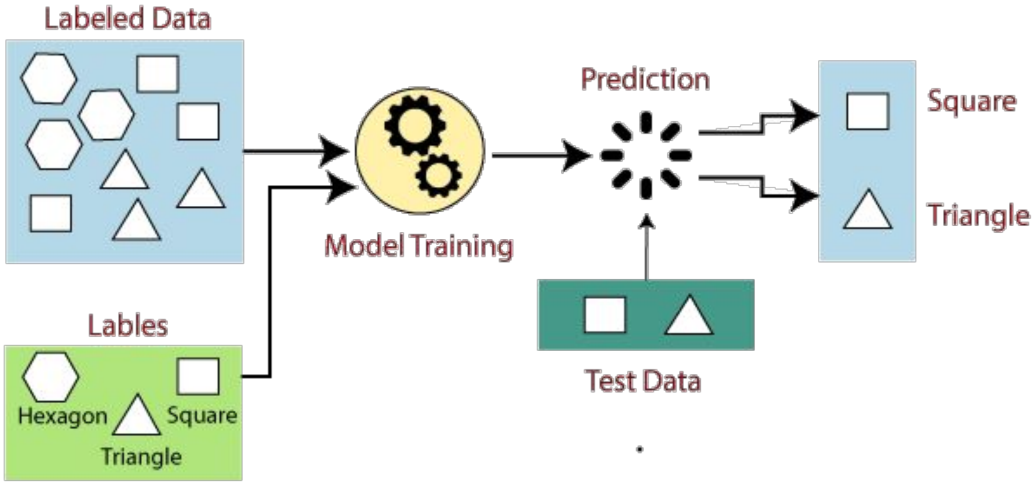
Given training data

+

Desired output

=

Predict



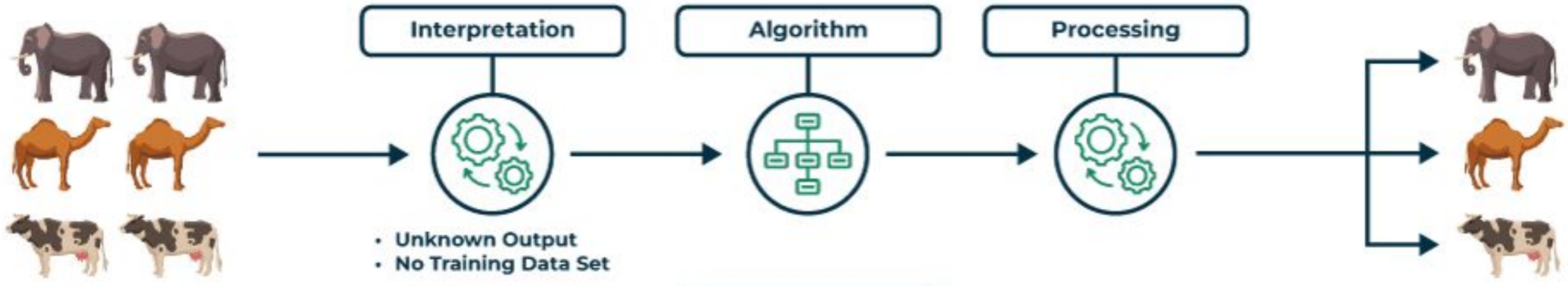
Unsupervised Learning

ROBOPACK

Lots of Training data (w/o desired output)

