Multi-armed bandits for dynamic movie recommendations

Making the best recommendations to anonymous audiences

Brian O'Gorman Follow
Apr 11, 2018 · 10 min read

Maybe they would like V for Vendetta?

Brian O'Gorman has a PhD in Physics from UT Austin, and was most recently a consultant at Princeton Consultants. He was an Insight Data Science Fellow in the winter of 2018 in New York. Here he describes his Insight project: a method for making recommendations to anonymous audiences using multi-armed bandit algorithms, and how they realize less regret than approaches like A/B testing.

. . .

Many online streaming services, like Netflix, Amazon Prime, Hulu, etc., recommend movies or TV shows to their users in order to increase the users' engagement with the service. If the service makes good recommendations, especially early in user adoption, then the users will watch more content. Incorporating methods that adaptively respond to new users, like multi-armed bandits, can provide a better service and generate more user engagement.

The typical strategies employed by these recommendation services are **collaborative filtering** and **content-based filtering**, and sometimes hybrid systems that combine the two. Both of these approaches require viewer-specific data in order to perform well. **Content-based filtering** focuses on the tastes of a single viewer, recommending content that is

similar to things the viewer has liked in the past. After gathering data on multiple users' preferences, a **collaborative filtering** strategy can make recommendations to a viewer based on content that has been liked by other viewers with similar tastes.

In each case, the need for collecting viewer-specific data diminishes the effectiveness of these systems until such data can be gathered. This is typically referred to as the "cold start problem." Furthermore, if the audience is anonymous, in the sense that it is not possible to identify returning viewers, then it is not possible to generate viewer-specific recommendations using these techniques because the necessary data cannot be acquired.

Multi-armed bandit algorithms and A/B testing strategies are two potential ways to confront these challenges. In this article, we will take a look at both methods, use them for a movie recommendation task, and compare the results.

. . .

Approaches for anonymous audiences

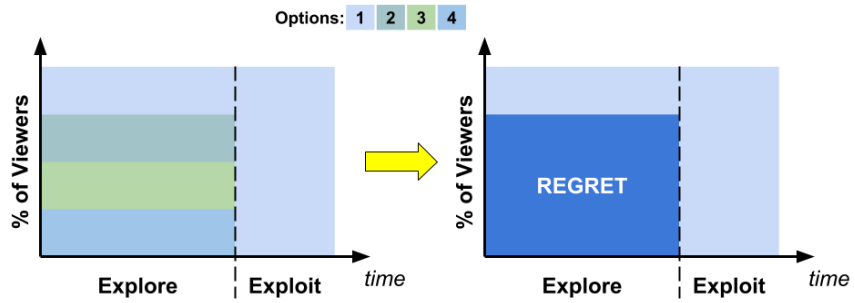
A/B tests

A standard approach for evaluating select variants in an online setting is A/B testing. As the name implies, the technique is an experiment to determine the performance of two options, "A" and "B." These could be ad banners or webpage formatting styles, for example. Option "A" generally represents what is currently in use and acts as a control to compare to choice "B," though this need not be the case. In a real setting, any number of alternative options can be tested at the same time.

During an A/B test, each variant is presented to an equal number of viewers to **explore** its performance. After the test concludes, the best option is identified and used exclusively, **exploiting** the knowledge that was gained from the test.

One drawback with A/B testing is that it incurs "**regret**," which is the concept that during the test, inferior options were presented to some viewers who may have had a better interaction with a better choice. For example, imagine performing an A/B test that compares different ad banners and finds that one banner leads to more conversions than the others. In this context, regret refers to the loss of conversions of viewers

that did not click on the inferior ad that they were shown, who would have clicked on the best ad if it had been presented to them.

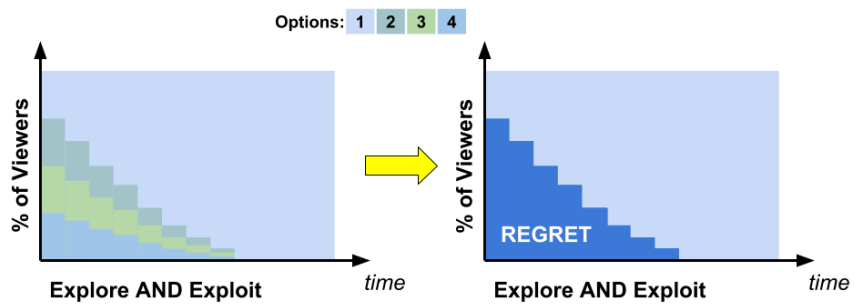


During an A/B test (left of the dashed line), equal percentages of viewers are presented with each of the options (1–4). After the test (right of the dashed line), the best option (1) is used exclusively. Since suboptimal variants (2–4) were used during the test, the exploration phase necessarily caused losses that are regrettable.

