

Deep Reinforcement Learning (DRL)

An introduction to using DRL in
robotics

@ENSEIRB-MATMECA – Bordeaux INP


Jean-Luc.Charles@ENSAM.EU



November 2022



An introduction to using DRL in Robotics

- 
- An introduction to get familiarised with...**
- Unsupervised / Supervised / **Reinforcement** learning.
 - DRL Training & operating of Artificial Neural Networks
 - PyTorch module for DRL training.
 - Applications of DRL to robotics.

An introduction to using DRL in Robotics



An introduction to get familiarised with...

- Unsupervised / Supervised / **Reinforcement** learning.
- DRL Training & operating of Artificial Neural Networks
- PyTorch module for DRL training.
- Applications of DRL to robotics.



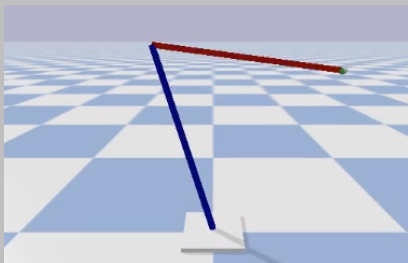
Ressources

- 2 hours of lecture and 1 × practical work Python session (4h) on **your laptop**
- Dedicated github repository with all the course material (PDF, notebooks...)

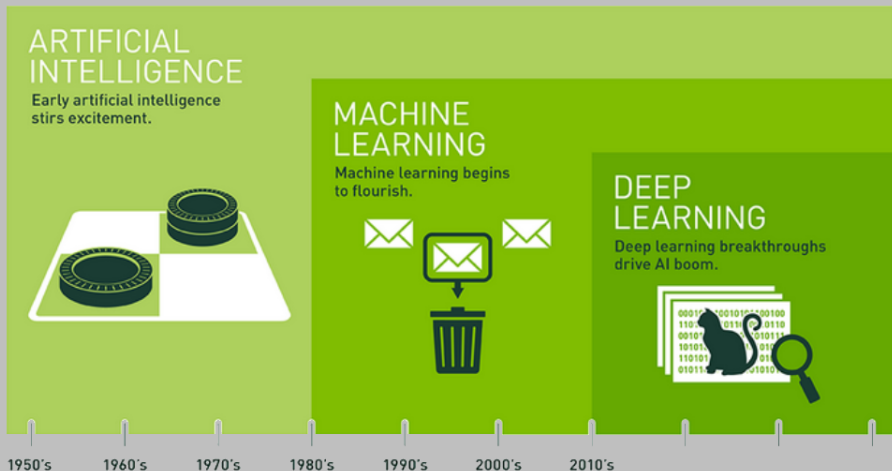
Practical Work: 1 × 4h

DRL practical work:

- Build the [PyBullet](#) simulation of a 2 DOF robot arm based on its URDF description.
- train a PPO (Proximal Policy Optimisation) network to drive the displacement of the end effector of a 2 DOF robot arm.



The historical way...



(from : developer.nvidia.com/deep-learning)

Artificial Intelligence ?

Artificial Intelligence¹: remains an ambiguous term with multiple definitions varying with time:

- *"...the science of making computers do things that require intelligence when done by humans."* [Alan Turing, 1940](#)
- *"the field of study that gives computers the ability to learn without being explicitly programmed."* [Arthur Samuel, 1960](#)
- *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E ."* [Tom Mitchell, 1997](#)
- Notion of *intelligent agent* or *rational agent*
"...agent that acts in such a way as to reach the best solution or, in an uncertain environment, the best predictable solution." [Stuart Russel, Peter Norvig, "Intelligence Artificielle" 2015](#)

¹ first used in 1956 by [John McCarthy](#), researcher at Stanford during the Dartmouth conference

Artificial Intelligences ?

Qualifiers often encountered:

Strong AI

Weak AI

General AI

Narrow AI

Artificial Intelligences ?

Qualifiers often encountered:

Strong AI

- Build systems that think exactly the same way that people do.
- Try also to explain how humans think... Whe are not yet here.

Weak AI

- Build systems that can behave like humans.
- The results will tell us nothing about how humans think.
- We already are there... We use it every day!
(anti-spam, facial/voice recognition, language translation...)

General AI

Narrow AI

Artificial Intelligences ?

Qualifiers often encountered:

Strong AI

- Build systems that think exactly the same way that people do.
- Try also to explain how humans think... We are not yet here.

Weak AI

- Build systems that can behave like humans.
- The results will tell us nothing about how humans think.
- We already are there... We use it every day!
(anti-spam, facial/voice recognition, language translation...)

General AI

- AI systems designed for the ability to reason in general.

