Multi-Arm Bandit Algorithm: Exploring Hyperparameters

Samuel Gerstein
College of Engineering and Computer Science
Florida Atlantic University
Boca Raton, USA
sgerstein2019@fau.edu

***Abstract*—This paper delineates the epsilon-greedy framework of a popular reinforcement learning tool, multi-arm bandits (MABs). MABs allow for the simple evaluation of stateless optimization problems by maximizing the action with the greatest average reward. The theoretical design of the framework is explored, including its various hyperparameters, in order to search for the optimal procedure.**

Additionally, a k-arm testbed is used in two experimental settings. The first experiment simulates MABs given the exact reward, as the second experiment applies MABs onto an advertisement optimization dataset. By exploring these characteristics of MABs, this paper hopes to further understand this paper hopes to understand how to adjust hyperparameters relative to setting of the bandits.

I. Introduction

Multi-arm bandits (MABs) are simple algorithms that can explore and/or exploit the rewards of various possible actions. The bandit is not aware of the exact reward of an action, nor the distribution of awards associated with the action. The bandit chooses an action or arm, receives an action value Q, and then updates the previous mean action value.

The means of how to pick an action is currently under debate, with the dueling priorities of exploration versus exploitation. Both exploration and exploitation are important to maximize the reward accumulated. By exploring each of the actions, the bandit can uncover other distributions that may be optimal. Some algorithms pick the greedy action, define an upper confidence bound (UCB), or choose to take a partially greedy approach with the ϵ-greedy algorithm. This paper will focus on the greedy and ϵ-greedy approaches.

The ϵ-greedy algorithm is popular in bandit implementations due to its balance of exploration and exploitation. With probability ϵ, the bandit will choose a random action. This allows the bandit to gather more information about each action's distribution. With probability 1-ϵ, the bandit will maximize the action-value, accumulating the most reward possible.

II. Algorithm

A. Pseudocode

```
A simple bandit algorithm

Initialize, for $a = 1$ to $k$:
    $Q(a) \leftarrow 0$
    $N(a) \leftarrow 0$

Repeat forever:
    $A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$
    $R \leftarrow bandit(A)$
    $N(A) \leftarrow N(A) + 1$
    $Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\big[R - Q(A)\big]$
```

Fig. 1. ϵ-Greedy Bandit Pseudocode [1]

The psuedocode for the ϵ-greedy bandit algorithm relies on an array-based implementation for each of the actions. This process idealistically repeats forever, but it is restricted to a number of iterations or a specific tolerance value in practice. Once an action-value is received for an arm, the average action-value is updated using [1]:

$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbf{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbf{1}_{A_i=a}} \tag{1}$$

B. Parameters

The parameters that can be tuned in the MAB algorithm include:

- Epochs: Number of iterations the bandit chooses an action.
- Epsilon (ϵ): Probability that the bandit chooses a random action.
- Discount Factor (γ): Rate that future steps will be discounted, $\gamma = 0$ in this implementation.

III. Experiment 1

A. Setup

This experiment models the bandit pseudocode for various values of ϵ. In this scenario, the exact action value q_* and reward at an iteration R_t are found by [1]:

$$q_*(a) \sim \mathcal{N}(0, 1) \tag{2}$$

$$R_t \sim \mathcal{N}(q_*(a), 1) \tag{3}$$

The q_* values generated were:

Arm	Value
1	0.22
2	0.66
3	-0.58
4	0.41
5	0.47
6	-2.0
7	0.28
8	-1.8
9	1.6
10	-0.72

The experiment was run for 10000 epochs, with $\epsilon = 0.1, 0.01$, and 0 (greedy).