=

Find patterns



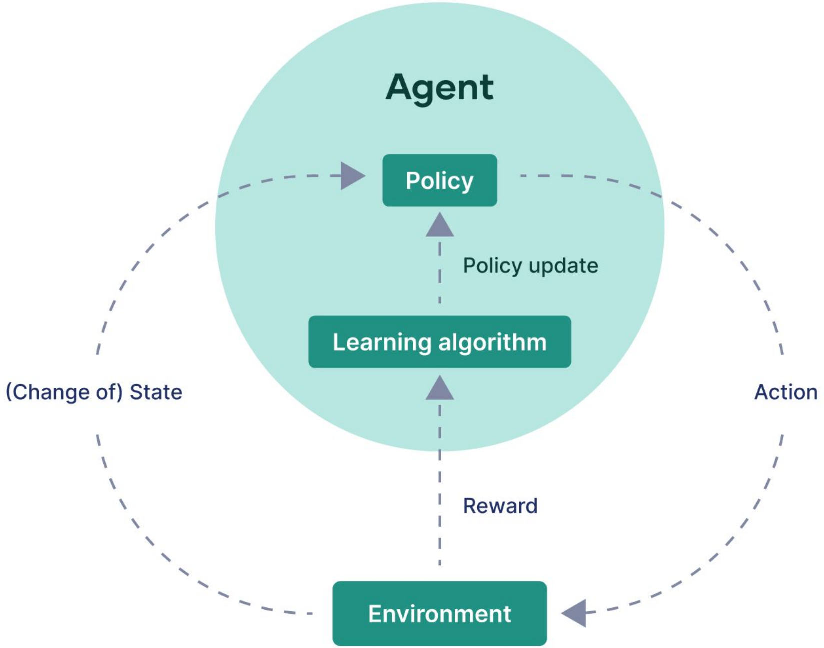
Reinforcement Learning

trains software to make decisions to achieve the most optimal results

almost no data
(just game parameters)

=

excel in the game



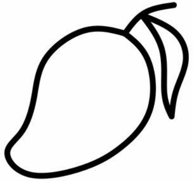
Let’s compare!

Supervised Learning

Data: (x, y)
x is data, y is label

Goal:
Learn function to map
x → y

Example:



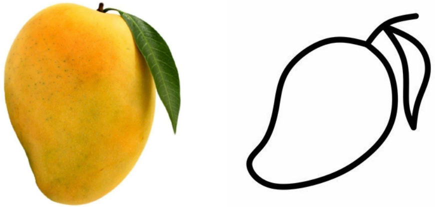
It’s a [‘mango’]

Unsupervised Learning

Data: x
x is data, no labels!

Goal:
Learn underlying structure

Example:



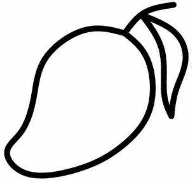
This thing is like the other

Reinforcement Learning

Data:
state-action pair

Goal:
maximize future rewards over
many time stamps

Example:



Eat it! ’coz that’s wot u live for

Reinforcement Learning

ROBOPACK

54 CHAPTER 3. FINITE MARKOV DECISION PROCESSES

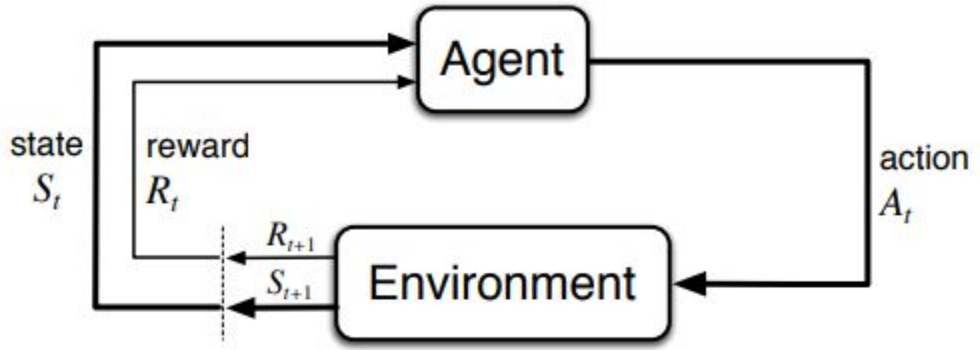


Figure 3.1: The agent–environment interaction in reinforcement learning.

Source: *Reinforcement Learning: An Introduction*, by Richard S. Sutton and Andrew G. Barto 2014-15 [(reference book link)]

Reinforcement Learning

Agent: takes action.

