Multi-Arm Bandit Optimization

1st Monika Kharadi
Computer Science and Engineering
CS20BTECH11026

2nd Sushma
Computer Science and Engineering
CS20BTECH11051

***Abstract*—Bandit optimization is a subfield of machine learning that involves developing algorithms to maximize rewards in a sequential decision-making scenario where actions have uncertain outcomes.**

I. Introduction

A bandit is defined as someone who steals money. One armed bandit is very similar to the slot machine, except it has k arms instead of one arm. So each time an agent inserts some coins, pulls an arm, and receives the reward associated with the chosen arm. That way, it's undoubtedly stealing a player's money; essentially, it is called a Bandit Problem. It can also be called Multi-armed Bandit Problem as it has multi-arms and dynamic is very similar to a slot machine (arms are referred to as actions). The probability distribution for the payoff corresponding to each arm is different and is unknown to the player. In MAB, the objective is to determine which distribution has the highest expected value and to design a sequence of actions to maximize the total expected reward over a certain period. There are many algorithms to solve MAB problems, but the challenge is balancing exploration and exploitation. Multi-Arm Algorithms are used in Website Optimization, Designing Clinical Trials, Games, Networking, and Online Ad Placement.

II. Bandit Settings

A Multi-arm Bandit Problem consists of a set of $\mathcal{K}$ probability distributions $\langle D_1 \dots D_K \rangle$ with associated Expected Values $\langle \mu_1 \dots \mu_K \rangle$ and Variances $\langle \sigma_1 \dots \sigma_K \rangle$.

III. Formulation

Let T denote the total number of game iterations (i.e., predictions) such that at iteration t, the player chooses x$_t \in \mathcal{K}$, where $\mathcal{K}$ be a convex action subset in R^d. After making a choice, a convex loss function (feedback) f_t: $\mathcal{K} \in R$ is revealed. Only the cost incurred to the player $f_t(x_t)$ for an action x_t is made available as feedback. Bandit Optimization aims to minimize the regret of $\mathcal{A}$ (Algorithm for BCO). The formal definition of Regret of $\mathcal{A}$ that predicted $x_1 \dots x_T$ to be

$$Reg_T(\mathcal{A}) = \sum_{t=1}^{T} f_t(x_t) - \min_{x \in K} \sum_{t=1}^{T} f_t(x) \tag{1}$$

Identify applicable funding agency here. If none, delete this.

In other words, a player's loss due to the number of rounds spent while learning is called **Regret**. Further, the total Expected Regret can be expressed as

$$Reg_T(\mathcal{A}) = \mathbb{E}\left[\sum_{t=1}^{T} f_t(x_t) - \min_{x \in K} \sum_{t=1}^{T} f_t(x)\right] \tag{2}$$

IV. Algorithms

There is a number of different algorithms for the Bandit Optimization Problem. Each algorithm represents a mathematical strategy for maximizing rewards over time. The following algorithms are discussed and implemented in the code.

A. Epsilon-Greedy

The ϵ-Greedy algorithm is a simple method to keep the balance between exploration and exploitation. With probability ϵ we choose a random arm with uniform probability and 1-ϵ probability we are doing exploitation(choosing arm with maximum expected mean)

$$p_i(t+1) = \begin{cases} 1 - \epsilon + \epsilon/k & \text{if } i = \arg max_{j=1 \dots K} \hat{\mu_j}(t) \\ \epsilon/k & otherwise \end{cases} \tag{3}$$

Only fixed values of ϵ are considered in the discussed experiments.

B. Softmax

In mathematical terms, the Softmax algorithm assigns a probability to each arm using a Boltzmann distribution, and based on the probabilities, the arm is selected randomly for exploitation. Given initial empirical means $\mu_1(0) \dots \mu_K(0)$,

$$p_i(t+1) = \frac{e^{\mu_i(t)/\tau}}{\sum_{j=1}^{k} e^{\mu_j(t)/\tau}} \tag{4}$$

where τ is a temperature parameter that controls the randomness of the choice. If τ approaches infinity, Softmax approaches a uniform strategy, a lower value of τ indicates that the probabilities are closer to their expected mean values and as τ tend towards zero, Algorithm is more inclined towards greedy strategy.