B. Results

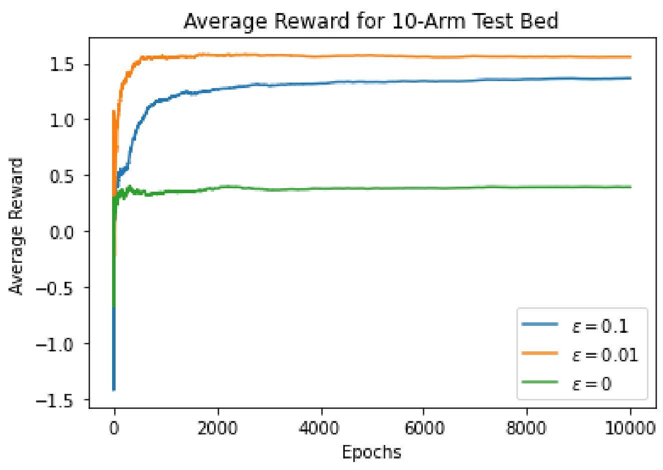


Fig. 2. Comparing average bandit reward for different values of ϵ

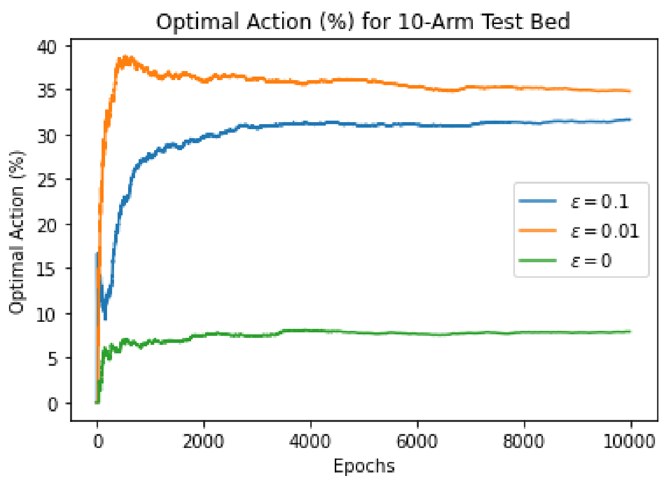


Fig. 3. Comparing percentage of optimal actions chosen for different values of ϵ

C. Discussion

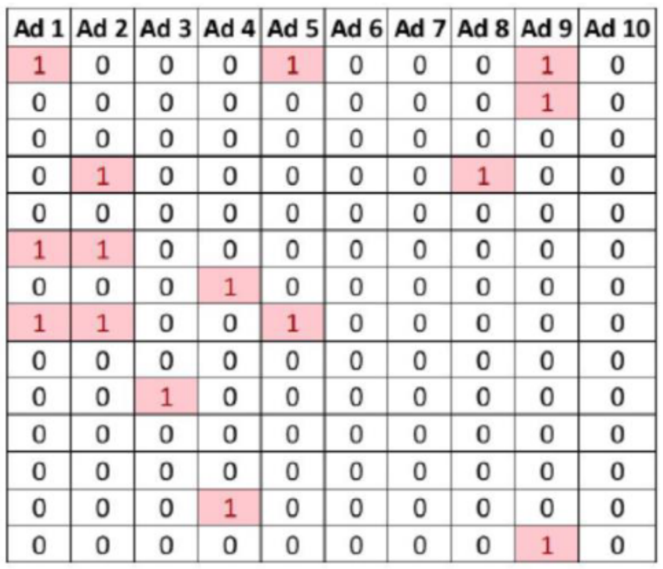
These graphs show the comparison between the epsilon values, and their ability to systemically maximize the reward in this environment. The data shows that $\epsilon = 0.01$ was the optimal hyperparameter, as it consistently converged to the highest average reward and percentage of optimal actions. The second best was the largest value of exploration, $\epsilon = 0.1$, while the least optimal $\epsilon = 0$. These results display how bandit algorithms must carefully walk the line between exploration and exploitation. This experiment validated this concept, with the two ϵ extrema performing the worst.

Although $\epsilon = 0.01$ was the ideal choice in this scenario, further researchers should explore the ϵ parameter relative to a non-I.I.D. (independently and identically distributed) reward distribution. In environments where actions have disproportionate rewards, the need for exploration may decrease.

IV. Experiment II

A. Setup

This experiment directly applies the bandit algorithm to simulate optimizing engagement with internet advertisements. This dataset contains 10 advertisements, along with the ads clicked by 10000 users. Due to the real-world nature of this dataset, the reward distribution is non-I.I.D.



Ad 1	Ad 2	Ad 3	Ad 4	Ad 5	Ad 6	Ad 7	Ad 8	Ad 9	Ad 10
1	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0

