Deep Reinforcement Learning (DRL)

A gentle introduction to using DRL in robotics

@ENSEIRB-MATMECA – Bordeaux INP

Jean-Luc.Charles@mailo.com

December 2023

An gentle introduction to using DRL in Robotics

An introduction to get familiarised with...

- Unsupervised / Supervised / Reinforcement learning.
- DRL Training & operating of a PPO neural network.
- Stable-Baselines3 & Gymnasium for DRL training with Python.
- Application of DRL to robotics.

An gentle introduction to using DRL in Robotics

An introduction to get familiarised with...

- Unsupervised / Supervised / Reinforcement learning.
- DRL Training & operating of a PPO neural network.
- Stable-Baselines3 & Gymnasium for DRL training with Python.
- Application of DRL to robotics.

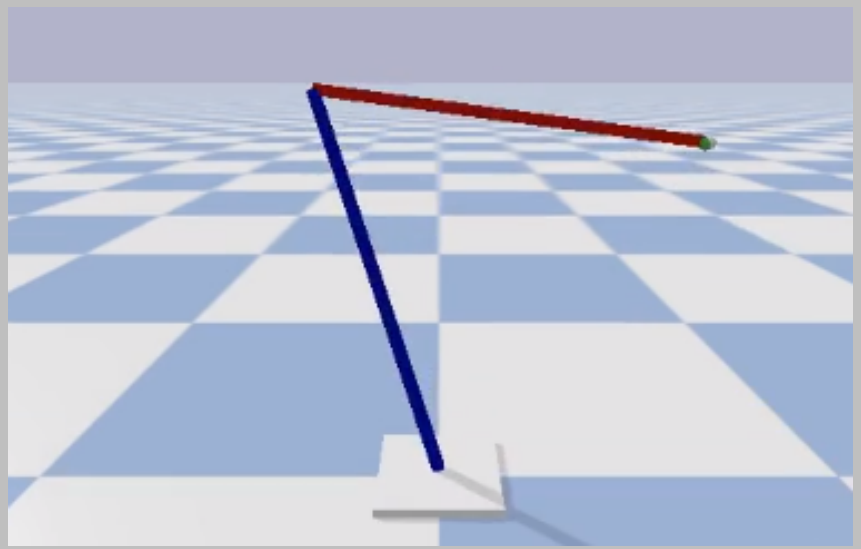
My profile

- I've been teaching Python programming (OO, Data processing & Scientific programing) at ENSAM
- I started to get interested in AI & ML in 2015
- Now I'm self-employed Data Scientist working on AI & ML: studies & practical work (ENSAM, ENSEIRB, ENSPIMA, PPU...).

DRL Practical Work (6h + personal work)

DRL practical work

- Use PyBullet to simulate a robot arm (2 DOF) based on its URDF model (same robot as in your previous *traditional control* session).
- Train a PPO (Proximal Policy Optimisation) neural network to drive the displacement of the end effector of the robot: to follow the green target moving on a diamond trajectory...



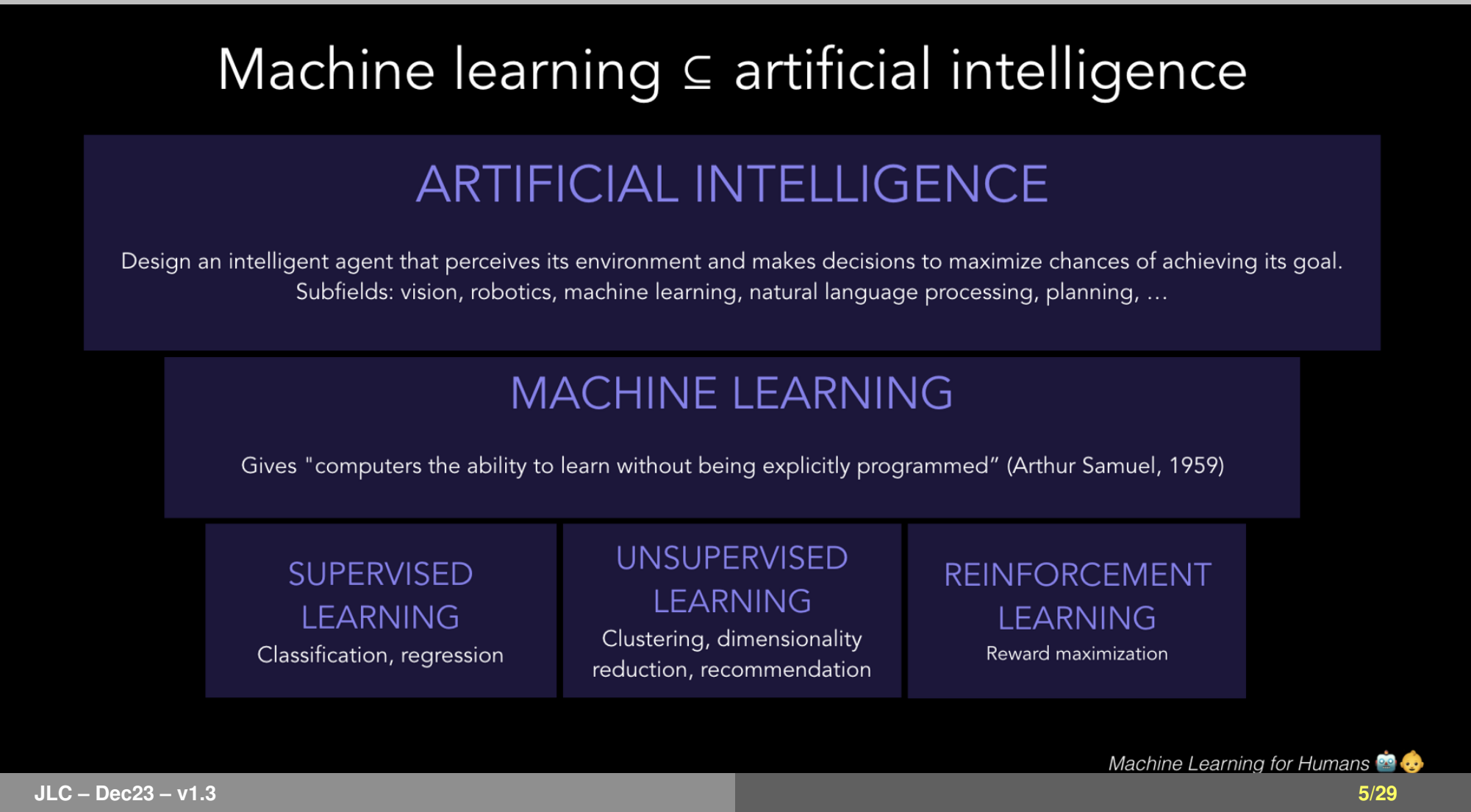
DRL Practical Work (6h + personal work)

Resources

- An introductory lesson (2h)
- Practical work on **your laptop** (Python, Jupyter notebooks...)
  4h + personnal work for training the network...
- Dedicated maintained GitHub repository (all the course material: PDF, notebooks...):
  github.com/cjlux/DRL_at_ENSEIRB-MATMECA

Machine Learning and AI

Page from medium.com/machine-learning-for-humans/...



