Multi-Armed Bandit Strategies

Natalie Hunt

galvanize

Objectives: answer the following:

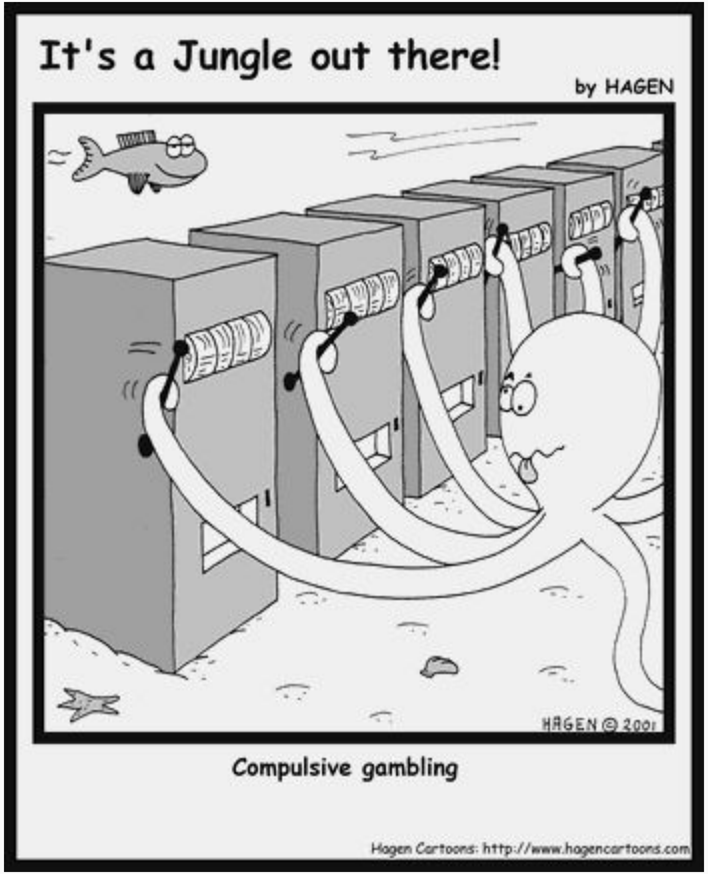
- What are exploitation, exploration, and regret in this context?
- How is this framework related to traditional A/B testing?

Optimizing rewards

- Terminology: a slot machine is called a "one-armed bandit"
- Imagine there are ***k*** slot machines (*bandits*), each with a ***probability of payout*** for a single pull

$$\{p_1, p_2, p_3, \cdots, p_k\}$$

- How do I find out which bandit has the highest probability? What's my best course of action? How do I make the most money from this situation?



Optimizing rewards: CTR again

- Say you've got ***k*** versions of your landing page, each with a click-through probability for a single visit

$$\{p_1, p_2, p_3, \cdots, p_k\}$$

- You send users at random to every page (*bandit*), but soon find that some pages are outperforming others. You don't like losing business by sending users to ugly pages (*bandits*), but you may be stuck waiting for statistical significance from your complicated multiple-hypothesis tests.
- Now your job depends on solving this problem.

Exploration vs Exploitation

- ***exploration***: collecting more data for each bandit to get a better sense of all the success probabilities
- ***exploitation***: using whichever bandit has performed the best so far
- Every strategy for optimization will have to balance exploration and exploitation.

Traditional A/B testing

galvanize

- Starts with ***pure exploration***: both bandits get the same number of users. This is the testing phase.
- Shifts to ***pure exploitation***: whichever bandit performed better is then shown to all users forever.

The Multi-Armed approach

- Show the best-performing site (*bandit*) **most** of the time (several strategies will be discussed for defining exactly how much time)
- As the experiment runs and users see more sites (*bandits*), update your beliefs about which site is best
- each site (***bandit_i***) will have:
    - a number of visits (*rounds* or *pulls*) n_i
    - a number of successes (*wins*) w_i
    - an observed success rate

$$\hat{p}_i = \frac{w_i}{n_i}$$

- Run until a clear victor emerges

Terminology: Regret

- We quantify our failure to pick the best bandit with ***regret***: the difference between the maximum expected reward (if we had picked the best bandit every time) and the expected reward of all the bandits we actually picked
- For each round (*user*), we pick a bandit and observe whether or not it resulted in a success
- Let p^* be the max of $\{p_1, p_2, p_3, \cdots, p_k\}$
- Let $p_{(t)}$ be the true success probability of the bandit chosen at time *t*
- Then our ***regret*** after T rounds is

$$r = Tp^* - \sum_{t=1}^{T} p_{(t)}$$

Regret

galvanize

$$r = Tp^* - \sum_{t=1}^{T} p_{(t)}$$

- We want a strategy that minimizes regret
- A **zero-regret strategy** is defined as one who's average regret per round, **r/T** , goes to **zero** in the limit where the number of rounds **T** goes to **infinity**.
- The interesting thing is that a zero-regret strategy does **not** guarantee that you will never choose a suboptimal outcome.
- Instead it guarantees that as you continue to play you will tend to choose the optimal outcome.
- Note that actually calculating regret requires knowing the true bandit probabilities

Strategies: epsilon-greedy

- **explore** with some fixed probability ϵ, usually 10% or less
  - generate a random number between 0 and 1. If it is less than ϵ, choose a random bandit
- **exploit** at all other times: choose the bandit that has the highest observed success rate so far
- for each bandit, update $\hat{p}_i$ after each round

Is this a zero-regret strategy?

