COMP-597: Reinforcement Learning - Assignment 1

Posted Thursday January 12, 2023
Due Thursday, January 26, 2023

The assignment can be carried out individually or in teams of two. Further instructions about how to submit will be provided by the TAs.

Bandit algorithms [100 points]

For this assignment, you will carry out some experimentation with bandit algorithms, in order to help you understand what we discussed in class, and to get used to the way in which we will run experiments for other assignments as well. You should submit a notebook with your code, results and explanations.

1. [5 points] Write a small simulator for a Bernoulli bandit with k arms. The probability of success p_i for each arm $i \in \{1, ...k\}$ should be provided as an input. The bandit should have a function called "sample" which takes as input the index of an action and provides a reward sample. Recall that a Bernoulli bandit outputs either 1 or 0, drawn from a binomial distribution of parameter p_k. Test your code with 3 arms of parameters $q_* = [0.5, , 0.5 - \delta, 0.5 - 2\delta$, with $\delta = 0.1$. Generate and save a set of 50 samples for each action. For the test, plot one graph for each action, containing the reward values obtained over the 100 draws, the empirical mean of the values, and the true q_* for each arm. Each graph will have an x-axis that goes to 50, two horizontal lines (true value and estimated value) and a set of points of value 0 and 1.

2. [5 points] Code the rule for estimating action values discussed in lecture 2, with a fixed learning rate α, in a function called `update`, and using the incremental computation of the mean presented in lecture 2, in a function called `updateAvg`. Using the previous data, plot for each action a graph showing the estimated q value as a function of the number of samples, using averaging as well as $\alpha = 0.01$ and $\alpha = 0.1$, and the true value. Each graph should have two curves and a horizontal line.

3. [10 points] Repeat the above experiment 100 times, starting with action value estimates of 0. Each run will still contain 100 samples for each action. Plot the same graph as above, but where the curves have the average and standard error over the 100 runs. Explain in 1-2 sentences what you observe. Which of the α values is better? How do they compare to averaging? If you wanted to optimize further, in what range of α would you look for better values?

4. [20 points] Code the ϵ-greedy algorithm discussed in class, with averaging updates, with ϵ provided as an input. You will run 100 independent runs, each consisting of 1000 time steps. Plot the following graphs:

   (a) The reward received over time, averaged at each time step over the 100 independent runs (with no smoothing over the time steps), and the standard error over the 100 runs

   (b) The fraction of runs (out of 100) in which the first action (which truly is best) is also estimated best based on the action values

(c) The instantaneous regret l_t (as discussed in lecture 3) (averaged over the 100 runs)

(d) The total regret L_t up to time step t (as discussed in lecture 3) (averaged over the 100 runs)

Generate this set of graphs, for the following values of ϵ: 0, 1/8, 1/4, 1/2, 1. Explain what you observe in the graphs and discuss the effect of ϵ you observe.

5. [5 points] For $\epsilon = 1/4$ and $\epsilon = 1/8$, plot the same graphs for $\alpha = 0.1$, $\alpha = 0.01$, $\alpha = 0.001$ and averaging. Explain in 2 sentences what you observe.

6. [20 points] Write a function that implements the UCB algorithm discussed in lecture 2. Set $c = 2$. Plot the same graphs as above for $\alpha = 0.1$, $\alpha = 0.01$, $\alpha = 0.001$ and averaging. Explain briefly the behavior you observe.

7. [20 points] Write a function that implements the Thompson sampling to be discussed in lecture 4. Plot the same graphs as above for $\alpha = 0.1$, $\alpha = 0.01$, $\alpha = 0.001$ and averaging. Explain briefly the behavior you observe.

8. [5 points] For each of the algorithms, pick the best hyper-parameter combination you have observed (explain how you decided what "best" means). Plot together the curves for this setting. Comment on the relative behavior of the different algorithms.

9. [10 points] Let us now consider a non-stationary problem. Let $\delta = 0.1$ and imagine that after 500 time steps, the parameter of actions 2 and 3 become $0.5 + \delta$ and $0.5 + 2\delta$ respectively. Run for each of the three algorithms a fixed value of $\alpha = 0.1$ and the averaging value estimation. For ϵ use values $1/4$ and $1/8$. Using these values, plot *only the reward graph* as above (you should have 2 lines for ϵ-greedy, one for UCB and one for Thompson sampling, for each learning rate setting). Explain what you see in the graph. Based on these results, which algorithm is best suited to cope with non-stationarity?