With regular use of A/B testing, regret quietly affects a company's bottom line over time, through things like lost conversions or less-than-ideal user experiences. For any company that regularly runs tests to find an optimal version, minimizing regret as much as possible could provide a significant advantage.

Bandit algorithms

Bandit algorithms can reduce the amount of regret that occurs with A/B tests because they continuously balance exploration with exploitation. After every new sample, the knowledge that was learned is used to make a better choice the next time around. Over time, the options that perform better are used more often than the underperformers, and eventually the best option wins out.



A bandit algorithm begins with equal percentages of viewers being presented with each of the options (1–4). As it learns from each experience, the algorithm begins showing the best option (1) more frequently, and the lesser options (2–4) less frequently, leading to less overall regret.

An additional advantage of bandit algorithms is that they can adjust to new options, because there is no "testing phase." If a new choice is added to the set of available variants, the bandit will begin to explore that option and exploit it if it performs better than the existing variants.

How the balance of exploration and exploitation is achieved depends on the particular bandit algorithm. One of the simplest bandit algorithms is the ε-Greedy algorithm, which uses a parameter ("ε") to control the percentage of time that a random option is used—corresponding to exploration. The remainder of the time, the option that has historically performed the best is used—corresponding to exploitation.

Another bandit is the Thompson Sampling algorithm, which is a probability matching algorithm. It tries to match the probability that a particular choice is used to the probability that that choice is the best one. To accomplish this, each tested option is treated as having an intrinsic probability of resulting in a positive user interaction. To make a selection for a viewer, each option's probability distribution is sampled and the one with the highest probability of having a positive interaction is used. After observing the response, the estimate of that option's probability distribution is updated for the next selection.

There are other bandit algorithms as well, such as the Softmax algorithm, which can be used to minimize negative interactions, at the cost of using the best option less often. Choosing which bandit algorithm to use depends on what is being tested and the priorities of the tester.

. . .

Recommending the best movie

Bandit algorithms and A/B tests are good choices for anonymous audiences, because they do not require viewer-specific data for identifying optimal variants. They are commonly used for selecting the best ad banners to display or choosing between website formatting styles. In principle, however, bandits can be used in any situation where there are a number of choices that each have a different rate of positive user interaction. Bandits provide the most benefit when there is a cost associated with making a suboptimal suggestion.

In this context, we can consider how an A/B test and a bandit algorithm would perform when making movie recommendations to an

anonymous audience, where collaborative and content-based filtering options are not possible. When a person watches a movie, she might enjoy the experience or she might not. From this viewpoint, bandit algorithms can be used to identify the "best" movie out of a collection of movies. In this scenario, the algorithm will make a recommendation to a potential viewer, and if the person watched and enjoyed the movie, that would be a positive interaction.

