Institute of Cognitive Science
&
Intuit Data Engineering and Analytics

Bachelor's Thesis

Comparing Models for A/B Testing

Andrea Suckro

September 30, 2015

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

Comparing Models for A/B Testing

A/B Testing is a business term that is tied to the concept of hypothesis testing and commonly used in marketing. Though it has been around for a while and many tools have been developed to do automated A/B Testing, the underlying statistics are widely discussed and each A/B Testing software implements different analytic approaches. I will start by describing the general procedure, then different methods will be compared on a simplified model, starting with more classical approaches and comparing these to the bandit algorithms popular in machine learning. In the discussion I will summarize pros and cons for the presented methods and outline a picture for future development.

Contents

Chapter 1

Introduction

Hypothesis testing has always been the fundamental tool of scientific research. Conducting test cases that vary the variable under introspection, recording the results and using statistic methods to determine significant results is the current methodology that shall ensure objective insights. This approach is more and more adapted for business research as well. In the context of marketing it is called A/B Testing and is used to gather insights about the customers behavior under varying conditions. The process is similar to the steps in a scientific set up: the participants are selected randomly from the customer base and are presented with a base condition (the unchanged product) or one or more variants of it. The customers feedback is then collected and used to analyze whether the new feature is a significant success or failure.

As computations on big sets of data are getting more tangible, automated A/B Testing becomes part of many online services or products. In fact by using online products like Google or Facebook it is highly likely that most of us have already been taking part in such an experiment and though that can raise some attention [Art14] most of these tests go by unnoticed. 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 setting.

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. Meanwhile 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 to 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 theoretical models found their way to the industry and started a discussion about whether this is the new, right way to do testing.

Experimenters hope for 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.

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 with a fixed number of buckets ranging from 2 to n. It is also defined what proportion of the users of the tested product shall take part in the test. This number is called the sampling rate and determines whether a user is taken into consideration before she is assigned to a bucket. The test is active for a preset amount of time, collecting the data to determine if there is a significant difference between the buckets.

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 percentages sum up to 100% .

Assignment An assignment determines the experience the user will be exposed to during the test. Once determined it stays fixed, meaning that a user will never be exposed to different buckets throughout one test.

Action A user with an assignment may perform an action that is measured. For example – the clicking on a specific button triggers 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

Encapsulating the process of administrating and measuring A/B Tests is the purpose of any A/B Testing system. Experimenters have the possibility to create and monitor their tests from such systems, allowing them to keep track of the tests that are running for their product and also keep a history for already finished tests. The following steps describe the different parts that are inherent to those testing platforms.

1.2.1 Creating a Test

A test is created with the above described properties. Once that is done, the experimenter has to tie the assignments a user gets to different experiences in the tested product. The A/B Testing service is ignorant towards the concrete differences that are tested – it only handles the administration of the assignments. The experimenter also needs to define when an action is created and send this to the A/B Testing service so that it can collect the action data and provide analysis on it.

1.2.2 Running the Test

For every new user who is visiting the tested product the A/B Testing service determines if she is part of the experiment based on the sampling rate. If she takes part in the testing the next

step for the A/B Testing service is to decide 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 buckets throughout the testing phase. Though it is not encouraged, experimenters can still update their tests and for example increase the sampling rate or disable whole buckets, if they are not useful anymore.

An example scenario will now illustrate the described process with a concrete implementation.

1.3 Example Scenario

Intuit is a software company that focuses on products in the financial and tax preparation sector as well as related services for small businesses, accountants and individuals. The data engineering and analytics group within Intuit has developed a framework to conduct A/B Tests called *jabba*. It is a web service that can be integrated by any online product and uses HTTP (Hypertext Transfer Protocol) requests to measure the characteristic numbers for a test. I will use this service as an exemplary implementation for an A/B Testing service throughout the thesis. A simple use case is now described in the following section.
Imagine a product team that wants to raise awareness of password security for their existing users. They discuss two ways to do that:

1. Sending out an email that informs the user of the importance of a strong password
2. Placing a banner on their products landing page that contains the same information

They create a test in *jabba* and specify that the test should be active for three weeks with a sampling rate of 5%. They generate two buckets: Bucket A for the email and Bucket B for the landing page. Each bucket get's 50% of the 5% of overall users that participate in the test. Once the test is started, the product makes a call to *jabba* for each user that logs in to the system. *Jabba* determines whether the user participates in the test and if so in which bucket. The product shows, depending on the returned assignment, whether the user experiences no changes (she falls outside the 5% sampling percentage), the email (Bucket A) or the security banner (Bucket B). If the user changes her password during the time the test is active it counts as an action that is sent to *jabba* as well. After the testing period is over *jabba* reports the performance of each bucket and whether one was significantly better than the other. Based on this, the product group can decide which feature they want to roll out for all their users.