C. Upper Confidence Bounds

The Upper Confidence Bound algorithm is based on the principle of optimism in the face of uncertainty. The action chosen at turn t is given by j_t such that

$$j_i(t) = \arg\max_{i \ldots n} \left(\hat{\mu_i} + \sqrt{\frac{2 \cdot \ln t}{n_i}} \right) \tag{5}$$

where n_i is the number of times the same action has been choosen in previous rounds. The LHS in argmax encourages exploitation, whereas RHS in argmax encourages exploration

V. Parameters

- Number of turns for each experiment: 1000
- Number of arms (K): 6

VI. Analysis of Algorithms

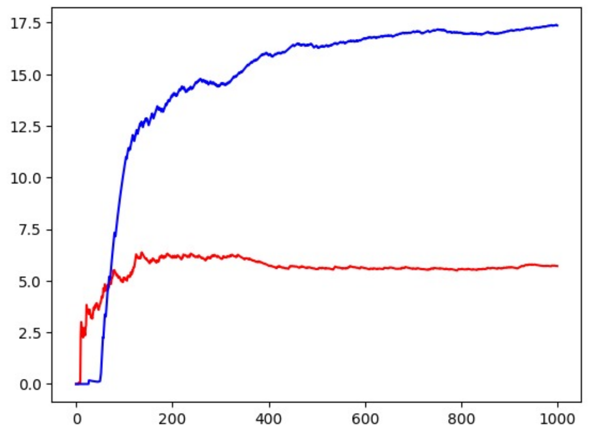

Fig. 1. Comparison between Random and Epsilon Greedy

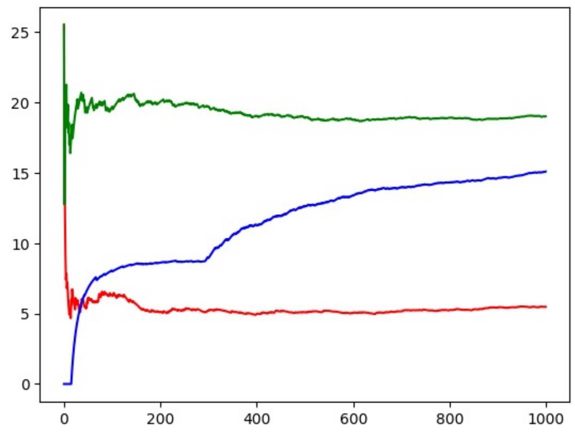

Fig. 2. Comparison of Random, Epsilon Greedy, and Softmax

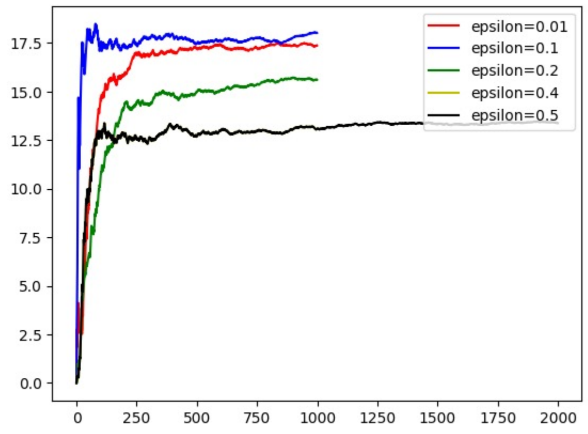

Fig. 3. Comparison for different values of epsilon in ϵ Greedy

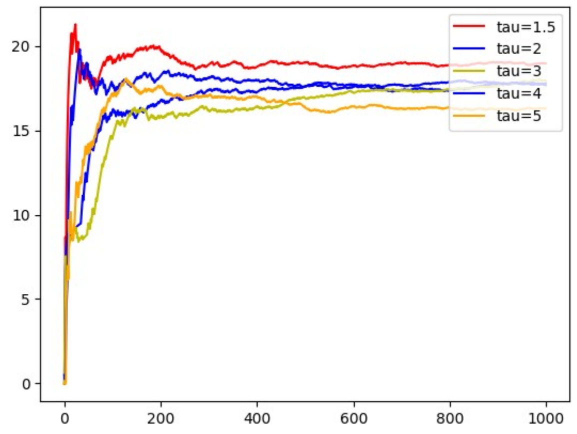

Fig. 4. Comparison for different values of τ in Softmax

VII. Conclusion

We evaluated the performance of Epsilon Greedy, Random, and Softmax algorithms by analyzing the changes in average reward. In general, the algorithm choice depends on the problem's requirements. Epsilon Greedy Algorithm is a good choice in case of large number of arms. A more exploratory algorithm like Random will be more effective if the expected . rewards are noisy. The Greedy algorithm might be a reasonable choice where there is limited number of computational resources. Multi-Arm Bandit Algorithms can also be used in Online Ad Placement where users click on published ads acts as a reward.

References

[1] https://colab.research.google.com/drive/1z9HL5cvA8xl-gvUW1fxKEg482r7gT2Y3?usp=sharingscrollTo=o44-udHv5xRL

[2] https://gibberblot.github.io/rl-notes/single-agent/multi-armed-bandits.html

[3] https://docs.frosmo.com/display/ui/Multi-armed+bandit+optimizationMultiarmedbanditoptimization-UCB1andUCB1-tuned

[4] https://arxiv.org/pdf/1402.6028.pdf