INF8953DE(Fall 2021) : Reinforcement Learning — Given on: Sep 9, 10:00 pm

Assignment #1 — Due on : Sep 29, 10:00 pm

- This is an individual assignment. However, you are allowed to discuss the problems with other students in the class. But you should write your own code and report.
- If you have any discussion with others, you should acknowledge the discussion in your answers by mentioning their name.
- All your solution, code, analysis, graphs, explanations should be done in a single Colab/Jupyter notebook.
- You have to submit the `pdf` copy of the Colab/Jupyter notebook on gradescope before the deadline.
- Submit the actual Colab/Jupyter notebook (.pynb file) on **Moodle**.
- Be precise with your explanations in the report. Unnecessary verbosity will be penalized.
- You are free to use libraries with general utilities, such as matplotlib, numpy and scipy for python. However, you should implement the algorithms yourself.
- If you have questions regarding the assignment, you can ask for clarifications in Piazza. You should use the corresponding tag for this assignment.
- Instructions on how to properly convert your jupyter notebook to pdf can be found here: https://courses.cs.duke.edu/fall18/compsci371d/homework/workflow.html.
- Instructions on how to submit the report in pdf on the Gradescope can be found here: https://www.youtube.com/watch?v=u-pK4GzpId0

1 Bandit Problem

Consider the 10-armed testbed (Sutton & Barto, 2018, Section 2.3), which is a set of n randomly generated 10-armed bandit problems. For each bandit problem, the action values, $q_*(a)$, a=1,...,10, are selected according to a normal (Gaussian) distribution with mean 0 and variance 1. When a learning method selects action A_t at time t, the actual reward, R_t, is selected from a normal distribution with mean $q_*(A_t)$ and variance 1. For this question, consider the number of independent runs n=20, each run consists of 1000 pulls.

1. Create a `class` for the bandit, which contains a function *pull* that takes the action as the input and returns the reward.
2. Implement the ϵ**-greedy** method on the 10-armed testbed described above, for three ϵ values: 0.1, 0.01, and 0. In two separate plots, plot (1) the averaged reward across the n=20 runs as a function of the number of pulls for all three ϵ's and (2) the total regret,

averaged across the n=20 runs, as a function of the number of pulls for all three ϵ's. Explain what you expected to see and what you actually saw.

3. Do you think setting $\epsilon = 0.9$ is a good strategy? Give reasons why it is or it is not. Plot the average reward and total regret plots for $\epsilon = 0.9$. Explain what you expected to see and what you actually saw.

4. Implement the **optimistic initial value** method on the 10-armed testbed described above, for initial action-value estimates $Q_1(a)$=+3 for all a. Compare its performance, measured by the average reward across n=20 runs as a function of the number of pulls, with the non-optimistic setting where $Q_1(a)$=0 for all a. For both optimistic and non-optimistic settings, ϵ=0. Explain what you expected to see and what you actually saw.

5. What happens when you set the initial action-value estimates $Q_1(a) = +100$ for all a? Plot the comparison of average reward for $Q_1(a) = +0$, $Q_1(a) = +3$, and $Q_1(a) = +100$. Explain what you expected to see and what you actually saw.

6. Implement the **UCB algorithm** on the 10-armed testbed described above, for three values of the parameters c: 0.2, 1, and 5. In a single plot, compare their performances as measured by the average reward across n=20 runs as a function of the number of pulls. Explain what you expected to see and what you actually saw.

7. Implement the **Gradient bandit** method with on the 10-armed testbed described above, for 4 settings: (1) reward baseline is the average reward, α=0.1, (2) reward baseline is the average reward, α=0.5, (3) no reward baseline, α=0.1, (4) no reward baseline, α=0.5. In a single plot, compare their performances as measured by the average reward across n=20 runs as a function of the number of pulls. Explain what you expected to see and what you actually saw.

8. How would you compare the performance of all the 4 different methods: **ϵ-greedy**, **optimistic initial value**, **UCB**, and **gradient bandits**? How would you rank them in terms of regret?

2 Dynamic Programming

Consider the tabular environment of 5x5 Gridworld. The nonterminal states are S={1,2,...,22,23}. There are four actions possible in each state, A={`up, down, right, left`}, which deterministically cause the corresponding state transitions, except that actions that would take the agent off the grid in fact leave the state unchanged. This is an undiscounted, episodic task. The reward is -1 on all transitions until the terminal state is reached. The terminal state is shaded in the figure (although it is shown in two places, it is formally one state). Suppose the agent follows the equiprobable random policy (all actions equally likely). Thus, for instance, $p(6, -1|5, \texttt{right})$=1, $p(12, -1|5, \texttt{right})$=0.

1. if π is the equiprobable random policy, what is $q_\pi(11, \texttt{down})$? What is $q_\pi(7, \texttt{down})$?

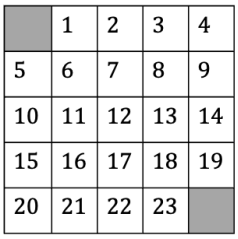

Figure 1: The 5x5 Gridworld

2. Draw a picture, similar to the right side of Figure 4.1 in Sutton & Barto 2018, to describe what you think the optimal policy would look like.

3. Implement tabular **policy iteration** and **value iteration** to find the optimal value function and optimal policy. You could either choose to use this package or design your own gridworld environment in python.

   (a) After every iteration (for every seed), run the estimated policy in the environment for 5 test episodes and report the mean over (i) cumulative reward per episode obtained by the agent, and (ii) the number of timesteps required to solve the task per episode of experience, both reported over 5 different seeds.

   (b) Visualize the final policies obtained using both policy iteration and value iteration. Are they similar to the optimal policy you expected in the second question of this section?