CS 4180/5180: Reinforcement Learning and Sequential Decision Making (Fall 2020) Lawson L.S. Wong
Northeastern University Due September 25, 2020

Exercise 1: Multi-armed Bandits

Please remember the following policies:

- Exercises are due at **11:59 PM** Boston time (EDT/EST).
- Submissions should be made electronically on Canvas. Please ensure that your solutions for both the written and programming parts are present. You can upload multiple files in a single submission, or you can zip them into a single file. You can make as many submissions as you wish, but only the latest one will be considered, and late days will be computed based on the latest submission.
- If you are unable to access Canvas, you may submit via the submission link on Piazza. In this case, please zip your submission into a single file, and follow the naming convention listed on the form:
`Ex[Num] - [FirstName] [LastName] [Version number].zip`
- Solutions may be handwritten or typeset. For the former, please ensure handwriting is legible. If you write your answers on paper and submit images of them, that is fine, but please put and order them correctly in a single .pdf file. One way to do this is putting them in a Word document and saving as a PDF file.
- You are welcome to discuss these problems with other students in the class, but you must understand and write up the solution **and code** yourself, *and* indicate who you discussed with (if any).
- Some questions are intended for CS 5180 students only. CS 4180 students may complete these for extra credit.
- Each exercise may be handed in up to two days late (24-hour period), penalized by 10% per day. Submissions later than this will not be accepted. There is no limit on the total number of late days used over the course of the semester.
- Contact the teaching staff if there are *extenuating* circumstances.

1. **1 point.** (RL2e 2.2) *Exploration vs. exploitation.*
Written: Consider a k-armed bandit problem with $k = 4$ actions, denoted 1, 2, 3, and 4. Consider applying to this problem a bandit algorithm using ε-greedy action selection, sample-average action-value estimates, and initial estimates of $Q_1(a) = 0$, for all a. Suppose the initial sequence of actions and rewards is $A_1 = 1, R_1 = -1, A_2 = 2, R_2 = 1, A_3 = 2, R_3 = -2, A_4 = 2, R_4 = 2, A_5 = 3, R_5 = 0$. On some of these time steps the ε case may have occurred, causing an action to be selected at random. On which time steps did this definitely occur? On which time steps could this possibly have occurred?

2. **1 point.** (RL2e 2.4) *Varying step-size weights.*
Written: If the step-size parameters, α_n, are not constant, then the estimate Q_n is a weighted average of previously received rewards with a weighting different from that given by Equation 2.6. What is the weighting on each prior reward for the general case, analogous to Equation 2.6, in terms of the sequence of step-size parameters?

3. **[CS 5180 only.] 2 points.** *Bias in Q-value estimates.*
Written: Recall that $Q_n \triangleq \frac{R_1 + \ldots + R_{n-1}}{n-1}$ is an estimate of the true expected reward q_* of an arbitrary arm a. We say that an estimate is *biased* if the expected value of the estimate does not match the true value, i.e., $\mathbb{E}[Q_n] \neq q_*$ (otherwise, it is *unbiased*).

   (a) Consider the *sample-average* estimate in Equation 2.1. Is it biased or unbiased? Explain briefly.

   For the remainder of the question, consider the *exponential recency-weighted average* estimate in Equation 2.5. Assume that $0 < \alpha < 1$ (i.e., it is strictly less than 1).

   (b) If $Q_1 = 0$, is Q_n for $n > 1$ biased? Explain briefly.
   (c) Derive conditions for when Q_n will be unbiased.
   (d) Show that Q_n is *asymptotically unbiased*, i.e., it is an unbiased estimator as $n \to \infty$.
   (e) Why should we expect that the *exponential recency-weighted average* will be biased in general?

For the remainder of this assignment, you will be implementing the 10-armed testbed described in Section 2.3, reproducing some textbook figures based on this testbed, and performing further experimentation on bandit algorithms. Some general tips:

- Read all the questions first before implementing your experimental pipeline. If you design the pipeline well, implementing experiments and plots after Q6 will be require fairly small changes.
- You are encouraged to use good software engineering practices (related to previous point), but it is not required as long as your code is readable.
- Some of the experiments will take a while. During development, you should use smaller numbers of steps and trials to make iteration much faster. Here are some possible settings:
  - Tiny: 100 steps, 20 trials – for fast iteration, should take < 1 second
  - Small: 100 steps, 200 trials – for development and debugging purposes, should take < 10 seconds
  - Medium: 1000 steps, 2000 trials – the setting used in the textbook, useful for verifying correctness of implementation and seeing initial trends, should take < 5 minutes
  - Large: 10^4 steps, 2000 trials – the final setting in this assignment, should take ~ 1 hour, consider running these overnight or while doing something else
- The above times were based on our implementation running on a single CPU thread of a Lenovo ThinkPad T480s (circa 2018) laptop. You can even consider multiprocessing, but that should not be necessary. You may consider adjusting the settings above of steps/trials in order to achieve fast iteration.
- If you are having difficulty running the required number of steps/trials even after trying to ensure your implementation is efficient, you may use the "Medium" setting above (1000 steps, 2000 trials), and indicate that you did this in your submission.

