Exercise session 1: Bandits

Guillaume Charpiat, Victor Berger

January 11, 2018

Exercise

We are considering the following problem: a 10-armed bandit for which each arm is a binomial distribution for which the two possible values and the probability of each outcome is chosen at random, at the start of the simulation.

Your task is to implement various bandit algorithms against this problem, using the python template provided on the course website (reminder: `https://www.lri.fr/~gcharpia/machinelearningcourse/`).

You are provided a random agent, that you can use as a template for your code. Your task is to implement an epsilon-greedy agent, an optimistic epsilon-greedy agent, a softmax agent, and a UCB agent[1] using this template. Then, you will compare their performances and upload your best agent on the test platform for final performance assessment (the link to the platform is on the course website).

Template description

The template is a zip file that you can download on the course website. It contains several files, two of them are of interest for you: `agent.py` and `main.py`.

`agent.py` is the file in which you will write the code of your agent, using the `RandomAgent` class as a template. Don't forget to read the documentation it contains. Note that you can have the code of your several agents in the same file, and use the final line `Agent = MyAgent` to chose which agent you want to run.

`main.py` is the program that will actually run your agent. You can run it with the command `python main.py`. It also accepts a few command-line arguments:

- `--niter N` will run your agent for N iterations against the same bandit and report the total cumulative reward

[1] As a reminder, the UCBAgent selects its action as $\arg\max_{k \in A} \left[\mu_{s_k,k} + \sqrt{\frac{2 \log t}{s_k}} \right]$ with t the number of iterations, s_k the number of times arm k was selected and $\mu_{s_k,k}$ the empirical estimator of the value of arm k.

- `--batch B` will run B instances of your agent in parallel, each against its own bandit, and report the average total cumulative reward
- `--verbose` will print details at each step of what your agent did. This can be helpful to understand if something is going wrong.

The running of your agent follows a general procedure that will be shared for all later practicals:

- The environment generates an observation
- This observation is provided to your agent via the `act` method which chooses an action
- The environment processes your action to generate a reward
- this reward is given to your agent in the `reward` method, in which your agent will learn from the reward

This 4-step process is then repeated several times.

Note that in this first exercise, there are no observations, as the environment is static. Thus the observation will always be `None`.

Grading

The final performance of your agent will be evaluated by running the following command on a pseudo-random[2] testbed:

```
python main.py --niter 1000 --batch 2000
```

Once you think your implementation is good, create a zip file containing your `agent.py` file and the `metadata` file provided in the template, and upload it to the platform. Your score will be computed and you can compare yourself to the rest of the class using the leaderboard. Your grade for this exercise will be based on this score.

[2]This means that uploading two times the exact same code will generate the exact same score