Strategies: UCB1

galvanize

- At round *t*, choose the bandit that maximizes the following expression:

$$\hat{p}_i + \sqrt{\frac{2\ln(t)}{n_i}}$$

$$n_i = \text{number of rounds played on bandit } i$$

$$\hat{p}_i = \frac{w_i}{n_i}, \quad w_i = \text{number of successes for bandit } i$$

$$t = \text{total number of rounds played so far}$$

Strategies: Softmax

- Here we create a ***probability*** of choosing a bandit according to the following formula

$$P(\text{choosing bandit i}) = \frac{e^{\hat{p}_i/\tau}}{\sum_{j=1}^{k} e^{\hat{p}_j/\tau}}$$

$$\tau = \text{“temperature” or “randomness” parameter, usually around } 0.001$$

- You then choose a bandit by sampling from this probability distribution
  - Coding tip: np.random.choice takes a parameter *p* for specifying probabilities. This is the fastest way to make a discrete random variable & probability mass function

Strategies: Bayesian bandit

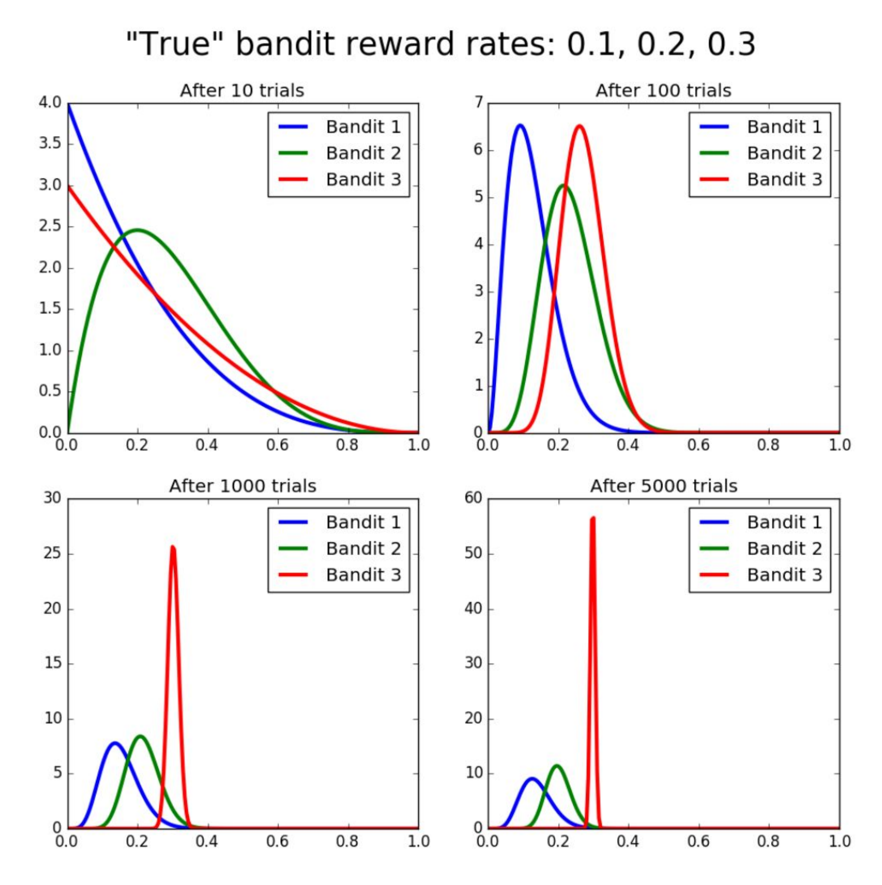
galvanize

- Use Bayesian updating to make a beta distribution for each bandit, where

$$\alpha = 1 + (\#\text{ of times that bandit has won})$$
$$\beta = 1 + (\#\text{ of times that bandit has lost})$$

- Then sample from each bandit's posterior distribution and play the one that gave you the highest probability

Bayesian bandits: simulation



Why multi-armed-bandit?

In pairs discuss the following:

- What advantage does multi armed bandit give us over standard a/b testing?
- Why doesn’t everyone use multi armed bandits?