Better Testing With Bandits

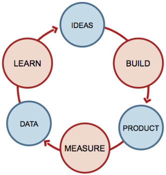
Paul Gribelyuk

April 12, 2014

Table of Contents

High-Level Overview of A/B Testing - aka Things We Already Know

- In this lean, "smaller is better", agile new world, we iterate over the Build→Measure→Learn (or MVP→Measure→Pivot) cycle



- We choose a metric by which to judge the success of our MVP (e.g. click-through rate, time spent on page, revenue, etc.)
- We want to quantify the effect a feature has on the chosen metric.

A/B Testing, A Definition

Definition (from Wikipedia)

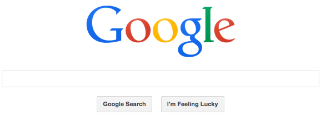
In marketing, A/B testing is a *simple* randomized experiment with two variants, A and B, which are the control and treatment in the controlled experiment. It is a form of statistical hypothesis testing. Other names include randomized controlled experiments, online controlled experiments, and split testing.

A/B Testing, A Definition

Definition (from Wikipedia)

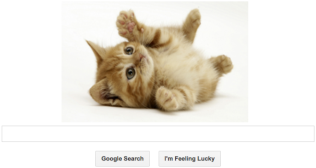
In marketing, A/B testing is a *simple* randomized experiment with two variants, A and B, which are the control and treatment in the controlled experiment. It is a form of statistical hypothesis testing. Other names include randomized controlled experiments, online controlled experiments, and split testing. *In online settings, such as web design (especially user experience design), the goal is to identify changes to web pages that increase or maximize an outcome of interest (e.g.,* ***click-through rate*** *for a banner advertisement).*

(emphasis mine)



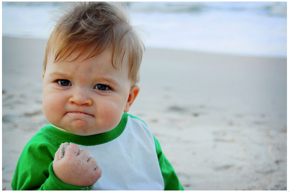
(a) Some Boring Site

(a) Some Boring Site



(b) I Can Haz Search

Figure: Progress



What can we test?

What can we test?

- Affiliate: Change wording, text (font and size), color, placement

What can we test?

- Lead Gen: Reorganize a page, remove forms fields, add form fields

What can we test?

- Email: Frequency, creative, wording, embedded video, reordering of contents

What can we test?

- Display: Change bidding strategy, change interactive nature of ad

What can we test?

- Search: Change keywords, change keyword bidding strategy, change search engines used

What can we test?

- Any publisher: change composition of displayed ads, sizes, positions, quantities, navigation images

What can we test?

- Products: ???

What can we test?

- Affiliate: Change wording, text (font and size), color, placement
- Lead Gen: Reorganize a page, remove forms fields, add form fields
- Email: Frequency, creative, wording, embedded video, reordering of contents
- Display: Change bidding strategy, change interactive nature of ad
- Search: Change keywords, change keyword bidding strategy, change search engines used
- Any publisher: change composition of displayed ads, sizes, positions, quantities, navigation images
- Products: ???

Example (Flipping a Coin)

I put on a Star Wars Tshirt and flip a coin 100 times, getting 48 heads. Next time, I wear a Star Trek (Original) Tshirt and flip the same coin 100 times, getting 52 heads. Did I just achieve and 8% boost in flipping heads?

Example (Flipping a Coin)

I put on a Star Wars Tshirt and flip a coin 100 times, getting 48 heads. Next time, I wear a Star Trek (Original) Tshirt and flip the same coin 100 times, getting 52 heads. Did I just achieve and 8% boost in flipping heads?

This is a silly example but illustrates the potential pitfalls in claims typically made in marketing materials.

The classical testing framework takes the following steps:

1. Specify a null hypothesis: that there is no difference between the two coin-flipping sessions
2. Make an assumption about the distribution of individual events: $P(heads) = P(tails) = \frac{1}{2}$ and $P(\text{k heads from n flips}) \sim Binomial(k, n)$
3. Compute the *test statistic* $T = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{(\frac{\sigma_1^2}{n_1} + \frac{s_2^2}{n_2})}}$
4. Test the hypothesis by considering how likely it is to have generated the test statistic (in our case, $T = 4$).

