The Epsilon Greedy Algorithm - a Performance Review

Riti Agarwal

Abstract **— Multi-Armed Bandit (MAB) is a class of reinforcement learning algorithms. A multi-armed bandit implementation has a agent (learner) that chooses between *k* different uncertain actions and receives a reward based on the chosen action. This paper focuses mainly on the Epsilon Greedy Algorithm in comparison to Thompson Sampling and UCB-1 (Upper Confidence Bound). It talks about the benefits of using bandit algorithms over A/B testing and evaluates the effectiveness of the 3 main solutions. It experimentally shows the best use cases for the Epsilon Greedy Algorithm - when the experimentation period is longer than that of A/B testing and you want to exploit the best performing variant. It also talks about when the algorithm does not provide statistically correct results - when the sample size, on each path of the experiment, is very small.**

Index Terms— **exploration, exploitation, regret, reward function, local maxima.**

I. Introduction

The Multi-armed bandit solution is a more intelligent way of doing A/B tests. It is based on a class of Machine Learning (ML) based reinforcement learning algorithms which dynamically allocate more traffic to variations of an experiment that are performing well, while allocating less traffic to the ones that are underperforming. It balances between exploration and exploitation simultaneously during the learning process, helping maximise the expected gain and reduces the amount of regret. The name comes from a gambler at a row of slot machines, who has to decide which machines to play and how much to play each of them, and whether to continue with the current machine or try a different machine. The multi-armed bandit problem also falls into a broader category of stochastic scheduling. The three most popular MAB algorithms are Epsilon Greedy, Thompson Sampling, and Upper Confidence Bound 1 (UCB-1). This paper will focus on the Epsilon Greedy Algorithm and the results it produces when the variations of an experiment have different payout rates.

II. A regular A/B test

To understand the benefits of multi-armed bandit algorithm, we first have to understand A/B testing.

The purpose of A/B testing is to determine the variant of an experiment that is truly more effective than another. In order to make accurate measurements, A/B tests must account for 2 key values: statistical power and statistical significance. Statistical power is the probability that the experiment will detect an effect where an effect is present while statistical significance is the measure of the degree of accuracy of the results. While conducting an A/B experiment, users are divided into 2 groups: the treatment group and the control group. The treatment group is given access to the one variant while the control group has access to another variant. The conversion rate of both these groups is measured for statistical significance. A balanced A/B test typically allocates equal traffic to each group, until reaching a significant sample size (pre-determined by the user). This leads to a high regret (decrease in potential rewards due to executing the learning algorithm instead of performing optimally from the start) as we cannot change traffic allocation during the course of the experiment, according to the results being yielded up until then. Therefore, we say that A/B testing is only an exploration algorithm.

Riti Agarwal, Computer Science Student, Inventure Academy, Bangalore, India

III. Benefits of MAB over A/B testing

Multi-Armed Bandit provides some key benefits over regular A/B testing: it helps conclude the experiment faster and it reduces the overall regret. The two aspects of MAB are exploration and exploitation. Exploration is an attempt to find the more successful variant while exploitation is maximising the reward function. These 2 facets of the experiment run simultaneously to make the MAB algorithms effective. The algorithm maximises the reward function by allocating a majority of the incoming traffic to the variant with a higher conversion rate, while allocating a smaller part of the traffic to the variants that need to be explored further. This way, the user maximises his/her reward function while the testing process is happening and the testing process concludes sooner, allowing the user to make better updates. In summary, the difference between MAB and A/B testing is that A/B testing only allows you to explore while the experiment is running and exploit after the experiment has concluded. MAB allows you to explore and exploit simultaneously, so you don't have to wait for the experiment to conclude before you can start exploiting.

Epsilon Greedy

Pros:

1. Exploits more- it is the most greedy algorithm available
2. Can give results with fewer samples than necessary for Thompson sampling
3. Easy to implement

Cons:

1. If the difference between the reward function of the two variants is very small, the algorithm will route most traffic to the initially winning variant, although this may not actually be the winning variant. Thus, the algorithm would need a lot more data to provide a set of statistically significant results. (the

algorithm might converge to a local maxima rather than a global maxima)

Thompson Sampling
Pros:
1. It is a more principled algorithm, which yields more accurate results even in cases where the difference in the payout rates of the two paths is very less. (always converges to a global maxima)

Cons:
1. It requires a lot of samples to converge to give significant results
2. It doesn't exploit as much as the epsilon greedy algorithm, so lesser cumulative rewards for the user
3. Harder to implement

UCB-1
Pros:
1. Performs consistently over time
2. Successful variant will continue to perform the best while least successful one will remain least popular
3. Once there is enough accumulated data, the algorithm exploits almost all the time
4. Reacts to best performing variation quickly

Cons:
1. Depending on how the algorithm is running, the distribution percentage of a variation may reach 0.
2. Overall performance may lose to Thompson Sampling
3. Hard to implement

IV. Epsilon Value

The epsilon value is set at the beginning of the experiment. It needs to be tuned to fit the needs of the experiment. There is no one value that works best for all experiments. The exploration probability is ϵ, while the exploitation probability is 1-ϵ. A higher ϵ value means higher regret, as there is lesser exploitation. A lower ϵ value means lesser regret, but the algorithm will find the best performing variant faster and it is more likely to be accurate.