Machine Learning for Humans

Branches of Machine Learning

Supervised learning

Labeled datasets are used to train algorithms:

- **Classification**
  - Images classification
  - Objects detection in images
  - Speech recognition...
- **Regression**
  - Predict a value...
- **Anomaly detection**
  - Spam detection
  - Manufacturing: finding known (learned) defects
  - Weather prediction
  - Diseases classification...
- . . .

Branches of Machine Learning

Unsupervised learning

Analyze and cluster **unlabeled datasets**:

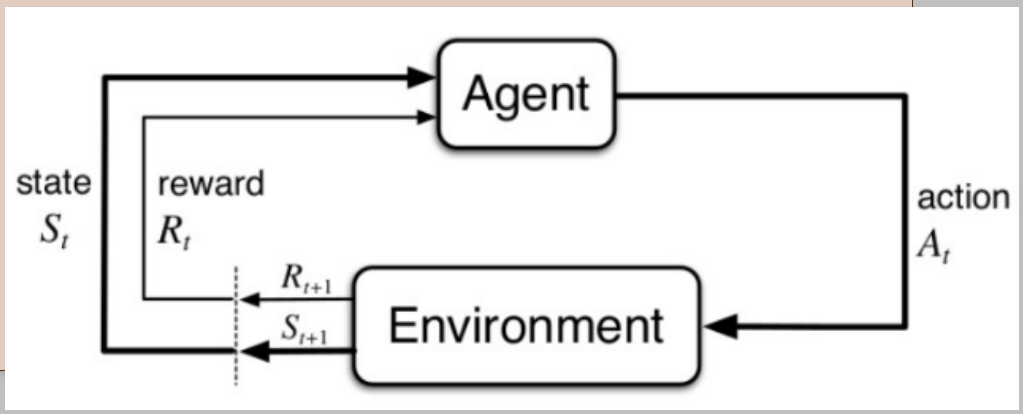
- **Clustering** & **Grouping**
  - Data mining, web data grouping, news grouping...
  - Market segmentation
  - Astronomical data analysis...
- **Anomaly Detection**
  - Fraud detecion
  - Manufacturing: finding defects even new ones
  - Monitoring abnormal activity: failure, hacker, fraud...
  - Fake account on Internet...
- **Dimensionality reduction**
  - Compress data using fewer numbers...
- . . .

Branches of Machine Learning

Deep Reinforcement Learning DRL

An Agent (the Neural Network) learns how to drive an environment by maximising a **reward**:

- **Control/command**
  - Controlling robots, drones, mecatronic systems
  - Factory optimization
  - Financial (stock) trading...
- **Decision making**
  - games (video games)
  - financial analysis...
- ...



[source: The Reinforcement Learning framework, Sutton & Barto, 2018]

Various approaches for ML algorithms

Supervised learning:

- Neural Networks
- Bayesian inference
- Random forest
- Decision Tree
- Support Vector Machine (SVM)
- K-Nearest Neighbor
- Linear regression
- Logistic regression
- ...

Unsupervised learning:

- Neural Networks
- Principal Composant Analysis
- Singular Value Decomposition
- K-mean & Prob. clustering
- ...

Reinforcement learning:

- Neural Networks (Q-learning, Actor-Critic, DDPG, PPO...)
- Monte Carlo
- SARSA
- ...

Various approaches for ML algorithms

Supervised learning:

- Neural Networks

Unsupervised learning:

- Neural Networks

Reinforcement learning:

- Neural Networks

Some applications of ML in *Computer vision*

Computer Vision

- Image Classification

Some applications of ML in *Computer vision*

Computer Vision

- Image Classification
- Object Detection

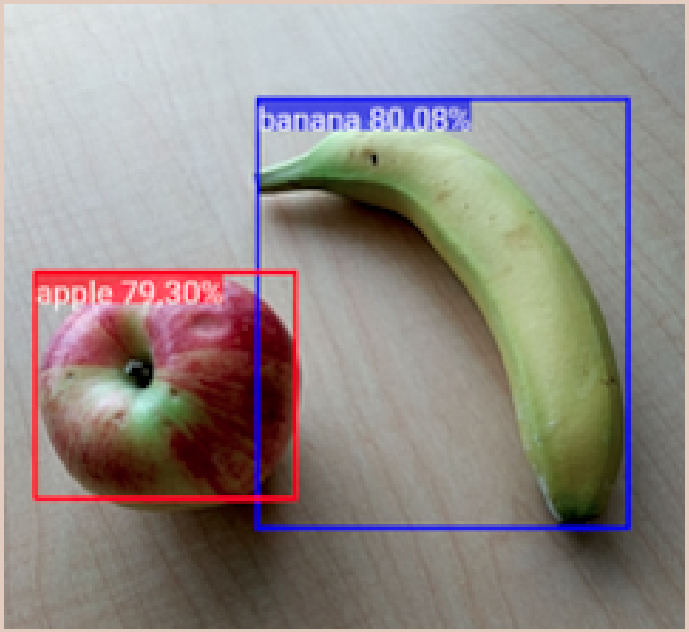


Image credit: Tensorflow

Some applications of ML in *Computer vision*

Computer Vision

- Image Classification
- Object Detection
- (Semantic) Segmentation

Some applications of ML in *Computer vision*

Computer Vision

- Image Classification
- Object Detection
- (Semantic) Segmentation
- Image Generation
  Les 10 Meilleurs Générateurs d'Images

Image credit: stylegan

Some applications of ML in *Computer vision*

Computer Vision

- Image Classification
- Object Detection
- (Semantic) Segmentation
- Image Generation
  Les 10 Meilleurs Générateurs d'Images
- Pose Estimation
- ...