4. **1 point.** *Implementing the* 10*-armed testbed for further experimentation in the remainder of the assignment.*
   Code: Implement the 10-armed testbed described in the first paragraph Section 2.3 (p. 28).
   Read the description carefully.
   Plot: To test that your testbed is working properly, produce a plot similar in style to Figure 2.1 by pulling each arm many times and plotting the distribution of sampled rewards. You can use any type of plot that makes this point effective, e.g., a violin plot, or a scatterplot with some jitter in the horizontal axis to show the sample density more effectively.

5. **1 point.** (RL2e 2.3) *Predicting asymptotic behavior in Figure 2.2.*
   Written: In the comparison shown in Figure 2.2, which method will perform best in the long run in terms of cumulative reward and probability of selecting the best action? How much better will it be? Express your answer quantitatively. (Compute what you expect the asymptotic performances to be for the lower graph, and possibly the upper graph if you want a small mathematical workout.)

6. **1 point.** *Reproducing Figure 2.2.*
   Code: Implement the ε-greedy algorithm with incremental updates. Note that in the graph: "All the methods formed their action-value estimates using the sample-average technique (with an initial estimate of 0)."
   Plot: Reproduce the curves shown in Figure 2.2, with the following modifications:

   - Extend the plots up to 10^4 steps (instead of 10^3 as shown), with 2000 independent runs (same as in text). Make sure that all methods are evaluated on the same set of 2000 10-armed bandit problems.
   - Add an extra constant upper bound line corresponding to the best possible average performance in your trials, based on the known true rewards.
   - For each curve, also plot confidence bands corresponding to (1.96$\times$ standard error) of your rewards. The standard error of the mean is defined as the standard deviation divided by $\sqrt{n}$: $\frac{\sigma}{\sqrt{n}}$
     This corresponds to a $\sim 95\%$ confidence interval around the average performance. In other words, our uncertainty in the average performance decreases as the number of trials increases. See the following for an example of plotting a confidence band in `matplotlib.pyplot`: `https://matplotlib.org/3.3.1/gallery/lines_bars_and_markers/fill_between_demo.html#example-confidence-bands`

   Written: Do the averages reach the asymptotic levels predicted in the previous question?

7. **2 points.** (RL2e 2.5) *Investigating nonstationary environments.*
Code: Design and conduct an experiment to demonstrate the difficulties that sample-average methods have for nonstationary problems. Use a modified version of the 10-armed testbed in which all the $q_*(a)$ start out equal and then take independent random walks (by adding a normally distributed increment with mean 0 and standard deviation 0.01 to all the $q_*(a)$ on each step).
Plot: Prepare plots like Figure 2.2 (with upper bounds and confidence bands as in Q6) for an action-value method using sample averages, incrementally computed, and another action-value method using a constant step-size parameter, $\alpha = 0.1$. Use $\varepsilon = 0.1$ and longer runs of 10^4 steps.

8. **[CS 5180 only.] 2 points.** *Reproducing and supplementing Figures 2.3 and 2.4.*
Code: Implement the ε-greedy algorithm with optimistic initial values, and the bandit algorithm with UCB action selection.
Plot: Reproduce the curves shown in Figures 2.3 and 2.4, with the following modifications:

   - Extend the plots up to 10^4 steps (instead of 10^3 as shown), with 2000 independent runs (same as in text). Make sure that all methods are evaluated on the same set of 2000 10-armed bandit problems.
   - Add an extra constant upper bound line corresponding to the best possible average performance in your trials, based on the known true rewards.
   - For each curve, also plot confidence bands corresponding to (1.96$\times$ standard error) of your rewards.
   - We noted in lecture that Figure 2.3 was unsatisfactory because both Q_1 and ε were changed simula-taneously, thereby confounding the results. Produce two additional curves for $(Q_1 = 5, \varepsilon = 0.1)$ and $(Q_1 = 0, \varepsilon = 0)$.
   - In Figure 2.2, both the average reward and % optimal action curves (against steps) were shown. For some unknown reason, only one was shown in each of Figures 2.3 and 2.4. Plot the average reward curves corresponding to Figure 2.3, and the % optimal action curves corresponding to Figure 2.4.

   Written: Observe that both optimistic initialization and UCB produce spikes in the very beginning. In lecture, we made a conjecture about the reason these spikes appear. Explain in your own words why the spikes appear (both the sharp increase and sharp decrease). Analyze your experimental data to provide further empirical evidence for your reasoning.

9. **[Extra credit.]** *Explore: Perform your own computational investigation.*
Code/plot/written: Choose and implement an idea to experiment with, excluding things directly covered above. Describe what you decided to do, the precise experimental conditions, and what you found (using plots and/or other statistics as evidence).
A non-exhaustive list of things to potentially experiment with:

   - Schedules for annealing ε or α
   - Modify the setup of the arms (number, reward distributions, etc.)
   - Experiment with other settings of nonstationarity (different from the gradual random walk in Q7)
   - Understanding and experimenting with gradient bandit algorithms (see Section 2.8, not covered yet)
   - Perform hyperparameter sweeps / grid searches to reproduce / improve on Figure 2.6
   - Introducing costs for switching between arms (because the agent is a robot that physically moves)