Chapter 2

Classic A/B Testing

2.1 Assumptions and Setup

This section will take a look at the common approach to A/B Testing as it is also implemented in *jabba* 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: Bucket 1 and Bucket 2. Users assigned to Bucket 1 generate actions with a probability q_1, for Bucket 2 this probability is q_2. The prior distribution for all q is a beta distribution $B(\alpha, \beta)$. This is a convenient choice because it is the conjugate family for the binomial likelihood.

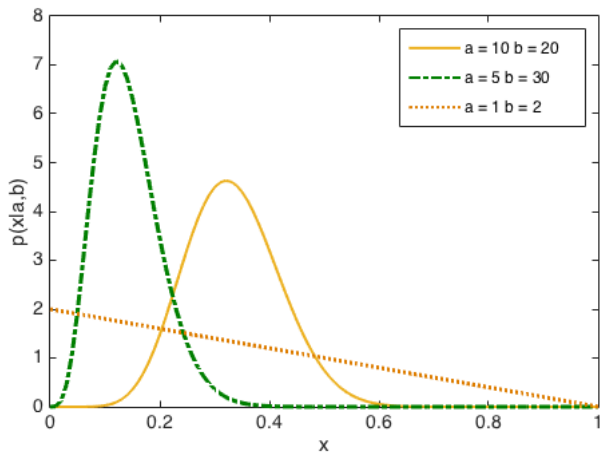

Figure 2.1: Different Beta Distributions

We also assume no prior knowledge from previous runs when creating the test. This is modeled by the uninformative prior $B(\alpha = 1, \beta = 1)$. N_1 users get assigned to Bucket 1 and N_2 to Bucket 2. They generate k_1 and k_2 actions. The posterior distributions for each individual

case is given by:

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

So each new assignment and each new action will directly alter the distribution parameters. It is important to notice that this model assumes actions to occur directly. The user is assigned to a bucket n, produces an action (k_n+1) or not (f_n+1) and leaves again. A user is only measured once – meaning that the model does not account for actions produced by returning users. A/B Testing tries to find the posterior probability for $P(q_1 > q_2|k_1, f_1, k_2, f_2)$, the probability that Bucket 1, given the collected data, will perform better than Bucket 2. The following section explains three different approaches to that conundrum.

2.2 Best Bucket

The following methods give an overview about strategies that can be implemented to find the best performing bucket among others.

2.2.1 Analytic

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

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

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:

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

Where I_{q_2} is the regularized incomplete beta function. 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 Cook shows that starting from a base case all following assignments can be computed by:

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

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 action or no-action 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 number of applicable symmetries in this case already reduce to:

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

This corresponds to the fact that $P(q_1 > q_2, q_3) = P(q_1 > q_3, q_2)$ and that the three possible states of g must sum up to 1. Where the symmetries for 2 buckets can be used for calculating subsequent states of the test, they can not be generalized for tests with more buckets.

2.2.2 Normal Approximation

Different approximations of a normal distribution to the beta distribution can be applied. One detailed derivation is described in "A normal approximation for beta and gamma tail probabilities" by Dieter and Hermann [AD84]. For larger samples as they are used in the A/B Testing case a simpler approximation can be chosen [LLC]. 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 normal distribution such a beta distribution is given by 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 two normal distributed variables can be solved by:

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

The gained normal distribution can now be used to perform several hypothesis tests on it. *Jabba* uses this for example to perform a two-tailed hypothesis test to differentiate the performance between buckets while other frameworks use a one tailed testing model. Another important choice in that case is the confidence interval, since it models how reliable the estimate of the action rate is. In *jabba* the confidence interval is chosen according to "Interval estimation for a binomial proportion" by Brown et al. [BCD01] the Agresti-Coull interval. The authors find that this interval works well for buckets containing $n \geq 40$ users which is a reasonable assumption, given that those tests are run online and rather deal with several thousand assignments. Other interval estimations may be used leading to more conservative estimates or stricter assumptions.