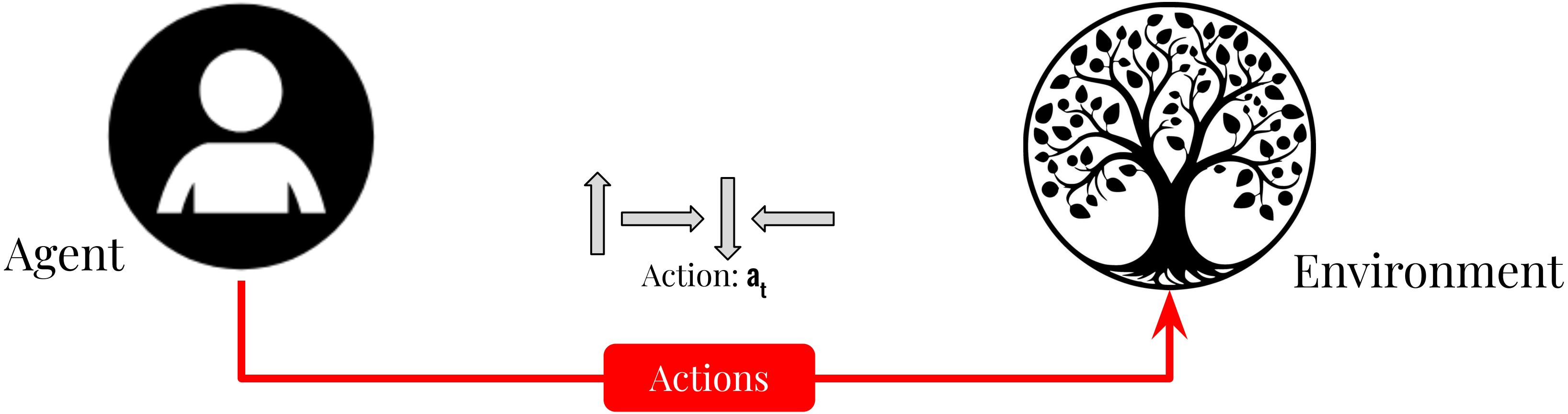
Reinforcement Learning

Agent

Environment

Environment: world in which the agent exists (or operates).

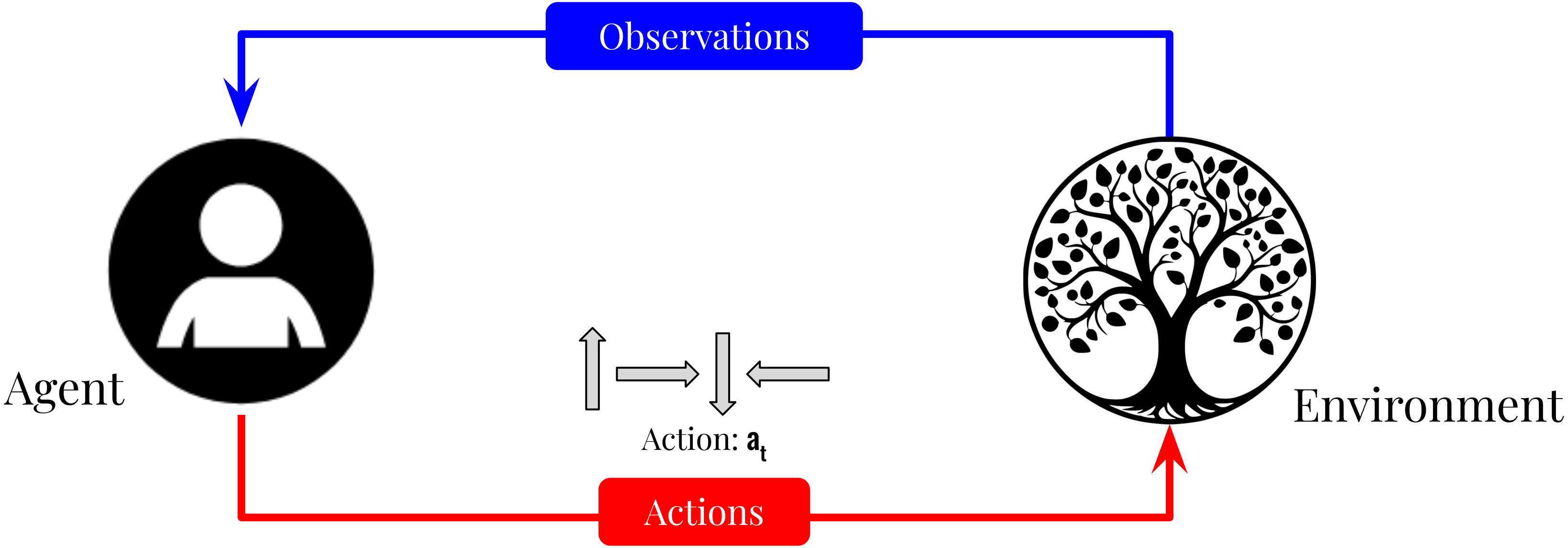
Reinforcement Learning



Action: a move that agent can make in the environment.