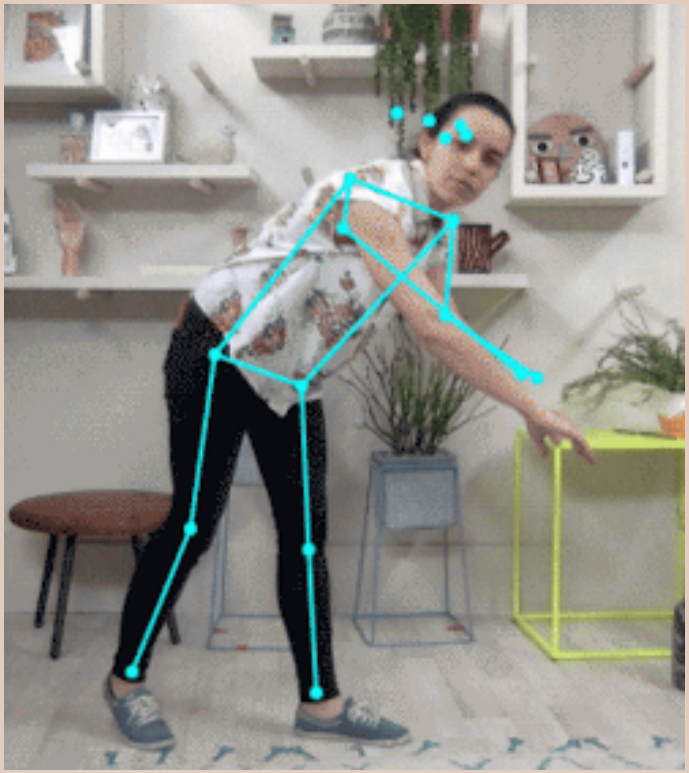


Image credit: Tensorflow-Pose Estimation

Neural Network Architectures

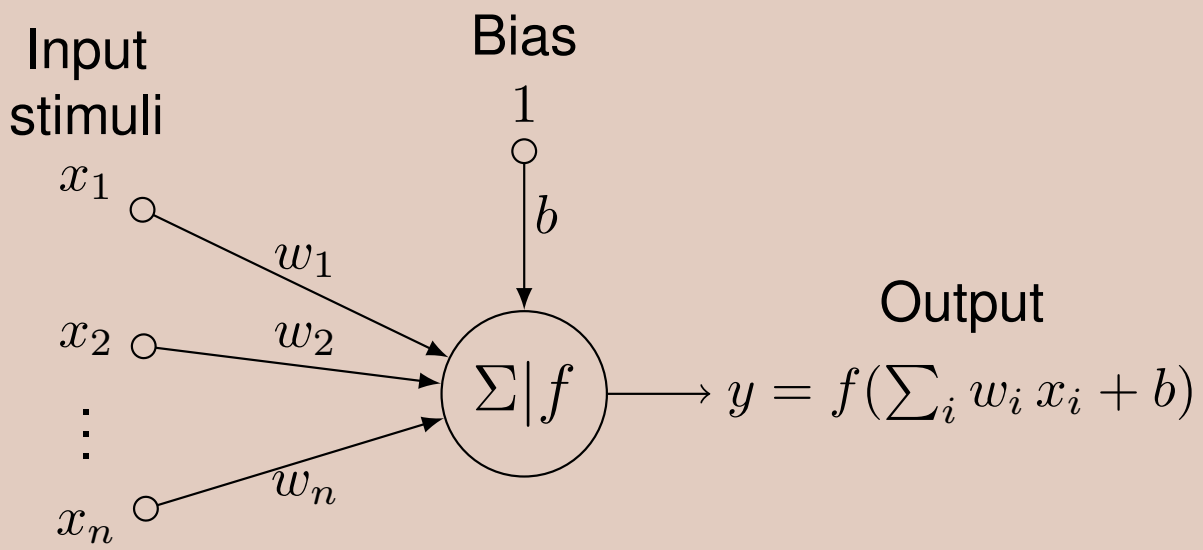
Many NN architectures for many applications (non-exhaustive list):

- **Feed Forward**: the simplest architecture made of successive layers of neurones, with *Feed Forward* and *Back Propagation* algorithms.
- **Convolutional** (CNN): Mostly used for analyzing and classifying images.
- **Recurrent** (RNN): Used for time series, like the Long Short-Term Memory (LSTM) algorithm.
- **Transformers** : Recently used for Natural Language Processing and then for image classification.
- **Auto Encoder** (AEN): Dimensionality reduction, Feature extraction, Denoising of data/images, Inputing missing data.
- **Generative Adversarial** (GAN): to generate text, images, music...
- **Large Language Model** (LLM): read text, sound, write books, images, speak, make music ...ChatGPT

[Synthetic Graphical chart: from Saul Dobilas on Medium]

Artificial Neural Networks

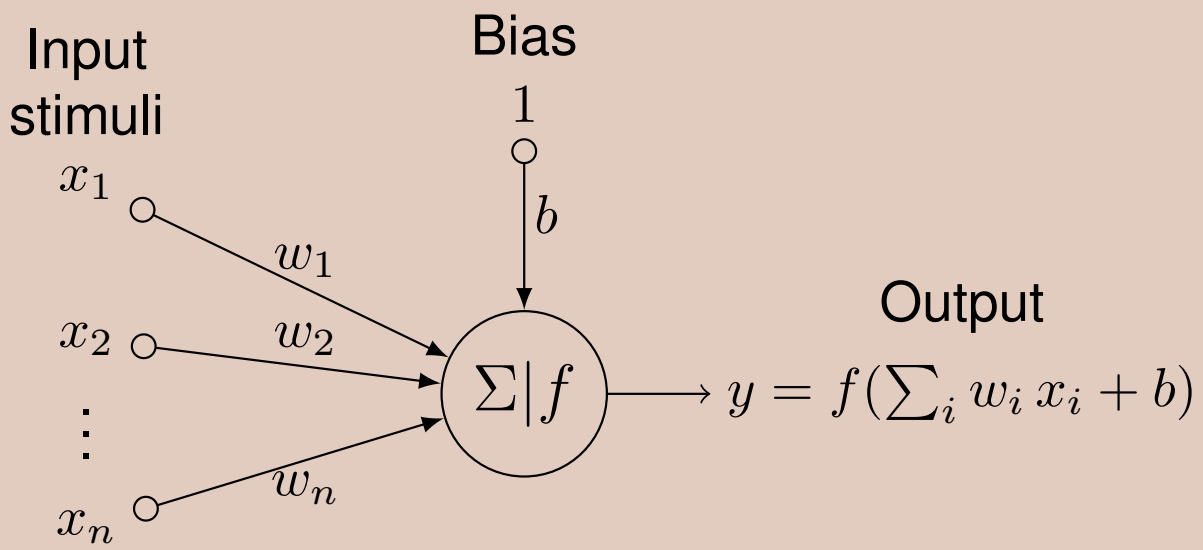
The Artificial neuron model



An **artificial neuron**:

Artificial Neural Networks

The Artificial neuron model

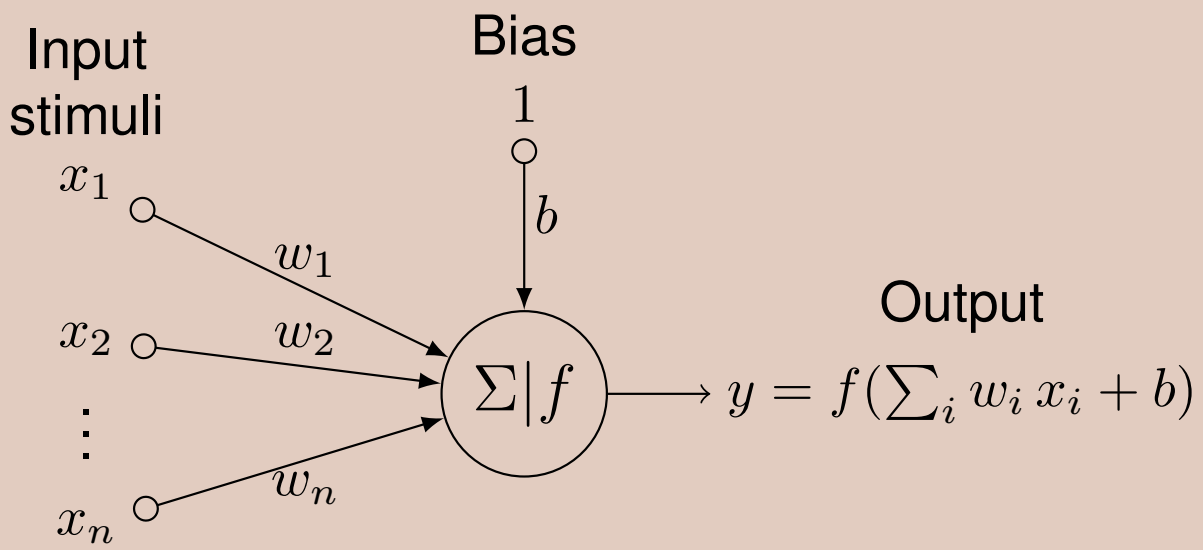


An **artificial neuron**:

- receives the input stimuli $(x_i)_{i=1..n}$ with **weights** (w_i)

Artificial Neural Networks

The Artificial neuron model

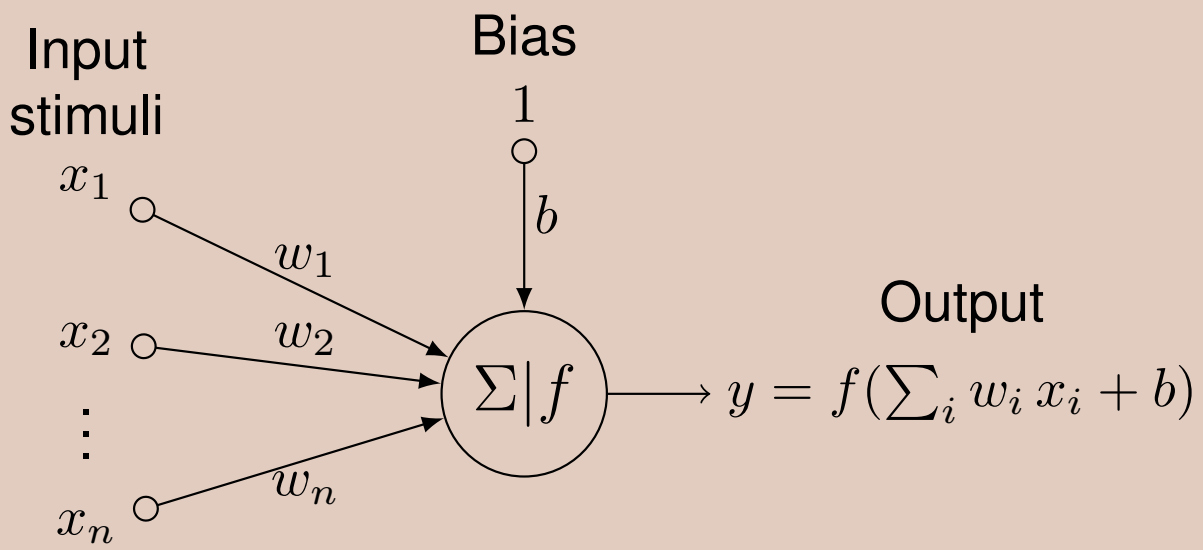


An **artificial neuron**:

- receives the input stimuli $(x_i)_{i=1..n}$ with **weights** (w_i)
- computes the **weighted sum** of the input: $\sum_i w_i\, x_i + b$

Artificial Neural Networks

The Artificial neuron model

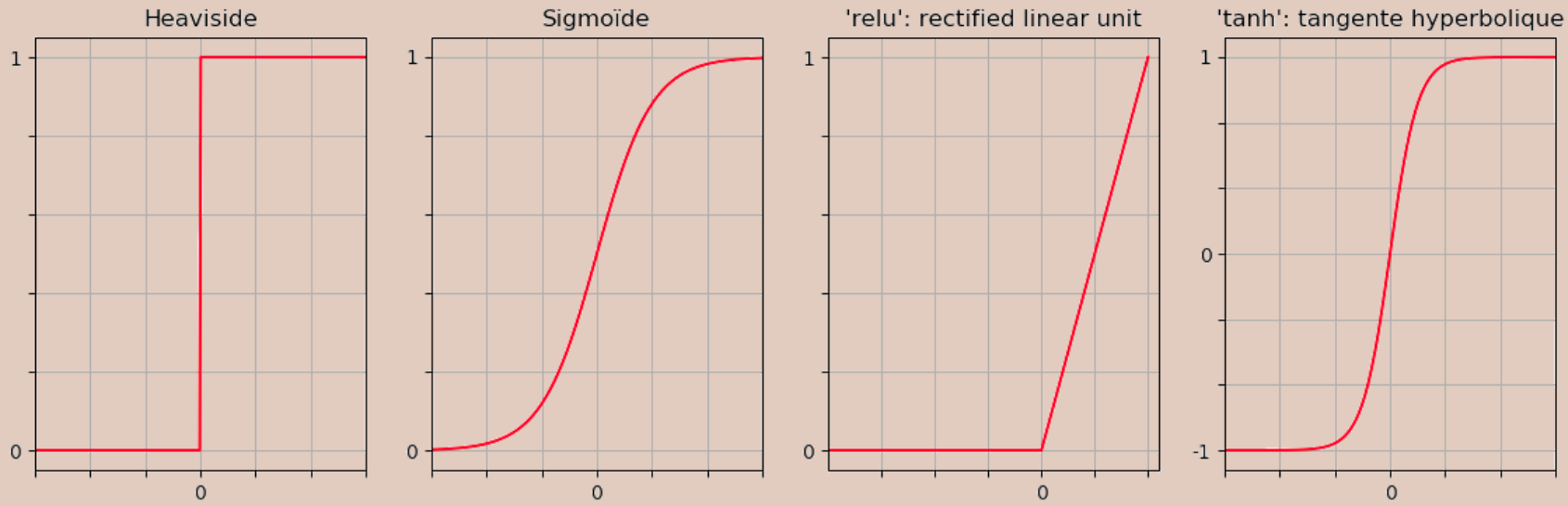


An **artificial neuron**:

- receives the input stimuli $(x_i)_{i=1..n}$ with **weights** (w_i)
- computes the **weighted sum** of the input: $\sum_i w_i\, x_i + b$
- computes its activation $f(\sum_i w_i\, x_i + b)$, using a non-linear **activation function** f.

