Alexander Booth
Found Visualize
October 26, 2017

Multi-Armed Bandit Testing

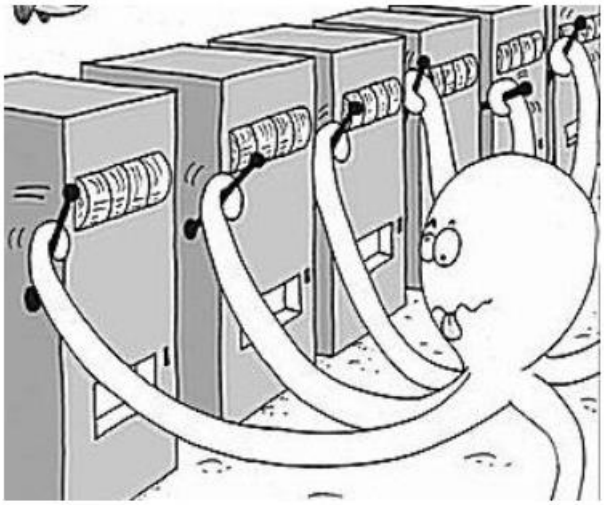
Purpose:
As we expand the role of testing, currently A/B testing, in the development life cycle of features on the website, explorations of other testing techniques could provide insight into more suitable methods.

While A/B Testing is currently the default method for optimizing our online conversions, there are frameworks other than A/B testing for selecting between competing options. One of the main alternative approaches is known as the Multi-Armed Bandits problem (MAB). Solutions to bandit problems have several nice features that make them a good choice for our optimization problems.

The Multi-Armed Bandit Problem:
Imagine the following scenario:

You are in a casino. There are many different slot machines (known as 'one-armed bandits', as they're known for robbing you), each with a lever (an arm, if you will). You think that some slot machines payout more frequently than others do, so you would like to maximize this. You only have a limited amount of resources – if you pull one arm, then you're not pulling another arm. Of course, the goal is to walk out of the casino with the most money. Question is, how do you learn which slot machine is the best and get the most money in the shortest amount of time?

Each time you play a poor performing machine, you lower your take that you walk out of the Casino with that night. In order to maximize how much money you walk out of the casino with, you will have to be efficient with how you collect your data.



This guy gets it^

What the analogy of a multi-armed slot machine captures well is the fact that it **costs to test your hypotheses**. A slot machine takes a coin to play. A website costs you a visit. Testing gains you knowledge, but will sometimes necessitate supplying your customers with an inferior product. In response, bandit algorithms are automated to move away from low scoring options and inherently prefer those that perform better.

Exploration/Exploitation Dilemma:
The optimal strategy to the above scenario is to start with a period of **exploration**, where you pull levers at random and gather information. When you have more information about what works and what doesn't, you shift to spending the majority of your time pulling the best lever (**exploitation**), but you keep exploring the other options in case your current best option is not actually the very best that exists.

This tradeoff between using currently available information and collecting more data is often referred to as the explore/exploit dilemma. You need to both explore in order to figure out what works and what does not, but to exploit, you have to take advantage of what you have learned. The MAB way of looking at this problem highlights that collecting data has a real cost, in terms of opportunities lost.

In statistical terms then, A/B testing consists of a short period of pure exploration, where you are randomly assigning equal numbers of users to Version A and Version B. It then jumps into a long period of pure exploitation, where 100% of your users are sent to the more successful version of the site. These phases are discrete; you do one, and then the other. You find out which machine is the best, and then you pump the handle on that one -- "forever".

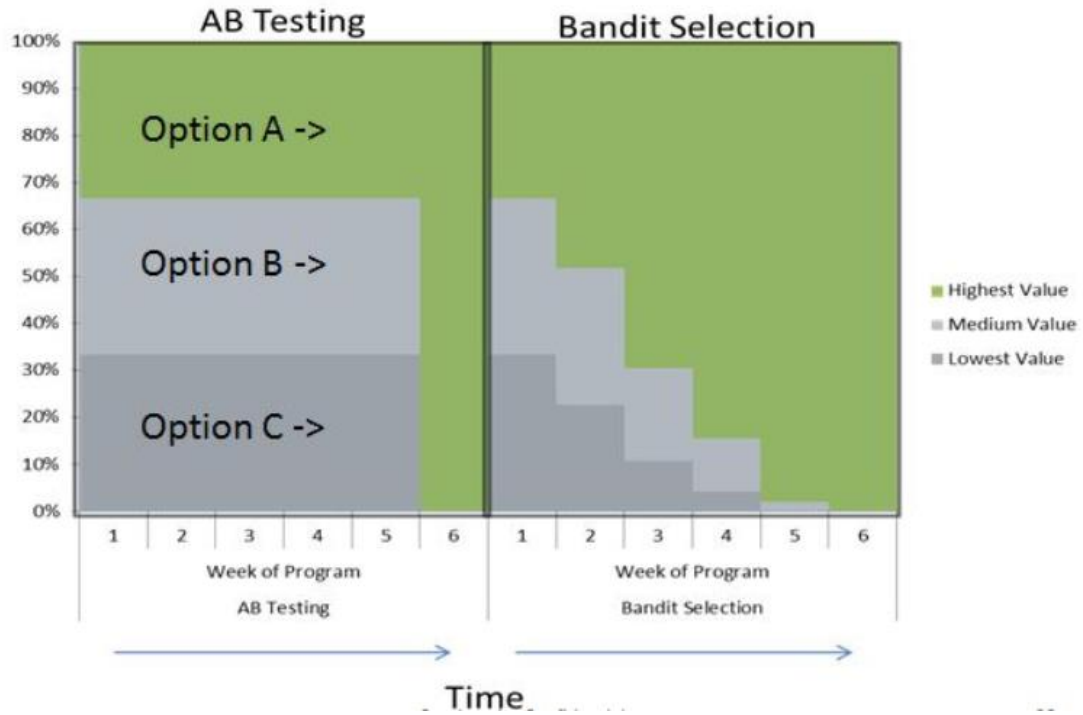
Bandit testing tries to solve the explore-exploit problem in a different way. Instead of two distinct periods of pure exploration and pure exploitation, bandit tests are adaptive, and simultaneously include exploration and exploitation, as mentioned above.