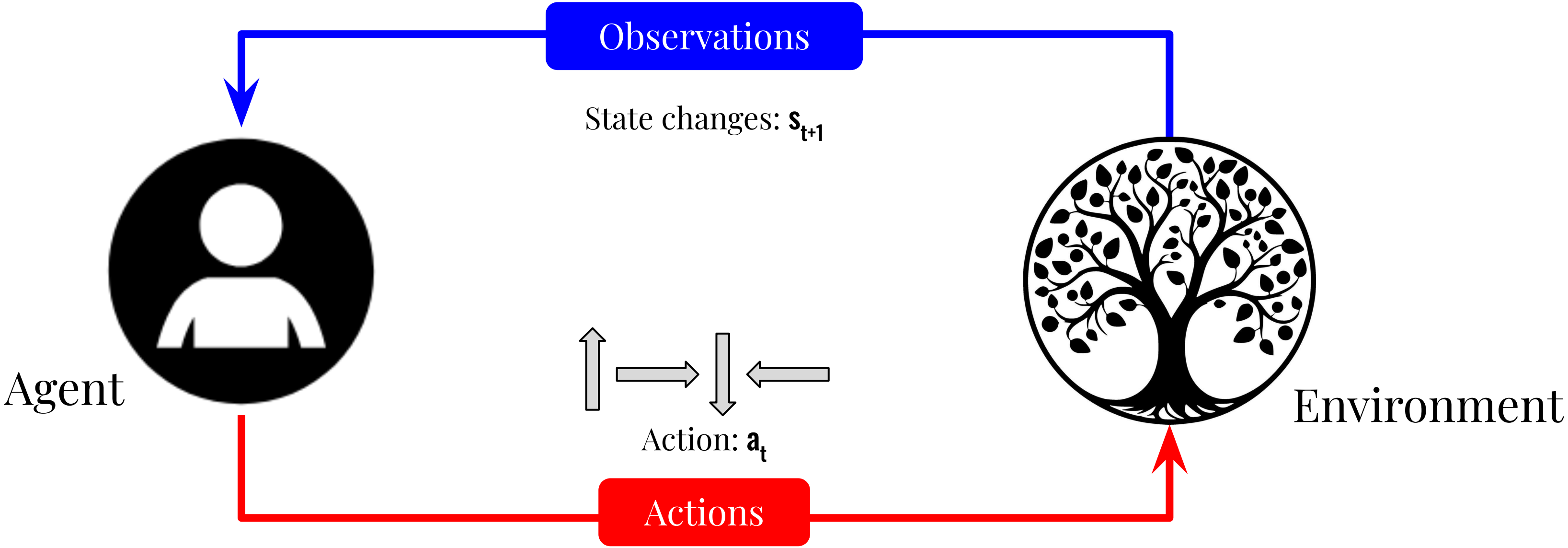
Action space A: a set of possible actions an agent can make in the environment.

