UNIVERSITÄT OSNABRÜCK

Institute of Cognitive Science

Intuit Data Engineering and Analytics

Bachelor's Thesis

Comparing Models for A/B Testing

Andrea Suckro

July 28, 2015

First supervisor:	Prof. Dr. Frank Jäkel
Second supervisor:	Mita Mahadevan

Comparing Models for A/B Testing

This thesis is centered around the frequently used method of automated A/B Testing. Though it has been around for a while and many tools have been developed to do A/B Testing, the underlying statistics are widely discussed and no current standard could be identified. I will start by describing the general procedure. Then different methods will be compared on a slightly simplified model, starting with the more classical approaches and later comparing those to the Bandit Algorithms popular in Machine Learning. The discussion will show pro's and con's for any method and sketch a useful picture for future development.

Contents

Chapter 1

Introduction

As computers and analytic on big sets of data become more tangible, A/B-Testing seems to become part of any online service or product. In fact by using online products like Google or Facebook it is highly likely that everyone of us has already been taking part in such an experiment and though sometimes that raises some attention [Art14] most of these tests go by unnoticed by the customer. The term itself originates from a simplified setting where only two variants are compared but is also used to describe settings with more scenarios that are referred to as Multivariate testing in the scientific literature.

Over the years more and more products came to the market, helping companies to conduct their own A/B-Testing, all the while still using the same traditional statistical model for the analysis of the collected data. In the mean time scientific research in the field of Machine Learning and Cognitive Science strived forward, trying to discover concepts like learning and decision making. New algorithms were developed and compared with actual human behavior (see for example the article 'Cheap but Clever: Human Active Learning in a Bandit Setting' by Shunan Zang and Angela J Yu [ZA13]). Slowly those

Experimenters hope for a realistic feedback by directly measuring the users behavior without additional work by the customer (as compared to filling out a survey for example). The gained insight should be used to more reliably model users future behavior and to extract behavior patterns possibly unknown prior to the experiment. This procedure is not bound to user studies as experiments can be conducted on back-end algorithms as well. Intuit has an own framework to conduct such test scenarios called *abntest*. It is a web service that can be integrated by any online product and uses http requests to measure the characteristic numbers for a test.

In the recent years the

1.1 Terminology

Certain terms are common in the framework of A/B-Testing and will therefore also appear repeatedly throughout this thesis. The following explanation will set them into context.

Test A test is created by a user with a defined number of buckets. A test is active for a preset amount of time, collecting the data with a sampling rate to determine if there is a significant difference between the buckets. The sampling rate determines what percentage of all the users participate in the test. This decision takes place before each user is assigned to a bucket.

Bucket A Bucket is a scenario that resides within a test. Buckets vary on a certain feature that is the discriminating factor to be measured by the overall test. This can be a new design or UI-feature, but basically it is not tied to a 'visible' change, but can also be concerned with internal mechanisms. The percentage of users that get assigned to a specific bucket can be freely administrated as long as the assignments sum up to 100% .

Assignment The assignment determines the experience the user will be exposed to during the test. Once determined it stays fixed.

Action A user with an assignment may perform an action that is measured. For example - the clicking on a specific button may trigger a recording of this action. The accumulated actions determine the success of a bucket compared to the others.

1.2 Workflow of a Testing System

Users have the possibility to create test cases for their projects. By encapsulating the whole process of performing and A/B Test the users can perform tests in any area of the product. The following flow describes the different parts that are inherent to most testing platforms.

1.2.1 Creating a Test

Creating the tests and measuring the metrics along the way. This is normally handled by a separate system. The users built in code to their products to show the users different experiences based on their assignment that is determined by the A/B Testing service. When creating the test the users define the number of different experiences, the control setting and how long the test should be run.

1.2.2 Running the Test

For any new user that is visiting the product the service determines if she is part of the experiment or not based on the sampling rate. If it is decided that she takes part in the testing the next step decides the bucket she is assigned to. The user will from now on always stay in the bucket to avoid shifting experiences from visit to visit.

1.2.3 Analyzing the Result

The service provides metrics and current states of the test cases throughout the testing phase. Though it is not encouraged, customers can still update their tests and for example increase the sampling rate or disable whole buckets, if they are not useful anymore.