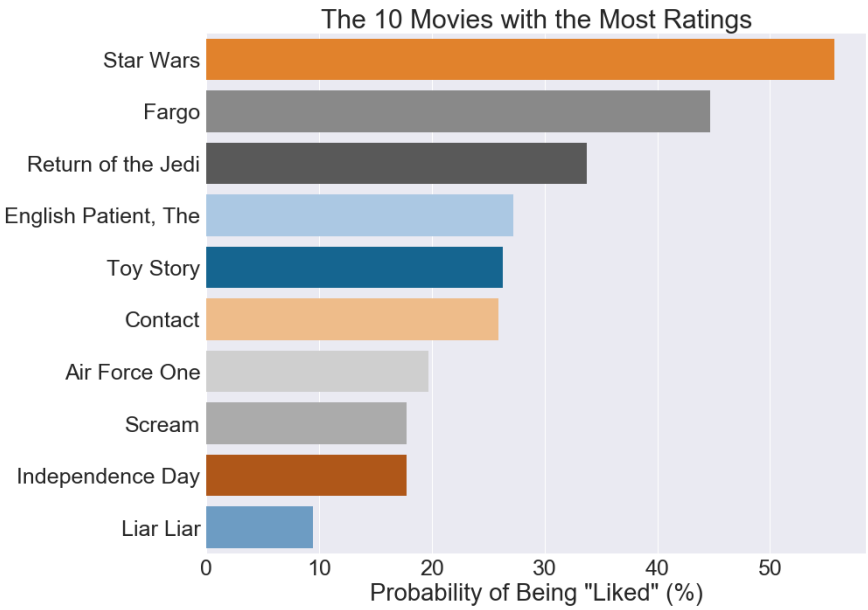
Replay for online tests

Both bandit algorithms and A/B tests are online algorithms, which means that they do not have a full set of data upfront to be trained on. Instead, they learn incrementally as the data is accrued. Their performance can be evaluated offline, however, through backtesting using a technique known as the "replayer" method. This works by predicting which option the algorithm would select for a viewer, and if there is a historical record of the viewer's interaction with that option, the result—whether the experience was positive or negative—is counted as if it had happened. If there is no historical record of that interaction, it is ignored. Using the replayer method, we can evaluate and compare bandit algorithms and A/B tests on a historical dataset of movie ratings.

Movie ratings data

The MovieLens 100K dataset contains 943 viewers and their ratings for 1,682 movies. The ratings are on a scale from 1 (worst) to 5 (best). To translate this discrete range into a simple positive or negative experience for the viewer, a rating threshold can be chosen to delineate whether the viewer "liked" or "disliked" the movie. Since a rating of 5 is the best, and clearly those who gave that rating liked the movie, it is a straightforward choice for the threshold. In this demonstration, if a viewer gave a movie a rating of 5, the user "liked" the movie, otherwise the user "disliked" the movie.

The movies in the dataset do not have the same number of ratings. As a consequence, more computational time would be needed to identify the best movie, given the constraints of the replayer method. In the interests of having a denser dataset, we will focus on the ten movies with the most ratings. These movies are summarized in the following figure, which shows the probability—based on the historical data—that a viewer liked the movie and gave it a rating of 5.



Ground truth probability for a movie to be "liked"—i.e., given a rating of 5 by a viewer. In this set of movies, Star Wars is the most liked movie and Liar Liar is the least liked.

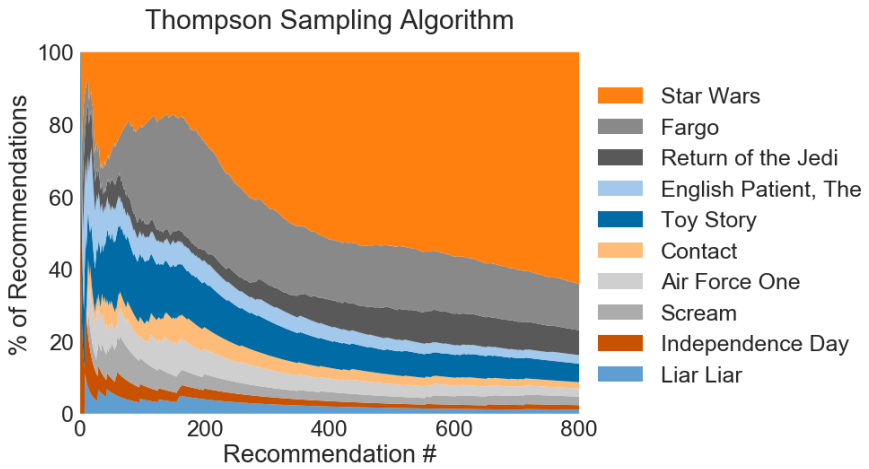
For example, the original *Star Wars* movie was liked by more than 50% of the viewers who rated it, whereas Liar Liar was liked by less than 10%. Based on this data, the bandits and A/B tests should determine that *Star Wars* is the best movie to recommend in the set.

. . .

Bandit results

As discussed earlier, bandit algorithms dynamically balance exploration and exploitation. They start out using all possible options, but continuously shift towards choosing the better options more often until one prevails as the best.

The following figure summarizes the first 800 recommendations made during a single replay of a Thompson Sampling bandit on the movie dataset. The figure shows each movie's percentage of the total recommendations made on the vertical axis, as a function of how many recommendations the algorithm made at that moment on the horizontal axis.



The percentage of time that each movie was recommended during the first 800 recommendations. At first, all movies are being recommended, but over time, the best performing option (Star Wars) is recommended more and more often.