There is a twist though the exploration phase should never stop. Even if deep down in your heart of hearts, you are positively certain that you have found the best possible option, you still never stop experimenting because the information gathered by experimenting is still valuable in helping to avoid the Change Over Time Dilemma.

Change Over Time Dilemma:
Returning to the slot machine example, suppose some of the machines' payouts change over time? If the payout changes at some point in time then the A/B test is a little less reliable because its discrete exploration and exploitation phases stop you from learning and earning at the same time, and only allows you to select one answer. Forever is a long time. Not expecting changes in success to occur is naïve. Since bandit tests have the option to never stop exploring, they are also more sensitive to changes over time, where a solution may suddenly not be as successful as once thought.

Regret:
Regret is crucial in understanding the drawbacks of discrete exploration/exploitation phases. Regret is the difference between the actual payoff and the payoff you would have collected had you played the optimal (best) options at every opportunity. Essentially, it's the opportunity cost.

In AB Testing, we see that each time we make a selection, each of the options gets the same chance to be selected. In the MAB approach, this is not the case. The probabilities for being selected at each turn depend on the current best guess off how valuable each option is. In the below case, Option A has the highest current estimated value, and so is getting selected more often than either options B or C, and likewise, option B has a higher estimated value than option C. These probabilities are fluid, and can change over time as the estimates are reinforced. As shown below, with each new week, the bandit reduces how often it selects the lower performing options and increases how often if selects the highest performing option.



With an A/B/C Testing approach we try out each of the three options with equal proportions until the test ends at week 5, and then select the option with the highest value. In this case, we have effectively selected the best option 44% of the time over the 6 week period (33% of the time through week 5, and 100% at week 6), and the medium and lowest performing options 28% of time each (33% of the time through week 5, and 0% at week 6).

If we the weight these selection proportions by the actual value of each option, we get an expect yield of about $1.31 per user over the 6 weeks (44%*$2 + %28*$1 + %28*$0.50=$1.31).

With a bandit approach, the bandit attempts to use what it knows about each option from the very beginning, so it continuously updates the probabilities that it will select each option throughout the optimization project.

Over the same 6 week period the bandit approach has selected the best option 72% of the time, the medium option 16% of the time, and the lowest performer on 12% of the time. If we weight by the

average value of each option we get an expected yield of about $1.66 per user (this proof is left to the reader). The bandit would have returned then a 27% lift over the A/B/C Testing approach ($1.66-$1.31)/$1.31) during the learning time of the project.

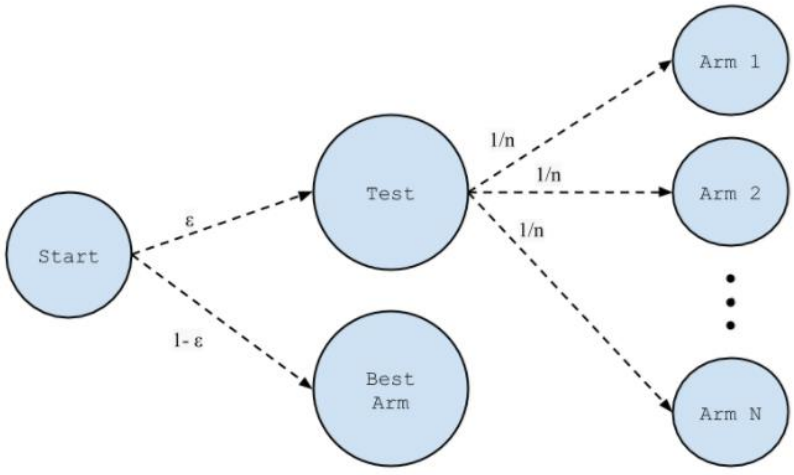
Regret is the regret you feel for finding out that you were wasting time on the wrong version, and could have made more by doing the right thing all along.

Intermission 1:
An A/B test is a simple type of MAB test, with two arms, a 50-50 exploration phase followed by a 100-0 exploitation phase. The purpose of testing, MAB or otherwise, is to minimize regret caused by the explore/exploit dilemma. Since success metrics can change over time, belief in the results of an A/B test "forever" is naïve.

Types of Bandits:

Epsilon-first:
A fixed period is spent exploring the arms, and from then on you pull the one you think is best. A/B testing is one way to do epsilon-first. This algorithm has *linear* regret on average. Since you're trying bad versions a fixed fraction of the time, the regret is linearly proportional to how many trials you have had. Further, there is a fixed probability that you'll make the wrong choice and from then on you will forever do the wrong thing. In practice the linear term is very small and so the regret tends to be acceptable in practice.

