CS 533: Intelligent Agents and Decision Making

HW #3

Aida Rahmattalabi

Problem Definition?

Part I: Create Bandit Algorithms

In this project we will implement several bandit algorithms and evaluate them in terms of their cumulative and simple regret on a number of bandit problems. We will implement three bandit algorithms, **Incremental Union**, **0.5_Greedy** and **UCB**. We create a common API to interface to a multi bandit problem.

The bandit problem has a function called **NumArms** that returns the number of arms, also the bandit algorithm would have a function called **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.

Part II: Bandit Design

For this assignment, we will always maintain that rewards are in the range [0, 1]. We will specify bandits via "a scaled binomial reward distribution (SBRD)" for each arm. 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 $E[Ra] = \rho_a \cdot r_a$. These two parameters 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.

For the experiments, we use two predefined Bandits. Also we define a new Bandit that will be described in the following.

The predefined Bandits can be described as:

Bandit #1. It consists of 10 arms. Nine of the arms 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. This bandit consists of 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.

Bandit #3. We define the bandit as a problem with 20 arms, this time all arms have frequent probability of giving a fixed reward, unlike the Bandit #2, each arm can be described with (0.5, i/20). This model indicates that arms have increasing probability of giving a reward and the expected value of each arm increases as the probability increases.

Following indicates the results for the given bandit problems and algorithms. We run T trials, where each trial involves initializing the algorithm and then using it to select a sequence of N arm pulls. T and N are chosen in a way 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

Part III: Evaluating Cumulative regret

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

Bandit #1:

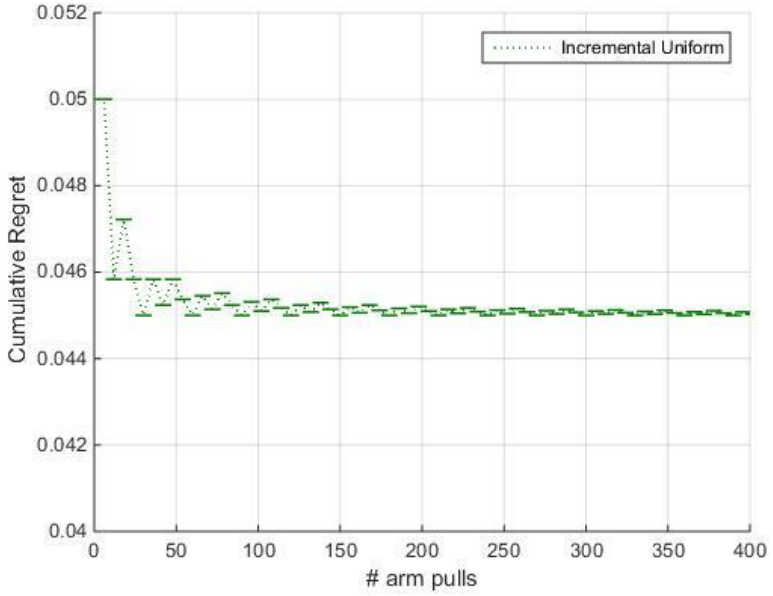

Figure 1: Cumulative Regret vs. arm pulls using Incremental Uniform- Bandit #1

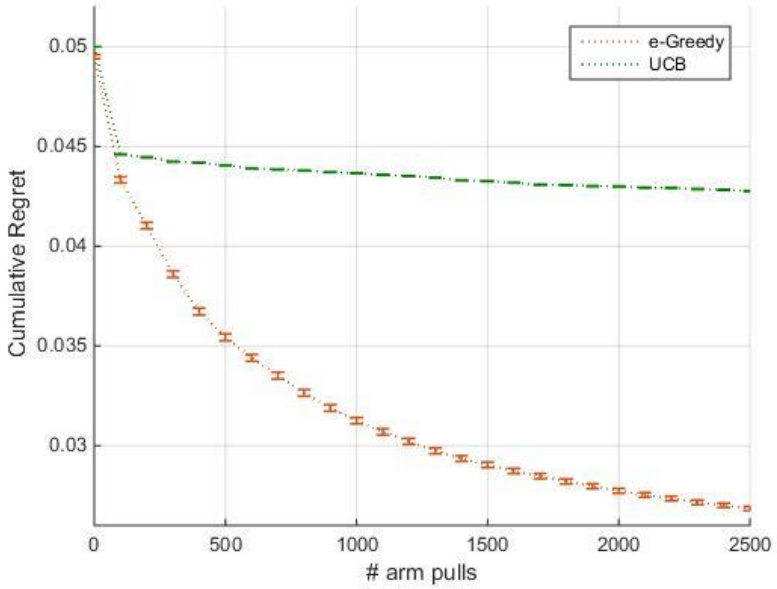

Figure 2: Cumulative Regret vs. arm pulls using 0.5_Greedy and UCB- Bandit #1

Bandit #2:

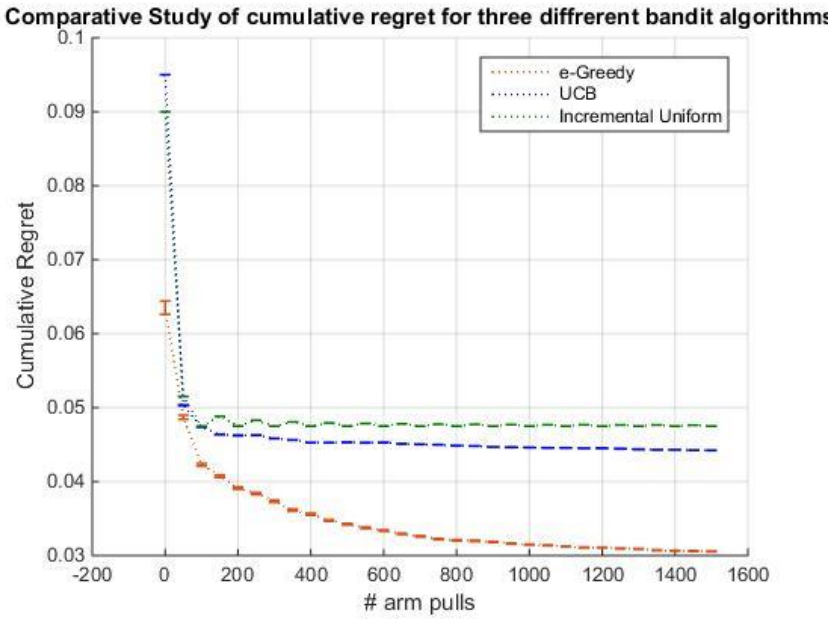

Figure 3: Comparative Cumulative Regret vs. arm pulls using three bandit algorithms- Bandit #2

Bandit #3:

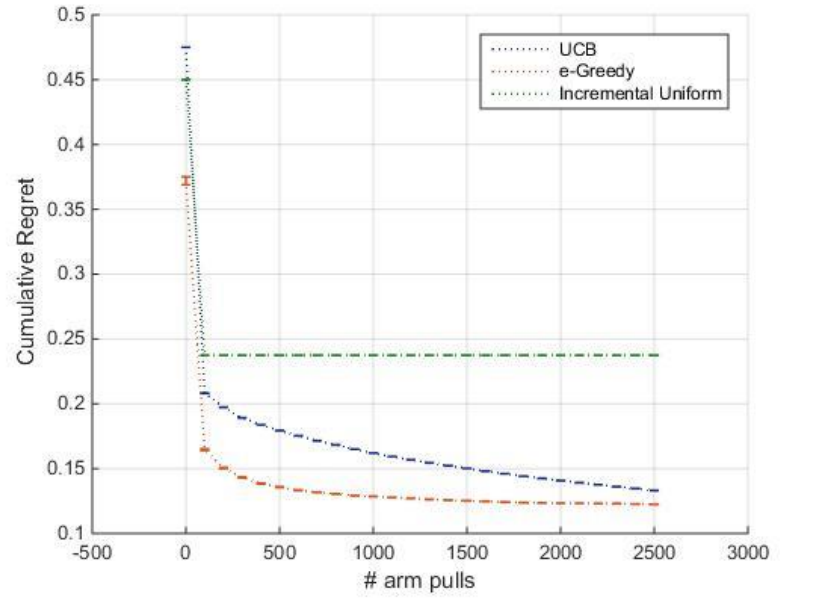

Figure 4: Comparative Cumulative Regret vs. arm pulls using three bandit algorithms - Bandit #3

Part IV: Evaluating Simple regret

In this section, we repeat part III, but instead of measuring cumulative regret we measure simple regret. The bandit algorithm output at each step will be the arm that it thinks is best in addition to suggesting an arm to pull (the two arms may or may not be different). Following is the results for the three bandit algorithms on the specified bandit problems

Bandit #1:

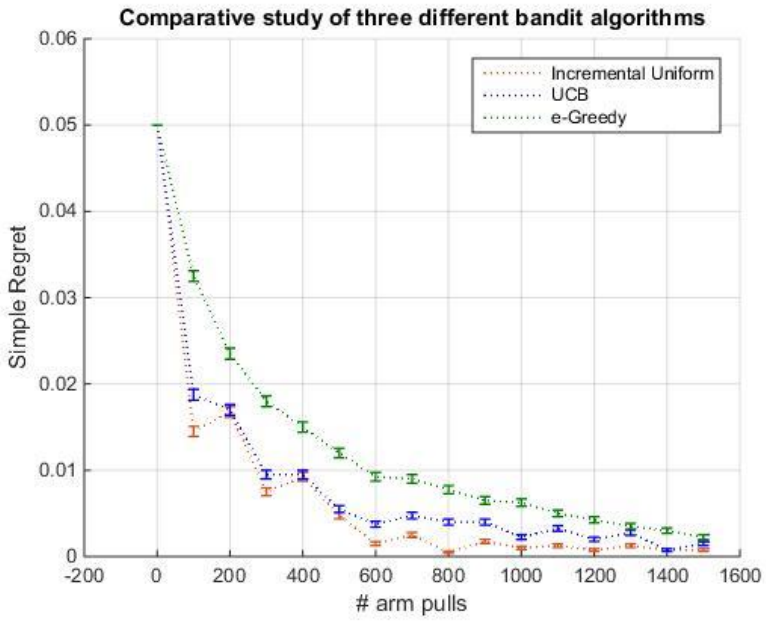

Figure 5: Comparative Simple Regret vs. arm pulls using three bandit algorithms - Bandit #1

