The Multi-Armed Bandit

huey sun

Motivation

It's 8PM on a Tuesday night. You told yourself "just one episode" at 4 as a reward for going to class, but now even Frary has closed and you no longer have Kit Harrington to distract you from your growling stomach. You pull out your phone to Yelp some Thai food, but the top results seem fishy: they only have a handful of reviews, all of which read like the handiwork of an unenthusiatic shill, and you've heard sketchy things about Yelp's business practices. You suddenly remember your buddy raving about this new place that opened up in the village, but then realize that this is the same guy who enjoys Kraft cheese on his salads. Time's ticking: you've got a list of potential spots from Google, an insatiable craving for Pad Thai, and a really annoyed stomach—what do you do?

Description

The Multi-Armed Bandit Problem (MABP) is an optimization problem where resources are allocated among choices with ambiguous or unknown properties, which can be better characterized as time passes or as resources are dedicated towards the choice. Imagine a slot machine (the bandit, as it is programmed to try to steal our money away) fixed with multiple levers (arms), each with a different payout. Without knowing the probability distribution of winning for any of the levers, we want to figure out the order in which we should pull them to maximize the amount of money we take home.

Compare this to our Thai Food example: faced with incomplete knowledge of each option, how should we proceed to maximize our satisfaction after going out, and how does that change after different nights of food experimentation?

We face the **Exploitation vs Exploration Dilemma**, which juxtaposes two different approaches to the problem. Let's say we know that Sanamluang is pretty good (7 out of 10 meals are superb), and we don't know too much about the other options at hand. The maximum exploitation approach would be to repeatedly choose the optimal current option, and pitch a tent outside of Sanamluang. On other hand, a fully exploratory approach would involve trying every single Thai place in a given radius before settling on the best one. As we might imagine, there are tradeoffs to both approaches: we could risk not ever finding that perfect 10 in the first, or risk eating a lot of mediocre Thai food in the second. There's a balance to negotiate, and our task is to find an algorithm that guides this decision-making.

The Math

Let's formally define the Multi-Armed Bandit problem. In the stochastic formulation, the bandit is represented as a set of K probability distributions $\langle P_1, P_2, ..., P_k \rangle$, with each P_k corresponding to an expected value μ_k and variance σ_k^2. Each distribution represents a lever in the machine, and initially, the distributions are unknown.

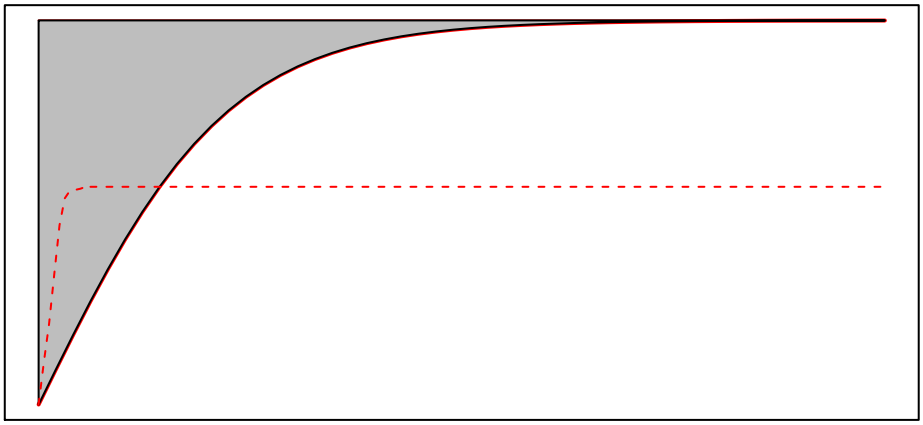
At each turn $t = 1, 2, ...$, the player chooses one of the levers, represented by its index $j(t)$, recieving a reward $r(t) \sim P_{j(t)}$. The Exploitation vs Exploration Dilemma reflects the two main goals the player has: to accumulate the maximum reward at each turn (exploit the current max) and to find the distribution with the highest expected value (exploring to find a better max). A bandit algorithm provides a way to select $j(t)$ at each turn.

We evaluate the preformance of a given algorithm by the total expected regret R_T:

$$R_T = T\mu^* - \sum_{t=1}^{T} \mu_{j(t)}$$

where $\mu^* = max_{i=1,...k}\mu_i$, the maximum expected value.

We can visualize regret (no, not my first-year haircut) as the shaded region below, with the x-axis representing turns and the y-axis representing rewards:



We can imagine μ^* as a constant line that represents the max reward attainable by the player at each turn. Consider $\sum_{t=1}^{T} \mu_{j(t)}$ as the white region under the curve, or the reward that you acculumate by playing. As we explore, $\mu_{j(t)}$ will eventually equal μ^*, but the process of getting there involves a cost of regret. A fully exploitative approach is demonstrated by the dashed line, where we continually pull our current best lever. While our regret is lower initially, our expected total reward ends up much lower as well. In our Thai food example, the gray region represents the cumulative regret of eating sub-optimal food (missing out on some really good stuff) while we were on the hunt for the best spot in LA.