Epsilon-greedy:
Perhaps the simplest and widely used bandit is epsilon-greedy. This is where we always keep track of the number of pulls of the lever and the amount of rewards we have received from that lever. 10% of the time, we choose a lever at random. The other 90% of the time, we choose the lever that has the highest expectation of rewards. Those values can obviously by changed.



This algorithm also has linear regret. Again, since we're trying bad versions a fixed fraction of the time, the regret is linearly proportional to how many trials we have had. This method also suffers from lack of statistical significance. Since the bad arms get increasingly less interactions, it takes more time to drop them from the test with confidence. However, it solves the change over time dilemma, and allows the insertion of new arms fairly easily.

Upper Confidence Bound UCB:
Fun fact: This is the bandit that the Washington Post uses

This algorithm can be summed up by the principle of optimism in the face of uncertainty. That is, despite our lack of knowledge in what actions are best we will construct an optimistic guess as to how good the expected payoff of each action is, and pick the action with the highest guess. If our guess is wrong, then our optimistic guess will quickly decrease and we'll be compelled to switch to a different action. But if we pick well, we'll be able to exploit that action and incur little regret. In this way we balance exploration and exploitation.

There are two ways an action could be the best:

- You have measured its mean payout to be higher than all the others, and you are certain of this with a high degree of statistical significance.
- You have not measured its mean payout very well, and the confidence interval is large.

In simplified terms, the version that you pick for the next pull is the one that makes avg(version) + sqrt(2*ln(trials_of_all_versions) / trials(version)) as large as possible. This technique has been proven to have logarithmic regret, meaning that the fraction of the time that you spend doing the wrong thing scales as the log of the number of times you're faced with the web page (or are faced with new users).

The key point here is that ln(trials_of_all_versions) grows very slowly relative to trials(worst_version). So while the exploration process continues forever, the fraction of time we spend exploring decreases exponentially.

UCB is acceptable, but not the best algorithm available. Simulations have been created showing that Thompson sampling has superior regret performance relative to UCB. Plus the Bayesian Bandit for sure has statistical significance.

The Bayesian Bandit (Thompson's Sampling for Bernoulli Bandits)
Fun Fact: This is Google Analytics' Bandit, as well as Microsoft's
Intuitively: It is a sequential model-based approach to solving problems, where we prescribe a prior belief over the possible objective functions and then sequentially refine this model as data are observed via Bayesian posterior updating.
"What is the probability that this is the best arm, given what I know now?"

Thompson sampling tends to shorten experiments while simultaneously making them less expensive to run for longer durations.

We have N possible layouts for a landing page, each of which has a click through rate θi. Unfortunately we do not know what θi is. We are following a Bayesian approach, so we will construct a probability distribution which represents our belief about what the actual value of θi is.

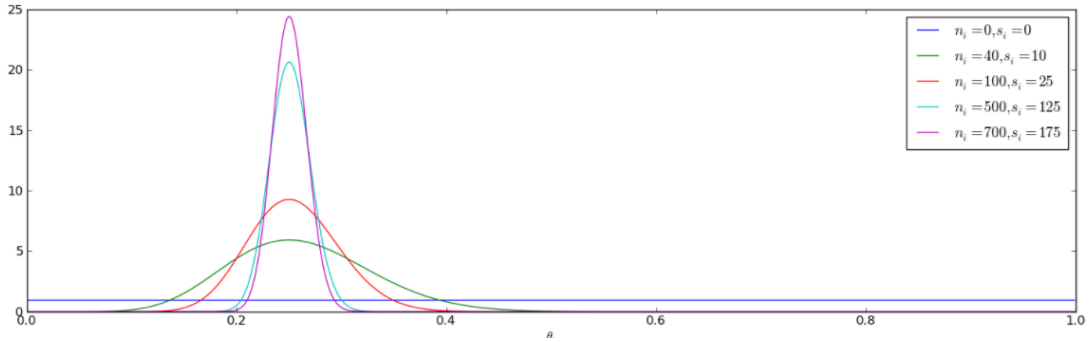
The basic idea behind Bayesian methods is to update our beliefs based on evidence. As we gather more data by showing different titles to other users and observing click throughs, we can incrementally narrow the width of the probability distribution.

As in all Bayesian inference, we need to choose a prior. The prior is something we believe to be true before we have any evidence - i.e., before we have shown the title to any visitors. This is just a starting point - after enough evidence is gathered, our prior will play a very minimal role in what we actually believe. Choosing a good prior is important both for mathematical simplicity, and because if your prior is accurate, you don't need as much evidence to get the correct answer.

The uniform Beta Prior is the beta distribution where alpha=beta=1 (Beta(1,1)) and is a good choice. Bayes himself used it! Other options are Beta(0,0), Beta(.5,.5) or whatever you want.