Chapter 2

Classic AB-Testing

2.1 Assumptions and Setup

In this section we will take a look at the common approach to A/B Testing as it is also implemented in Intuit's own framework and other available solutions. To find a common ground for comparison the problem will be abstracted and to a certain degree simplified. This leads to the following definition:

A given test consists of two buckets. One is generating actions with a probability $q1$ the other generates them with a probability q_2. The prior distribution for all q is a Beta distribution $Beta(\alpha, \beta)$. This is a convenient choice because it is the conjugate family for the binomial likelihood. We also assume no prior knowledge when creating the test case (uninformative prior $Beta(\alpha = 1, \beta = 1)$). N_1 customers get assigned to Bucket 1 and N_2 to Bucket 2. They generate

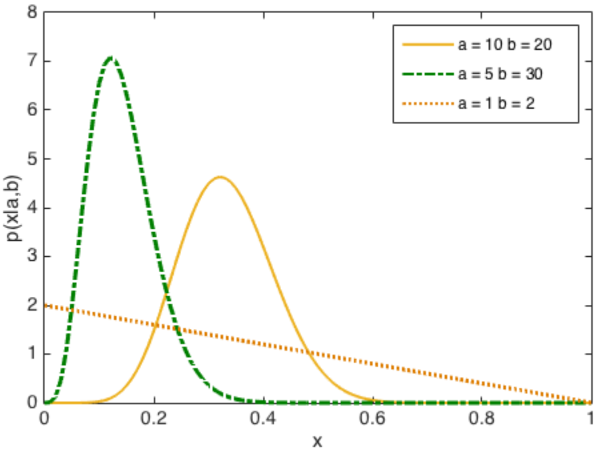


Figure 2.1: Different Beta Distributions

k_1 and k_2 actions. The posterior distributions for each individual case is:

$$\begin{aligned}
q &= f(k|N) \propto f(N|k)f(k)\\
f_n &= N_n - k_n\\
q_1 = Beta(k_1+\alpha, f_1+\beta) &= \frac{x^{k_1+\alpha-1}(1-x)^{f_1+\beta-1}}{B(k_1+\alpha, f_1+\beta)}\\
q_2 = Beta(k_2+\alpha, f_2+\beta) &= \frac{x^{k_2+\alpha-1}(1-x)^{f_2+\beta-1}}{B(k_2+\alpha, f_2+\beta)}
\end{aligned}$$

So each new assignment and each new event will directly alter the distributions parameters. It is important to notice that this model assumes events to occur directly. The user enters the experiment, produces an event (for example clicks on a specific button) or not and leaves again. A user is only measured once - meaning that the model does not account for events produced by returning users. A/B Testing tries to find an answer to the posterior for $P(q_1 > q_2|k_1, f_1, k_2, f_2)$ the probability that Bucket 1 performs better than Bucket 2 given the collected data. The following section explains three different approaches to that question.

2.2 Best Bucket

Finding the best Bucket among the others can be achieved in several ways. The following methods give an overview about typical strategies that are used in different frameworks.

2.2.1 Analytic

In general for two random variables X, Y with corresponding probability density function f_X, f_Y and cumulative density function F_X, F_Y it holds:

$$\begin{aligned}
P(X \geq Y) &= \iint_{[x>y]} f_X(x)f_Y(y)\,dy\,dx\\
&= \iint_{-\infty}^{x} f_X(x)f_Y(y)\,dy\,dx\\
&= \int_{-\infty}^{\infty} f_X(x)F_Y(x)\,dx
\end{aligned}$$

In our case with the two given buckets (q_1, q_2) and the Beta Distribution that has only non-zero values in the range from 0 to 1 this formula can be simplified to:

$$P(q_1 \geq q_2) = \int_0^1 Beta_{q_1}(k_1, f_1)I_{q_2}(k_2, f_2)\,dx$$

Where I_{q_2} is the regularized incomplete beta function. Depending on how often this has to be calculated during the experiment, this can become costly (depending on the precision again). In the paper "Numerical Computation of Stochastic Inequality Probabilities" the author John D. Cook [Coo08] uses symmetries of the distribution to arrive at a set of equations that can be used to calculate the problem recursively.

