Bayesian A/B Testing

Beta Distribution and Multi-Arm Bandit

A/B Testing
(frequentist)

- Define a metric (CTR, for example)
- Determine parameters of interest for study (number of observations, power, significance threshold, and so on)
- Run test, without checking results, until number of observations has been achieved
- Calculate p-value associated with your hypothesis test
- report p-value and suggestion for action

A/B Testing
(frequentist)

- Can you say “it is 95% likely that site A is better than site B”?
- Can you stop test early based on surprising data?
- Can you update the parameters of your test while it is running?

A/B Testing
(bayesian)

- Define a metric (CTR, for example)
- Run test, continually monitor results
- At any time calculate a probability that site A has better results on the defined metric than site B
- Suggest courses of action based on this probability

A/B Testing

(bayesian)

- Can you say “it is 95% likely that site A is better than site B”?
- Can you stop test early based on surprising data?
- Can you update the parameters of your test while it is running?

Bayes Theorem

Posterior Probability

Likelihood of Observations

Prior Probability

$$\Pr(\theta|y) = \frac{\Pr(y|\theta)\Pr(\theta)}{\Pr(y)}$$

Normalizing Constant

- **prior:** initial belief
- **likelihood:** likelihood of data given outcome
- **posterior:** updated belief

Bayes Theorem

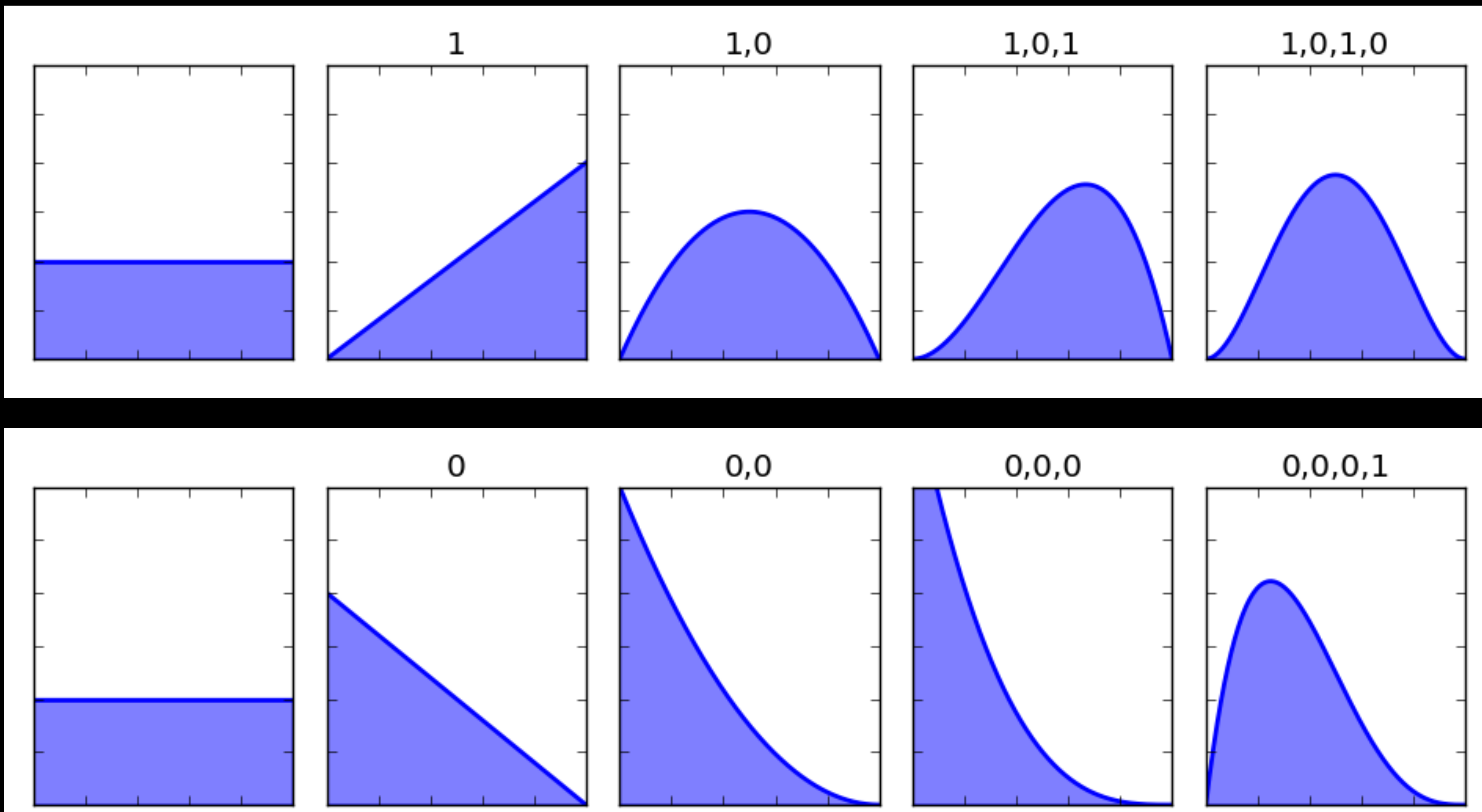
posterior ∝ prior x likelihood

Beta Distribution

$$\frac{p^{\alpha-1}(1-p)^{\beta-1}}{\mathrm{B}(\alpha,\beta)}$$

- p: conversion rate (between 0 and 1)
- α, $\boldsymbol{\beta}$: shape parameters
  - α = 1 + number of conversions
  - $\boldsymbol{\beta}$ = 1 + number of non conversions
- Beta Function ($\mathbf{B}$) is a normalizing constant
- $\alpha = \boldsymbol{\beta} = 1$ gives the *uniform distribution*

The Distribution



1 = conversion   0 = non conversion

Binomial (Likelihood)

$$\binom{n}{k} p^k (1-p)^{n-k}$$

- p: conversion rate (between 0 and 1)
- n: number of visitors
- k: number of conversions

Conjugate Priors

posterior ∝ prior × likelihood

beta ∝ beta × binomial

THE MATH:

$$
\begin{aligned}
\text{posterior} &\propto \text{prior} \times \text{likelihood} \\
&= \frac{p^{\alpha-1}(1-p)^{\beta-1}}{B(a,b)} \times \binom{n}{k} p^k (1-p)^{n-k} \\
&\propto p^{\alpha-1}(1-p)^{\beta-1} \times p^k (1-p)^{n-k} \\
&\propto p^{\alpha+k-1}(1-p)^{\beta+n-k-1}
\end{aligned}
$$