The classical testing framework takes the following steps:

1. Specify a null hypothesis: that there is no difference between the two coin-flipping sessions
2. Make an assumption about the distribution of individual events: $P(heads) = P(tails) = \frac{1}{2}$ and $P(\text{k heads from n flips}) \sim Binomial(k, n)$
3. Compute the *test statistic* $T = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{(\frac{\sigma_1^2}{n_1} + \frac{s_2^2}{n_2})}}$
4. Test the hypothesis by considering how likely it is to have generated the test statistic (in our case, $T = 4$).

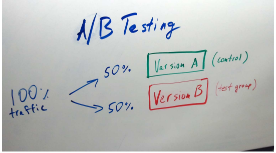

AND NOW BACK TO
OUR REGULARLY
SCHEDULED
PROGRAMMING

We talked about what we can test, so how do we go about doing it?

- Let's collect data about the original (the control) and the feature (the variable) side-by-side

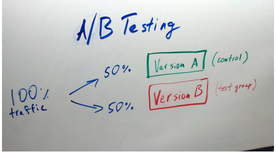
We talked about what we can test, so how do we go about doing it?

- Let's collect data about the original (the control) and the feature (the variable) side-by-side
- This is how conventional "A/B" testing is done



We talked about what we can test, so how do we go about doing it?

- Let's collect data about the original (the control) and the feature (the variable) side-by-side
- This is how conventional "A/B" testing is done



- When rolling out a major rewrite of an application, this is typically used to measure improvement. We can start at a 90%/10% split and adjust from there according to our performance metric

Examples of Useful Performance Measures:

Site Redesign Avg. Time Spent, Degree of Site Exploration (percentage of links clicked)

Mailing List Format Clickthrough Rate, Forwarding Rate

Product Description $ Revenue

In each case, we run some form of a t-test test to determine whether the feature had a statistically-significant impact on the measure we are trying to optimize

We will need a bit of math to demonstrate the limitations of conventional A/B testing:

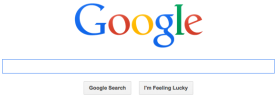
Theorem (Central Limit Theorem)

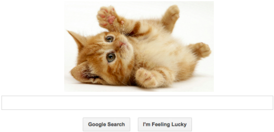
If X_i are normal independently distributed variables with mean μ and variance σ^2 and $S_n = \frac{1}{n}\sum_{i=1}^{n} X_i$, then S_n has mean μ and variance $n\sigma^2$

Thus, if we make these (strong) assumptions about the distribution of the measurements of feature A and feature B, then we can see that to double our confidence in the conclusion, we would need to quadruple the number of samples.

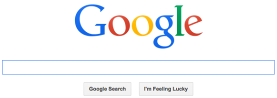
What if we wanted to test many?

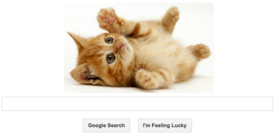
What if we wanted to test many?

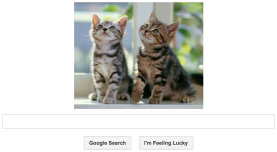




What if we wanted to test many?







What if we wanted to test many?

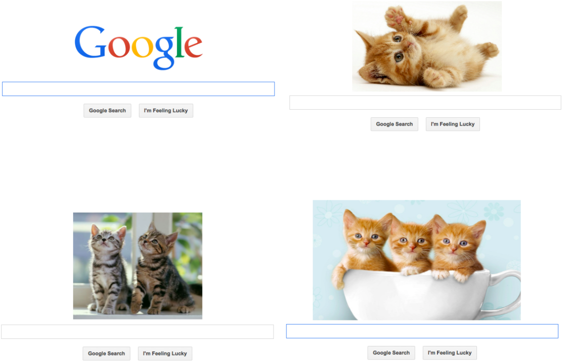
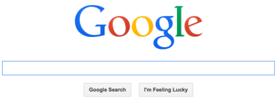
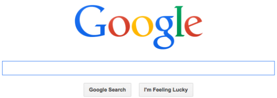


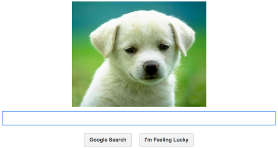
Figure: Which one will the users love the most?