Reinforcement Learning



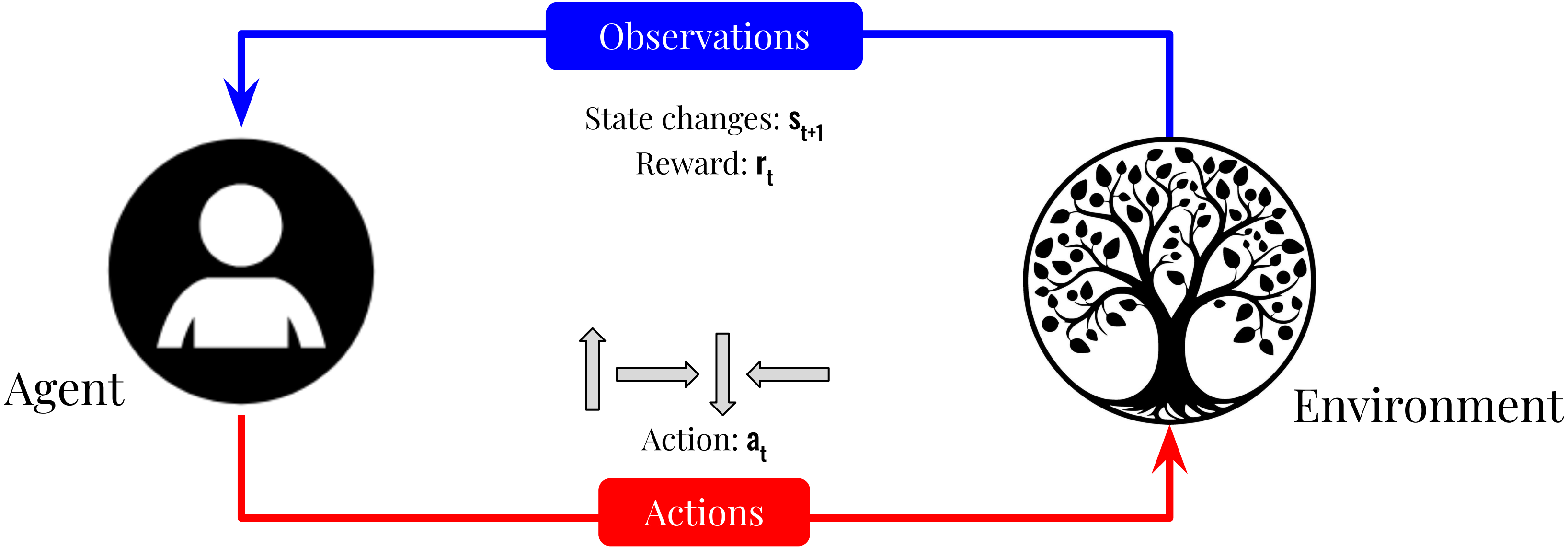
Observations: of the environment after taking actions.

Reinforcement Learning



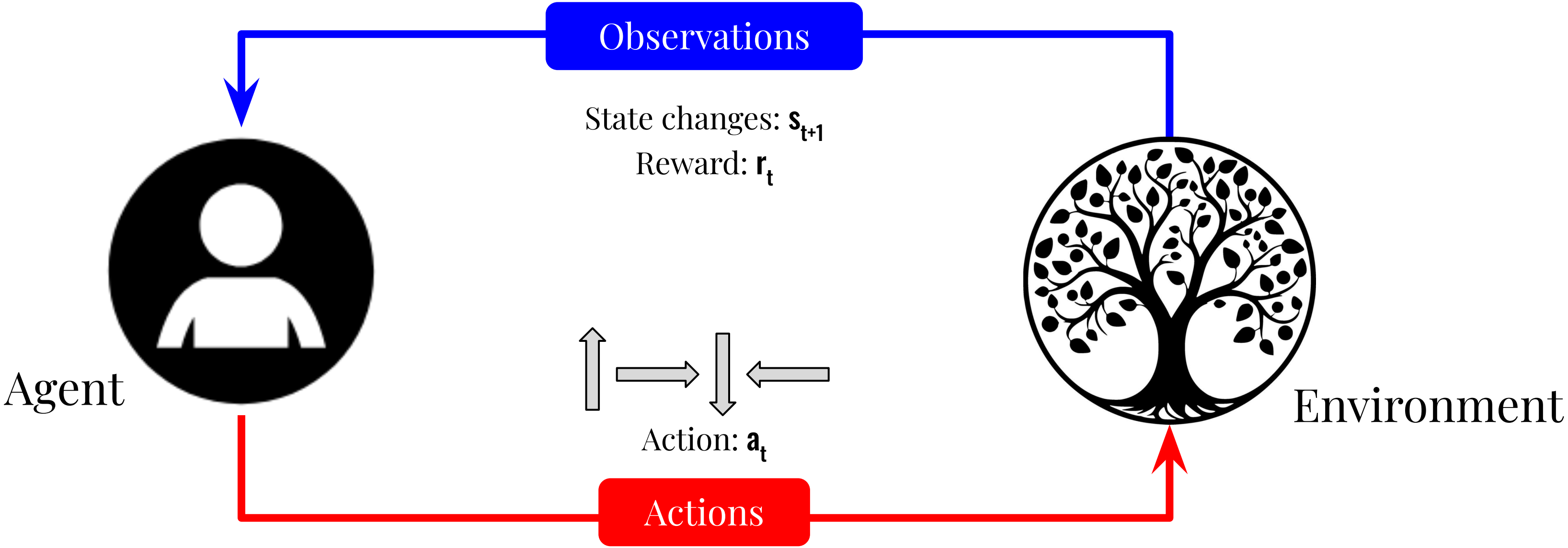
State: a situation that the agent perceives.

