COMP-579: Reinforcement Learning - Assignment 1

Posted Tuesday January 14, 2025
Due Wednesday, January 29, 2025

The assignment can be undertaken individually or in teams of two. Alongside this document, you have been provided with a partially completed .IPYNB (Jupyter Notebook) file. Your task is to complete the .IPYNB file by adding the necessary code, displaying results, and providing explanations for questions. While following the notation of predefined classes and functions is not mandatory, it is recommended. Also, you may adapt notations to better align with your thought process and coding style. Ensure that you only submit the completed .IPYNB file for evaluation. Your submission should include a comprehensive set of code, results, and explanations. If you have any questions or require clarification, feel free to reach out for assistance on the Ed. We will try to answer your questions within 24 hours. Good luck with your assignment!

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 Gaussian bandit with k arms. The mean reward μ_i for each arm $i \in \{1, \ldots, k\}$ should be provided as an input, and assume a fixed variance σ^2 for all arms. 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 Gaussian bandit outputs rewards drawn from a normal distribution $\mathcal{N}(\mu_i, \sigma^2)$ for the selected arm.

   Test your code with 3 arms of parameters $\mu_* = [0.5, 0.5 - \delta, 0.5 + \delta]$, with $\delta = 0.2$ and $\sigma^2 = 0.01$. 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 50 draws, the empirical mean of the values, and the true μ_* for each arm. Each graph will have an x-axis that goes to 50, two horizontal lines (true value and estimated value), and the set of reward values obtained from the samples.

2. [5 points] Code the rule for estimating action values with a fixed learning rate α, in a function called `update`, and using the incremental computation of the mean, in a function called `updateAvg`. Additionally, implement an approach with an improved decaying learning rate $\alpha_t = \frac{\alpha_0}{(1+\lambda t)^p}$, where α_0 is the initial learning rate, λ is a decay factor, and p is a power parameter, in a function called `updateDecaying`.

   Using the previous data, plot for each action a graph showing the estimated q value as a function of the number of samples, using the following approaches:

   (a) Incremental mean computation (via `updateAvg`).

   (b) Fixed learning rate $\alpha = 0.01$ (via `update`).

   (c) Fixed learning rate $\alpha = 0.1$ (via `update`).

(d) Decaying learning rate $\alpha_t = \frac{\alpha_0}{(1+\lambda t)^p}$ with $\alpha_0 = 0.5$, $\lambda = 0.01$, and $p = 0.5$ (via `updateDecaying`).

Each graph should have four curves (one for each method) and a horizontal line representing the true value of q. The x-axis should represent the number of samples, and the y-axis should represent the estimated q value.

3. [10 points] Repeat the above experiment 100 times, starting with action value estimates of 0. Each run will 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 ($\alpha = 0.01$, $\alpha = 0.1$) performs better? How do these methods compare to incremental averaging and decaying learning rate? When is it beneficial to use a decaying learning rate? 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. Additionally, implement a decaying ϵ such that $\epsilon_t = \frac{\epsilon_0}{1+\lambda t}$, where ϵ_0 is the initial ϵ value and λ is a decay rate constant. Note that if multiple actions have maximum value, you should choose randomly among them (and not always pick, e.g., the one with the lowest index). 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 third action (which truly is best) is also estimated best based on the action values.

   (c) The instantaneous regret l_t (averaged over the 100 runs).

   (d) The total regret L_t up to time step t (averaged over the 100 runs).

   Generate this set of graphs for the following values of ϵ: 0, 1/8, 1/4, 1/2, 1, and for a decaying ϵ with $\epsilon_0 = 1/2$ and $\lambda = 0.1$. Note that the x-axis for the graph will go up to 1000, and you should have on each graph a set of 6 curves, one for each value of ϵ. Explain what you observe in the graphs and discuss the effect of ϵ, including the decaying ϵ.

5. [5 points] For $\epsilon = 1/4$, $\epsilon = 1/8$, and decaying ϵ with $\epsilon_0 = 1/2$ and $\lambda = 0.1$, plot the same graphs for $\alpha = 0.1$, $\alpha = 0.01$, $\alpha = 0.001$, and incremental averaging. Explain in 2-3 sentences what you observe.

6. [20 points] Write a function that implements the gradient bandit algorithm discussed in class. The algorithm should use a baseline to normalize the rewards and update preferences for actions based on the gradient ascent formula. Plot the same graphs as above for $\alpha = 0.1$, $\alpha = 0.01$, $\alpha = 0.001$ and a decaying learning rate defined as $\alpha_t = \frac{\alpha_0}{(1+\lambda t)^p}$, with $\alpha_0 = 0.5$, $\lambda = 0.01$, and $p = 0.5$. Explain briefly the behavior you observe.

7. [20 points] Write a function that implements the Thompson sampling to be discussed in class. Plot the same graphs as above. 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 $\delta = 0.2$ and assume that after 500 time steps, the parameters of actions 2 and 3 change to $0.5 + 2\delta$ and $0.5 - 2\delta$, respectively. Implement the gradient bandit algorithm with fixed values of $\alpha = 0.1$ and $\alpha = 0.01$ for value estimation. For the ϵ-greedy method, use $\epsilon = \frac{1}{4}$ and a decaying ϵ defined as $\epsilon_t = \frac{\epsilon_0}{1+\lambda t}$, where $\epsilon_0 = \frac{1}{2}$ and $\lambda = 0.1$, along with a fixed value of $\alpha = 0.1$ and incremental averaging.

   Plot *only the reward graph*, showing the average reward over 1000 time steps for each algorithm. Include:

   - Four lines for the ϵ-greedy algorithm (two for $\epsilon = 1/4$, and two for decaying ϵ),
   - Two lines for the gradient bandit algorithm,
   - One line for Thompson sampling.

   Explain briefly what you observe in the graph. Based on these results, which algorithm is best suited to cope with non-stationarity? Discuss how each algorithm responds to the parameter change at $t = 500$ and its ability to adapt.