Bayesian A/B Testing

Beta Distribution and Multi-Arm Bandit

A/B Testing
(frequentist)

- define a metric
- 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 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 test while it is running?

A/B Testing
(Bayesian)

- define a metric
- run test, continually monitor results
- at any time calculate probability that A > B or vice versa
- suggest course of action based on probabilities calculated

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 test while it is running?

Bayes Theorem

$$\overset{\text{Posterior Probability}}{\Pr(\theta|y)} = \frac{\overset{\text{Likelihood of Observations}}{\Pr(y|\theta)}\,\overset{\text{Prior Probability}}{\Pr(\theta)}}{\underset{\text{Normalizing Constant}}{\Pr(y)}}$$

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

Bayes Theorem

posterior $\propto$ prior x likelihood

likelihood

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

prior

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

- p: conversion rate (between 0 and 1)
- α,β: shape parameters
  - α = 1 + number of conversions
  - β = 1+ number of non conversions
- beta function (B) is for normalization
- α = β = 1 gives the uniform distribution

Conjugate Priors

posterior ∝ prior x likelihood

beta ∝ beta x binomial

Conjugate Priors

posterior ∝ prior × likelihood

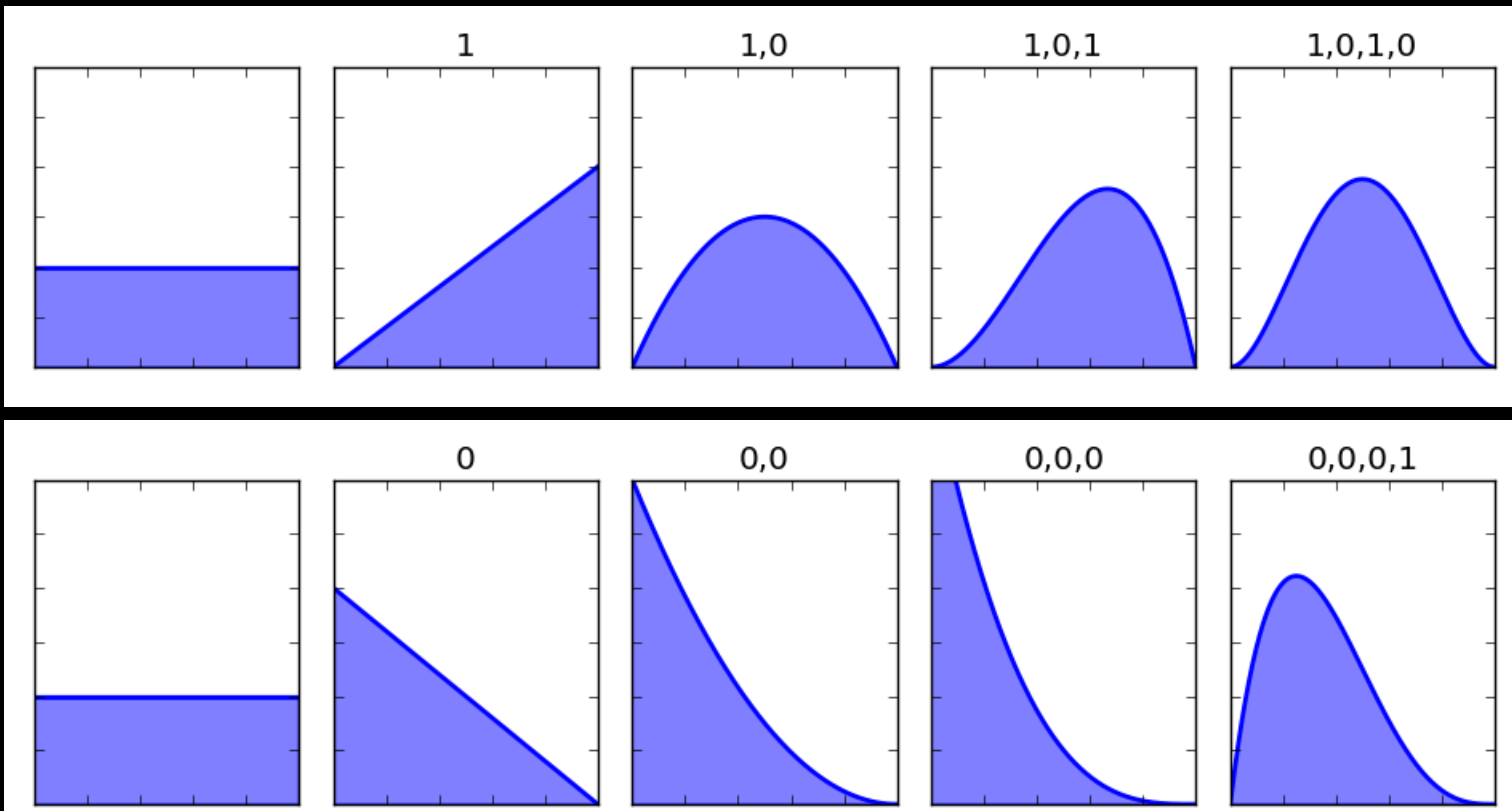
beta ∝ beta × binomial

THE MATH:

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

result is a beta distribution with shape parameters:
α + k and β + n - k

evolution of the posterior

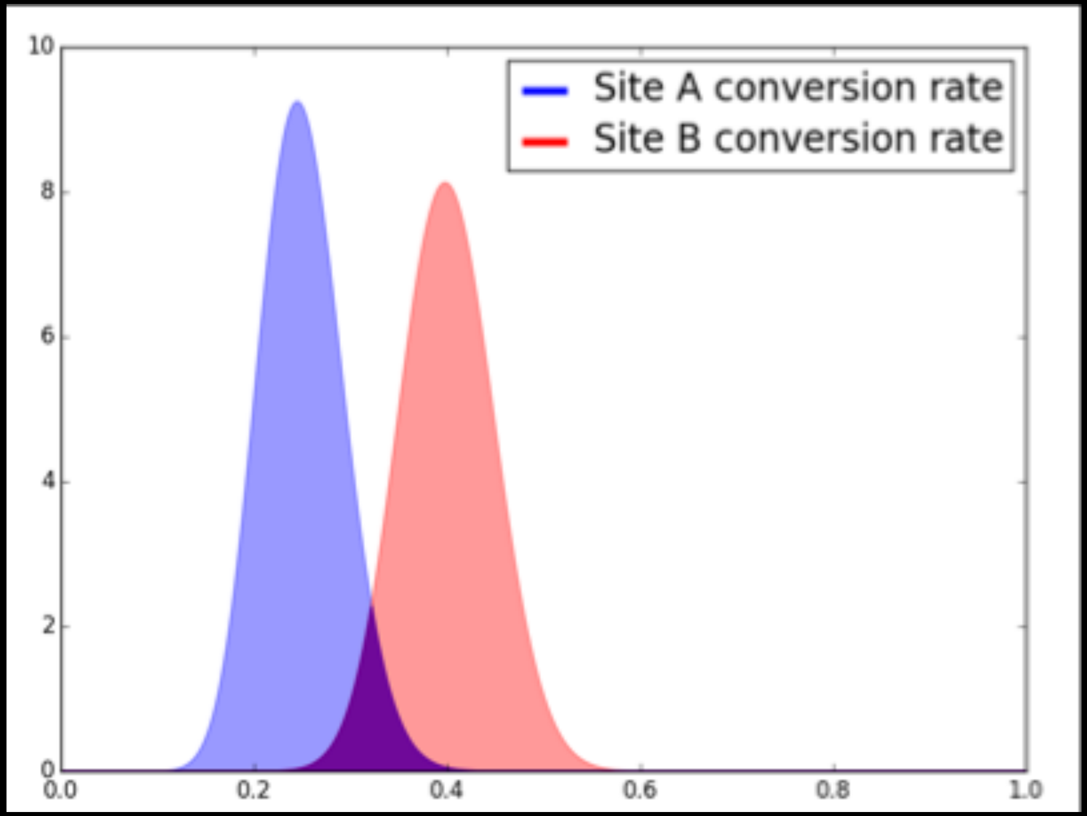


1 = conversion

0 = non conversion

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 repeatedly from each posterior distribution
- count how often site A has a high rate than site B