$$g(k_1, f_1, k_2, f_2) = P(q_1 > q_2)$$
$$h(k_1, f_1, k_2, f_2) = \frac{B(k_1 + k_2, f_1 + f_2)}{B(k_1, f_1)B(k_2, f_2)}$$

From that one could calculate a base case for a small sample and then continue with:

$$\begin{aligned} g(k_1 + 1, f_1, k_2, f_2) &= g(k_1, f_1, k_2, f_2) + h(k_1, f_1, k_2, f_2)/k_1 \\ g(k_1, f_1 + 1, k_2, f_2) &= g(k_1, f_1, k_2, f_2) - h(k_1, f_1, k_2, f_2)/f_1 \\ g(k_1, f_1, k_2 + 1, f_2) &= g(k_1, f_1, k_2, f_2) - h(k_1, f_1, k_2, f_2)/k_2 \\ g(k_1, f_1, k_2, f_2 + 1) &= g(k_1, f_1, k_2, f_2) + h(k_1, f_1, k_2, f_2)/f_2 \end{aligned}$$

This makes sense if the value of $P(q_1 > q_2)$ needs to be computed at any time step, since the difference to the previous result can only be an additional click or non-click by a new user. For an arbitrary number n of buckets the formula above resolves to:

$$P(q_1 > max_{i>1}q_i) = \int_0^1 Beta_{q_1}(k_1, f_1) \prod_{i=2}^{n} I_{q_i}(k_i, f_i)\, dx$$

In another paper [CN06] Cook and Nadarajah evaluate symmetries for a test with three buckets. The symmetries already reduce to:

$$\begin{aligned} &g(k_1, f_1, k_2, f_2, k_3, f_3) = g(k_1, f_1, k_3, f_3, k_2, f_2) \\ &g(k_1, f_1, k_2, f_2, k_3, f_3) + g(k_2, f_2, k_3, f_3, k_1, f_1) + g(k_3, f_3, k_1, f_1, k_2, f_2) = 1 \end{aligned}$$

Which corresponds to the rather trivial fact that $P(q_1 > q_2, q_3) = P(q_1 > q_3, q_2)$ and the three possible states of $g(...)$ must sum up to 1. Where this method of computation for two buckets is fast, it can not be generalized for more buckets.

2.2.2 Normal Approximation

For larger samples one could approximate the Beta Distribution with a normal distribution. This means that the following equation should be fulfilled for $B(k, f)$:

$$\frac{k+1}{k-1} \approx 0, \frac{f+1}{f-1} \approx 0$$

The Gaussian Distribution to a given Beta Distribution has the following shape:

$$B(k, f) \approx N\left(\frac{k}{k+f}, \sqrt{\frac{kf}{(k+f)^2(k+f+1)}}\right)$$

The inequality for this case then could be solved by:

$$\begin{aligned}
P(X > Y) &= P(0 > Y - X) \\
&= P(0 > \mu_Y - \mu_X + (\sigma_X^2 + \sigma_Y^2)^{\frac{1}{2}} Z) \\
&= P\left(Z < \frac{\mu_X - \mu_Y}{(\sigma_X^2 + \sigma_Y^2)^{\frac{1}{2}}}\right) \\
&= \Phi\left(\frac{\mu_X - \mu_Y}{(\sigma_X^2 + \sigma_Y^2)^{\frac{1}{2}}}\right)
\end{aligned}$$

Another path is starting directly with a Gaussian distribution. Intuit uses a Gaussian approximation to perform a two tailed hypothesis test to differentiate the performance between buckets. Other providers use a one tailed testing model. While the later leads to faster results it neglects the possibility that a Bucket performs worse than the control setting. The confidence interval for the conversion rate is chosen according to "Interval estimation for a binomial proportion" by Brown et al [BCD01] the Agresti-Coull interval. This works well for buckets containing $n \geq 40$ user which is a reasonable assumption. Other interval estimations may be used leading to more conservative estimates or stricter assumptions.

