Analysis of Epsilon-Greedy and UCB Algorithms in Stochastic Multi-Armed Bandits

Georgios Lamprinakis 2018030017

03/23/2023

Abstract

This report examines and compares the Epsilon-Greedy and Upper Confidence Bound (UCB) algorithms for the stochastic multi-armed bandit problem. We analyze the performance of each algorithm in terms of cumulative regret, cumulative reward, and regret per iteration, using a custom-built Python code. The code generates plots for various time horizons and number of arms, providing insights into the performance and scalability of each algorithm.

1 Introduction

Multi-armed bandits are a well-known class of decision-making problems that involve making decisions under uncertainty. In a stochastic multi-armed bandit setting, the objective is to maximize the total reward obtained over a series of actions by selecting the best arm to play, without knowing the true reward distribution in advance. Two popular algorithms for tackling this problem are Epsilon-Greedy and UCB, both of which balance exploration and exploitation in different ways.

2 Code Overview

The provided code consists of two parts: the definition of the StochasticBanditsEnvironment class, and the implementation of the Epsilon-Greedy and UCB algorithms. The StochasticBanditsEnvironment class simulates a stochastic multi-armed bandit environment with a specified number of arms, each with a randomly assigned reward distribution. The main goal is to maximize the total reward by choosing the optimal arm to play.

2.1 Epsilon-Greedy Algorithm

The Epsilon-Greedy algorithm balances exploration and exploitation by making random actions with a probability epsilon, and exploiting the best-known action otherwise. The provided code computes the epsilon value based on the current time step and the number of arms using a function that scales as $t^{-1/3} \cdot (k \cdot \log(t))^{1/3}$. The algorithm then updates the action-value estimates using a running average and computes the regret at each time step.

2.2 UCB Algorithm

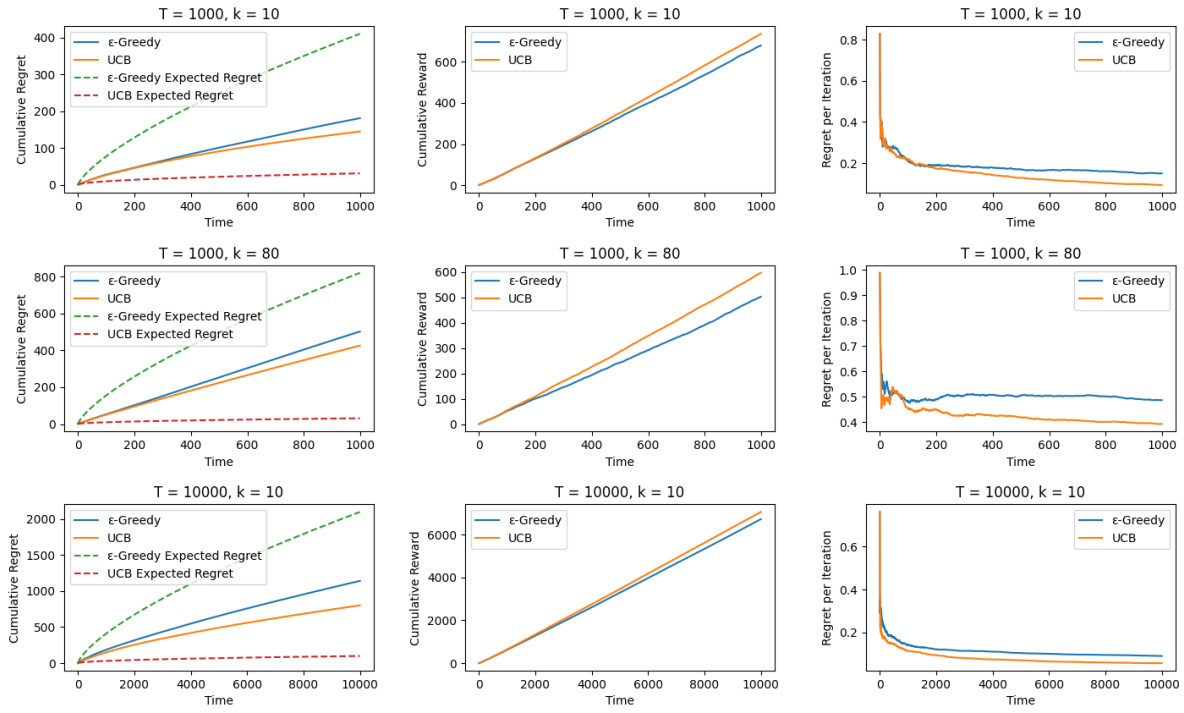
The UCB algorithm, on the other hand, balances exploration and exploitation by taking into account the uncertainty in the action-value estimates. The provided code computes the UCB values for each arm by adding a confidence interval to the estimated mean reward. The action with the highest UCB value is then selected. The algorithm updates the action-value estimates using a running average and computes the regret at each time step.

3 Experimental Setup

To evaluate the performance of the Epsilon-Greedy and UCB algorithms, the code generates plots for various combinations of time horizons (T) and number of arms (k). The chosen values are T = [1000, 1000,

10000] and k = [10, 80, 10]. For each combination, the code simulates the stochastic bandit environment and computes the regret and rewards for both algorithms. The results are plotted as cumulative regret, cumulative reward, and regret per iteration.

4 Results and Discussion



The generated plots provide insights into the performance and scalability of the Epsilon-Greedy and UCB algorithms. The cumulative regret plots show the difference between the optimal reward and the actual reward at each time step, while the cumulative reward plots display the total reward accumulated over time. Lastly, the regret per iteration plots highlight the per-step performance of each algorithm.The results show that both algorithms have different trade-offs and performance characteristics. While the Epsilon-Greedy algorithm has a faster initial convergence, the UCB algorithm tends to achieve better performance in the long run, as it effectively balances exploration and exploitation. The scalability of each algorithm, however, depends on the specific problem instance and the chosen parameters.When changing the values of T (number of time steps) and k (number of bandit arms), we can expect different results in terms of regret and reward.As T increases, the algorithms have more time to explore and exploit the bandit arms, which can lead to better performance in terms of cumulative reward and lower cumulative regret. However, increasing T can also increase the computation time required to run the algorithms.Similarly, as k increases, the algorithms have more bandit arms to choose from, which can make it harder to identify the optimal arm and can increase the amount of exploration required. This can lead to higher cumulative regret and lower cumulative reward. However, having more bandit arms can also make the problem more interesting and challenging.