2.2.3 Sampling

Another method that is more useful for many different buckets is sampling. The following Figure 2.2 shows a possible situation for a test with many buckets and a high conversion rate for illustrative purposes. One algorithm for determining the best bucket among others is algorithm 1. The accuracy of this method depends on the number of samples that are drawn from each bucket.

Algorithm 1: Determining the best bucket by sampling

Input: buckets, samples
Output: probabilities

probabilities ← zeroes ;
for 1 ... samples **do**
    num ← `maxarg(draw(buckets))` ;
    probabilities[num] ← probabilities[num] + $\frac{1}{\text{samples}}$;
end

A simulation in MatLab was conducted to find a reasonable number for the two bucket test. For this, random beta distributions have been evaluated using sampling and the basic analytic method described in 2.2.1. For a two-digit exact approximation of the sampling method 20.000 draws were necessary.

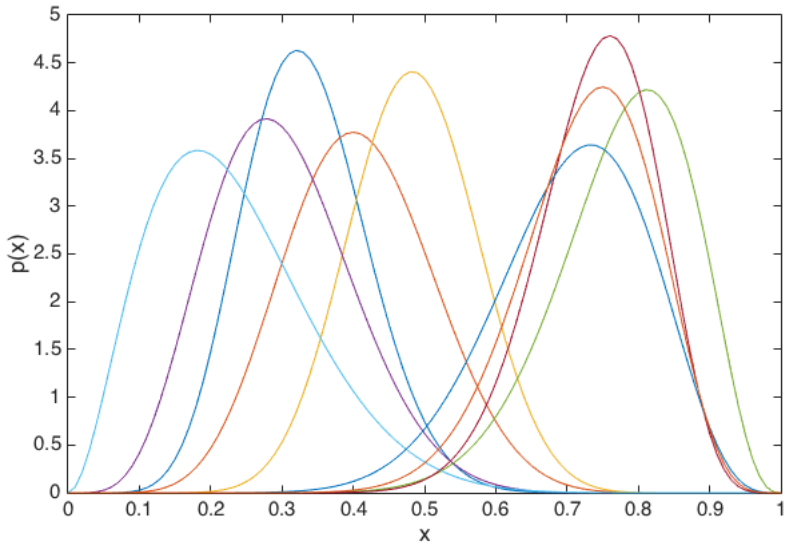

Figure 2.2: Beta distributions representing the bucket performance during the test

2.2.4 Discussion

The described methods where evaluated for comparing two bucket performances. A generalization to n buckets depends on the chosen method to evaluate the difference between them.

Analytic

The analytic approach gives an exact value to the probability that one bucket is performing better than the other, independent of the number of buckets involved in the test. And although the presented symmetries do not generalize to more than two buckets, this method is still superior to the normal approximation in the sense that it is cleaner from the theoretical standpoint, leading to less assumptions about the problem and an easier application.

Approximation

When applying the standard techniques of significance testing some general aspects have to be kept in mind. 2.2.2 already touched on the difference between one- and two-sided hypothesis testing. J. M. Bland et al. [BB+94] describe in their article "Statistics Notes: One and two sided tests of significance" the implication both testing frameworks have on the significance of a result. And though A/B Testing as described here is not tied to clinical trials the underlying issue is the same: when testing one sided the experimenter ignores the possibility that a new feature may perform worse than the base implementation. This can lead to the implementation of features that do not really have a significant effect on the customers behavior. The whole method of significance testing is furthermore again and again source of misconceptions and confusion that also apply to the result of any A/B Test. Goodman describes the common fallacies in his paper

"A dirty dozen: twelve p-value misconceptions" [Goo08]. The experimenter has therefore to be very careful when interpreting her results.

Sampling

Sampling is very easy to implement and, depending on the needed precision, can be very fast as well. It is more useful if the majority of tests have a lot of different buckets that need to be compared frequently.

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 overall best strategy is. In the following section several approaches will be described and evaluated against each other.

2.3.1 Equal

For each new user, the assignment alternates equally between the buckets. That means that very bad performing buckets are chosen as often as very well performing ones.