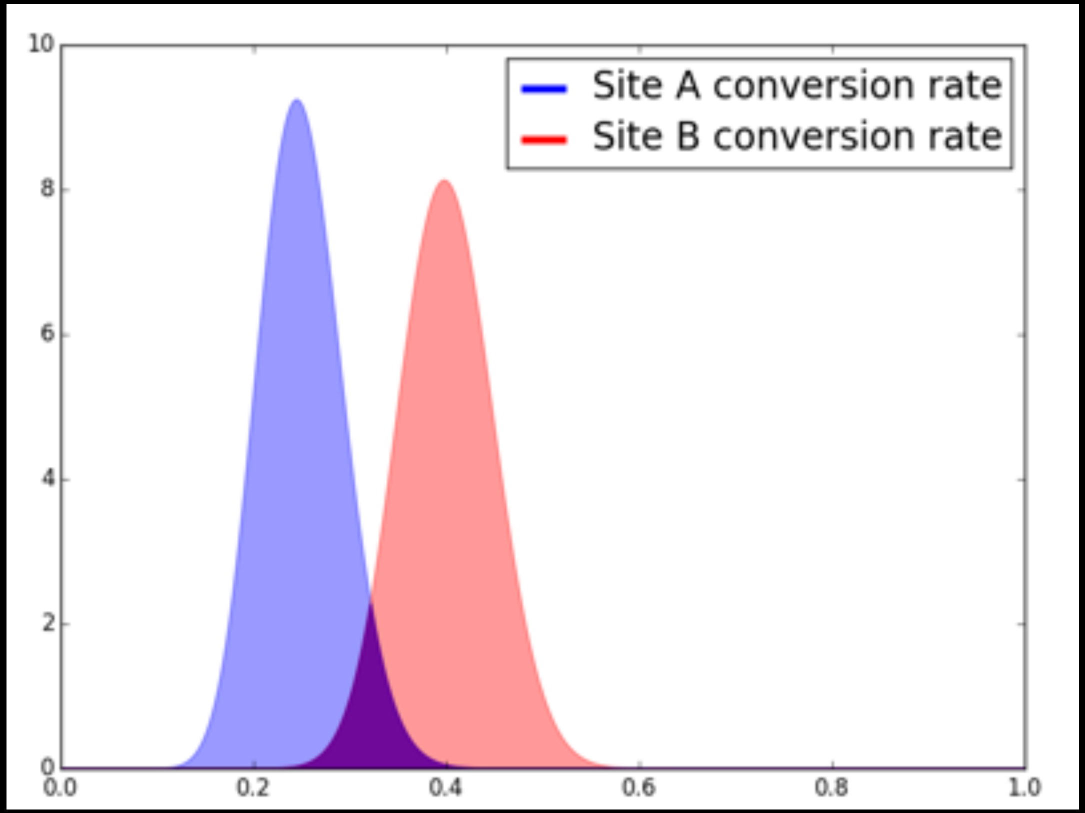
The result is a Beta Distribution with these shape parameters:

$\alpha + k$ and $\boldsymbol{\beta} + n - k$

A/B Testing

- We want to know if this is true:
  conversion rate of site A > conversion rate of site B

- We can also answer if this is true:
  conversion rate of site A > conversion rate of site B + 5%



Method:

- Sample a large number from both distributions

- Count the percent of times site A wins

The code

```
num_samples = 10000

A = np.random.beta(1 + num_clicks_A,
                   1 + num_views_A - num_clicks_A,
                   size=num_samples)

B = np.random.beta(1 + num_clicks_B,
                   1 + num_views_B - num_clicks_B,
                   size=num_samples)

### The probability that A wins:
print np.sum(A > B) / float(num_samples)

### The probability that A > B + 0.5%:
print np.sum(A > (B + 0.05)) / float(num_samples)
```