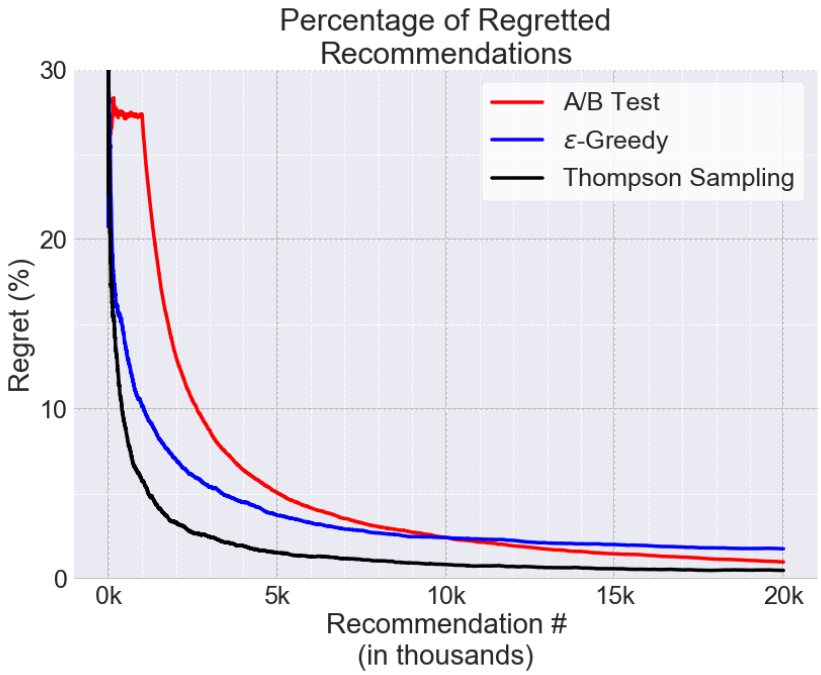
In the beginning, all options are being explored, but over time the best one emerges. Of the first 250 recommendations:

- 91 (~36%) were *Star Wars*
- 63 (~25%) were *Fargo*
- 27 (~11%) were *Toy Story*

After 500 recommendations, *Star Wars* has already accounted for more than 50% of all of the recommendations made by the algorithm; after 1,000 recommendations, its share has surpassed 70%.

Bandit and A/B test comparisons

The cumulative regret experienced over a course of 20,000 recommendations made by three different algorithms (Thompson Sampling, ε-Greedy and an A/B test) is compared in the following figure. The A/B test had a testing phase of 1,000 recommendations. The ε-Greedy algorithm used a value of 0.05 for the parameter ε. All of the experiments were averaged over 20 iterations.



Observed regret as a function of number of recommendations made for three different algorithms: Thompson Sampling (black), ε-Greedy (blue) and an A/B test (red).

The A/B test (red) demonstrates an average performance during the testing period and only after the test has concluded does it show improvement. This makes sense because all ten movies are recommended in equal portions during the testing phase. After the test concludes and *Star Wars* has been identified as the best movie to recommend, it begins to suggest *Star Wars* exclusively and the regret begins to decrease to zero. (It does not immediately drop to zero because we are observing the *cumulative* regret.)

The ε-Greedy results (blue) are significantly better in the short-term than the A/B test, but not as good in the long-term—after ~10,000 recommendations. This is due to the simplistic approach that this particular ε-Greedy algorithm uses to balance exploration and exploitation: the parameter ε specifies the fraction of time that the bandit spends exploring, but that fraction remains constant. As a result, the ε-Greedy bandit will continue to explore at the same rate, even after the best movie has been found, leading to suboptimal performance in the long run. (There are ε-Greedy algorithms that adjust the value of ε over time to account for this, for example, the ε-Decreasing algorithm.)

The Thompson Sampling results (black) are the best of them all. This bandit performs better than the ε-Greedy bandit because it dynamically adjusts the rate at which it explores—rather than using a constant rate.

In the beginning, it explores more often, but over time, it explores less often. As a result, this bandit quickly identifies the best movie and exploits it more frequently after it has been found, leading to high performance in both the short- and long-term.

. . .

Summary

The experiments of movie recommendations clearly demonstrate that the bandit algorithms can reduce regret. The observed regret from the bandit recommendations quickly falls below that of the A/B test, which makes a large quantity of suboptimal recommendations during the testing phase and therefore requires a long recovery time.

Different situations require different bandit algorithms. Depending on the circumstances and the specific needs at hand, it may be better to use a bandit that converges faster—if short-term performance is critical—or one that operates more slowly but results in better long-term performance. While bandit algorithms are a powerful tool, they do have their drawbacks compared to traditional A/B testing. Since bandit algorithms reduce the usage of inferior options, it takes longer to establish a statistically significant measurement of their performance, which can be an important consideration. Ultimately, these trade-offs must be weighed in order to select the right approach for the given situation.

For additional details about the implementation and analysis presented in this article, please refer to my GitHub repository.

. . .