2.3.2 Random

The assignment is based on a draw from a uniform distribution that yields the next bucket. In the long run the assignments are equally distributed as in the for the equal assignment, but can fluctuate in the beginning depending on the concrete draws.

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.

$$w_1 = \left[\frac{k_1 + 1}{k_1 + f_1 + 1}, \frac{f_1}{k_1 + f_1 + 1}\right]$$
$$w_2 = 1 - w_1$$

The expected entropy with the assignment given to the first bucket, denoted by q_1^*, can then be calculated:

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

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. The results of this method can be found in Figure 2.3.

2.3.4 Soft Entropy

There is a problem with the before described method though. When comparing its performance with the random and uniform assignments (see Figure 2.3) it performs worse. This effect appears because the described entropy algorithm optimizes greedily. It starts off with a small offset between the buckets and keeps on focusing on them without trying the other bucket. Different solutions to this situation are possible. The used one in this implementation was adding a softmax algorithm as described in Sutton and Barto's chapter "Softmax Action Selection" [SB98] to the entropy assignment.

$$p(q_1) = \frac{e^{\frac{H(q_1^*, q_2)}{T}}}{e^{\frac{H(q_1^*, q_2)}{T}} + e^{\frac{H(q_1, q_2^*)}{T}}}$$

The parameter T called temperature modulates how strong differences between $H(q_1^*, q_2)$ and $H(q_1, q_2^*)$ shall be weighted when calculating the probability that the next assignment goes to Bucket 1 ($p(q_1)$). A high temperature can therefore be used to make the selection of a bucket with higher estimated entropy more likely.

2.3.5 Discussion

The used algorithms for the bucket assignment do not differ that much for a two bucket test, see Figure 2.3. This can be seen in two ways. Either the differences between the presented algorithm do not matter for a test with only two buckets, or equal and random assignment strategies are already very efficient in what they are doing.