Reinforcement Learning



Reward: feedback that measures the success/failure of the agent's action.

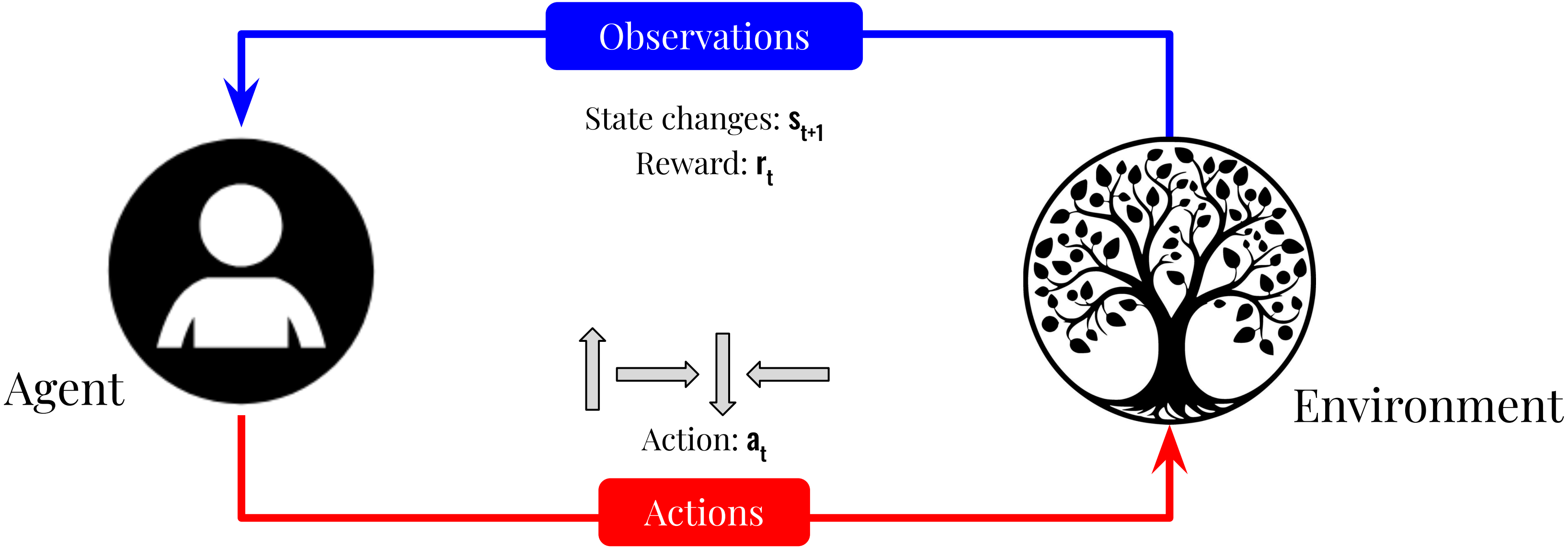
Reinforcement Learning



Total Reward
(**Return**)