If the prior is $f_{\alpha i,\beta i}(\theta i)$, then the posterior distribution is posterior=$f_{\alpha i+si,\beta i+ni-si}(\theta i)$

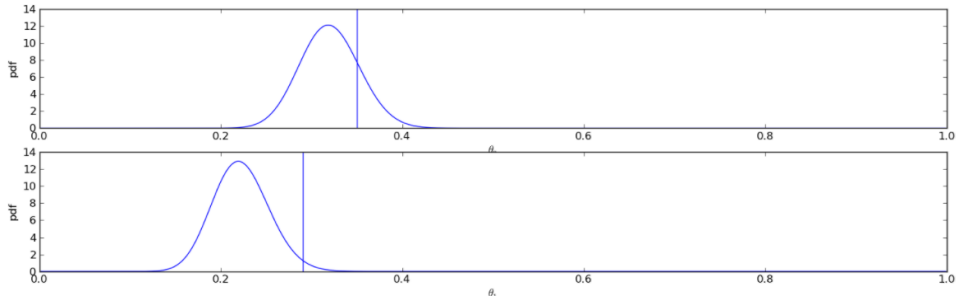
The key idea here is that to update our probability distribution describing θi, we need only update the parameters of our beta distribution. So what does this mean in practice? As we run more experiments, our probability distribution on where θi lives becomes sharper:



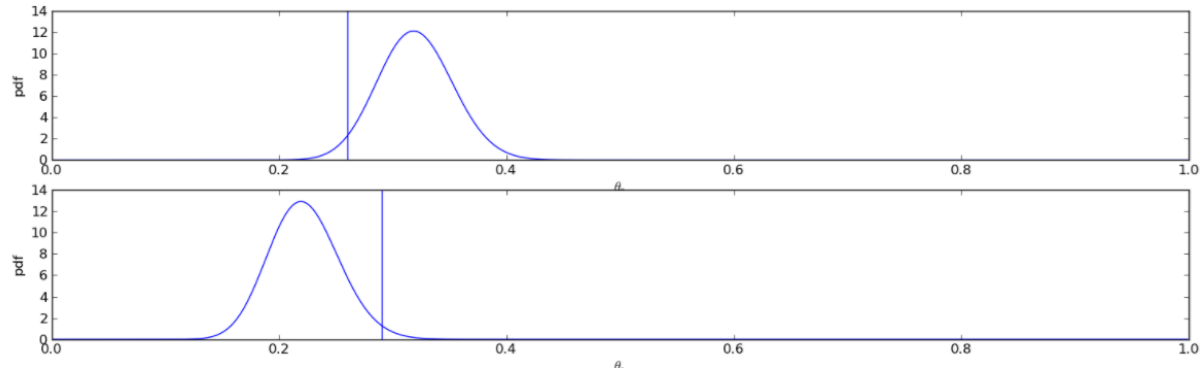
Before we run any experiments, θi could be anything (as represented by the blue line). Once we have run 700 experiments, yielding 175 click throughs, we are reasonably confident that θi lives roughly between 0.2 and 0.3.

What we've done so far is figured out how to estimate what our click through rates actually are based on empirical evidence. But that doesn't actually give us a method of optimizing them yet. To optimize clicks, we should use a Monte Carlo sampling method.

The ultimate goal of the bandit algorithm is to display to the user whichever landing page layout has the highest CTR. One method of estimating the CTRs of the landing page is to sample the posterior distribution. I.e., suppose we have two possible pages, from which we have drawn n0=200, s0=64 and n1=180, s1=40. Then one possible set of samples we might observe is this:



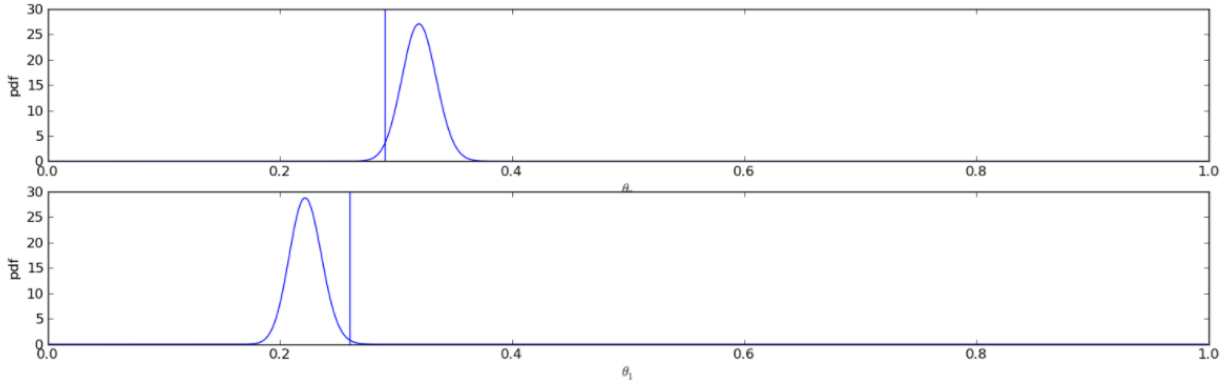
For page 0, our sample of θ0 has worked out to be 0.35, while our sample of θ1 is only 0.28. Since θ0=0.35>θ1=0. 28, we will display page 0 to the user. However, there was no guarantee that things worked out this way. It was possible, although less likely, that θ1 could come out larger than θ0:



In this case, we would have displayed page 1 to the user rather than page 0.

The net result is that for overlapping probability distributions, we will display the title with the larger expected CTR the majority of the time. But occasionally, we will draw from the other distributions simply because it is within the realm of possibility that they are greater.

As we gather more data our probability distributions will become narrower and a clear winner will become apparent. When this occurs, we will almost surely choose the winner:



Intermission 2:

> ***Epsilon-Greedy***: *the rates of exploration and exploitation are fixed*
>
> ***Upper Confidence Bound***: *the rates of exploration and exploitation are dynamically updated with respect to the UCB of each arm*
>
> ***Thompson sampling***: *the rates of exploration and exploitation are dynamically updated with respect to the entire probability distribution of each arm*

FAQ from a skeptical reader:

Are the results from the Bayesian bandit statistically valid?
Yes. The bandit uses sequential Bayesian updating to learn from each day's experimental results, which is a different notion of statistical validity than the one used by classical testing. A classical test starts by assuming a null hypothesis. For example, "The variations are all equally effective." It then accumulates evidence about the hypothesis, and makes a judgement about whether it can be rejected. If you can reject the null hypothesis you've found a statistically significant result.

