Reinforcement Learning: Assignment 1

Alexander Y. Shestopaloff

May, 2025

In this assignment, your goal will be to experiment with some simple bandit learning algorithms by implementing them from scratch in a variety of scenarios and comparing their performance in terms of (1) the average per-step reward as well as (2) the proportion of time the optimal action, i.e., the one with the highest expected reward, is taken.

You should provide a report including plots describing your results and a link to an online repository with commented and reproducible code. You should also provide a readme file that can make it easy for anyone to replicate the results in this assignment. Note that modern machine learning conferences e.g., NeurIPS, and some of the leading journals, e.g. International Journal of Forecasting require submission of documented code for reproducibility of reported results.

Each part is worth **25 marks**. You will be graded on the correctness, clarity of exposition and reproducibility of your results. The assignment is due before **11:59pm June 18th**. It may be done individually or in pairs. Both participants in a pair will receive the same grade and no preference will be given to people working individually or in pairs.

1 Part 1

We start by setting up a simple bandit problem with stationary reward distributions. Consider the so-called k-armed testbed, with $k = 10$, with normally distributed rewards. Generate a set of ten iid means $\mu_1, \ldots, \mu_{10}$ from a $N(0, 1)$ distribution and suppose that the arms 1 through 10 have $N(\mu_i, 1)$ reward distributions where $i = 1, \ldots, 10$.

Our goal will be to study the performance of using the different learning methods we discussed for bandit algorithms. These are as follows.

- greedy with non-optimistic initial values, i.e. 0.
- epsilon-greedy with different choices of epsilon. Here, you will need to explain how you choose the value of ϵ. One option is to use *pilot runs*: small-scale experiments to try a grid of several settings at low computation cost, tracking the evolution of the rewards curve for each setting to pick a setting that gives good results.
- optimistic starting values with a greedy approach. You may assume you know the means of each of the reward distributions to help you set the optimistic initial values. For concreteness, set the initial action values to the 99.5th percentile of the normal distribution with the highest μ_i.
- gradient bandit algorithm. Try different learning rates α and determine a good one through some pilot runs.

In all cases, tiebreaking should be done by choosing an arm uniformly at random. Run each algorithm for 2000 time steps. Repeat for a total of 1000 simulations and report (1) the average reward acquired by the algorithm at each time step, averaged over the 1000 simulations and (2) the percentage of time the optimal action is taken by the algorithm at each time step. Note that each of the 1000 simulations muse use a separate, independent random stream.

Comment on which of the methods performs the best. Can you comment why? What did you do to tune each of the methods?

2 Part 2

Now consider non-stationary modifications of the problem above. In the real world, changes to rewards can be gradual, abrupt or a combination of both. Think of the brightness of an image changing gradually, as daylight does, or abruptly, as if a light switch is turned on.

2.1 Gradual changes

Try applying (1) a *drift* to each μ_i you generated above to make it time-variable

$$\mu_{i,t} = \mu_{i,t-1} + \epsilon_{i,t}$$

where $\epsilon_{i,t}$ is iid $N(0, 0.01^2)$ and separately (2) a *mean-reverting change*

$$\mu_{i,t} = \kappa\mu_{i,t-1} + \epsilon_{i,t}$$

where $\kappa = 0.5$, $\epsilon_{i,t}$ is iid $N(0, 0.01^2)$, and $\mu_{i,1} = \mu_i$. Note that we use the mean-variance parametrization of the normal distribution throughout.

2.2 Abrupt changes

At $t = 501$ randomly permute the means corresponding to each of the reward distributions. Explore what happens when you keep running the algorithms as-is. Also, check what happens with a hard reset, i.e., when you reset all action values to 0 at $t = 501$. In other words, you have foreknowledge of the changepoint.

Compare action-value based methods with the gradient bandit algorithm on this problem in terms of the average per-step reward and the proportion of time the optimal action is taken on 2000 time steps. Like above, use an average over 1000 simulations. Do not forget to use the *same* 10 seeds for the noise processes driving the drift on each μ_i and the *same* random permutation for each of the simulations, else the 1000 simulations will not be comparable to one another (think why?). You may use pilot runs to get an idea of the sort of rewards a method produces to set the parameters before running the 1000 repetitions.