CS533: Intelligent Agents and Decision Making

Investigating Bandit Algorithms and Objectives

You can use any programming language for the project, including MATLAB. In addition to the write-up materials indicated below, you should also send all of your code to me when you submit the assignment. Submit the project via Canvas.

In this project you will implement several bandit algorithms and evaluate them in terms of their cumulative and simple regret on a number of bandit problems.

1 Part I: Create Bandit Algorithms

You should implement the following three bandit algorithms:

1. **Incremental Uniform.** This algorithm repeatedly loops through the arms pulling each arm once each time through the loop. The number of pulls for any two arms will never differ by more than 1. The average rewards for each arm are tracked and for the simple regret objective, the arm with the best average reward is returned as the best arm.

2. **UCB.** This is the UCB algorithm from the notes. In the case of simple regret, when asked to return the best arm, UCB returns the arm that accumulated the largest average reward (it ignores the exploration term of the UCB rule for picking the best arm).

3. **ϵ-Greedy.** This is the ϵ-Greedy algorithm from the course notes, where $0 < \epsilon < 1$ is a parameter to the algorithm. The arm that currently looks best is selected with probability ϵ and otherwise a random arm is selected from among the other arms. Note that if there are k arms, then the $(1/k)$-Greedy algorithm will behave very much like Incremental Uniform, it is just a randomized version of that approach.

You should implement the bandit algorithms so that they have a common API for interfacing to a multi-armed bandit problem. A suggested interface to a bandit problem would be as follows: 1) The bandit problem would have a function *NumArms* that returns the number of arms, 2) The bandit algorithm would have a function *Pull(a)* where a is the index of an arm to pull and the function would return a reward drawn from the distribution associated with that arm.

2 Part II: Bandit Design

You need to design several different bandits that match your standard bandit API. For this assignment we will always maintain that rewards are in the range $[0, 1]$ and we will specify bandits via "a scaled binomial reward distribution (SBRD)" (I made this name up) for each arm as we now describe.

An SBRD for arm a is specified by a tuple (r_a, ρ_a) where $r_a \in [0, 1]$ and $\rho_a \in [0, 1]$. When arm a is pulled it returns a reward of r_a with probability ρ_a and otherwise with probability $1 - \rho_a$ returns a reward of 0. The expected reward of arm a is thus $E[R_a] = \rho_a \cdot r_a$. With these two parameters you can control both the magnitude and the frequency of non-zero rewards received at an arm. A multi-armed bandit problem with k arms is simply a sequence of k SBRDs, one for each arm.

You should design and code an implementation of SBRD bandits in a way that you can specify the bandit parameters. For your experiments you will used the following two bandits and one that you design for a total of three bandits. You should write up a brief description of the bandit problem that you design, indicating why it is an interesting test case.

- **Bandit #1.** Create a bandit with 10 arms. Nine of the arms should have parameters $(0.05, 1)$ meaning they always return 0.05 as the reward. The remaining arm should have parameters $(1, 0.1)$ meaning that 10% of the time it returns a reward of 1. This later arm has twice the expected reward of the others.
- **Bandit #2.** Create a bandit with 20 arms. The i'th arm (for $i = 1, \ldots, 20$) should have parameters $(i/20, 0.1)$. This models a situation where all arms give infrequent rewards that range the entire spectrum of magnitudes.

3 Part III: Evaluating Cumulative Regret

In this section you will evaluate the different bandit algorithms on the three bandit problems defined above in terms of the cumulative regret objective.

For each pair of algorithm A and bandit B, you will produce a curve that shows the average cumulative regret of the algorithm versus number of arm pulls. Each such curve should be produced as follows. For a given A and B you will run T trials, where each trial involves initializing the algorithm and then using it to select a sequence of N arm pulls. For each pull you will record the cumulative regret up until that point. In this assignment we will calculate expected regret based on the observed rewards. The cumulative regret after N arm pulls is $N \cdot R^* - \sum_{i=1}^{N} r_i$, where R^* is the optimal expected reward (only known to the evaluation code) and r_i is the reward obtained at time step i. Thus each trial will produce a graph that is cumulative regret versus number of pulls. After running T trials you will produce a final graph by averaging each of the T graphs produced during the trials. This will provide a graph that estimates the expected cumulative regret of the algorithm as the number of pulls increases.

You should select T and N on a per problem basis in order to satisfy the following conditions: 1) T should be large enough so that the curves are relatively smooth, and 2) N should be large enough so that the algorithms appear to have converged in terms of their performance. It has been common in the past for students to use values of N that are far too small. You should at least let N grow into the millions to ensure convergence.

You should produce curves for at least the following algorithms on each of the bandits: Incremental Uniform, UCB, 0.5-Greedy. You are free to experiment with other values of ϵ, though this is not required. You should write a short summary of the main observations and try to explain any significant differences in performance that you observe.

4 Part IV: Evaluating Simple Regret

Repeat part III, but instead of measuring cumulative regret you should measure simple regret. Note that this requires that your bandit algorithm output at each step the arm that it things is best in addition to suggesting an arm to pull (the two arms may be different). As a reminder the simple regret at time i is $R^* - E[r_i]$, where again R^* is the optimal expected reward and $E[r_i]$ is the expected reward of the arm that was chosen as best by the algorithm at time i (this may be different than the arm that was pulled). Note that the bandit algorithm does not have knowledge of R^* or $E[r_i]$, that knowledge is only available to the code doing the evaluation.