Statistical significance exists to keep you from making a type I error. In the context of website optimization, a type I error means picking a new variation that is really no different from the original, performance wise. You'd like to avoid type I errors (they're errors, after all), but in this context they are far less costly than type II errors. For us, a type II error means failing to switch to a better arm, which is costly because it means you're losing conversions.

Bayesian updating asks the question "What is the probability that this is the best arm, given what I know now?" Hypothesis testing asks "What is the probability that I would see this outcome if all the arms were equal?" Both are valid questions, but the Bayesian question is easier for most people to understand, and it naturally balances between type I and type II errors by taking advantage of information from your experiment as it becomes available.

Classical hypothesis tests make you wait until you've seen a certain number of observations before you look at your data, because the probability question they need to answer becomes too complicated otherwise. If you've got a poorly performing arm in your experiment, then classical tests impose a heavy opportunity cost. So if both methods are valid, why not use the one that saves you time and money, and skip the complicated, expensive one that makes you wait to look at your experimental results?

Is it always shorter than a classical test?
The bandit can yield results much faster than classical testing, at less cost, and with just as much statistical validity, but there may be occasional experiments that take longer than expected just by chance.

Prove it.

Comparison against A/B tests:

Motivation: Example from web optimization

We want to make (good) decisions under uncertainty

Setting

- We want to redesign a website
- We have multiple proposals
- The website has space for 2 large pictures and we have 51 of them.
- There are 51*50 = 2550 possible ways

What should we do?

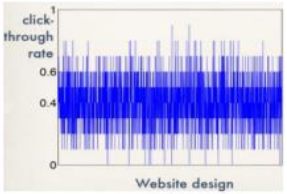
- The industry standard: A/B Test

Motivation: Example from web optimization

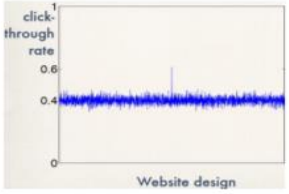
We want to make (good) decisions under uncertainty

Problems: Comparing many users requires time

- 2.55k designs
- CTR=click-through rate
- Best test has CTR=0.6
- Rest have CTR=0.4
- 10 users per design
- 25k users overall
- 2.5 days (10k visits / day)



- 2.55k designs
- CTR=click-through rate
- Best test has CTR=0.6
- Rest have CTR=0.4
- 500 users per design
- 1.25M users overall
- 4 months(10k visits / day)

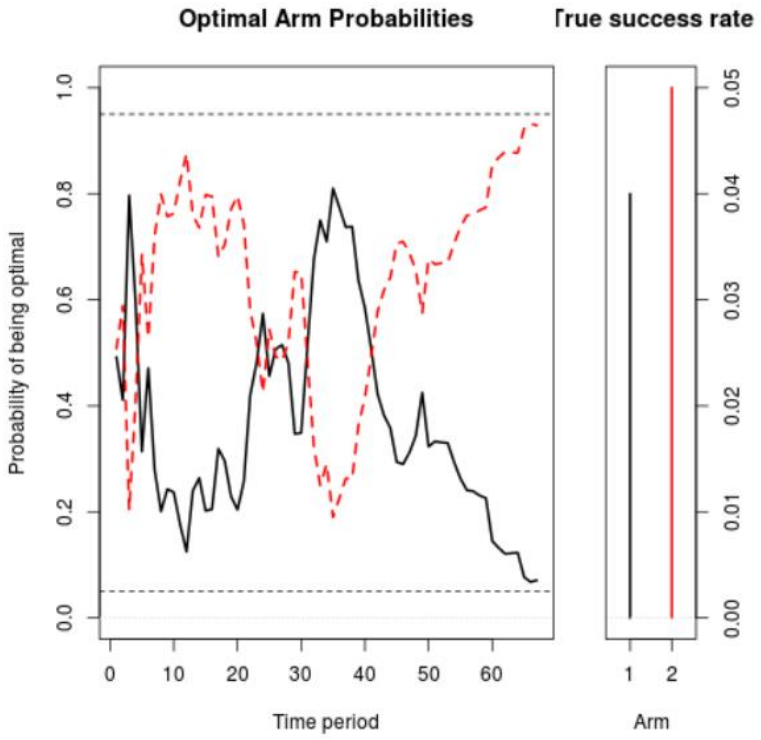


Comparison against A/B tests:

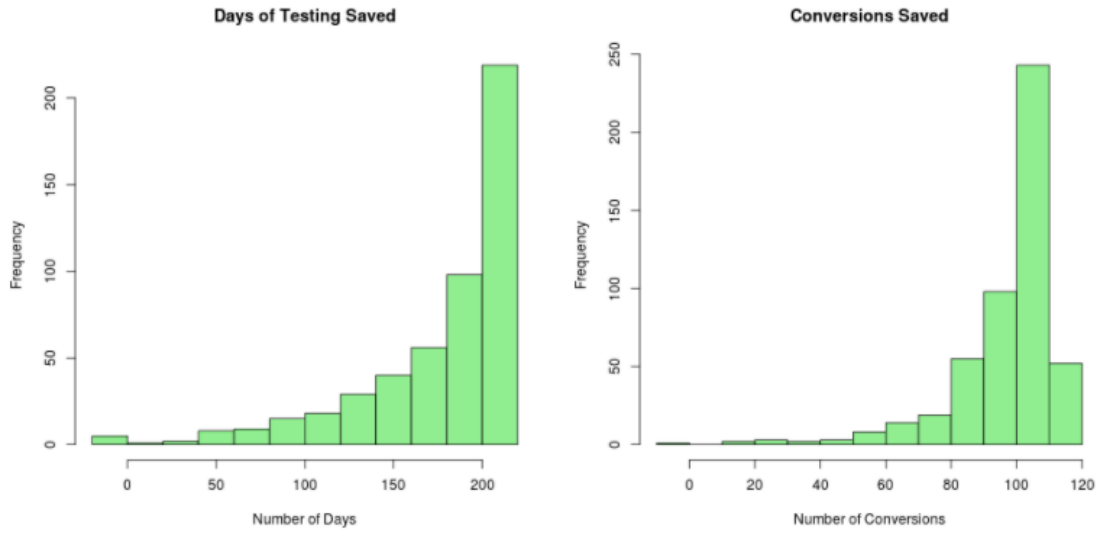
Suppose you've got a conversion rate of 4% on a handcrafted landing page. You experiment with a new version of the page that actually generates conversions 5% of the time (dynamic summarization). You don't know the true conversion rates of course, which is why you're experimenting, but let's suppose you'd like your experiment to be able to detect a 5% conversion rate as statistically significant with 95% probability. A standard power calculation tells you that you need 22,330 observations (11,165 in each arm) to have a 95% chance of detecting a .04 to .05 shift in conversion rates. Suppose you get 100 sessions per day to the experiment, so the experiment will take 223 days to complete. In a standard experiment you wait 223 days, run the hypothesis test, and get your answer.

Now let's manage the 100 sessions each day through the multi-armed bandit. On the first day about 50 sessions are assigned to each arm, and we look at the results. We use Bayes' theorem to compute the probability that the variation is better than the original. Let's suppose the original got really lucky on the first day, and it appears to have a 70% chance of being superior. Then we assign it 70% of the traffic on the second day, and the variation gets 30%. At the end of the second day we accumulate all the traffic we've seen so far (over both days), and re-compute the probability that each arm is best. That gives us the serving weights for day 3. We repeat this process until a set of stopping rules has been satisfied.

Below shows a simulation of what can happen with this setup. In it, you can see the serving weights for the original (the black line) and the variation (the red dotted line), essentially alternate back and forth until the variation eventually cross the line of 95% confidence. (The two percentages must add to 100%, so when one goes up the other goes down). The experiment finished in 66 days, so it saved 157 days of testing.



Of course this is just one example. Re-running the simulation 500 times shows how well the bandit fares in repeated sampling. The distribution of results is shown below. On average the test ended 175 days sooner than the classical test based on the power calculation. The average savings was 97.5 conversions.



Out of the 500 experiments shown above, the bandit found the correct arm in 482 of them. That's 96.4%, which is about the same expected error rate as the classical test (95%). There were a few experiments where the bandit actually took longer than the power analysis suggested, but only in about 1% of the cases (5 out of 500).

Stopping the experiment

It is recommended that the bandit run for at least two weeks. After that, you can keep track of two metrics.

The first is the probability that each variation beats the original. If we're 95% sure that a variation beats the original then you can declare that a winner has been found. Both the two-week minimum duration and the 95% confidence level can be adjusted by the user.

The second metric that we monitor is the "potential value remaining" (PVR) in the experiment. At any point in the experiment there is a "champion" arm believed to be the best. If the experiment ended "now", the champion is the arm you would choose. The "value remaining" in an experiment is the amount of increased conversion rate you could get by switching away from the champion. The whole point of experimenting is to search for this value. If you're 100% sure that the champion is the best arm, then there is no value remaining in the experiment, and thus no point in experimenting. But if you're only 70% sure that an arm is optimal, then there is a 30% chance that another arm is better, and we can use Bayes' rule to work out the distribution of how much better it is.

The PVR is related to the distribution of regret because it is obtained by summarizing an upper quantile of the regret distribution, such as the 95th percentile. PVR is the value per play that might be lost if the experiment ended at time t. Because businesses experiment in the hope of finding an arm that provides

greater value, a sensible criterion for ending the experiment is when PVR falls below a threshold of practical significance.

It is recommended to end the experiment when there is at least a 95% probability that the PVR in the experiment is less than 1% of the champion's conversion rate. That is a 1% improvement, not a one percentage point improvement. So if the best arm has a conversion rate of 4%, then we end the experiment if the value remaining in the experiment is less than .04 percentage points.