$$R_t = \sum_{i=t}^{\infty} r_i$$

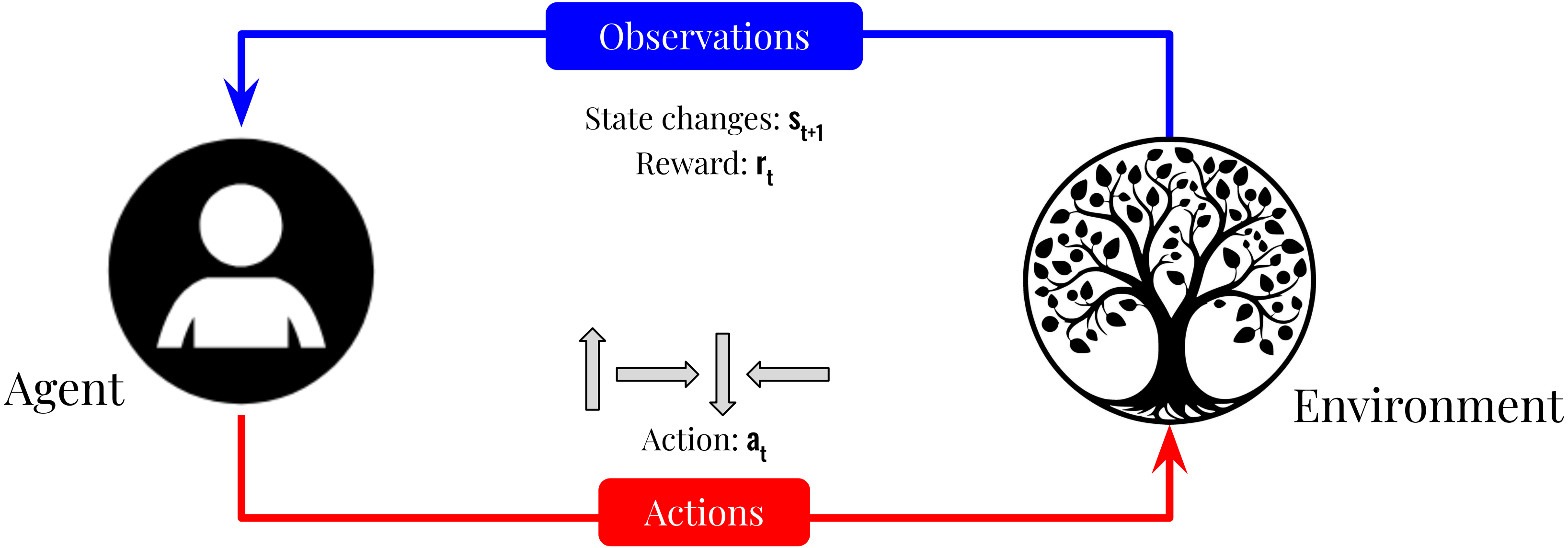
Reinforcement Learning



Total Reward
(**Return**)

