Global Objective

During this practical session you will implement several Bandit algorithms and the needed framework to compare them on artificial data.

Thereafter you have up to **Tuesday 29, November at 8pm** to submit a short report which resumes your opinion about these algorithms (see requirements bellow).

This submission has to be done through Moodle corresponding link.

Source Code

Requisites

- Python3
- `numpy` and `scipy` for tensor manipulation
- `matplotlib` for graphics
- Jupyter-Notebook for the tutorial

Initial

Moodle contains an initial version of the source code. It has been split in three files:

- *player.py*: some bandit algorithms implemented as objects with three functions

    - `choose_next_arm() -> unsigned int` : choose the next arm to pull
    - `update(unsigned int: arm, float: reward) -> void` : update stored informations given that the arm `arm` was pulled * `restart() -> void` : erase player's memory

- *arm.py*: some standard arms implemented as objects with two functions

    - `draw() -> float` : return a random value given the probability distribution corresponding to the arm
    - `mean() -> float` : return the expected reward corresponding to the probability distribution of the arm

- *exp.py*: a few useful functions to run experiments and plot results.

Note that for each setting (set of arms & algorithm), the algorithm is run against the environnement `nb_games` times and the logs (matrices) at index `[i,j]` contains the logged value at time-step `j` during game `i`. We store

- the index of the chosen arm,
- the reward obtained,
- the expected reward of the chosen arm,
- the expected reward of the best arm.

The four corresponding matrices are stored in a dictionary (take a look at line 44).

Mandatory Job During the Practical Session

In almost chronological order:

Phase 0: Discover the APIs

- Read *tutorial.ipynb*, fill the missing code lines, and run.
- Take a look at already implemented players and arms.

Phase 1: UCB1

- Implement the UCB1 strategy.
- Compare ε_n-greedy, UCB1, and Thompson Sampling behavior
    - focus on the cumulative regret curve (given time-step)
    - fix the environment to two arms {Bernoulli(0.2), Bernoulli(0.5)}, with horizon 300
    - optimise the ε_n-greedy parameter and the UCB parameter
    - plot results of the selected parameters
- Do the same job against the set of two arms {Bernoulli(0.2), Bernoulli(0.9)}, with horizon 300 * are the best parameters the same ? How behave the parameters selected during previous optimization
- Idem with horizon 10,000

Bonus

- Implement Bernoulli arms.
- Compare Algorithms with more divers environments (more arms, Bernoulli arms...).
- Look for the worst set of 10 arms for Explore Then Commit (with horizon 300). How behave UCB1 and ε_n-greedy against that setting ?
- Look for the worst set of 10 arms for UCB / ε_n-greedy.

Report Contain

- Compare ε_n-greedy, UCB1, and Thompson Sampling behavior on your personal set of arms given in Moodle.
- Submit the corresponding report in pdf format.
- The report should
  - express the choice of hyper-parameter for ε_n-greedy strategy,
  - compare all the algorithms,
  - include comments regarding the behavior of these algorithms.