Approaches

There are various approaches to handling exploration. Let's initially consider the purely greedy approach (no exploration, maximum exploitation). The empirical mean $\hat{\mu}_i(t)$ at turn t for arm i (the i_{th} lever, restaurant) is defined as:

$$\hat{\mu}_i(t) = \frac{\sum_{\tau=1}^{t} r(\tau)\mathbb{1}[j(\tau) = i]}{\sum_{\tau=1}^{t} \mathbb{1}[j(\tau) = i]}$$

Essentially, the numerator sums the rewards $r(t)$ each time the i_{th} arm is chosen and the denominator counts the number of times we choose it, giving us the average reward at turn (or time) t. Thus, we want to pull the lever i that maximizes $\hat{\mu}_i(t)$:

$$j(t) = arg \max_{i=1...k} \hat{\mu}_i(t)$$

After initially sampling a few levers, the purely greedy approach continually selects its next move $j(t)$ by looking at the best choice so far. However, as demonstrated in the previous graph, there is a high likelihood (unless you happened to start at the max) that you get stuck somewhere mediocre. Thus, the regret may be unbounded, as we may never get to the lever with the best reward! Exploration is crucial—but how should we handle it?

The ε-Greedy Algorithm

The ε-Greedy Algorithm utilizes random exploration. We use variable ε to control how often we explore. For example, with $\varepsilon = .1$, we would exploit the current max by using our formula for $j(t)$ 90% of the time, and roll a dice to explore another option 10% of the time.

Thus, instead of the probability of choosing an arm $p_i(t)$ being solely determined by whether or not it is the best overall choice so far, we have:

$$p_i(t+1) = \begin{cases} 1 - \varepsilon + \varepsilon/k & i = j(t) \\ \varepsilon/k & i \neq j(t) \end{cases}$$

The i_{th} lever is chosen with a probability of ε/k (where k is the number of levers) if it is not the current best (the probability that you will explore divided by the number of options), and $1 - \varepsilon$ (with the additional exploration probability) if it is the best lever so far. Thus, we are introducing some random exploration into our algorithm. However, as ε is held constant, we can only guarantee a linear bound on our expected regret, which is not ideal.

Looking at the previous graph, one might ask: why don't we just explore until we find the max? Shouldn't we try to guarantee that our current lever is the best one before committing? Exploiting the max doesn't seem nearly as effective as exploring in the long run.

It's a valid question, and one that gets to the heart of the dilemma. The graph, for simplicity, assumes that exploring *always* increases your reward at each turn. In actuality, we would expect the reward line to be far more wobbly: you'll definitely run into a bad restaurant or two while you look around. As we are seeking to minimize the cumulative regret (alternatively, maximize the total cumulative reward), and not just the regret at some far-off turn, we need to consider the costs as we go, or the "opportunity cost" of not taking the best option. The ε-Greedy Algorithm gives us an algorithm to quantify this tradeoff.

However, at this point, our exploration process is still quite naive, as it doesn't take into account our current knowledge of each arm—we explore them randomly with ε/k frequency, indiscriminant of how they are preforming. Thus, even if we are quite certain that an arm is terrible, we will continue to explore it, which increases our regret: ideally, we want some method of exploring our options in a smarter way.

Upper-Confidence Bounds

We can address this by factoring how uncertain we are about a particular lever into how much we want to explore it. Upper-Confidence Bounds take an **optimistic** approach: in the face of uncertainty, we seek to select from the lever until we're relatively certain of how it behaves. At this point, we either determine that it is suboptimal, in which case we never pull the lever again, or we determine that it maximizes our reward, which is what we are looking for.

One way to implement this is by augmenting our decision function to take into account the number of times we have pulled a particular lever, divided by the total number of turns. Thus, our algorithm becomes:

$$j(t) = arg \max_{i=1\ldots k} \left(\hat{\mu}_i(t) + \sqrt{\frac{2 \ln t}{n_i}} \right)$$

where

$$n_i = \sum_{\tau=1}^{t} \mathbb{1}[j(\tau) = i]$$

Basically, our greedy algorithm now also wants to prioritize levers that are rarely examined. Once we have explored an option enough times, $\hat{\mu}_i(t)$ dominates, and we are back to evaluating its mean reward. This allows us to explore across different levers more dynamically.

