Lab Session 1

Introduction

In this Lab exercise you'll be introduced to:
1. A small warmup exercise with numpy, Matplotlib.
2. Best practise to documenting your code.
3. Bandit Algorithms

Instructions

In this lab session you will have to implement Bandit algorithms, Bandits (aka slot machines) and the learning loop of the agents. The code is divided in 4 modules, and you will have to complete the code in the 3 first ones:

Module 1: launcher.py

This module controls the hyperparametrs, main loops, and plotting.

General functionning

You are given two possibilities for launch, using the `launch_type` variable:
1. *multiple_agents*: comparing multiple agents on the k-armed bandit problem. Fill the list of agent classes you want to compare, `agents`; you can change the labels and plot file names.
2. *spectrum*: comparing values of a hyperparameter for an algorithm on the k-armed bandit problem. The `spectrum` variable should look like a list \['hparam_name', \[list of values the hparam should take\]\]; you can change the labels and plot file names.

Note:- you cannot run the code before you implement at least
* Gaussian_Bandit
* KBandit
* run_bandit

TODO

You only have to fill in the run_bandit function. The details of the function are in its documentation.

Module 2: bandit.py

This module contains the *Slot Machines* of the problem.
A Slot Machine is actually called a "bandit" because it steals money from the gambler's pockets, apparently.

General functionning

The *Bandit* class is implemented: it is the abstraction of a machine that you can simply pull, and get a reward.
All the subsequent classes should be child classes of Bandit.

TODO

You need to implement the following classes:
* *Gaussian_Bandit* - the machine's reward distribution is a Gaussian N(m,1) around a fixed mean m~N(1,0)
* *Gaussian_Bandit_NonStat* - a Gaussian_Bandit whose mean moves~N(0,std=0.01) after each pull (of any other arm also)
* *KBandit* - a k-armed bandit, i.e. a set of k Gaussian_Bandits
* *KBandit_NonStat* - a KBandit, but with Non Stationary Bandits, i.e. the distributions move every pull.

All the classes should be children (or grand-) of the abstract Bandit class, to keep a clean **object-oriented code**.

Once the KBandit and run_bandit class and function have been implemented, you can try to run the Random_Agent (given as an example) by selecting `**launch_type='mutiple_agents'**` and only un-commenting the `**Random_Agent**` from the list of `**agents**`. You can set as output file_name, for example **'random_agent'**.
Do the results make sense?

Module 3: agents.py

This module contains the agents, or algorithms, that are meant to solve the Bandit problems implemented above.

General functionning

The *Bandit_Agent* abstract class is implemented and contains the formalism necessary to run an agent. Such an agent should be able to act, learn and reset itself.
The *Random_Agent* class was implemented to show an example, but note how it does not need to implement the `learn` method, by definition.

TODO

As soon as a class is implemented, you can test it by comparing it to just the Random Agent (or another class you implemented) in the `agents` list.
The agent classes you will have to implement are:
* *EpsGreedy_SampleAverage* - estimating Q values with sample averages and selecting actions epsilon-greedily
* *EpsGreedy*`- fixed learning rate instead of Sample Averages
* *OptimisticGreedy* - high initial Q values; full greedy action selection
* *UCB* - selecting action greedily according to Q + U where U is a count-based bonus on rarely tried action
* *Gradient* - keeping up advantages vector H for each action, action selection samples from softmax(H)

Module 4: utils.py

Contains the plotting functions, some useful functions like the softmax or my_random_choice.
You don't need to do anything here. Enjoy!

Submitting

Please record your plots in the folder of the same name (the `file_name` variable controls this in addition to the plot name). We ask that you at least provide a comparison of all agents (both action and average reward) (file_name='plots/agent_comparison').
Once done, please zip your lab_session1 folder and submit it on blackboard.