Artificial Neural Networks

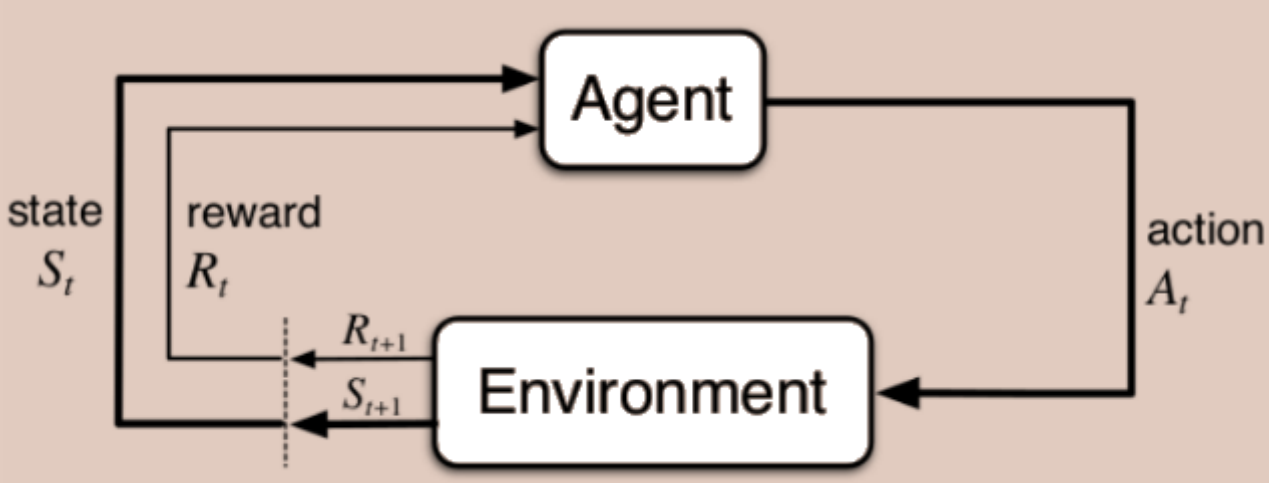
Common activation functions



- Introduces a non-linear behavior
- Sets the range of the neuron output: $[-1, 1]$, $[0, 1]$, $[0, \infty[$...
- The bias b sets the activation threshold of the neuron.

Reinforcement Learning

RL main ingredients



[Source: *Reinforcement Learning - An Introduction*, Richard S. Sutton and Andrew G. Barto]

The **Agent** learns how to take the right **action** on the **Environment** (the system to be controlled) in order to maximize its expected long-term cumulative **reward**, also called the **return**.

Reinforcement Learning

The **Agent**

- At each time step the **Agent**:
  - gets an observation o_t of the Environment state s_t
  - executes an action a_t on the Environment
  - gets a reward (scalar feedback signal) from the Environment

Reinforcement Learning

The **Agent**

- At each time step the **Agent**:
  - gets an observation o_t of the Environment state s_t
  - executes an action a_t on the Environment
  - gets a reward (scalar feedback signal) from the Environment
- Agent's **action space** can be:
  - **discrete**: on/off, left/right...
  - **continuous**: force/velocity applied....

Reinforcement Learning

The **Agent**

- At each time step the **Agent**:
  - gets an observation o_t of the Environment state s_t
  - executes an action a_t on the Environment
  - gets a reward (scalar feedback signal) from the Environment
- Agent's **action space** can be:
  - **discrete**: on/off, left/right...
  - **continuous**: force/velocity applied....
- Agent's **Goal**: to maximize the expected **return** of the Environment (return = long-term cumulative reward)

Reinforcement Learning

The **Environment**

The **Environment** is what the Agent wants to monitor:

Reinforcement Learning

The **Environment**

The **Environment** is what the Agent wants to monitor:

- Receives an **action** a_t from the Agent

Reinforcement Learning

The **Environment**

The **Environment** is what the Agent wants to monitor:

- Receives an **action** a_t from the Agent
- Takes a new **state** s_{t+1} under the Agent's action a_t

Reinforcement Learning

The **Environment**

The **Environment** is what the Agent wants to monitor:

- Receives an **action** a_t from the Agent
- Takes a new **state** s_{t+1} under the Agent's action a_t
- Emits a **reward** r_{t+1} (scalar):
  - maps the pair (a_t, s_{t+1}) to a scalar (number) giving the desirability of the new state s_{t+1}
  - sent by the Environment in response to the agent action
- The Environment state is modelized as a Partially Observable Markov Process (POMP)

Reinforcement Learning

The **Environment**

The **Environment** is what the Agent wants to monitor:

- Receives an **action** a_t from the Agent
- Takes a new **state** s_{t+1} under the Agent's action a_t
- Emits a **reward** r_{t+1} (scalar):
  - maps the pair (a_t, s_{t+1}) to a scalar (number) giving the desirability of the new state s_{t+1}
  - sent by the Environment in response to the agent action
- The Environment state is modelized as a Partially Observable Markov Process (POMP)

☹ The design of an **efficient reward function** is one of the difficult task to achieve in RL...

Reinforcement Learning

What makes RL different from other Machine Learning paradigms?

- There is no supervisor, only a reward signal

Reinforcement Learning

What makes RL different from other Machine Learning paradigms?

- There is no supervisor, only a reward signal
- Feedback is delayed, not instantaneous

Reinforcement Learning

What makes RL different from other Machine Learning paradigms?

- There is no supervisor, only a reward signal
- Feedback is delayed, not instantaneous
- Time really matters (training is not based on a static data set)

Reinforcement Learning

What makes RL different from other Machine Learning paradigms?

- There is no supervisor, only a reward signal
- Feedback is delayed, not instantaneous
- Time really matters (training is not based on a static data set)
- Agent's actions affect the subsequent data it receives.

RL: Model-free versus Model-based algorithms

Model-free RL algorithms

- No explicit representation of the environment
- Learning rely only on experiences using **trial and error**
- The most commonly used (car driving, robot control...)
- **sample inefficient** $\rightsquigarrow$ needs millions of samples...
- Exples: Monte Carlo, SARSA, Q-learning, Actor-Critic....