We can also consider the variance of the probability distributions:

$$j(t) = arg\max_{i=1\ldots k}\left(\hat{\mu}_i(t) + \sqrt{\frac{\ln t}{n_i} min\left(\frac{1}{4}, V_i(n_i)\right)}\right)$$

where

$$V_i(t) = \hat{\sigma}_i^2(t) + \sqrt{\frac{\ln t}{n_i}}$$

In this iteration of the UCB, our uncertainty is manifested in how varied the rewards are for a given lever: the higher the variance, the more heavily we prioritize picking from that distribution. Thus, once we are certain enough that a lever is consistently trash, we will no longer pick it!

The Upper-Confidence Bound approach guarantees that our regret has a bound of $O(\log n)$, which matches the theoretical optimal regret multiplied by some constant. Thus, the UCB is said to *solve* the problem.

Other Ideas

There are various other approaches to the MAB problem:

- **Decaying ε Algorithm**

Decaying-ε is a variant of the ε-Greedy Algorithm where ε decreases (reducing exploration) as the number of turns go up. However, this iteration of the problem still suffers from the drawbacks of random exploration.

- **Softmax (Boltzman Exploration)**

We pick the levers with a probability proportional to the average reward:

$$p_i(t+1) = \frac{e^{\hat{\mu}_i(t)/\tau}}{\sum_{j=1}^{k} e^{\hat{\mu}_i(t)/\tau}}$$

When $\tau \to \infty$, selection is uniform, and when $\tau \to 0$, selection becomes purely greedy. This is a little like simulated annealing!

- **Thompson Sampling**

Thompson sampling is a Bayesian algorithm that uses probability matching. The means μ_i are maintained as Beta distributions, and at each turn t, the algorithm uses the Bayesian inference to update our information and select the optimal lever to play.

Applications

Besides satisfying late-night cravings, Multi-Armed Bandit problems arise quite frequently in real life.

Game Design

With the sheer number of games out there to play, game designers are facing substantial competition, and providing a subpar experience can quickly lead to abandonment. Thus, developpers are under pressure to maximize playtime or "enjoyment," measured through user interaction within a game. Some games such as Infinite Mario feature scaling difficulty levels (exploration) selectively chosen over the defaults (exploitation), balanced to keep the user interested.

On-Click Advertising

Online advertising seeks to maximize the amount of clicks they get, often with incomplete information about the current user (for now... Zuck is always watching). Advertisers are faced with a decision: should they show a new ad (explore) or try something that has been previously successful (exploit). MAB algorithms are used alongside A/B testing, a tournament-style of testing where options are compared side-by-side, with the winner moving on. However, if you have a low conversion rate on your advertisements, accumulating the data for A/B testing can take a long time—and regret may be much higher than a MAB algorithm!

Clinical Trials

Clinical trials are the quintessential example of a bandit problem, and has motivated much of the research in developping new algorithms. In a trial, we want to find the most effective treatment (exploration) while making sure as many patients as possible benefit (exploitation). In traditional experiments, a group will be split in half, and different treatments will be administered to each group. However, *adaptive* strategy has arisen from MAB theory, allowing for better care of the participants.

A 2000 study *Algorithms for the Multi-Armed Bandit Problem* by Volodymyr Kuleshov and Doina Precup (McGill University) demonstrated that "bandit-based treatments would have allowed at least 50% more patients to be successfully treated, while significantly reducing the number of adverse effects and increasing patient retention." Along with these benefits, "the best treatment... [can still be] identified with a high level of statistical confidence!" Thus, the MAB approach is a great alternative to classic clinical trial design.

Concluding Thoughts

Multi-Armed Bandit Problems are optimization problems where the choices are not fully understood. By balancing exploration and exploitation, algorithms can help us pick the optimal choice at each turn to minimize cumulative regret. Upper-Confidence Bounds provide a guaranteed $O(\log n)$ bound on regret, and thus **solve** the problem! This is super cool, because we see a bunch of MABs in everyday life, from choosing where to eat with your friends to managing research projects in the math department!

Sources

My most utilized resources:

https://www.cs.mcgill.ca/~vkules/bandits.pdf

https://www.cs.ubc.ca/~hutter/nips2011workshop/papers_and_posters/Agrawal-Goyal-TS-report.pdf

---

Algorithms, descriptions, and general understanding:

https://towardsdatascience.com/solving-the-multi-armed-bandit-problem-b72de40db97c

https://www.analyticsvidhya.com/blog/2018/09/reinforcement-multi-armed-bandit-scratch-python/

https://www.searchenginepeople.com/blog/16072-multi-armed-bandits-ab-testing-makes-money.html

https://lilianweng.github.io/lil-log/2018/01/23/the-multi-armed-bandit-problem-and-its-solutions.html#case-study

https://arxiv.org/pdf/1707.02038.pdf

https://www.optimizely.com/optimization-glossary/multi-armed-bandit/

https://people.ok.ubc.ca/rlawrenc/research/Students/KR_16_Thesis.pdf

---

Thanks for reading! Hope you enjoyed :-)