2.2.3 Sampling

Another method that would be more useful for many different buckets is sampling. The following Figure 2.2 shows a a possible situation after a test run (the conversion rate is higher than realistic for illustrative purposes). An algorithm for determining the best Bucket among others would

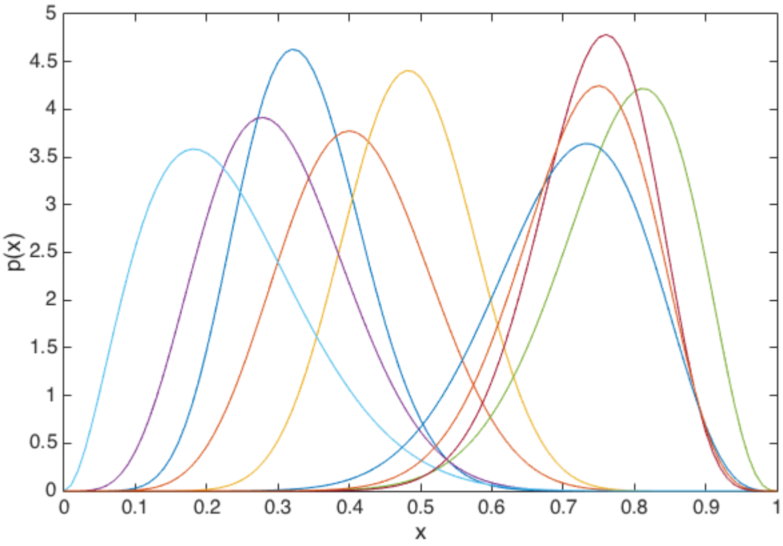


Figure 2.2: Beta distributions representing the Bucket performance during a test

look like Algorithm 1 The accuracy of this method depends on the number of samples that

```
Input: buckets, samples
Output: probabilities
probabilities ← zeroes ;
for 1 to samples do
    num ← maxarg(draw(buckets)) ;
    probabilities[num] ← probabilities[num] + $\frac{1}{\text{samples}}$ ;
end
```

Algorithm 1: Use sampling for determining the best Bucket

are drawn from each Bucket. A simulation in MatLab with two random generated Buckets and comparing it to the basic Analytic method described above resulted in 20.000 draws for a two-digit exact approximation.

2.2.4 Discussion

The generalization would again depend on the chosen method to evaluate the inequality.

Analytic

(Analytic discussion here)

Approximation

When applying the standard techniques of significance testing some general aspects have to be kept in mind. no evaluation of the results in between p - value walk paper still there? wait through a few cycles, if the software users span around different time-zones, wait for the weekend etc. testing in two directions to avoid over confidence. really check if the approximation is justifiable for your set of data.

Sampling

Sampling is very easy to implement and depending on the precision can be very fast as well. It is more useful if the majority of tests have a lot of different variants that need to be compared. Once the used technology is decided, a direct test between this approach and the analytic solution should be conducted.

2.3 Best Assignment

From the previous section it is clear that the assignment strategy is crucial for a meaningful result of the test. But besides the fact that the assignment should not result in inhomogeneous groups it is not obvious what the best strategy is. Informative rule The following section will compare several approaches.

2.3.1 Uniform

For each new user we alternate the assignment uniformly between the buckets. So for two buckets the assignment alternates between bucket 1 and bucket 2.

2.3.2 Random

The assignment is based on a random draw that yields the next bucket.

2.3.3 Entropy

Another approach one could think of is distributing the assignments in a way that they minimize the entropy of the inequality across buckets. This makes sense because the desired state at the end of the experiment starting from $P(q_1 \geq q_2) = P(q_2 \geq q_1) = 0.5$ - a high entropy, should be changed to $P(q_1 \geq q_2) \ll P(q_2 \geq q_1) \vee P(q_2 \geq q_1) \ll P(q_1 \geq q_2)$ - a low entropy. The entropy of $P(q_1 \geq q_2)$ is given by:

$$H(q_1, q_2) = -P(q_1 \geq q_2) \cdot \log_2(P(q_1 \geq q_2)) - P(q_2 \geq q_1) \cdot \log_2(P(q_2 \geq q_1))$$

Now it has to be determined which assignment is 'better' in the sense of entropy minimization. Consider the following case where a recorded event in a bucket would lead to a huge decrease in the entropy, but it is very unlikely that the user will actually show the behavior. In this case another assignment would still yield the higher information gain. The following factors are used to weight the entropy.

$$\begin{aligned} w_1 &= [\frac{k_1 + 1}{k_1 + f_1 + 1}, \frac{f_1}{k_1 + f_1 + 1}] \\ w_2 &= 1 - w_1 \end{aligned}$$

The expected entropy with the assignment given to the first bucket can then be calculated by:

$$E(H(q_1*, q_2)) = w_1 \cdot E(q_1^+, q_2) + w_2 \cdot E(q_1^-, q_2)$$

where q_1^+ assumes the user to click and q_1^- not. The formulas for the second Bucket can be calculated accordingly. When $E(H(q_1*, q_2)) < E(H(q_1, q_2*))$ the next assignment goes to the first bucket.