RL: Model-free versus Model-based algorithms

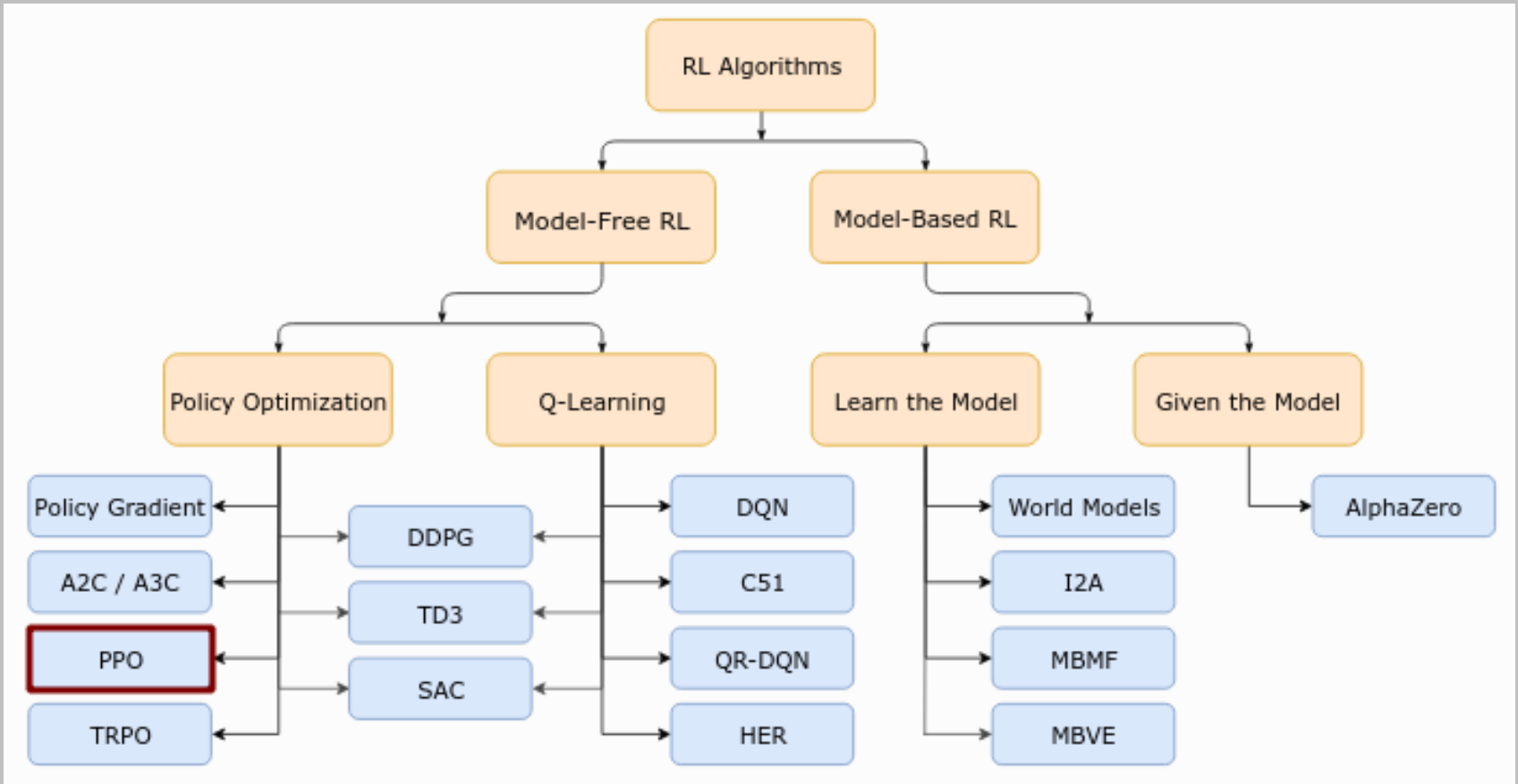
Model-free RL algorithms

- No explicit representation of the environment
- Learning rely only on experiences using **trial and error**
- The most commonly used (car driving, robot control...)
- **sample inefficient** $\rightsquigarrow$ needs millions of samples...
- Exples: Monte Carlo, SARSA, Q-learning, Actor-Critic....

Model-based RL algorithms

- The agent has access to a **model** of the environment and uses this model to predict future states and rewards
- A true model of the environment is usually not available
- Applies to environment like Chess, Go...

A non-exhaustive, but useful taxonomy of RL algorithms (Source: spinningup.openai.com)



PPO : Proximal Policy Optimization algorithm

ref. Proximal Policy Optimization Algorithms, J. Schulman et al. 2017

DRL reference sites

Stable-baselines3

A set of reliable implementations of Reinforcement Learning algorithms in **PyTorch**.

- github.com/DLR-RM/stable-baselines3
- docs:
  spinningup.openai.com
  stable-baselines3.readthedocs.io
- Easy installation as a Python module:
  `pip install stable-baselines3`

DRL reference sites

Gymnasium

- stable-baselines3 works with the Gymnasium workbench (maintained fork of OpenAI's Gym)

DRL reference sites

Gymnasium

- stable-baselines3 works with the Gymnasium workbench (maintained fork of OpenAI's Gym)
- Main class from `Gymnasium`: the abstract class Env

DRL reference sites

Gymnasium

- stable-baselines3 works with the Gymnasium workbench (maintained fork of OpenAI's Gym)
- Main class from `Gymnasium`: the abstract class Env
  - Derive this class and re-define its main `methods`:

`step`	updates an environment with actions
`reset`	resets the environment, required before calling step.
`render`	renders the environments to help visualisation ($\rightsquigarrow$ we will use PyBullet rendering instead)
`close`	closes the environment.

DRL reference sites

Gymnasium

- stable-baselines3 works with the Gymnasium workbench (maintained fork of OpenAI's Gym)
- Main class from `Gymnasium`: the abstract class Env
  - Derive this class and re-define its main `methods`:

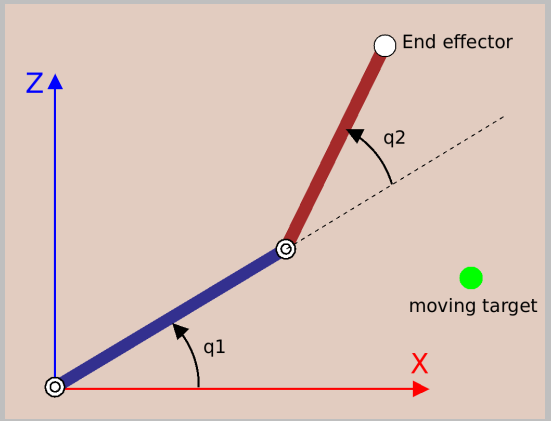
    `step` updates an environment with actions
    `reset` resets the environment, required before calling step.
    `render` renders the environments to help visualisation
    ($\rightsquigarrow$ we will use PyBullet rendering instead)
    `close` closes the environment.

  - define/use the `attributes`:

    `action_space` the Space object defining valid actions
    `observation_space` the Space object defining valid observations
    `np_random` the random number generator for the environment.

The DRL practical work 1/2

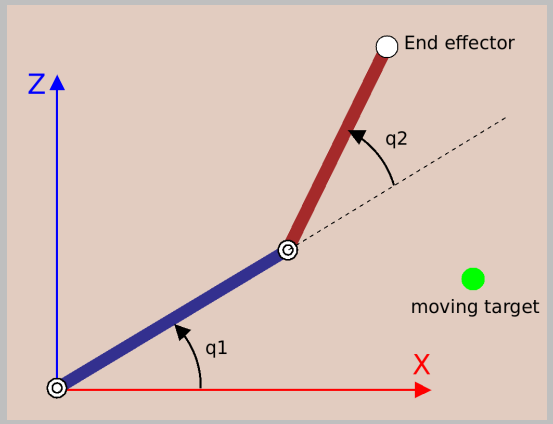
- Goal: train a **PPO** neural network to drive the end effector of a 2 DOF robot arm (follow the green moving target...)



L1=L2=1m; M1=M2=1kg

The DRL practical work 1/2

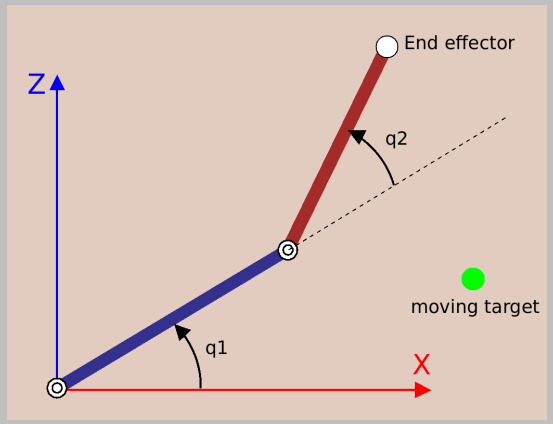
- Goal: train a **PPO** neural network to drive the end effector of a 2 DOF robot arm (follow the green moving target...)
- The training of the neural network involves the simulation of the robot with the PyBullet (thanks to its URDF file).