$$R_t = \sum_{i=t}^{\infty} r_i = r_t + r_{t+1} + \ldots + r_{t+n} + \ldots$$

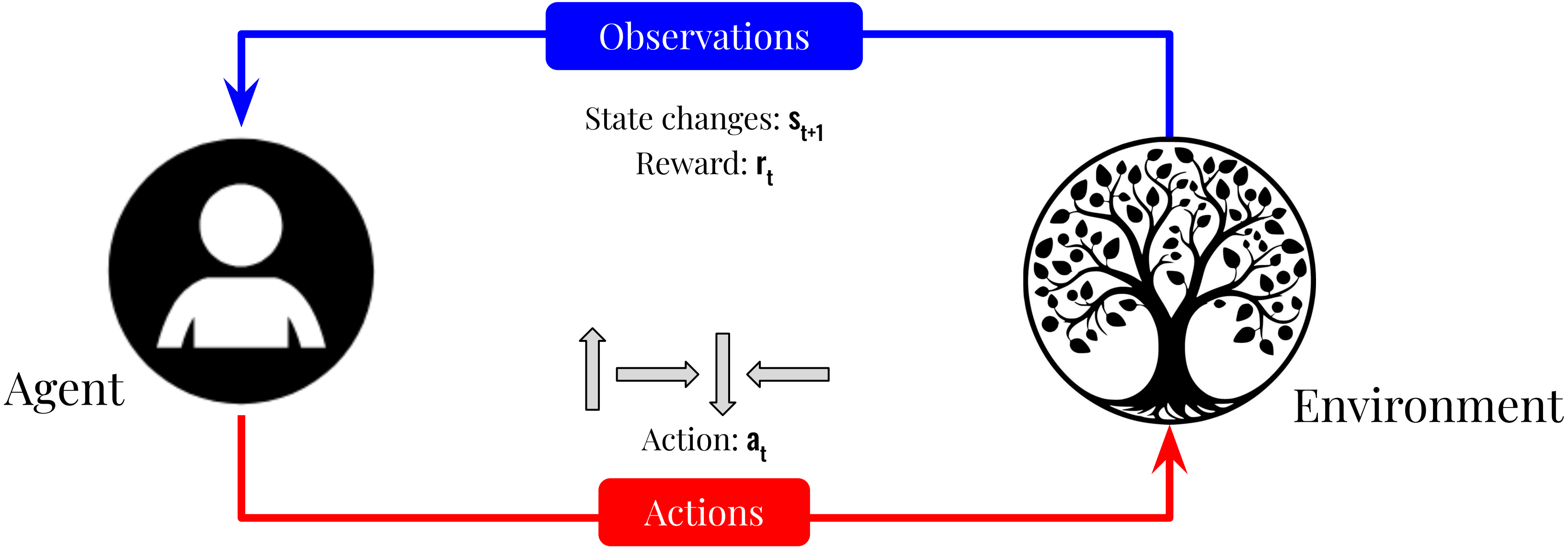
Reinforcement Learning



Discounted
Total Reward
(**Return**)

$$R_t = \sum_{i=t}^{\infty} \gamma^i r_i$$

Reinforcement Learning



Discounted Total Reward (**Return**)

$$R_t = \sum_{i=t}^{\infty} \gamma^i r_i = \gamma^t r_t + \gamma^{t+1} r_{t+1} + \ldots + \gamma^{t+n} r_{t+n} + \ldots$$

Defining the Q-function

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots$$

Total reward, R_t is the discounted sum of all rewards obtained from time ***t***.

$$Q(s_t, a_t) = E[R_t \mid s_t, a_t]$$

The Q-function captures the expected total future reward an agent in state, **s**, can perceive by executing a certain action, **a**

How to take action given a Q-function?

$$Q(s_t, a_t) = E[R_t \mid s_t, a_t]$$

(state, action)

The agent needs a **policy** $\boldsymbol{\pi(s)}$, to infer the **best action to take** at its state, s

Strategy: the policy should choose an action that maximizes future reward

$$\pi^*(s) = \arg\max_a Q(s, a)$$

Deep Reinforcement Learning Algorithms

Value Learning

Find **Q(s,a)**

$$\mathbf{a} = \underset{\mathbf{a}}{\mathbf{argmax}}\, \mathbf{Q(s,a)}$$

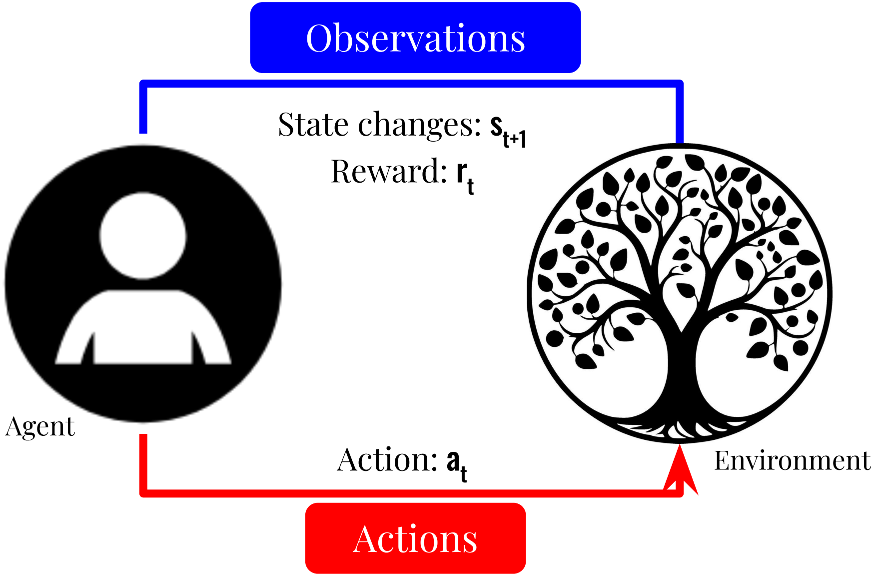
Policy Learning

Find $\boldsymbol{\pi}\mathbf{(s)}$

Sample $\mathbf{a} \sim \boldsymbol{\pi}\mathbf{(s)}$

Training Policy Gradient

Reinforcement Learning Loop



Case Study – Self Driving Car

Agent:	vehicle
State:	camera, lidar, etc
Action:	steering wheel angle
Reward:	distance traveled

Get in touch!

https://www.linkedin.com/in/keivalya/

@keivalya

keivalya.pandya@bvmengineering.ac.in