2.3.4 Soft Entropy

But there is a problem with the before described method. When comparing it's performance with the Random and Uniform assignments Figure 2.3 it performance worse. This effect is appears because the described algorithm optimizes greedily. It start off with a small offset between the Buckets and keeps on focusing on them without trying the other Bucket. What can be done to correct this behavior? Different solutions are possible. The used one in this implementation was adding a Soft-Max Algorithm to the Entropy Assignment to allow for small deviations from the rule.

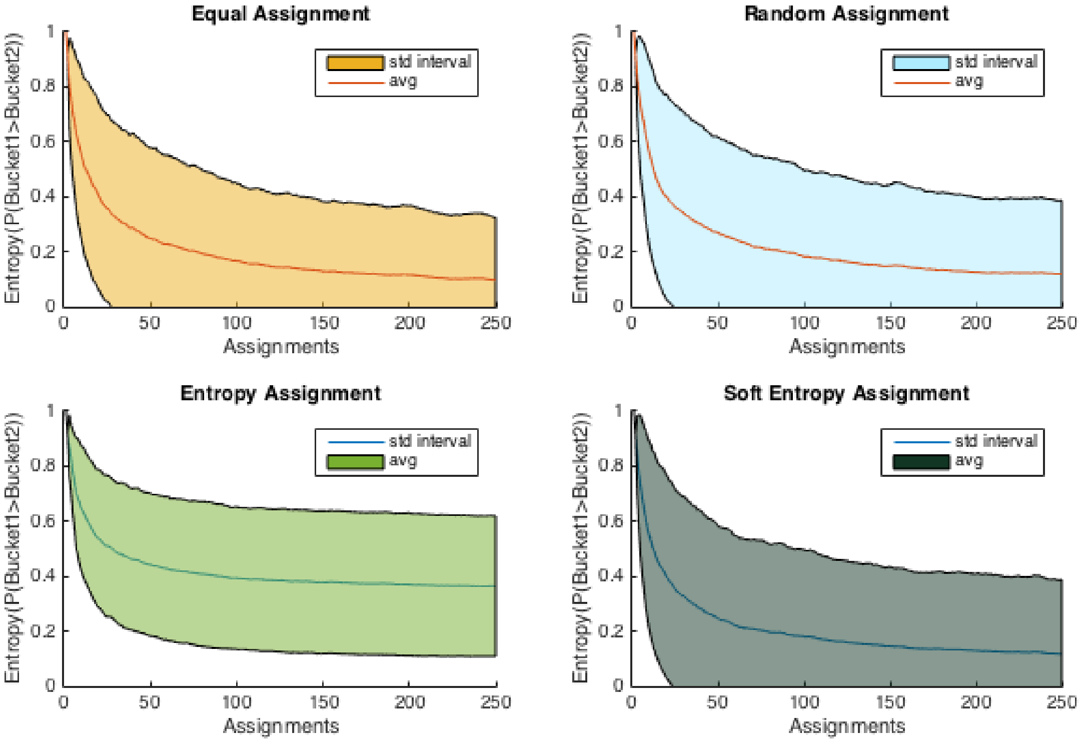


Figure 2.3: Comparison of Assignment Strategies

2.3.5 Discussion

Yeah!

Chapter 3

Bandit AB-Testing

3.1 Underlying Idea

The previous chapter dealt with optimization concerning classical AB-Testing. The following will describe a fundamentally different approach in which there is no separation between the exploration and the exploitation phase. This is motivated by the idea that valuable costumers could be lost during exploration. Basically this means that we want to already maximize our profit while gaining information about the other buckets. When combining exploration and exploitation one has to choose on each step if a known good bucket should be used again or if one invests in other unknown buckets to gain more information about their reward function. This balancing is at the core of the now described algorithms.