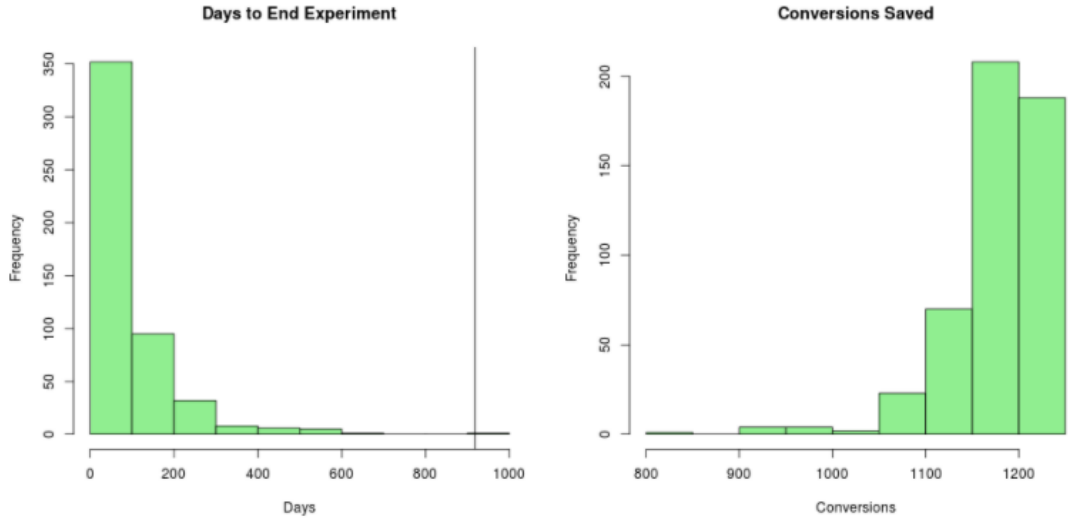
Ending an experiment based on the PVR is also nice because it handles ties well. For example, in an experiment with many arms, it can happen that two or more arms perform about the same, so it does not matter which is chosen. You wouldn't want to run the experiment until you found the optimal arm (because there are two optimal arms). You just want to run the experiment until you're sure that switching arms won't help you very much. Any arm from the set of potential winners can then be chosen going forward.

Multivariate Testing

We use the term "multivariate testing" to describe an experiment with more than one experimental factor. For instance, we have more than one idea for how to improve our landing pages, and usually have more than one variation that we'd like to test. This also spans the scenario of one test spanning multiple pages. Luckily, the multi-armed bandit's edge over classical experiments increases as the experiments get more complicated.

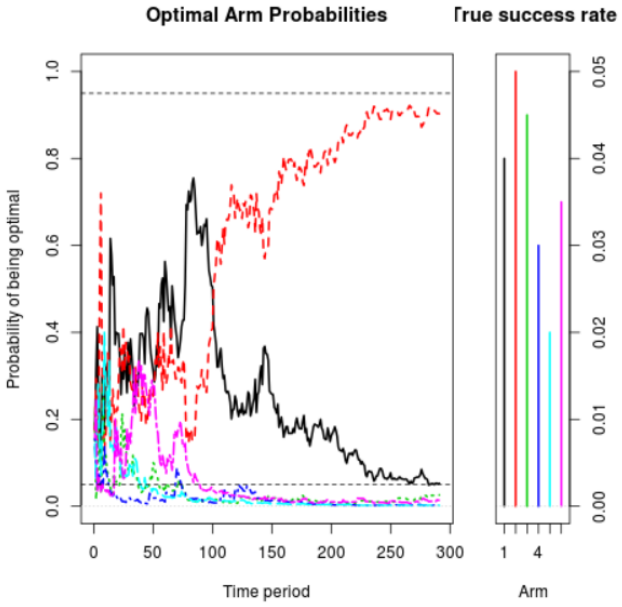
Let's assume that we have 5 variations plus the original. We're going to do a calculation where you compare the original to the largest variation, so we need to do some sort of adjustment to account for multiple comparisons. The Bonferroni correction is an easy adjustment that we currently implement in scenarios like this. It can be implemented by dividing the significance level of the hypothesis test by the number of arms. Thus we do the standard power calculation with a significance level of .05 / (6 - 1), and find that we need 15,307 observations in each arm of the experiment. With 6 arms that's a total of 91,842 observations. At 100 sessions per day the experiment would have to run for 919 days (over two and a half years). In real life it usually wouldn't make sense to run an experiment for that long, but we can still do the thought experiment as a simulation.

Now let's run the 6-arm experiment through the bandit simulator. Again, we will assume an original arm with a 4% conversion rate, and an optimal arm with a 5% conversion rate. The other 4 arms include one suboptimal arm that beats the original with conversion rate of 4.5%, and three inferior arms with rates of 3%, 2%, and 3.5%. The distribution of results, relative to a Bonferroni adjusted power calculation for a classical experiment, is shown below. The left panel shows the number of days required to end the experiment, with the vertical line showing the time required by the classical power calculation. The right panel shows the number of conversions that were saved by the bandit.



The average experiment duration is 88 days (vs. 919 days for the classical experiment), and the average number of saved conversions is 1,173. There is a long tail to the distribution of experiment durations, but even in the worst cases, running the experiment as a bandit saved over 800 conversions relative to the classical experiment. The cost savings are partially attributable to ending the experiment more quickly, and partly attributable to the experiment being less wasteful while it is running.

The next graphic shows the history of the serving weights for all the arms in the first of our 500 simulation runs. There is some early confusion as the bandit sorts out which arms perform well and which do not, but the very poorly performing arms are heavily down weighted very quickly. In this case, the original arm has a "lucky run" to begin the experiment, so it survives longer than some other competing arms. But after about 50 days, things have settled down into a two-horse race between the original and the ultimate winner.



Once the other arms are effectively eliminated, the original and the ultimate winner split the 100 observations per day between them. Notice how the bandit is allocating observations efficiently from an economic standpoint, as they are flowing to the arms most likely to give a good return, as well as from a statistical standpoint, since they are flowing to the arms that we most want to learn about.