Or What if We Got Other Ideas?

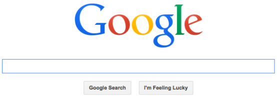


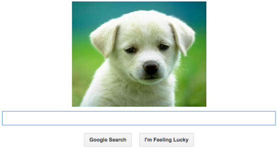
Or What if We Got Other Ideas?

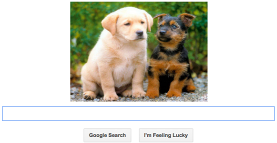




Or What if We Got Other Ideas?







Or What if We Got Other Ideas?

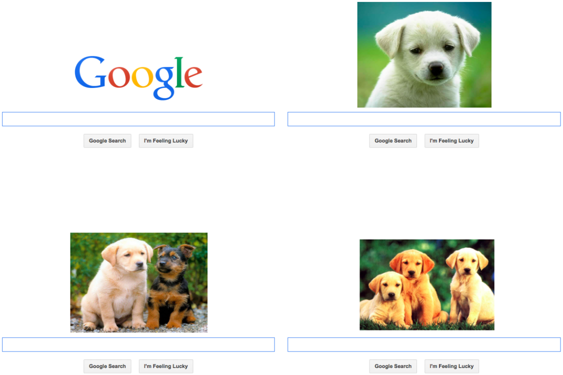


Figure: So many choices, so little time!

Outline of Process

We will label the versions by the number of cats they have: $C_0, C_1, C_2, C_3, \ldots C_n$

1. Run A/B test for C_0 vs C_1, the winner becomes the new control, $C_{control}$

Note that at each turn we are wasting valuable time and potentially losing users due to an inferior number of cats. Alternatively, we could run them all side-by-side, but it would take just as many points to attain *statistical significance* as specified by our test statistics and the CLT.

Outline of Process

We will label the versions by the number of cats they have: $C_0, C_1, C_2, C_3, \ldots C_n$

1. Run A/B test for C_0 vs C_1, the winner becomes the new control, $C_{control}$
2. Run A/B test for $C_{control}$ vs C_2, the winner becomes new control $C_{control}$

Note that at each turn we are wasting valuable time and potentially losing users due to an inferior number of cats. Alternatively, we could run them all side-by-side, but it would take just as many points to attain *statistical significance* as specified by our test statistics and the CLT.

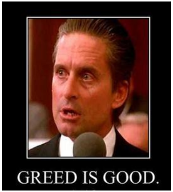
Outline of Process

We will label the versions by the number of cats they have: $C_0, C_1, C_2, C_3, \ldots C_n$

1. Run A/B test for C_0 vs C_1, the winner becomes the new control, $C_{control}$
2. Run A/B test for $C_{control}$ vs C_2, the winner becomes new control $C_{control}$
3. ad infinatum

Note that at each turn we are wasting valuable time and potentially losing users due to an inferior number of cats. Alternatively, we could run them all side-by-side, but it would take just as many points to attain *statistical significance* as specified by our test statistics and the CLT.

Greedy Algorithms



. . . well, it's definitely *better*

Definition (Wikipedia to the rescue)

A greedy algorithm is an algorithm that follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum. In many problems, a greedy strategy does not in general produce an optimal solution

ϵ-greedy

We are now in the world of bandit strategies! The idea is that we run the $C_0/C_1/C_2/\ldots$ versions side-by-side in the following way:

1. Initialize all variants to be equally likely to be selected for display for a user (i.e. if we have 10 of them, each one gets chosen with $p = 0.1$)
2. Keep track of the running performance of all the variants
3. Choose the best one with probability $1 - \epsilon$
4. Choose equally from all the variants with probability ϵ
5. Break ties by choosing randomly, or one with the lowest ID, etc.
6. Rinse and repeat until a clear winner emerges.