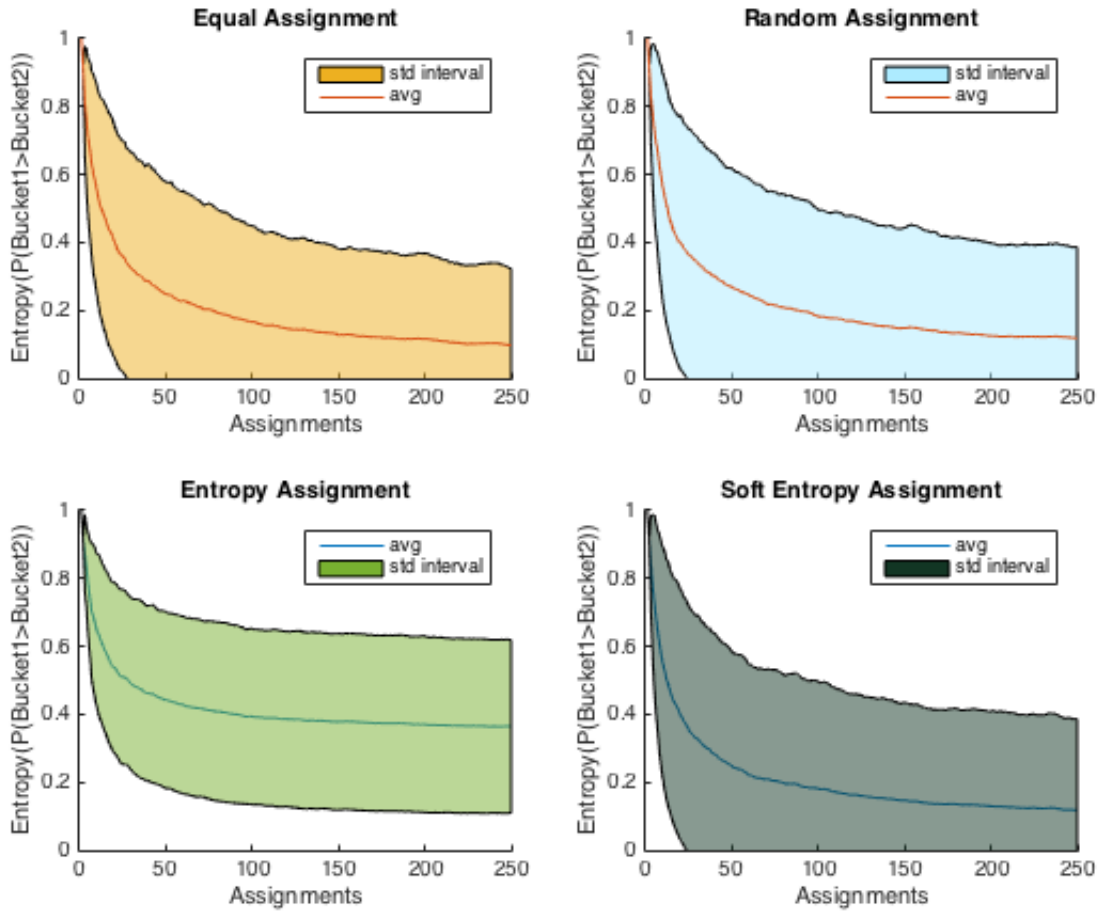

Figure 2.3: Comparison of Assignment Strategies
The plots show the performance of the different assignment strategies for randomized buckets. The entropy is used as a measurement of how sure we get over the number of assignments that one bucket is better/worse than the other.

Chapter 3

Bandit A/B Testing

3.1 Underlying Idea

The previous chapter dealt with optimizations concerning classical A/B Testing. The following will describe a fundamentally different approach. Classical A/B Testing focuses solely on gaining information about the different buckets, enabling teams to implement the best performing solution. In the bandit setting this separation between the exploration and the exploitation phase is abrogated. This is motivated by the idea that valuable customers can be lost during exploration. Basically this means that we want to already maximize our profit while gaining information about the different 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 individual reward function. This balancing is the core of the now described algorithms.

3.1.1 Bandit algorithms

Bandit algorithms originate from machine learning where they serve learning agents like autonomous robots 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 unknown and the agent can only estimate the true reward function by learning from the generated outcomes. Over the whole experiment any agent's objective is to maximize its received reward. The name of this class of algorithms stems from their relation to the famous casino slot machines called one-armed bandits. Each action can be imagined as the pull of one lever in a row of many of those machines. Each bandit has a hidden reward function and a player wants to maximize her earned money over the course of the evening. The rule for updating the estimate about each machines reward function can be roughly formalized as:

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

where the *OldEstimate* is what the player assumed about the reward that is generated by playing this machine. *Reward* is the value the payer observed after choosing the action and the

StepSize is a value that modulates the learning rate. Based on those information a new estimate is formed about the machine. 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 and n the number of times a specific machine has been played. Successive plays of machine i yield reward $X_{i,1}, X_{i,2}, ...$ which are independent and identically distributed according to an unknown reward distribution with unknown expectation μ_i. The machines among themselves are also independent.

3.1.2 Reformulating A/B Testing

Each new user is the chance to either explore or exploit what is known about the buckets of the test. Each assignment becomes the pull on a lever. Since the setup is ignorant towards returning users, the assignments are independent and identically distributed. The A/B Test describes a stationary environment which means our *StepSize* is $\frac{1}{k}$ where k denotes the total number of assignments for one test. No prior knowledge is assumed before running the test. This translates to no data or knowledge that is not deducible from the single trials, in machine learning such information is called *side information*. The reward value does not differ among the different levers. It is always 1, although it could be in fact another fixed value. One of the later described algorithms 3.2.3 does require the reward of a lever to be bounded by 1, but for the sake of A/B Testing the concrete value does not matter and can be adapted as seen fit – it only has to be greater than 0 and the same for each bucket. What differs though are the real distributions underlying the variations represented by the buckets. These are of the form $P_B(y|w)$ where w are the unknown parameters that define the distributions function. By assigning users random observations $D_B = (y_1, y_2..y_t)$ are generated. These observations are used to estimate the difference between the buckets. The test ends after a given number of assignments – this number is called the *horizon*.

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 randomly. For example if $\epsilon = 0.1$ then every tenth assignment would not necessarily be to the so far best performing bucket. This means that even after convergence for the bucket probabilities 10% of the assignments will not always hit the optimal bucket. This can be formulated for n buckets as follows:

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

A simulation of 1000 tests, with two random buckets and a horizon of 1000 and varying ϵ result in Figure 3.1 for the performance of this algorithm. Note 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' of time (the algorithm does not change if the experiment is almost over and only $m \ll n$ choices left) the epsilon-greedy performs equally well if the environment changes and users behavior change later on.