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)
```

to sum up

- the bayesian framework has signifiant advantages over the frequentist in the context of A/B testing
  - can answer the questions we want to ask
  - allows for more flexibility in the testing framework

- beta distributions are a natural fit for A/B testing
  - conjugate to both Bernoulli and binomial distributions
  - maps naturally onto a probability

Multi-Arm Bandit

Smarter A/B Testing

boring A/B testing

- to start, use pure exploration, assign equal numbers to each group
- then: pure exploitation, stop the experiment and send all users too more successful version of the site

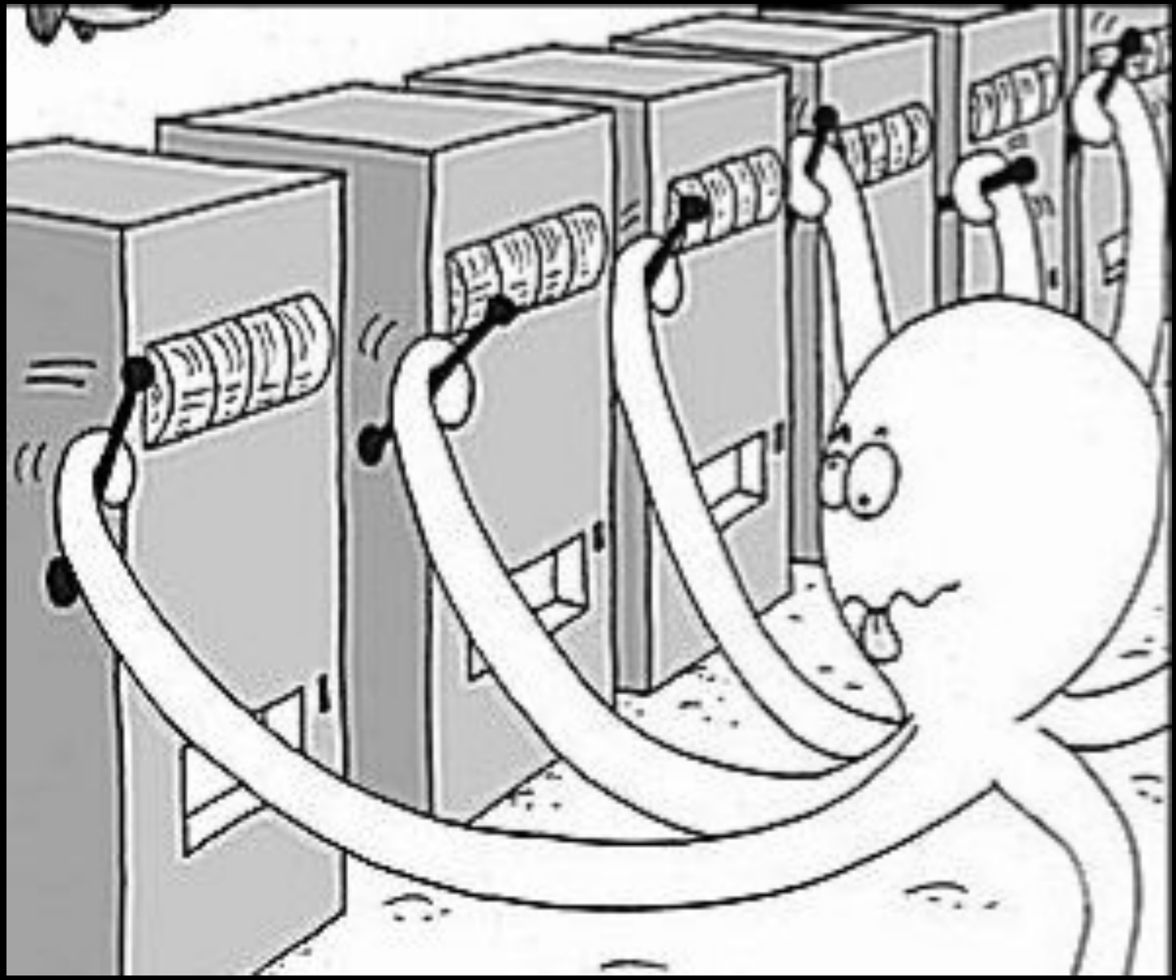
boring A/B testing

- but, every user you send to the worse site during the test is lost money
- can we minimize our loses during the exploration phase?

enter the bandits

- start exploiting the likely best solution before ending the exploration phase

- each version of the site is a bandit
- how do we pick which bandit to play



regret

- we can evaluate the mulit-armed bandit by thinking about **regret**
- we want to minimize the time we spend on non-optimum payouts