V. Example of epsilon greedy implementation

Let's say you are an app developer and you want to see whether a red, yellow, green or blue button attracts more users. You don't want to waste resources and missing out on sales, nor do you want to miss out on the possibility of a great revenue booster. So you decided to use a MAB algorithm, and chose epsilon greedy. Let's assume you set the ϵ value to 0.1. This means the algorithm will route 10% of the traffic equally between the red, yellow, green and blue buttons. It will route 90% of the traffic to the button that attracts the most users. There was 10% exploration, and 90% exploitation.

VI. Experiment

I wanted to find out how long each epsilon value took to converge and whether it converged to the correct variation or not, when given variations with different kinds of payout rates. To do this, I used the Epsilon Greedy Algorithm with epsilon values of 0.1, 0.2 and 0.3 and measured the trial at which it converged for variations with payout rates of [0.01-0.02-0.03-0.04], [0.1-0.2-0.3-0.4], [0.1-0.3-0.5-0.7]. I also measured whether the algorithm converged to the correct variation. We ran the algorithm for 20,000 trials.

These were my results:

Epsilon Value 0.1			
Variation payout rate	0.01-0.02-0.03-0.04	0.1-0.2-0.3-0.4	0.1-0.3-0.5-0.7
Trial number at which it converged	229-434	16129	2478
Did it converge to correct arm	NO (converged to 0.03 arm)	YES	YES

As evident, with a low epsilon value and a small difference between payout rates, the algorithm converges to a variation that doesn't necessarily have the highest payout rate. This is because the third variation started doing well in the beginning, and since the algorithm is so "greedy", it started routing most of the traffic to this variation, without exploring the others as much. In these cases, when there is not much difference between the payout rates, it is better to use A/B testing, as this yields more statistically accurate results. When the difference between the payout rates is a little more [0.1-0.2-0.3-0.4], it converges to the variation with the highest payout rate but takes a slightly longer time to do so. With a much higher difference in payout rates, the algorithm converges to the variation with the highest payout rate in a shorter period of time. In these use cases, the epsilon greedy algorithm is a better choice as it significantly reduces regret.

Epsilon Value 0.2			
Arm Payout rate	0.01-0.02-0.03-0.04	0.1-0.2-0.3-0.4	0.1-0.3-0.5-0.7
Trial number at which it converged	-	2524	2119
Did it converge to correct arm	NO	YES	YES

With an epsilon value of 0.2, there is 80% exploitation and 20% exploration, so with a higher exploration percentage the algorithm did not converge to a variation with a lower payout rate. In the first scenario, with little difference in the payout rates, the algorithm was unable to converge; it kept routing different amounts of traffic to the variations. In the second case, when the payout rates were not as close together, the algorithm converged much faster than it did with an epsilon value of 0.1. This is due to the higher exploration. When the payout rates are very far apart, the algorithm converges early enough, just like before but the trail number at which it converged doesn't change as much.

Epsilon Value 0.3			
Arm Payout rate	0.01-0.02-0.03-0.04	0.1-0.2-0.3-0.4	0.1-0.3-0.5-0.7
Trial number at which it converged	396	626	3599
Did it converge to correct arm	YES	YES	YES

In this scenario, with an epsilon value of 0.3, there is 70% exploitation and 30% exploration. With this much exploration, the algorithm converged in a short period of time even when the difference in payout rates was minimal. One thing we notice about this is that when the difference between the payout rates increase, the trial number at which the algorithm converges also increases. This is because with such high exploration, the algorithm hits "local maximas" - a short period of time where the algorithm converges to a variation with a lower payout rate. In the second scenario, when the payout rates were [0.1-0.2-0.3-0.4], the algorithm had a local maxima from trail numbers 36 to 87, before it moved on to finding the correct maxima. A similar thing happened with the third scenario, with payout rates of [0.1-0.3-0.5-0.7]. It hit local maxima between trail number 39 and 443.

VII. Conclusion

1. The Epsilon Greedy Algorithm is a very useful algorithm when you want to exploit a lot and the difference in payout rates is a lot. It works well with relatively smaller samples as well, as compared to Thompson sampling.
2. When the difference between payout rates is very small and the sample size is not big enough, the algorithm may converge to local maxima.
3. It is important to select the correct epsilon value, a very high epsilon value can lead to more regret, as there is less exploitation and higher chance of a local maxima occurring.

References

[1] https://github.com/KaleabTessera/Multi-Armed-Bandit
[2] https://towardsdatascience.com/comparing-multi-armed-bandit-algorithms-on-marketing-use-cases-8de62a851831
[3] https://www.optimizely.com/optimization-glossary/multi-armed-bandit/
[4] https://towardsdatascience.com/beyond-a-b-testing-multi-armed-bandit-experiments-1493f709f804
[5] https://frosmo.com/multi-armed-bandit-optimization-makes-testing-faster-and-smarter-with-machine-learning/
[6] https://vwo.com/blog/multi-armed-bandit-algorithm/
[7] https://docs.frosmo.com/display/ui/Multi-armed+bandit+optimization#Multiarmedbanditoptimization-Multi-armedbanditalgorithms
[8] https://imaddabbura.github.io/post/epsilon-greedy-algorithm/#:~:text=epsilon%2DGreedy%20Algorithm%20works%20by,N%20of%20selecting%20any%20option.

Riti Agarwal https://www.linkedin.com/in/riti-agarwal-0310051a9/