L1=L2=1m; M1=M2=1kg

The DRL practical work 1/2

- Goal: train a **PPO** neural network to drive the end effector of a 2 DOF robot arm (follow the green moving target...)
- The training of the neural network involves the simulation of the robot with the PyBullet (thanks to its URDF file).

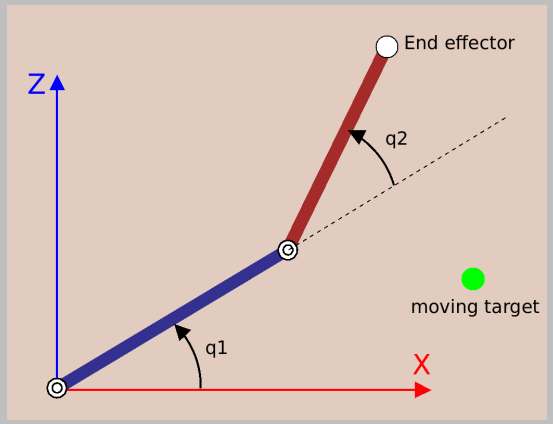


L1=L2=1m; M1=M2=1kg

- The PyTorch Python module provides classes to instanciate and train the PPO neural network.

The DRL practical work 1/2

- Goal: train a **PPO** neural network to drive the end effector of a 2 DOF robot arm (follow the green moving target...)
- The training of the neural network involves the simulation of the robot with the PyBullet (thanks to its URDF file).



L1=L2=1m; M1=M2=1kg

- The PyTorch Python module provides classes to instanciate and train the PPO neural network.
- To ensure the robustness, repeatability & maintainability of Python developments, the work is done within a **Python Virtual Environment** (PVE).

The DRL practical work 1/2

1. Clone the repository (or download its ZIP archive)
   github.com/cjlux/DRL_at_ENSEIRB-MATMECA

The DRL practical work 1/2

1. Clone the repository (or download its ZIP archive)
   github.com/cjlux/DRL_at_ENSEIRB-MATMECA

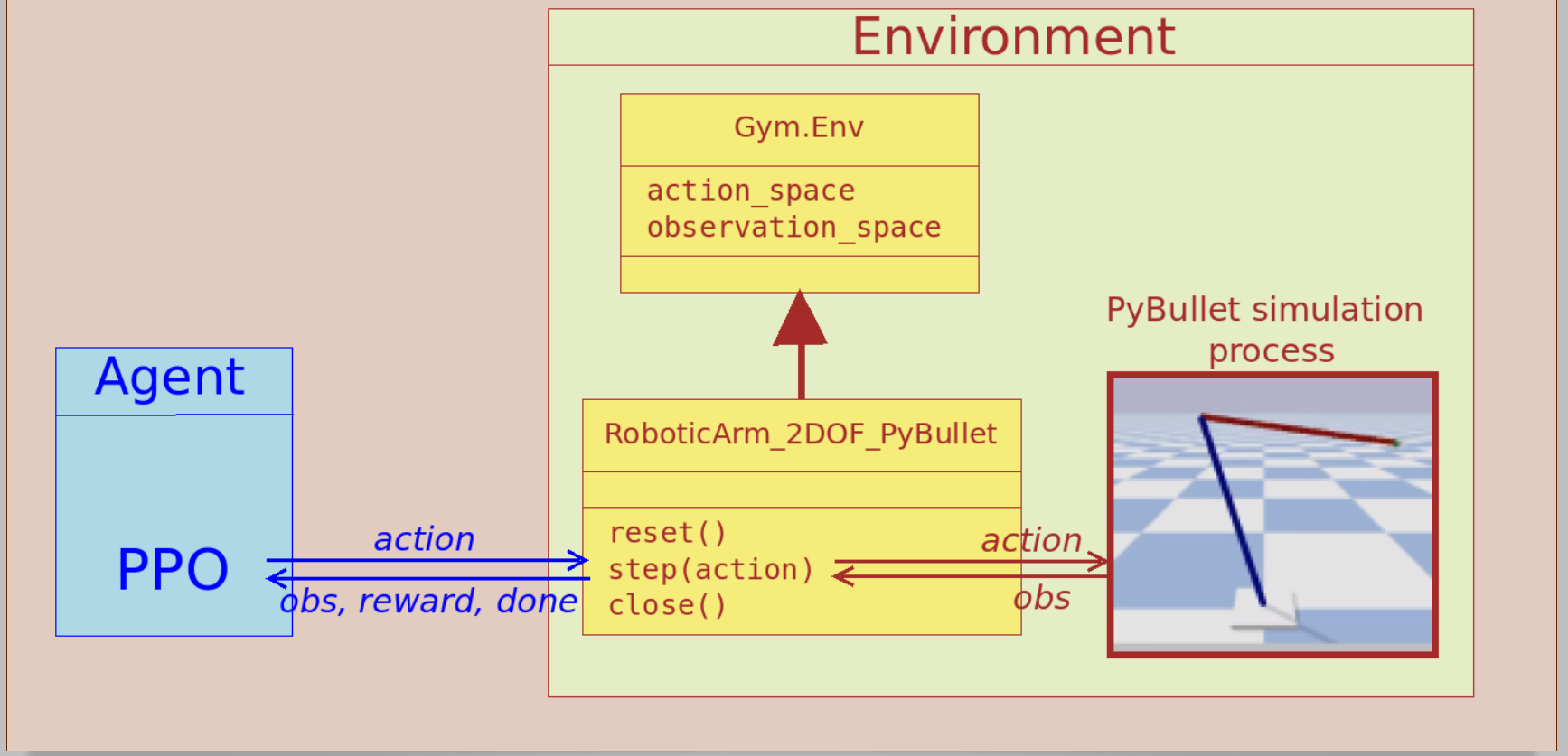
2. Follow `Create_PVE.pdf` to create, populate & activate the PVE `drl`

The DRL practical work 1/2

1. Clone the repository (or download its ZIP archive) github.com/cjlux/DRL_at_ENSEIRB-MATMECA
2. Follow `Create_PVE.pdf` to create, populate & activate the PVE `drl`
3. Within the PVE `drl` run `jupyter notebook` or `jupyter lab` (or whatever IDE you want that is 'PVE aware') to load and run the notebook `1-Getting_started.ipynb`.

The DRL practical work 2/2

PPO training and Gym.Env



The DRL practical work 2/2

The class: **RoboticArm_2DOF_PyBullet**

The **RoboticArm_2DOF_PyBullet** class defines the Environment corresponding to the robot.

