K Armed Bandit Notes

Colin Manko[1][*]

September 27, 2020

Abstract

Quick overview of the neccessary formulas and concepts for K Armed Bandits. I'm reading Reinforcement Learning by Sutton and Barton.

Introduction

The K Armed Bandit can model a situation where an agent faces k actions, $a \in A$, each with a distribution of possible rewards, $r \in R$, and their corresponding probabilities of occurring, $p(r_t|a_t = a)$. The goal of this situation is to maximize the summation of rewards through time step T, $\sum_{t=1}^{T} R_t$. We do not transition to another state after selecting an action. Instead, we repeatedly face the same situation T times and we keep track of a_n, the number of times action a was selected. As $a_n \to \infty$, we can cacluate an accurate reward expectation, this assumes the probability ditributions over $r \in R|A$ are stationary, or non-changing.

The reward expectation for an action is defined as follows:

$$q_*(a) \doteq \mathbb{E}[R_t|A_t = a]$$

A few interesting parameters and situations arise.

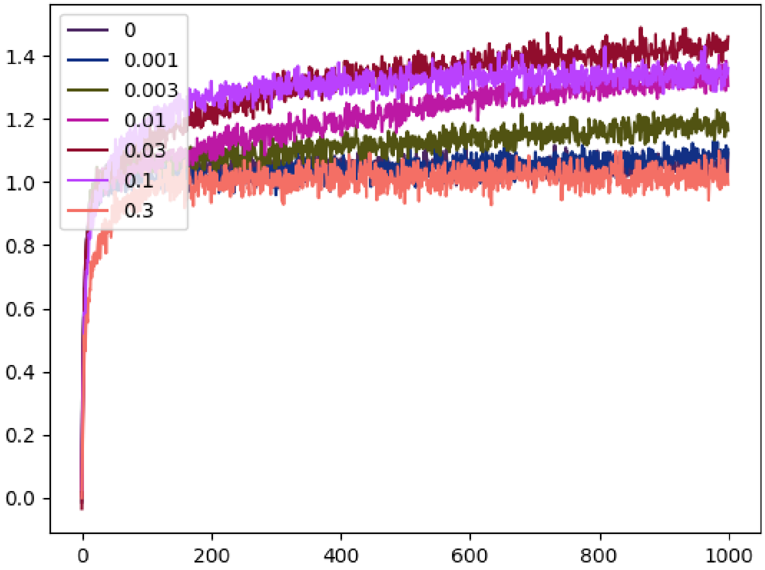
- You can choose the number of actions, k.
- For each a in A,$ you can choose the number of rewards, R_n.
- For each $a \in A$ you can choose a distribution of $R_t|A_t = a$ and the associated probability distribution. For now, we assume all time steps have the same probability distribution for each $R_t|A_t = a$.
  - The probability distribution for each reward can be known or can be figured from data.

Of course, having one reward per action would make finding the optimal action trivial. You can still use a reinforcement learning system to learn that, if for example you don't know $q_*(a)$.

[*] Corresponding author – colin@colinmanko.com

In order to learn this, your reinforcement system cannot always choose the optimal action. Meaning, the system also has to *explore* in order to find the optimal actions. Balancing *exploring* and *exploiting* (taking the *greedy* action) is a delicate process. One needs to be careful that the method of doing so does not violate any conditions surrounding stationarity, etc.

Here are a few simulations over epsilon greedy versus just greedy:



0
0.001
0.003
0.01
0.03
0.1
0.3
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000