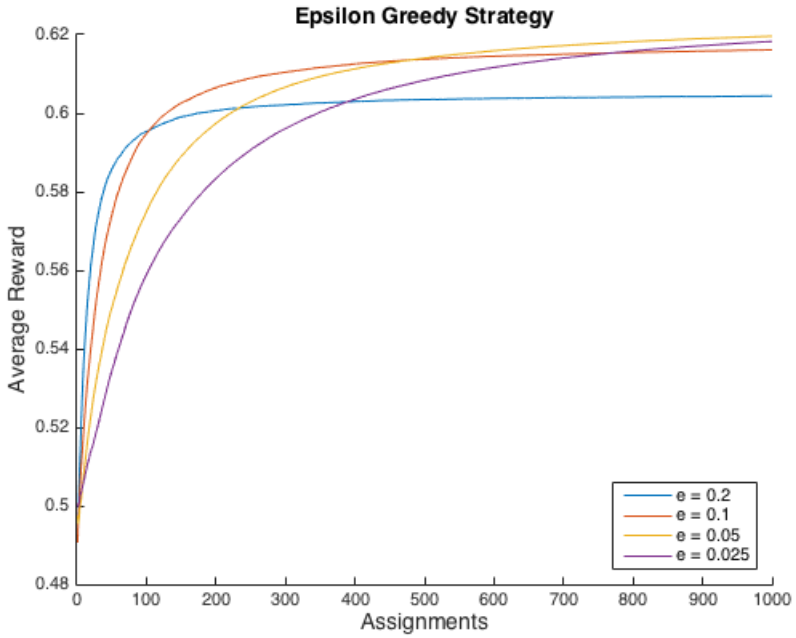

Figure 3.1: Epsilon Greedy

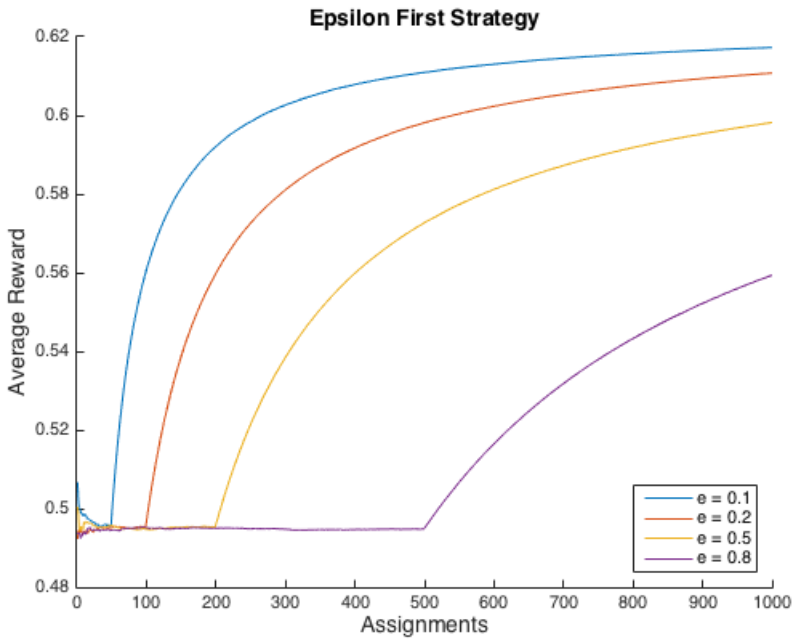

Figure 3.2: Epsilon First

3.2.2 Epsilon-first

Another algorithm that is closely related is called the ϵ-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 exploits. This resembles the case where the tested change with the highest payoff is implemented and from then on permanently presented to all users, see Figure 3.2. Since the assigned bucket is fixed after $\epsilon \cdot n$ steps this algorithm cannot react to changes that later on happen in the environment.

3.2.3 Unified Confidence Bound

The concrete formulas in this section are taken from "Finite-time analysis of the multiarmed bandit problem" by Auer et al. [ACBF02], but the algorithm itself is known across the community. UCB (which stands for *Uniform Confidence Bound*) does not explore for a fixed number of assignments, but dynamically changes the chances for less 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). We choose to play the next machine that maximizes:

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

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. This is given by our assumption that the concrete value of the individual rewards does not matter as long as they are the same across buckets. The fraction of time that is spent 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 samples" [Tho33]. To determine the next bucket to be played, the buckets probability distribution itself is used. The result of that sampling is used to update the buckets underlying distribution as described in 2.1.

