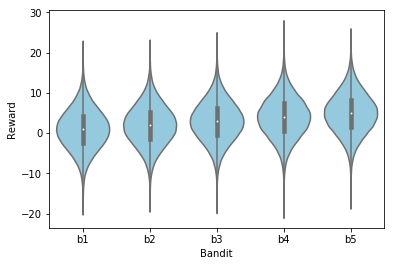
The multi-armed (also called k-armed) bandit is an introductory reinforcement learning problem in which an agent has to make n choices among k different options. Each option delivers a (possibly) different reward from an **unknown distribution** which usually doesn’t change over time (i.e. it is stationary). If the distribution changes over time (i.e. it is not stationary), the problem gets harder because previous observations (i.e. previous games) are of little usefulness. In either case, the goal is to maximized the total reward obtained.

This article reviews one (of many) simple solution for both a stationary and a non-stationary 5-armed bandit across 1000 games. Note that only some remarks of the full code will be showcased here, for the fully functional notebook, please see this github repository.

First of all, let’s just define the 5 stationary bandits shown in the image below and that will become the agent’s options.

[bandit code]



For the sake of simplicity, each of them follows a normal distribution with mean 1, 2, 3, 4 and 5 and all of them have a standard deviation of 5. This means that a reward of, let’s say 3, in a single is highly likely in all of the bandits, but the expected reward of 1000 games will vary hugely from one bandit to the other. In fact, if one always chooses the bandit b1, the expected reward is 1000, while if one does the same with the bandit b5 then the expected reward rises up until 5000 (a 5x increase).

Of course, if one knew these distributions beforehand then the problem will be trivial: just choose the bandit with the highest expected value and stick with it. This is where the trade-off between exploration and exploitation comes in: given that one has imperfect information, it is necessary to keep exploring the environment (i.e. trying different bandits) in order to gain knowledge about the best option to choose or it will be very probable to get stuck in a local optima (pure exploitation, a greedy algorithm), but if one only explores, then the information obtained is not being used and no optima gets achieved (pure exploration).

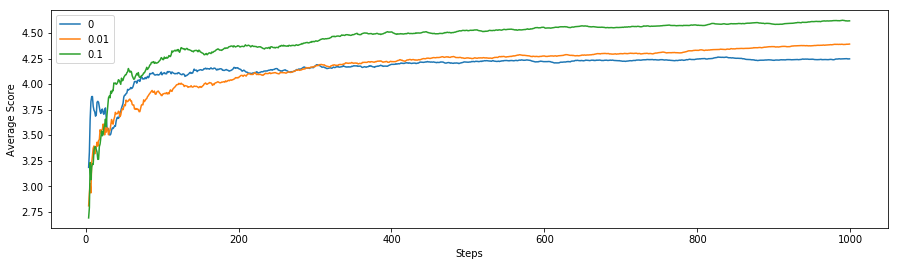
Just to make sure the above concept gets understood, suppose that after trying each bandit once the following outcome is:

Then a pure exploitation algorithm will pull the worst bandit () for a long time (until the average of gets below 0, which may be never). On the contrary, a pure exploration algorithm will sample the five bandits uniformly and its expected value will be of:

Which is a suboptimal result.

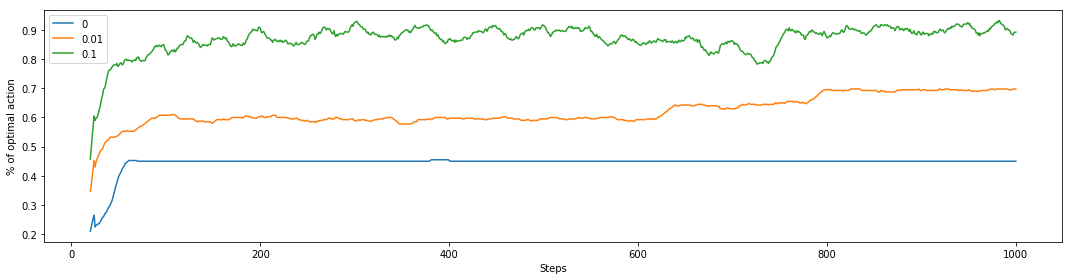
A simple solution is to combine this two approaches and just be greedy but explore in ε% of the time. In the figure below one can see an implementation of this for ε = 0 (pure exploitation), 0.01 and 0.10. The greedy algorithm dominates in the early phases of the search, but those that explore quickly realize that there are better bandits and outperform the greedy algorithm.