Multi-Arm Bandit

Smarter A/B Testing

Traditional A/B Testing

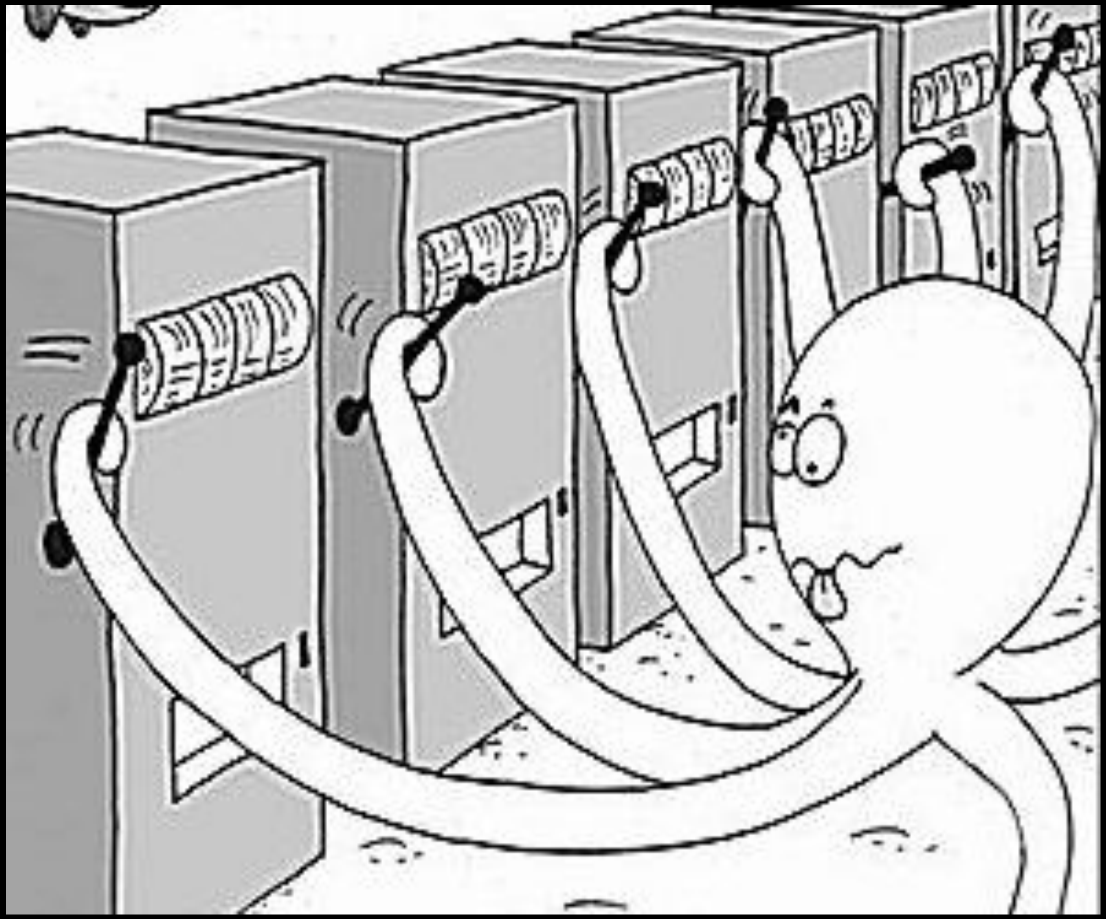
- First: pure *exploration*, in which you assign equal numbers of users to Group A and Group B
- Second: pure *exploitation*, in which you stop the experiment and send all your users to the more successful version of your site.

Multi-Arm Bandit

- Start *exploiting* the likely best solution *before* you're done *exploring*

- versions of the site
  = bandits (slot machines)

- How to pick which one to play….



Regret

- To evaluate a multi-arm bandit algorithm, *minimize* regret

- Regret is a measure of how often you chose a suboptimal bandit

$$\text{regret} = \sum_{i=i}^{k} (p_{\text{opt}} - p_i)$$
$$= k \cdot p_{\text{opt}} - \sum_{i=1}^{k} p_i$$

Epsilon-Greedy Algorithm

- Explore with probability epsilon (often 10%)
- Exploit the rest of the time (play the bandit with the best performance so far)

UCB1 Algorithm (Upper Confidence Bound)

- Choose the bandit where this value is the largest

$$p_A + \sqrt{\frac{2 \log N}{n_A}}$$

where $n_A =$ number of times bandit A has been played

and $N =$ total number of times any bandit has been played

Softmax Algorithm

- Choose the bandit randomly in proportion to its estimated value:

$$\frac{e^{p_A/\tau}}{e^{p_A/\tau} + e^{p_B/\tau} + e^{p_C/\tau}}$$

Bayesian Bandit Algorithm

- Model each of the bandits with a beta distribution with shape parameters:

  α = 1 + number of times bandit has won

  $\boldsymbol{\beta}$ = 1 + number of times bandit has lost

- Take a random sample from each bandit's distribution and choose the bandit with the highest value.

Bayesian Bandit Simulation
3 bandits, payouts of [0.05,0.1,0.2]