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 its performance increased. A comparison between Thompson Sampling and the UCB-Algortihm described in 3.2.3 can be found in 3.3. Here the UCB is weaker in the first part of assignments but slightly outperforms the Thompson sampling in the long run. A paper that is taking a look at the general performance of Thompson sampling versus UCB for more arms is "An empirical evaluation of Thompson sampling" by Chapelle and Li [CL11]. They show that in the general case Thompson sampling outperforms UCB.

Algorithm 2: The usage of Thompson sampling

Input: buckets

```
draws ← zeroes ;
foreach i in buckets do
  | draws (i) ← draw(buckets[i])
end
nextBucket ← maxarg(draws) ;
rew = reward(nextBucket);
if rew == 1 then
  | updateBucketSuccess(nextBucket);
else
  | updateBucketFailure(nextBucket);
end
```

3.2.5 Discussion

The section started with a weaker Bandit algorithm: epsilon-greedy. The major drawback on that was the fact that the exploration phase did not adapt to the gained insights of the different buckets. But still: for this simplified scenario it works well compared to the other algorithms. This is due to two major restrictions that were imposed on the problem. First only two buckets were used in the A/B Test, for more buckets the performance of epsilon-greedy would reduce since the number of non-optimal played buckets $\epsilon \cdot \frac{n-1}{n}$ grows for increasing number n of buckets. Another assumption was that the reward functions of the buckets do not change throughout the test. This is especially important for epsilon-first since this method is not sensitive to changes that occur after the initial exploration phase.

UCB and Thompson Sampling are stronger algorithms in the way that they account for changes in the users behavior and their exploration phase is dynamically bound to the information that are collected during the test. This is not taken advantage of in the explored A/B Testing scenario of this thesis since the reward functions do not change with the trials, but can be powerful in any real application.

Bandit algorithms are very useful when they keep on running for an unspecified time. 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.

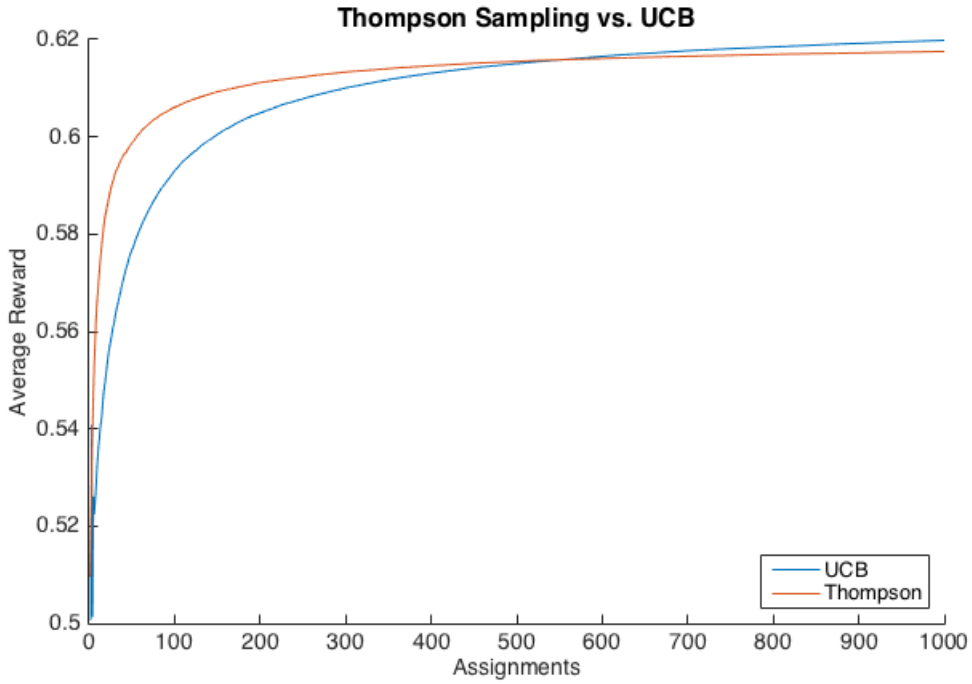

Figure 3.3: Comparison for Thompson sampling and UCB
The graph averages 1000 runs with random bucket distributions and a horizon of 1000.

