Reinforcement Learning: Assignment 1

Alexander Y. Shestopaloff

May, 2024

In this assignment, your goal will be to experiment with some simple bandit learning algorithms by implementing them from scratch and comparing their performance in terms of (1) the average accumulated 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, require submission of documented code for reproducibility of reported results.

Each part is worth 25 marks. You will be graded on correctness, clarity and reproducibility of your results. The assignment is due on June 13th. 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 looking at 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 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 $i = 1, \ldots, 10$. You goal is to learn the action values corresponding to each of the 10 arms, i.e., the expected rewards $q^*(a)$ for $a = 1, \ldots 10$ using the different methods we discussed for doing so.

- greedy with non-optimistic initial values. Initialize the action value estimates to 0 and use the incremental implementation of the simple average method.
- 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*: to take a few example bandit problems and try a few settings of epsilon on these problems to track the evolution of the rewards curve and picking one 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.
- gradient bandit algorithm. Try different learning rates α and determine a good one through some pilot runs.

Repeat this for a total of 1000 different bandit problems (i.e., 1000 sets of ten mean parameters) and report (1) the average reward acquired by the algorithm at each time step (except for the gradient bandit algorithm) at each time step and (2) the percentage of time the optimal action is taken by the algorithm, again, at each time step.

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* change

$$\mu_t = \mu_{t-1} + \epsilon_t$$

where ϵ_t is $N(0, 0.001^2)$ and separately (2) a *mean-reverting change*

$$\mu_t = \kappa\mu_{t-1} + \epsilon_t$$

where $\kappa = 0.5$ and ϵ_t is $N(0, 0.01^2)$ to the mean parameters as you do the training. Note that we use the mean-variance parametrization of the normal distribution throughout.

2.2 Abrupt changes

At each time step, with probability 0.005, permute the means corresponding to each of the reward distributions.

2.3 Evaluation

To understand the importance of being able to adapt to non-stationarity, compare the (1) optimistic greedy method, (2) ϵ-greedy with a fixed step size (3) ϵ-greedy with a decreasing step-size, for example, the simple average estimator. Comparing methods on non-stationary problems requires different metrics to be used. Since we cannot compare the peformance at a point in time (due to non-stationarity of the rewards distributions) we will look at the distribution of the rewards attained after some large number of steps (say, $10,000$ or $20,000$). You should therefore run the algorithm on $1,000$ repetitions of the non-stationary problem (use the same starting means but a different seed for each run) and report the distribution of the average reward attained at the terminal step over the $1,000$ repetitions. The algorithm which produces the most favourable distribution of rewards at the terminal step is then a preferable one. Use box plots to present the terminal reward distributions.

You may use pilot runs to get an idea of the sort of rewards a method produces to set the parameters before running the $1,000$ repetitions. Explain how you do so in your report.

3 Additional Practice

Finally, as an additional practice for how bandits can be used for more complex problems, take a look at the papers: `https://proceedings.mlr.press/v151/duran-martin22a/duran-martin22a.pdf` and also this paper `https://proceedings.mlr.press/v232/chang23a/chang23a.pdf` which is more on the learning methodology itself. The key idea here is to convert a classification problem into a bandit problem where you need to learn a classifier online to maximize cumulative reward. (A similar logic also drives recommender systems). A good exercise would be to pull the code (freely available for both papers) and attempt to run it on your machine, and perhaps on a new data problem.