SomehowImLearning

```
   _____                    _                  _____ _            _
_
  / ____|                  | |                |_   _( )          | |
(_)
 | (___   ___  _ __ ___   ___| |__   _____      __   | | |/ _ __ ___   | |
___  __ _ _ __ _ __  _ __   __ _
  \___ \ / _ \| '_ ` _ \ / _ \ '_ \ / _ \ \ /\ / /   | |   | '_ ` _ \  | |    /
_ \/ _` | '__| '_ \| | '_ \ / _` |
  ____) | (_) | | | | | |  __/ | | | (_) \ V  V /   _| |_  | | | | | | | |___|
__/ (_| | |  | | | | | | | (_| |
 |_____/ \___/|_| |_| |_|\___|_| |_|\___/ \_/\_/   |_____| |_| |_| |_|
|______\___|\__,_|_|  |_| |_|_|_| |_|\__, |

__/ |

|___/

@AUTHORS : Pierre LAGUE & François MULLER
@ESTABLISHMENT : University of Lille, France
```

Welcome,

This repository aims to demonstrate the efficiency of known reinforcement learning algorithms on hard-coded environments.

It is an initiative of the authors following the class "Reinforcement Learning" led by Pr. Philippe Preux at the University of Lille.

The key points to remember from this repository :

- explicitely coded algorithms and environments
- easy use
- insights on the performance of the algorithms on the environments compared to state-of-the-art models
- we're just two chill guys coding stuff and somehow it's working

Explanation of the structure

> TODO : once we have all of our programs (algo, models, results) we use the `tree` command and describe each of the folders or files

Introduction to the algorithms

> TODO : short context to markovian decision problems
> TODO : short introduction to each algorithms (math ++ ?)
> N.B : take it chronologically (markovian decision problems, discretized q learning, tabular q learning, continuous NFQ, DQN, Direct policy, etc.)

How to use the repo

The root

At the root of the repository, you will find a python script `simulation_environment.py`. This script simulates an environment with a specific model, for a certain amount of time. (e.g. `python simulation_environment.py <env_name> <model_name> <simulation_time>`).

- For the car simulation :
    - it ends once the car has gotten a positive result
    - you can monitor its speed, reward, position and movements on the hill
- For the pendulum simulation :
    - it lasts a certain time (cli argument)
    - you can monitor rewards, angular position, angular velocity, actions

The `algorithms/environments` folder

In this folder you will find an implementation of each of our environments. The structure of the environment is purely indicative, but change it and the programs wont work so good after that. We tried to make it as general as possible (define your envs features and at least a `reset` and a `step` function).

> We highly encourage you to try and develop new environments (and visualisations if you want) and then open an issue to inform us.

> TODO : litteraly the rest

Results for now

NFQ (offline)

Car Environment

- *Setup 1* :
    - nb_episodes : 10
    - episodes_per_iter : 200
    - evaluation_episodes : 10
    - gamma : 0.99
    - lr : 0.001
    - hidden_layers : (5, 5)

- **Training Time : 544.59 seconds (~9min)**
- **Iterations to first success : 17**
- model : `nfq_car_model_10_ep.pkl`

Pendulum Environment

- *Setup 1* :
  - nb_episodes : 150
  - episodes_per_iter : 5000
  - evaluation_episodes : 10
  - gamma : 0.99
  - lr : 0.001
  - hidden_layers : (5, 5)
  - **Training Time : 140.84 seconds (~2min30)**
  - model : `nfq_pendulum_model_150_ep.pkl`

DQN (online)

Car Environment

- *Setup 1* :
  - nb_episodes : 500
  - iterations_per_ep : 200
  - target_network_update : 5
  - gamma : 0.99
  - lr : 0.001
  - hidden_layers : (5, 5)
  - epsilon_decay : 0.9995
  - **Training Time : 402.24 seconds (~7min)**
  - **Iterations to first success : 17**
  - model : `dqn_car_500_ep.pth`

Pendulum Environment

- *Setup 1* :
  - nb_episodes : 50000
  - iterations_per_ep : 5000
  - target_network_update : 5
  - gamma : 0.99
  - lr : 0.001
  - hidden_layers : (5, 5)
  - epsilon_decay : 0.9995
  - **Training Time : 305.56 seconds (~5min)**
  - model : `dqn_pendulum_50000_ep.pth`