Conclusion

In general Bandit Based optimization can produce far superior results to regular A/B testing. Bandit algorithms are a great way of optimizing many factors of your website. There are many good options, including UCB. However, Bayesian Bandit is the best. It's simple to implement, straightforward to use, and very importantly it's also straightforward to extend. It's also important to note that the theoretical properties of the Bayesian Bandit, including logarithmic regret have been proven.

Multi-armed bandit experiments can be a substantially more efficient optimization method than traditional statistical experiments. The Thompson sampling (Bayesian Bandit) heuristic for implementing multi-armed bandit experiments is simple enough to allow flexible reward distributions that can handle the kinds of issues that arise in real applications.

Business and science have different needs. Traditional experiments were created to address uncertainty in scientific problems, which is a naturally conservative enterprise. Business decisions tend to be more tactical than scientific ones, and there is a greater cost to inaction. The two business sectors that dominated the 20th century, agriculture and manufacturing, happen to have economic structures that align with scientific conservatism, which partly explains why they succeeded so well using the tools of science. In those sectors a type I error is costly because it means a potentially expensive change to a production environment with no accompanying benefit.

In the service economy type I errors are nearly costless, so artificially elevating them over expensive type II errors makes very little sense. When paired with the fact that the proportion of type I errors is bounded by the proportion of true null hypotheses, the significance vs. power framework underlying traditional statistical experiments seems a poor fit to the modern service economy. By explicitly optimizing value, multi-armed bandits match the economics of the service industry much more closely than traditional experiments, and should be viewed as the preferred experimental framework. This is not to say that there is no room for traditional experiments, but their use should be limited to high level strategic decisions where type I errors are truly important.

Appendix

Sources:

Examples in Use

Washington Post's UCB Bandit:

https://developer.washingtonpost.com/pb/blog/post/2016/02/08/bandito-a-multi-armed-bandit-tool-for-content-testing/

Google's Bayesian Bandit

https://support.google.com/analytics/answer/2844870?hl=en#2
https://support.google.com/analytics/answer/2846882
https://support.google.com/analytics/answer/2847021?hl=en&ref_topic=1745207

Scholarly Articles

Algorithms for the multi-armed bandit problem
http://www.cs.mcgill.ca/~vkules/bandits.pdf

Analysis of Thompson Sampling for the Multi-armed Bandit Problem
http://proceedings.mlr.press/v23/agrawal12/agrawal12.pdf

Multi-armed bandit experiments in the online service economy
https://www.gsb.stanford.edu/sites/gsb/files/oit_05_16_scott.pdf

Collaborative Filtering as a Multi-Armed Bandit
https://hal.inria.fr/hal-01256254/file/main.pdf

Bayesian Bandit

https://dataorigami.net/blogs/napkin-folding/79031811-multi-armed-bandits
https://www.chrisstucchio.com/blog/2013/bayesian_bandit.html
https://gist.github.com/stucchio/5383015#file-beta_bandit_test-py
http://www.machinelearning.ru/wiki/images/f/f8/Rodrigo_Rivera_Bayesian_context.pdf

UCB

https://jeremykun.com/2013/10/28/optimism-in-the-face-of-uncertainty-the-ucb1-algorithm/
https://books.google.com/books?id=xnAZLjqGybwC&pg=PA52&source=gbs_selected_pages&cad=2#v=onepage&q&f=false

Epsilon-Greedy (The easiest one)

http://blog.yhat.com/posts/the-beer-bandit.html
http://stevehanov.ca/blog/index.php?id=132
https://vwo.com/blog/multi-armed-bandit-algorithm/
https://blogs.mathworks.com/loren/2016/10/10/multi-armed-bandit-problem-and-exploration-vs-exploitation-trade-off/

Wikipedia
https://en.wikipedia.org/wiki/Multi-armed_bandit
https://en.wikipedia.org/wiki/Beta_distribution#Bayes.27_prior_probability_.28Beta.281.2C1.29.29

Bayesian Theory for A/B Testing
https://en.wikipedia.org/wiki/Beta_distribution#Bayes.27_prior_probability_.28Beta.281.2C1.29.29
https://www.chrisstucchio.com/blog/2013/bayesian_analysis_conversion_rates.html

A/B Testing vs MAB
Intro to MAB/Fair Well Rounded Arguments on Both Sides
https://conversionxl.com/blog/bandit-tests/ <-READ THIS ONE
https://conductrics.com/balancing-earning-with-learning-bandits-and-adaptive-optimization/
https://www.searchenginepeople.com/blog/16072-multi-armed-bandits-ab-testing-makes-money.html

Totally All In, Pro MAB:
http://stevehanov.ca/blog/index.php?id=132
https://endouble.com/blog/multi-armed-bandits-smart-alternative-ab-testing/
https://www.retentionscience.com/multi-armed-bandit/

Criticisms (Only mention Epsilon-Greedy):
https://vwo.com/blog/multi-armed-bandit-algorithm/ (These guys sell A/B testing software)
https://www.quora.com/What-are-the-practical-drawbacks-of-multi-armed-bandit-testing-as-applied-to-conversion-optimization/answer/Andrew-Anderson-5?srid=8XxW&share=1

A/B Testing is easier technically, MAB is easy to mess up. Use Caution if You Don't Understand. Defaulting to A/B is not a bad thing.
https://bentilly.blogspot.com/2012/09/ab-testing-vs-mab-algorithms-its.html
https://www.chrisstucchio.com/blog/2012/bandit_algorithms_vs_ab.html