Narrow AI

- AI systems designed for specific tasks.

Artificial Intelligence

already a reality:

- Runs in much of our present technology (smartphone apps...)
- Powered by rapid advances in data storage, computer processing power.
- Powered by **free dataset acces via Internet** and **code publishing as open source** environments.
- Rate of acceleration is already astounding.
- Will likely shape our future more powerfully than any other innovation this century.

Machine Learning and AI

Page from [medium.com/machine-learning-for-humans/...](https://medium.com/machine-learning-for-humans/)

Machine learning \subseteq artificial intelligence

ARTIFICIAL INTELLIGENCE

Design an intelligent agent that perceives its environment and makes decisions to maximize chances of achieving its goal.
Subfields: vision, robotics, machine learning, natural language processing, planning, ...

MACHINE LEARNING

Gives "computers the ability to learn without being explicitly programmed" (Arthur Samuel, 1959)

SUPERVISED LEARNING

Classification, regression

UNSUPERVISED LEARNING

Clustering, dimensionality
reduction, recommendation

REINFORCEMENT LEARNING

Reward maximization

Branches of Machine Learning

Supervised learning applications

labeled dataset is used to train algorithms to classify data:

- **Classification**

- Images classification
- Objects detection
- 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 application

Analyze and cluster **unlabeled datasets**:

- **Clustering & Grouping**

- Data mining, web data grouping, news grouping...
- Market segmentation
- DNA grouping
- Astronomical data analysis...

- **Anomaly Detection**

- Fraud detection
- 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

Reinforcement learning

An agent learns how to drive an environment by maximising a **reward**:

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

Various approaches for ML algorithms

Supervised learning:

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

Unsupervised learning:

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

Reinforcement learning:

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

Various approaches for ML algorithms

Supervised learning:

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

Unsupervised learning:

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

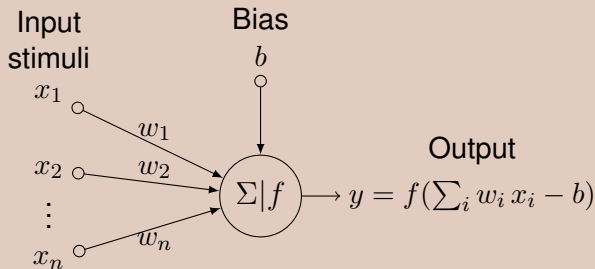
Reinforcement learning:

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

The following deals only with **Artificial Neural Networks**.

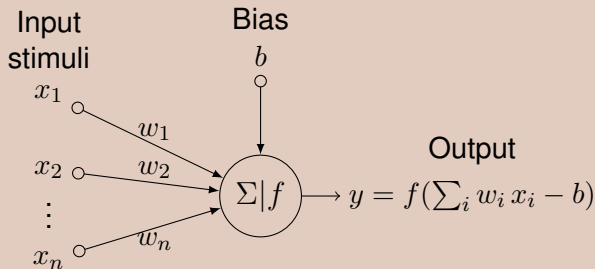
Artificial Neural Networks

The Artificial neuron model



Artificial Neural Networks

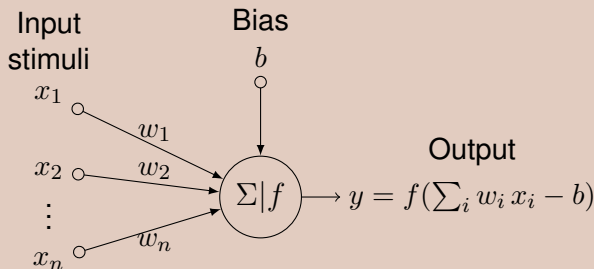
The Artificial neuron model



An **artificial neuron**:

Artificial Neural Networks

The Artificial neuron model

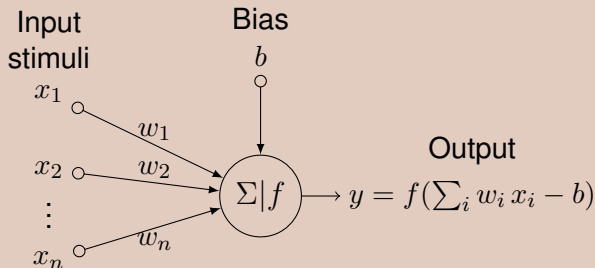


An **artificial neuron**:

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

Artificial Neural Networks

The Artificial neuron model

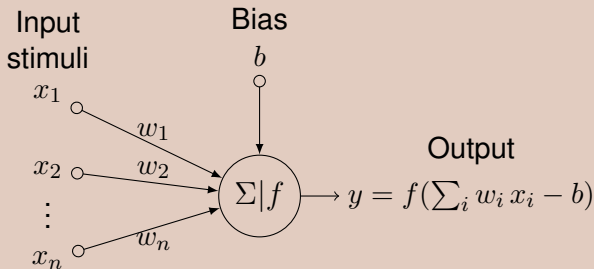


An **artificial neuron**:

- 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

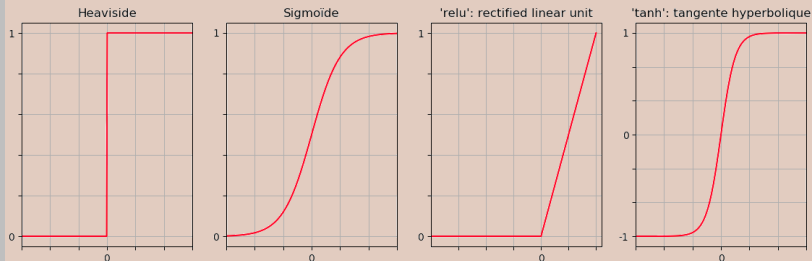


An **artificial neuron**:

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

Artificial Neural Networks

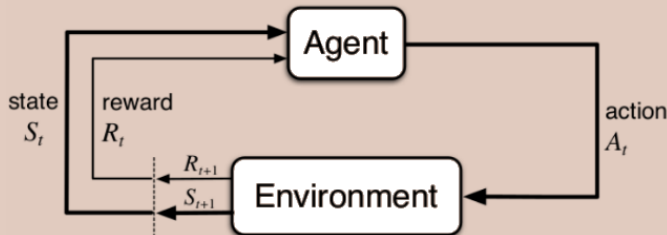
Common activation functions



- 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](#) by Richard S. Sutton and Andrew G. Barto)

The **agent** (the neural network) learns how to take the right **action** on the **environment** (the system to be controlled) in order to maximize its long-term **reward**.

RL ingredients: **Agent**

The **Agent** is the **algorithm**:

RL ingredients: **Agent**

The **Agent** is the **algorithm**:

- Monitors the **Environment**
- Decides which **action** to be taken
- Action can be
 - discrete**: on/off, left/right...
 - continuous**: force/velocity applied....
- Goal: maximize the total reward it receives in the long run.

Discrete versus **continuous** action involves different algorithms for the learning stage

RL ingredients: **Environment**

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

RL ingredients: **reward** funtion

RL ingredients: **Environment**

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

- Receives **actions** from the Agent.
- Takes a new **state** under the Agent's action.
- Gives back its new **state** and computed **reward** to the Agent.
- Modelized as a **Partially Observable Markov Process**.

RL ingredients: **reward** function

RL ingredients: **Environment**

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

- Receives **actions** from the Agent.
- Takes a new **state** under the Agent's action.
- Gives back its new **state** and computed **reward** to the Agent.
- Modelized as a **Partially Observable Markov Process**.

RL ingredients: **reward** function

- Maps each (state, action) pair of the environment to a number indicating the intrinsic desirability of that state.
- The environment sends a **scalar value** as a **reward** in response to the agents's action.

Model-free versus Model-based DRL algorithms

Model-free

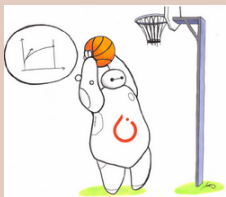
- No explicit representation of the environment
- Learning rely only on experiences using **trial and error**.
- Examples : Monte Carlo, SARSA, Q-learning, and Actor-Critic algorithms.
- Applies to car driving, robot control...

Model-based algorithms

- The agent has access to (or learns) a **model** of the environment.
- Applies to environment like Chess, Go...