- State of the environment:
  $(q_1, \dot{q}_1, q_2, \dot{q}_2, x_{targ}, z_{targ}, x_{ee}, z_{ee})$
  targ: target, ee: End Effector
- Action given by the agent: $(\text{Torque}_{motor1}, \text{Torque}_{motor2})$
- End of episode:
  - terminated: if the environment succeded (the robot end effector is close to the green target)
  - truncated: if the environment time step reaches `max_episode_steps`

The DRL practical work 2/2

PPO training and the Gym.Env

Training is done over `tot_steps` time steps:

PPO hyperparameters:

```
policy = 'MlpPolicy'
tot_steps  = 5000000
save_freq  = 100000
nb_steps   = 2048
nb_epochs  = 10
batch_size = 256
headless   = True
```

The DRL practical work 2/2

PPO hyperparameters:

```
policy = 'MlpPolicy'
tot_steps  = 5000000
save_freq  = 100000
nb_steps   = 2048
nb_epochs  = 10
batch_size = 256
headless   = True
```

PPO training and the Gym.Env

Training is done over `tot_steps` time steps:

- Run training episodes...
  - the Agent chooses and sends `action` to the Environment
  - the Environment steps and returns its new state & reward:
    `(state, reward, terminated, truncated, ....)`

The DRL practical work 2/2

PPO hyperparameters:

```
policy = 'MlpPolicy'
tot_steps  = 5000000
save_freq  = 100000
nb_steps   = 2048
nb_epochs  = 10
batch_size = 256
headless   = True
```

PPO training and the Gym.Env

Training is done over `tot_steps` time steps:

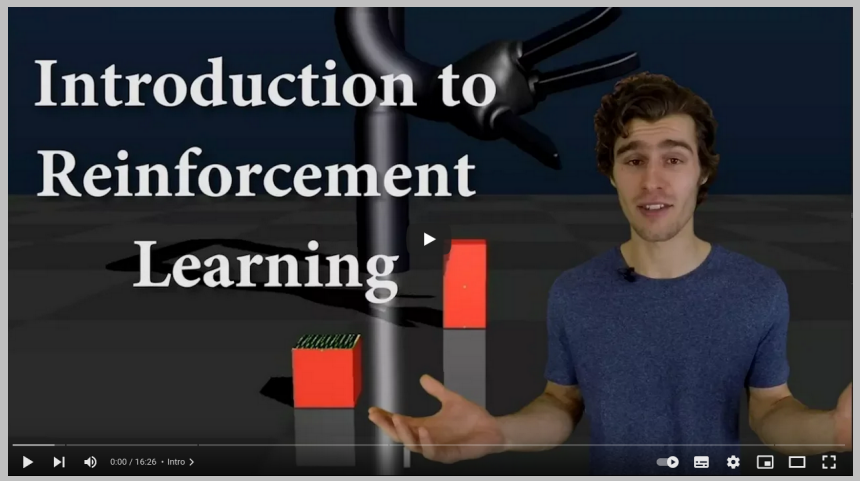
- Run training episodes...
  - the Agent chooses and sends `action` to the Environment
  - the Environment steps and returns its new state & reward: `(state, reward, terminated, truncated, ...)`
- Every `nb_steps` steps PPO is updated `nb_epochs` times using `batch_size` experiences `(state, action, newstate, reward)`.

Now, within the PVE `drl` load and run the notebook `2-DRL_training.ipynb`.

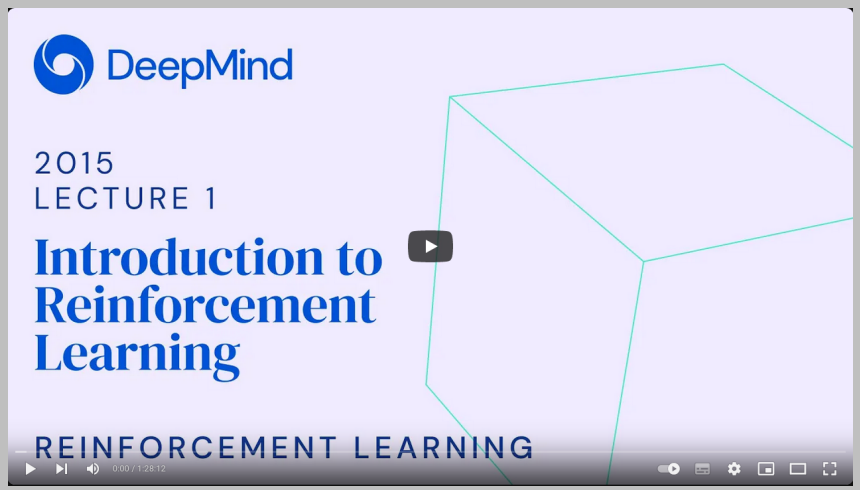
Papers – Books

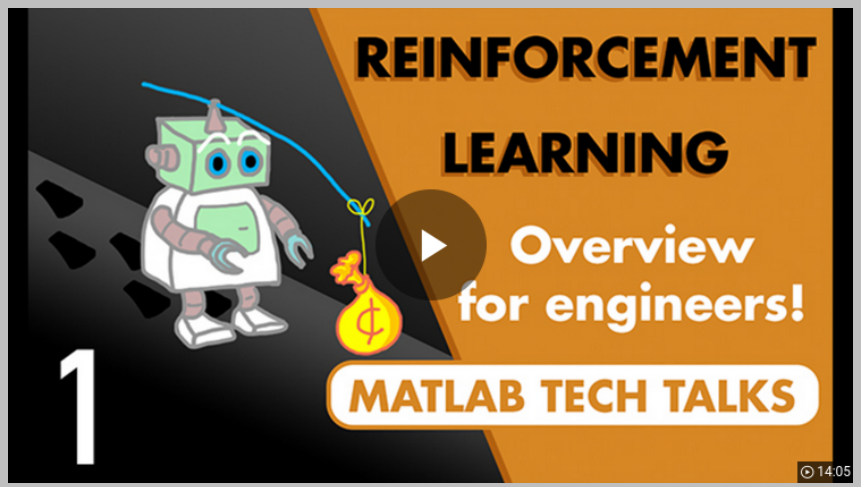
- [1] *Reinforcement Learning - An Introduction*, Richard S. Sutton, Andrew G. Barto, (2018), MIT Press, ISBN 0262039249 - Free PDF here
- [2] Stable-Baselines3: Reliable Reinforcement Learning Implementations, Journal of Machine Learning Research 22 (2021), pages 1-8
- [3] *Artificial Intelligence: A Modern Approach (4th Edition)*, By Stuart Russell & Peter Norvig. Pearson, 2020. ISBN 978-0134610993. aima.cs.berkeley.edu
- [4] *Intelligence artificielle – Une approche moderne – 4e éd.*, By Stuart Russell & Peter Norvig. Translated by L. Miclet, F. Popineau, & C. Cadet. Paris: Pearson Education France, 2021. ISBN 978-2326002210.
- [5] *Stanford Encyclopedia of Philosophy*, plato.stanford.edu/entries/artificial-intelligence
- [6] *Deep Learning.*, Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016), MIT Pres, ISBN 9780262035613

Videos - DRL


Introduction to Reinforcement Learning


Proximal Policy Optimization
We're releasing a new class of reinforcement learning algorithms, Proximal Policy Optimization (PPO), which perform comparably or better than state-of-the-art approaches while being much simpler to implement and tune. PPO has become the default reinforcement learning algorithm at OpenAI because of its ease of use and good performance.
Proximal Policy Optimization


DeepMind
2015
LECTURE 1
Introduction to Reinforcement Learning
REINFORCEMENT LEARNING


REINFORCEMENT LEARNING
Overview for engineers!
1
MATLAB TECH TALKS
14:05

Videos - Introducing ML