Exploration vs Exploitation

Every time we are *greedy*, we are displaying the winner, i.e. **exploiting** what we have learned about the environment we are in.

Exploration vs Exploitation

The rest of the time, we choose to pick uniformly from the available bandits, i.e. **exploring** the environment to gain more familiarity (statistical confidence)

Exploration vs Exploitation

Every time we are *greedy*, we are displaying the winner, i.e. **exploiting** what we have learned about the environment we are in. The rest of the time, we choose to pick uniformly from the available bandits, i.e. **exploring** the environment to gain more familiarity (statistical confidence) This is a technique more generally used in robotics, where a robot senses (explores) its environment and then acts (exploits) based on the obtained knowledge. This technique is the basis for learning unfamiliar environments (think Google driverless cars), as well as path planning (what is the optimal route between two points in this paritially-unfamiliar environment?).

Within the ϵ-greedy framework, we could improve the performance if we know something about the data:

- ϵ-first Start by exploring only, then switch to exploiting; if there are N visitors to the site, begin with ϵN exploration steps followed by $(1-\epsilon)N$ exploitation steps

Within the ϵ-greedy framework, we could improve the performance if we know something about the data:

- ϵ-first Start by exploring only, then switch to exploiting; if there are N visitors to the site, begin with ϵN exploration steps followed by $(1-\epsilon)N$ exploitation steps
- ϵ-decreasing Start with a high ϵ, and gradually reduce it to 0 over time.

Within the ϵ-greedy framework, we could improve the performance if we know something about the data:

- ϵ-first Start by exploring only, then switch to exploiting; if there are N visitors to the site, begin with ϵN exploration steps followed by $(1-\epsilon)N$ exploitation steps
- ϵ-decreasing Start with a high ϵ, and gradually reduce it to 0 over time.
- ϵ-adaptive: Adapt ϵ base on how much the performance values are fluctuating (variance)

Within the ϵ-greedy framework, we could improve the performance if we know something about the data:

- ϵ-first Start by exploring only, then switch to exploiting; if there are N visitors to the site, begin with ϵN exploration steps followed by $(1-\epsilon)N$ exploitation steps
- ϵ-decreasing Start with a high ϵ, and gradually reduce it to 0 over time.
- ϵ-adaptive: Adapt ϵ base on how much the performance values are fluctuating (variance)
- More, a field of active research

Can We Do Better? Lerning to Live with Regret

Definition (Regret)

The regret incurred by a specific strategy is the difference between the optimal strategy (only known in hindsight) and that strategy. More formally, if we have N rounds (N users hitting the site)

$$\rho = N\mu^* - \sum_{i=1}^{N} \hat{r}_i$$

with μ^* representing the (unknown) optimal strategy and $\hat{r}_i$ is the reward from our chosen strategy at a given time i.

It's clear that A/B testing does not minimize regret since we wind up alienating a lot of users during our discovery process. It's also clear that ϵ-greedy strategies are better, but by how much?

How I Stopped Worrying and Learned to Love the Beta Distribution

Definition (Beta Distribution)

This is a two-parameter probability distribution over the interval $[0, 1]$, and because of this, it is used to model "the probability of probabilities". The two parameters are labeled α and β and skew the density function either to the right (higher α) or to the left (higher β). The density function is written as:

$$Beta_{\alpha,\beta}(x) \sim x^{\alpha-1}(1-x)^{\beta-1}$$

How I Stopped Worrying and Learned to Love the Beta Distribution

Definition (Beta Distribution)