[play code]

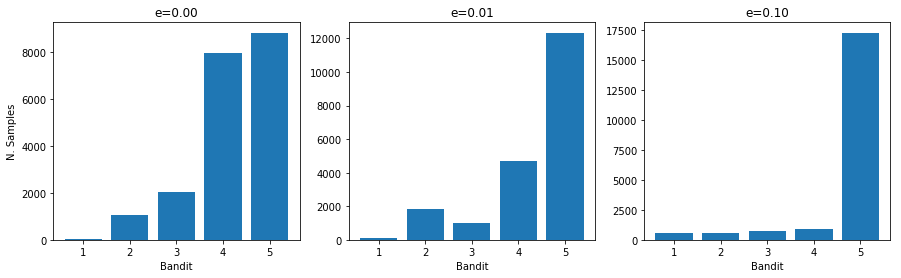


As the problem is stationary, once we are confident on being sampling the best bandit, one needs no longer exploration, hence, in the limit, the ε = 0.01 algorithm will be the best performer. Nevertheless, if one has confident in the stationarity of the problem, then the best strategy will be to do an initial search (ε = 0.10) and then switch to exploitation mode (ε = 0)!

The figure below shows that by playing this game 20 times, the mixed algorithms consistently sampled the best bandit a higher percent of the time. Notice how ε = 0.10 looks stagnated but ε = 0.01 keeps increasing with time.



Another way to look at it is to check the number of samples per bandit across the 20 games, and as shown below, the greedy algorithm gets usually confused between the bandit 4 and 5, while ε = 0.10 finds the best bandit quite easily.



Non stationary bandit

In real life, it is common to find distributions that change over time. In this case, the problems gets harder to solve just because previous observations are less useful because they might not be reflecting the truth about the current state of the bandit. An easy (but very restrictive) way of handling this is just to take into account a number X of previous observations, which is the approach that is being used here. Note that this arises many issues, such as:

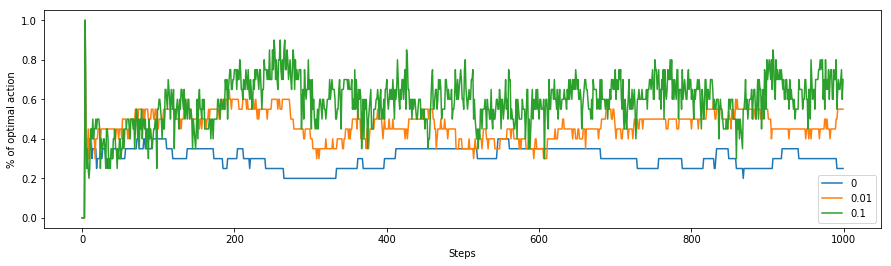
* Changes may be more frequent than X, in which case information is getting mixed.
* Changes may be less frequent than X, in which case information is being wasted.
* This approach assumes that the new distribution of the bandit has nothing to do with the previous distribution, which is often not the case.

Alternative approaches such as using exponential weighted means, weighted block means or even fitting a time series model to find indicator of when a distribution changes and tune the exploration rate accordingly may be more accurate given the problem’s nature.

In order to tweak these bandits into being non-stationary a simple probabilistic mutation on the distribution’s mean and standard deviation is used (1% mutation probability), but changes may also come every X number of steps or with a non-uniform probability or they might even change the shape of the distribution.

[mutation code]

In this case, one would expect a pure exploitation algorithm to perform very poorly. This intuition is confirmed with the figure below, which shows that the ε = 0 algorithm only chooses the best bandit about 30% of the time, which is not much better than a purely random choice (20%). ε = 0.10 manages to discover when the best bandit changes and keeps being the best performance algorithm. On the other hand, ε = 0.01 is not improving anymore since its low exploration rate doesn’t allow it to find the changing best bandit quickly.



Finally, if one needs to remember something about this article it should be: just like the k-bandit problem, real world problems in which the true nature is unknown need a mixture of exploration and exploitation to be addressed efficiently.