RL reference sites

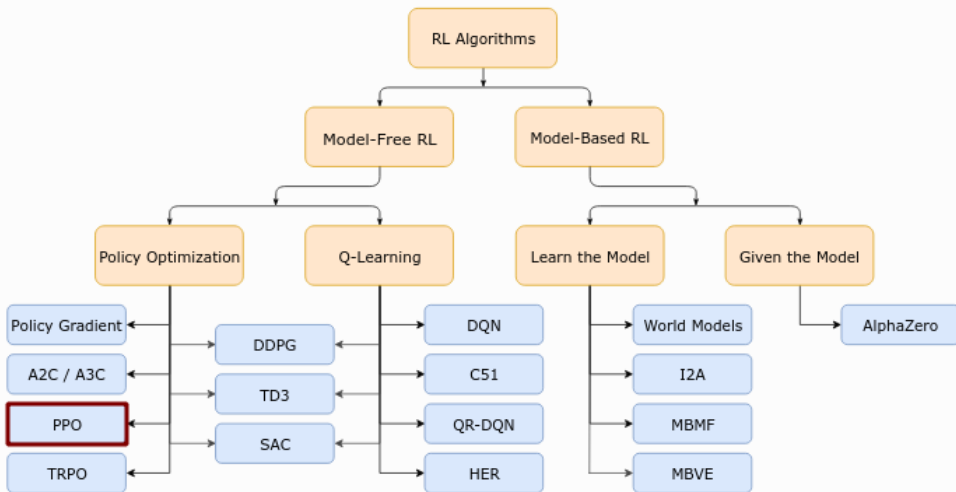
Stable-baseline3



A set of reliable implementations of reinforcement learning algorithms in **PyTorch**.

- github.com/DLR-RM/stable-baselines3
- docs:
stable-baselines3.readthedocs.io
spinningup.openai.com
- install:
`pip install stable-baselines3`

A Taxonomy of RL Algorithms (Source: spinningup.openai.com)



The DRL practical work

- Goal: train a **PPO** (Proximal Policy Optimisation) neural network to drive the position of the end effector of a 2 DOF robot arm.

The DRL practical work

- Goal: train a **PPO** (Proximal Policy Optimisation) neural network to drive the position of the end effector of a 2 DOF robot arm.
- The training of the network involves the simulation of the robot with the [PyBullet](#) simulator.

The DRL practical work

- Goal: train a **PPO** (Proximal Policy Optimisation) neural network to drive the position of the end effector of a 2 DOF robot arm.
- The training of the network involves the simulation of the robot with the [PyBullet](#) simulator.
- The robot is described in the simulator thanks to its [URDF](#) file.

The DRL practical work

- Goal: train a **PPO** (Proximal Policy Optimisation) neural network to drive the position of the end effector of a 2 DOF robot arm.
- The training of the network involves the simulation of the robot with the [PyBullet](#) simulator.
- The robot is described in the simulator thanks to its [URDF](#) file.
- The [PyTorch](#) Python module provides classes to instantiate and train the PPO neural network.

The DRL practical work

- Goal: train a **PPO** (Proximal Policy Optimisation) neural network to drive the position of the end effector of a 2 DOF robot arm.
- The training of the network involves the simulation of the robot with the **PyBullet** simulator.
- The robot is described in the simulator thanks to its **URDF** file.
- The **PyTorch** Python module provides classes to instantiate and train the PPO neural network.
- To ensure the robustness and maintainability of Python developments, the work is done in a **Python Virtual Environment** (PVE).

The DRL practical work

- The pedagogical material of the practical work is on the Git repository github.com/cjlux/DRL_at_ENSEIRB-MATMECA

The DRL practical work

- The pedagogical material of the practical work is on the Git repository github.com/cjlux/DRL_at_ENSEIRB-MATMECA
- Session sequencing :

The DRL practical work

- The pedagogical material of the practical work is on the Git repository github.com/cjlux/DRL_at_ENSEIRB-MATMECA
- Session sequencing :
 - 1 From the Git repository download the file [Create_PVE.pdf](#)

The DRL practical work

- The pedagogical material of the practical work is on the Git repository github.com/cjlux/DRL_at_ENSEIRB-MATMECA
- Session sequencing :
 - 1 From the Git repository download the file [Create_PVE.pdf](#)
 - 2 Following the PDF document, create & activate the PVE [pydr1](#), download the repository and install the required Python modules.

The DRL practical work

- The pedagogical material of the practical work is on the Git repository github.com/cjlux/DRL_at_ENSEIRB-MATMECA
- Session sequencing :
 - 1 From the Git repository download the file [Create_PVE.pdf](#)
 - 2 Following the PDF document, create & activate the PVE `pydrl`, download the repository and install the required Python modules.
 - 3 Within the PVE `pydrl` run `jupyter notebook` or `jupyter lab` or whatever you want to open the notebook
[1-Getting_started_with_pybullet.ipynb](#).

The DRL practical work

- The pedagogical material of the practical work is on the Git repository github.com/cjlux/DRL_at_ENSEIRB-MATMECA
- Session sequencing :
 - 1 From the Git repository download the file [Create_PVE.pdf](#)
 - 2 Following the PDF document, create & activate the PVE `pydr1`, download the repository and install the required Python modules.
 - 3 Within the PVE `pydr1` run `jupyter notebook` or `jupyter lab` or whatever you want to open the notebook [1-Getting_started_with_pybullet.ipynb](#).
 - 4 Play with the notebook...