3.1.1 Bandit Algorithms

Bandit algorithms originate from Machine Learning where they serve learning agents to show sensible behavior while exploring the environment. In each time step an agent is forced to make a choice among several actions. An action will lead to a reward. The reward function for any action is not known from the beginning and the agent can only estimate the true reward function of any action by trying this action. The objective is of course to maximize the received reward over the whole experiment. The name of this class of algorithms stems from their relation to the famous casino slot machines. Each action can be imagined as one lever in a row of many one-armed bandits. Each machine has a hidden reward function and a player wants to maximize their earned money at the end of the evening. The update rule for each step can be roughly formalized as:

$$NewEstimate \leftarrow OldEstimate + StepSize[Target - OldEstimate]$$

A K-armed bandit problem is defined by random variables $X_{i,n}$ for $1 \leq i \leq K$ and $n \geq 1$ where each i is the index of a slot machine. Successive plays of machine i yield reward $X_{i,1}, X_{i,2}, ...$ which are independent and identically distributed according to an unknown law with unknown expectation μ_i. The machines are also independent.

3.1.2 Reformulating AB-Testing to Bandits

Each new user is our chance to either explore or exploit what we know about the buckets of the test. Each assignment becomes the pull on a lever. Since we only measure each user once, the assignments are independent and identically distributed. We deal with a stationary environment which means our *StepSize* is $\frac{1}{k}$ where k denotes the number of assignments. Since we use no prior knowledge there is also *no side information*.

Terms

We want to find the real distributions underlying the variations we made in the buckets. Assume this distributions to be of the form $P_B(y|w)$ where w are the unknown parameters that describe the distribution. By assigning users we generate random observations $D_B = (y_1, y_2..y_t)$.

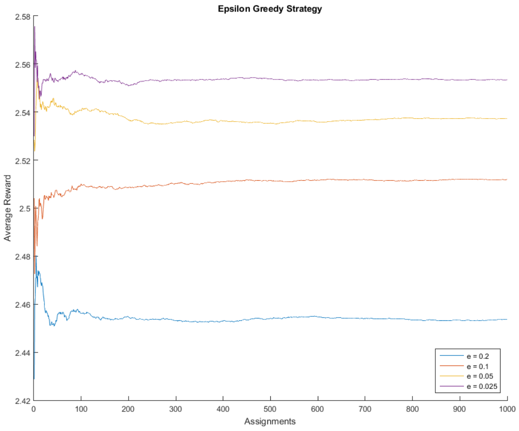
3.2 Bandit algorithms

3.2.1 epsilon-greedy

One of the simplest algorithms exploits always the best performing bucket (with the highest conversion rate) except for ϵ's fraction of cases where the next bucket is chosen uniformly. For example: if $\epsilon = 0.1$ every tenth assignment would not be to the best performing bucket. This also means that even after convergence for the bucket probabilities 10% of the assignments would not always hit the optimal bucket. For n buckets this would be:

$$P(exploration) \cdot P(\neg bestmachine|exploration) = \epsilon \cdot \frac{n-1}{n}$$

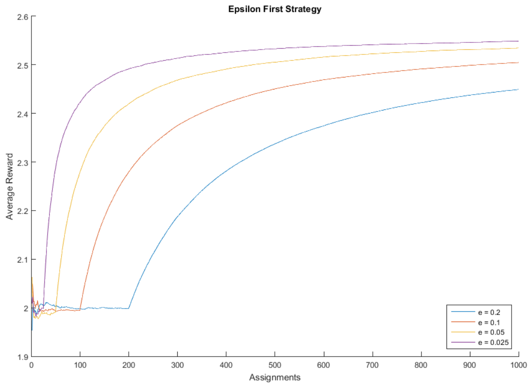
times a not optimal result. A simulation with 1000 times 1000 assignments and varying ϵ gives the following plot for this algorithm: Not that this algorithm does not track or account for the uncertainty of the estimates it has about the buckets. And since it has no 'sense' for time (the Algorithm does not change if the experiment is nearly over and only n choices left) the



epsilon-greedy performs equally well if the environment changes and users behavior change later on.