Fig. 4. Sample of Experiment II Ad Dataset

B. Results

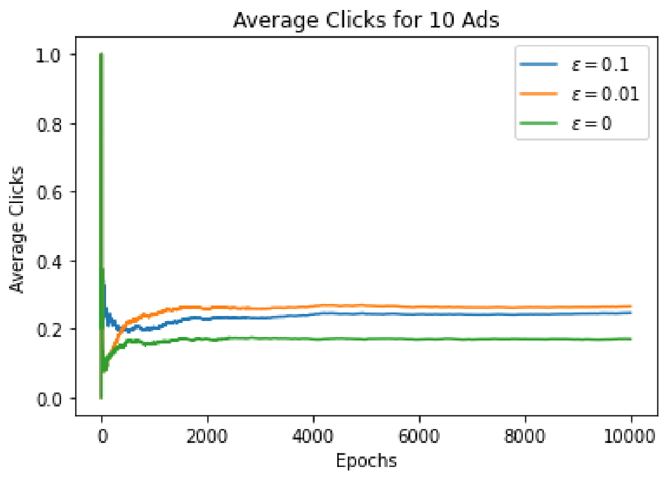


Fig. 5. Comparing average ad clicks for different values of ϵ

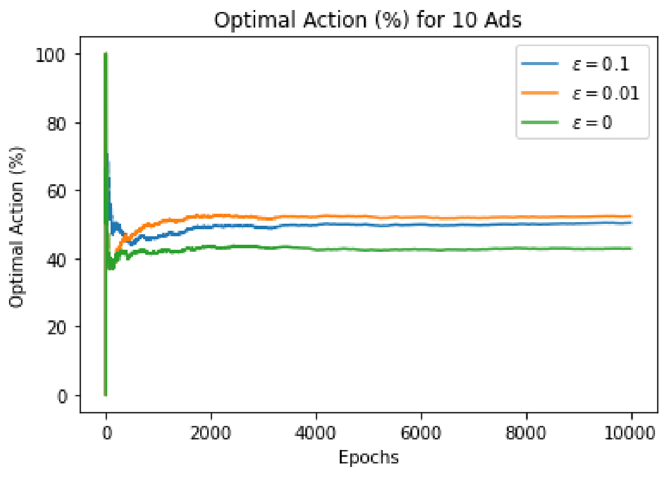


Fig. 6. Comparing optimal ad chosen for different values of ϵ

C. Discussion

This experiment again shows the accuracy of $\epsilon = 0.01$ on a dataset with varying reward distribution. The average clicks converged to 0.2, or two clicks for every 10 users an ad is shown to. The optimal action converged to 50%, meaning that 50% of ads chosen by the bandit were then clicked by the user. The graphs originally exhibited non-monotonic behavior at the first epoch due to the rewards being restricted to 0 or 1, but stabilized with further iterations.

From Figure 7, it can be seen that the optimal advertisement was Ad 5. with the total clicks approaching approximately 2500 in the optimal $\epsilon = 0.01$ case. The additional ϵ-greedy case also had an identical conclusion, but lost clicks by over-exploring Ad 8. The greedy case over-exploited Ad 1, thus unaware of the greater clicks given by Ad 8.

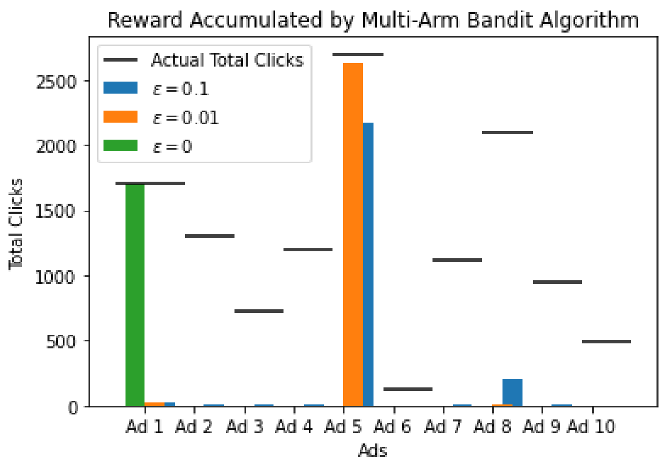


Fig. 7. Comparing total clicks for each ad for different values of ϵ, compared to the true clicks

V. Conclusion

This paper explored both the theoretical and practical characteristics of the multi-arm bandits algorithm (MABs). An ϵ-greedy algorithm was introduced, with the advantage of tuning exploration versus exploitation in a bandit setting.

A simulation was completed for $epsilon$-values 0, 0.01, and 0.1 on an independently and identically distributed (I.I.D.) reward distribution and a non-I.I.D. advertisement optimization dataset. In both cases, $\epsilon = 0.01$ was found to be the optimal parameter, converging to the highest average reward and optimal action percentage in both environments. The bandit correctly chose the optimal ad in this case, displaying the possible applications of bandits in online-policy evaluation.

This paper hopes to build an understanding of basic bandit algorithms, to then explore applications and possible advancements in MABs. One promising technique that can be combined with bandit algorithms is the federated multi-arm bandit, with multiple bandit clients running in parallel. Additionally, bandits can also be helpful in optimizing net metering returns in smart solar microgrids.

References

[1] R. Sutton and A. Barto, Reinforcement learning. The MIT Press.

[2] S. Shawl, "Epsilon-Greedy Algorithm in Reinforcement Learning - GeeksforGeeks", GeeksforGeeks, 2022. [Online]. Available: https://www.geeksforgeeks.org/epsilon-greedy-algorithm-in-reinforcement-learning/. [Accessed: 11- Feb- 2022].

[3] A. Choudhary, "Multi Armed Bandit Problem & Its Implementation in Python", Analytics Vidhya, 2022. [Online]. Available: https://www.analyticsvidhya.com/blog/2018/09/reinforcement-multi-armed-bandit-scratch-python/. [Accessed: 11- Feb- 2022].

Code can be accessed at: https://tinyurl.com/bd2b953u