Chapter 4

Discussion

Although Bandit and regular A/B Testing algorithms have a lot in common they do not really solve the same problems which makes a clear decision upon which is better difficult to decide. The Bandit algorithms make more sense in an ongoing test setting, where the surrounding conditions can change through the trial and a test is not finished after a fixed short time period. This also entails that users have to cope with possibly more changes from time to time. In fact when coming back to the limitations that were set at the beginning of the thesis more points come up that may cause troubles.

4.1 The Limitations of the Used Restrictions

In the beginning of the thesis we applied some simplifications to the problem of A/B Testing. The number of buckets was restricted to two which is not harmful since no algorithm takes this as a hard precondition, although it can affect optimization 2.2.1. This means that the described methods work for more buckets as well. Specifically for the Bandit algorithms it makes sense to use UCB or Thompson sampling with more buckets, since they should outperform epsilon-first and epsilon-greedy in this setting.

Another factor is that for a realistic A/B Test it is not clear when the user is producing an event. She may come to the tested page but not click the specified link or button. This means that until the user really performs an action we can only speak of a non-click event with a certain probability. For modeling this, probability data from previous tests could be utilized, making it more and more likely that the user will not click as more time passes by. Users can also provide more than one event, making the samples no longer independent. It is not really clear how the Bandit algorithms have to be adapted in a way that they support this setting.

The described methods make not use of a prior so far. Using previous test runs could provide a distribution over the differences between buckets. This distribution is most likely shaped similar to an exponential decay function, making small differences between buckets more likely than huge differences. This makes sense, because no introduced change to a certain product will profoundly change the users' behavior. This prior can be easily integrated in the Analytic and Bandit's approaches, since the used beta distributions would not start with a flat prior, but the values estimated from previous similar test runs.

4.2 Duration of the Test

Merely comparing the duration of an A/B Test, one could argue that Bandit algorithms are useful if they keep on running in the background of a product/service and are not turned off 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. This is a theoretically solid 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 this application will not just have one test running at a time. The different buckets have to be maintained in the code and dragged along through several releases as some customers might still get that experience. Which brings up another point: if the Bandit A/B Test finds that one bucket is clearly favorable over the other – why keep it from a small percentage of the over all user base?

This affects not only users but customer service in a running business. How to handle requests that are made upon different buckets by the user? The tests themselves may not drastically change the application, but many little tests can blur the whole picture.

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 Final thoughts

Given the discussed restrictions of the evaluated Bandit algorithms I think it is too early to adopt them to A/B Testing. In real world applications it makes sense to have the tests bound to a certain time interval and later on deciding for one feature to be rolled out to all the customers. Having said that, I also think that using the normal approximation as it is done so far in many A/B Testing systems is not a good solution either. The proposed model with Beta distributions could be easily used to incorporate data from previous A/B Tests, which are unaccounted for in the regular setting. The interpretation of the test results would also become more easy since the concrete probabilities can be reported to the experimenter and she can then decide (given some reasonable extra assumptions, that can not be accounted for by the theory – like differences between day and night, workday or weekend) when the test is over.

Future research in this could find good methods to extract a useful prior from previous tests, that is not too 'strong' to be overcome by an exceptional case, but not too 'weak' to unnecessarily prolong other tests. One could also use data from past tests to identify customers' behavioral patterns and account for that with an estimate of the test duration.

List of Figures

List of Algorithms

Bibliography

[ACBF02] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[AD84] Dieter Alfers and Hermann Dinges. A normal approximation for beta and gamma tail probabilities. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 65(3):399–420, 1984.

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

[BB+94] J Martin Bland, DG Bland, et al. Statistics notes: One and two sided tests of significance. *Bmj*, 309(6949):248, 1994.

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

[CL11] Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pages 2249–2257, 2011.

[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.

[Goo08] Steven Goodman. A dirty dozen: twelve p-value misconceptions. In *Seminars in hematology*, volume 45, pages 135–140. Elsevier, 2008.

[LLC] EpiX Analytics LLC. Model assist version 3.0. *How to use ModelAssist for @Risk*, pages R–M0323–A.

[SB98] R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*, volume 116. Cambridge Univ Press, 1998.

[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, September 30, 2015