This is a two-parameter probability distribution over the interval $[0, 1]$, and because of this, it is used to model "the probability of probabilities". The two parameters are labeled α and β and skew the density function either to the right (higher α) or to the left (higher β). The density function is written as:

$$Beta_{\alpha,\beta}(x) \sim x^{\alpha-1}(1-x)^{\beta-1}$$

Pictorially

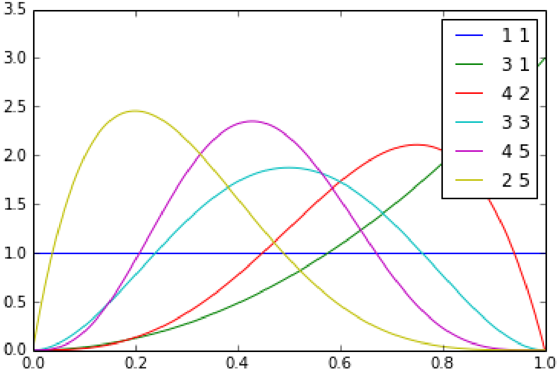


Figure: Examples of Different α and β values

Why do we need this mess?

Bandits Algorithm - Thompson Sampling

- Initialize each bandit with its very own Beta distribution with parameters $\alpha = 1$ and $\beta = 1$ (the uniform distribution).

Bandits Algorithm - Thompson Sampling

- Initialize each bandit with its very own Beta distribution with parameters $\alpha = 1$ and $\beta = 1$ (the uniform distribution).
- In the context of testing performance, α represents number of conversions and β represents number of non-conversions

Bandits Algorithm - Thompson Sampling

- Initialize each bandit with its very own Beta distribution with parameters $\alpha = 1$ and $\beta = 1$ (the uniform distribution).
- In the context of testing performance, α represents number of conversions and β represents number of non-conversions
- In each round:

Bandits Algorithm - Thompson Sampling

- Initialize each bandit with its very own Beta distribution with parameters $\alpha = 1$ and $\beta = 1$ (the uniform distribution).
- In the context of testing performance, α represents number of conversions and β represents number of non-conversions
- In each round:
  - Sample from each of the Beta distributions and pick the bandit which produced the highest value

Bandits Algorithm - Thompson Sampling

- Initialize each bandit with its very own Beta distribution with parameters $\alpha = 1$ and $\beta = 1$ (the uniform distribution).
- In the context of testing performance, α represents number of conversions and β represents number of non-conversions
- In each round:
  - Sample from each of the Beta distributions and pick the bandit which produced the highest value
  - Serve the corresponding site

Bandits Algorithm - Thompson Sampling

- Initialize each bandit with its very own Beta distribution with parameters $\alpha = 1$ and $\beta = 1$ (the uniform distribution).
- In the context of testing performance, α represents number of conversions and β represents number of non-conversions
- In each round:
  - Sample from each of the Beta distributions and pick the bandit which produced the highest value
  - Serve the corresponding site
  - If the bandit succeeds at conversion, update α by 1, otherwise, update β by 1

Bandits Algorithm - Thompson Sampling

- Initialize each bandit with its very own Beta distribution with parameters $\alpha = 1$ and $\beta = 1$ (the uniform distribution).
- In the context of testing performance, α represents number of conversions and β represents number of non-conversions
- In each round:
    - Sample from each of the Beta distributions and pick the bandit which produced the highest value
    - Serve the corresponding site
    - If the bandit succeeds at conversion, update α by 1, otherwise, update β by 1
- ???

Bandits Algorithm - Thompson Sampling

- Initialize each bandit with its very own Beta distribution with parameters $\alpha = 1$ and $\beta = 1$ (the uniform distribution).
- In the context of testing performance, α represents number of conversions and β represents number of non-conversions
- In each round:
  - Sample from each of the Beta distributions and pick the bandit which produced the highest value
  - Serve the corresponding site
  - If the bandit succeeds at conversion, update α by 1, otherwise, update β by 1
- ???
- Profit - e.g. mathematically proven to minimize regret!!!

Further Reading - Links to Resources (Standing on the Shoulders of Giants

- Statistical Analysis and A/B Testing
- Reinforcement Learning book
- Selecting Statistical Tests
- Background on Statistical Tests
- Meetup Event
- Meetup Event Slides
- https://github.com/pavelgrib/bandits