Bandit #2:

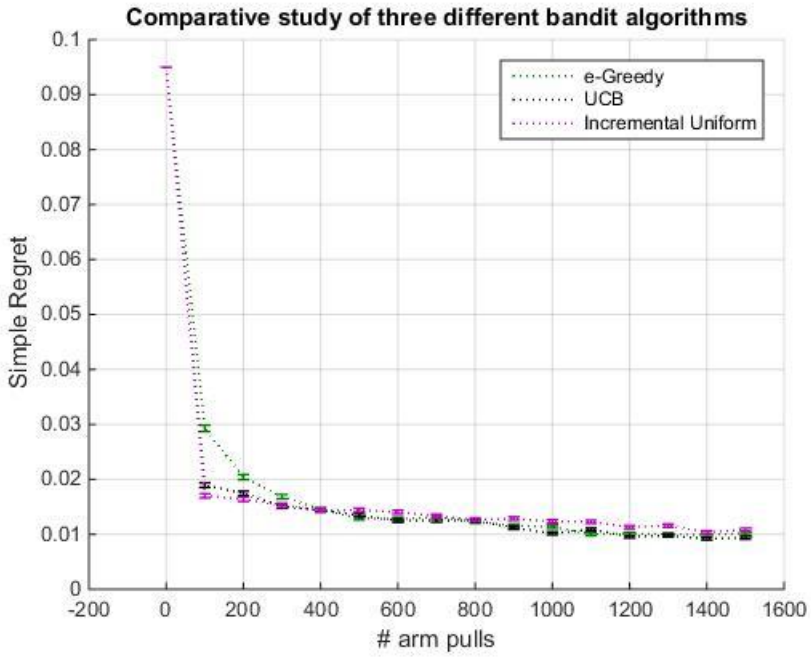

Figure 6: Comparative Simple Regret vs. arm pulls using three bandit algorithms - Bandit #2

Bandit #3:

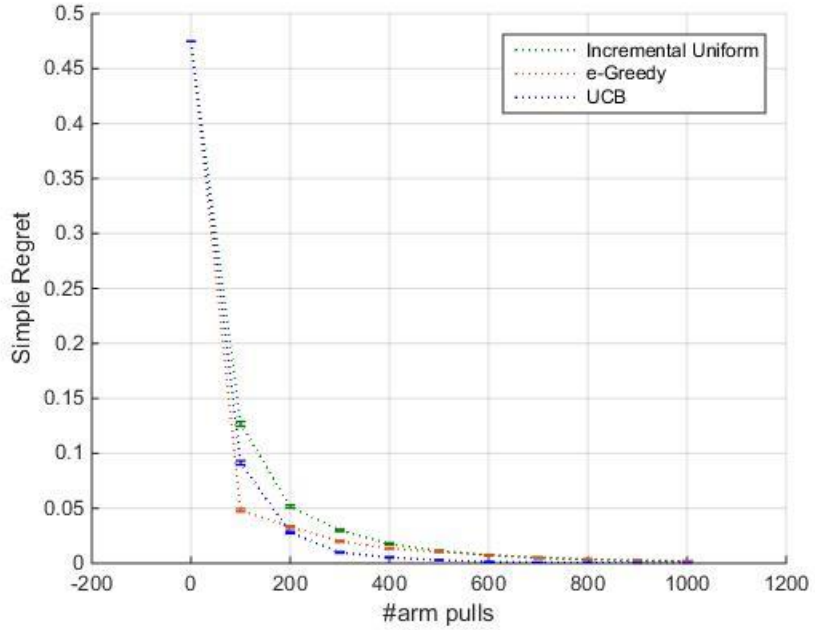

Figure 7: Comparative Simple Regret vs. arm pulls using three bandit algorithms - Bandit #3

Discussion:

Results indicate that for cumulative regret objective, incremental uniform performs the worst, while e_greedy (with e = 0.5) has the best long term performance. However, considering the convergence speed, UBC performs comparably well. e_greedy, however, often requires more time to converge to the final value in cumulative-regret case. This can be due to the fact that e-greedy algorithm takes random choices half of the time (e = 0.5) and this encourages more exploration resulting in better performance in longer time.

In simple-regret case, there is not a huge difference between different bandit algorithms. However, Incremental uniform has performed slightly better (faster) in Bandit#1 and Bandit#2. In evaluating simple-regret, it is useful to have high underlying exploration. Incremental Uniform frequently pulls every arm and this can partially explain why incremental uniform suits well to bandits with simple-regret objective.