3.2.2 epsilon-first

Another algorithm that is closely related is called the $\epsilon - first$ algorithm. This is close to A/B-Testing since the exploration phase proceeds the exploitation phase for a finite number of steps and afterwards just exploitation happens. This would mimic the case where the tested change with the highest payoff is implemented and from then on permanently presented to all customers.



3.2.3 UCB

This algorithm does not explore for a fixed number of assignments, but dynamically changes the chances for not so good performing buckets to be chosen. The Hoeffding's inequality gives a confidence bound that models how sure we are about the estimated payoff across multiple plays. This way - the longer we do not play a machine the more likely it will be played in the future (but only once). The probability for each machine is defined as:

$$mean(b_n) + \sqrt{\frac{2 \log N}{n_p(b_n)}}$$

Here b_n is one of n buckets and $n_p(b_n)$ is the number of times this bucket has been played. N corresponds to the number of assignments across all buckets. This algorithm requires $mean(b_n)$ to be bounded by 1 so that the confidence bound overpowers the first term. The fraction of time that is spend on exploring decreases exponentially.

3.2.4 Thompson sampling

This fairly simple algorithm was originally developed by William R. Thompson in his paper "On the likelihood that one unknown probability exceeds another in view of the evidence of two

```
Input: buckets
Output: nextAssignment
draws ← zeroes ;
foreach i in buckets do
    draws (i) ← draw(buckets[i])
end
nextAssignment ← maxarg(draws) ;
```

Algorithm 2: Use Thompson sampling to determine the Bucket for each Assignment

samples" [Tho33] To determine the next bucket to be played, the buckets probability distribution itself is used.

They generate numbers that naturally balance between exploitation and exploration, because less performing buckets are demoted and winners promoted by the information already collected. This method is as well resistant to changes in the environment since they 'notice' the change by playing a less performing Bucket, that will then be played more often if his performance increased.

3.2.5 Discussion

Bandit algorithms can be very useful when they keep on running even after the experimental phase is over. In fact, since most of them distribute new users in a way that over the time most of them experience the 'better' variant, they replace the deployment cycle for new features by rolling them out slowly to the customer base.

Chapter 4

Discussion

Although Bandit and regular A/B Testing algorithms have a lot in common they do not really solve for the same problem which makes a clear decision upon what is better very hard.

4.1 A different Mindset

The Bayesian approaches gives subjective values based on the so far collected data but does not care about significance.

4.2 Duration of the test

Just comparing the duration of the test. One could argue that Bandit algorithms are useful if they keep on running in the background of a product/service and not turned of at a certain time, hence rendering the question for statistical significance useless. And since they are adapting dynamically one could react to changes occurring later in time. A good idea, but it leads also to several implications that are possibly not easy to be resolved.

Imagine a reasonable sized application or service. If testing proofs valuable an application will not just have one test running at a time.

No statistical model can compensate for a test that is not run long enough to cover different behavioral patterns of the targeted users during time. Meaning that for any useful test the cycles that are important have to be identified first (work-weekend,day-night,differences in timezone) and the duration adapted to account for those.

4.3 A reasonable prior

List of Figures

List of Algorithms

Bibliography

[Art14] Charles Arthur. Facebook emotion study breached ethical guidelines, researchers say. *The Guardian*, 2014.

[BCD01] Lawrence D Brown, T Tony Cai, and Anirban DasGupta. Interval estimation for a binomial proportion. *Statistical science*, pages 101–117, 2001.

[CN06] John D Cook and Saralees Nadarajah. Stochastic inequality probabilities for adaptively randomized clinical trials. *Biometrical journal*, 48(3):356–365, 2006.

[Coo08] John D Cook. Numerical computation of stochastic inequality probabilities. Technical report, UT MD Anderson Cancer Center Department of Biostatistics, 2008.

[Tho33] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, pages 285–294, 1933.

[ZA13] Shunan Zhang and J Yu Angela. Cheap but clever: Human active learning in a bandit setting. *Ratio*, 12(13):14, 2013.

Declaration of Authorship

I hereby certify that the work presented here is, to the best of my knowledge and belief, original and the result of my own investigations, except as acknowledged, and has not been submitted, either in part or whole, for a degree at this or any other university.

Osnabrück, July 28, 2015