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

regret

- regret lets us evaluate in the abstract the behavior of bandit algorithms
- it is not generally something you calculate as it requires knowledge of the true probability

epsilon-greedy

- explore with a fixed probability
  - epsilon
  - often 10% or less
- if you do not explore, play the bandit with the best performance seen so far

ucb1

- calculate a probability for each bandit
- use the bandit with the maximum of the following equation

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

where n_A = number of times bandit A has been played

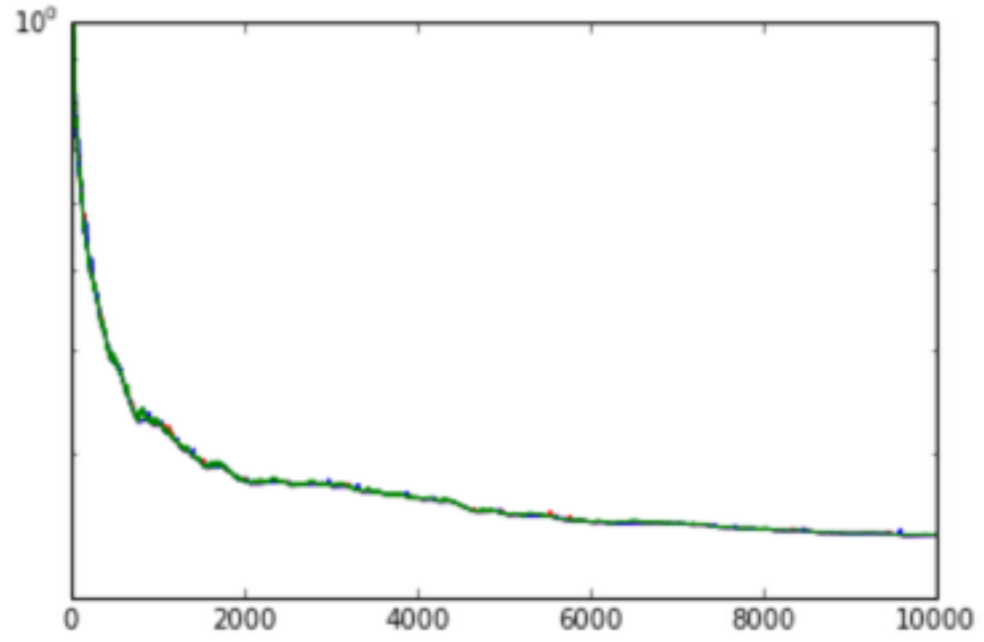
and N = total number of times any bandit has been played

simulation of ucb1 bandits payouts are [0.05, 0.1, 0.2]

In [62]:
```
probs = [0.05,0.1,0.2]
played_bandits = play_bandits(bandits,probs)
colors = ['r','b','g']
for i,bandit in enumerate(played_bandits):
    plt.plot(bandit['ucb'],color=colors[i])
    print i,len(bandit['plays']),max(bandit['plays'])
plt.yscale('log')
plt.ylim(.2,1)
```

```
0 528 9633
1 707 9639
2 8765 9999
```

Out[62]: (0.2, 1)



softmax

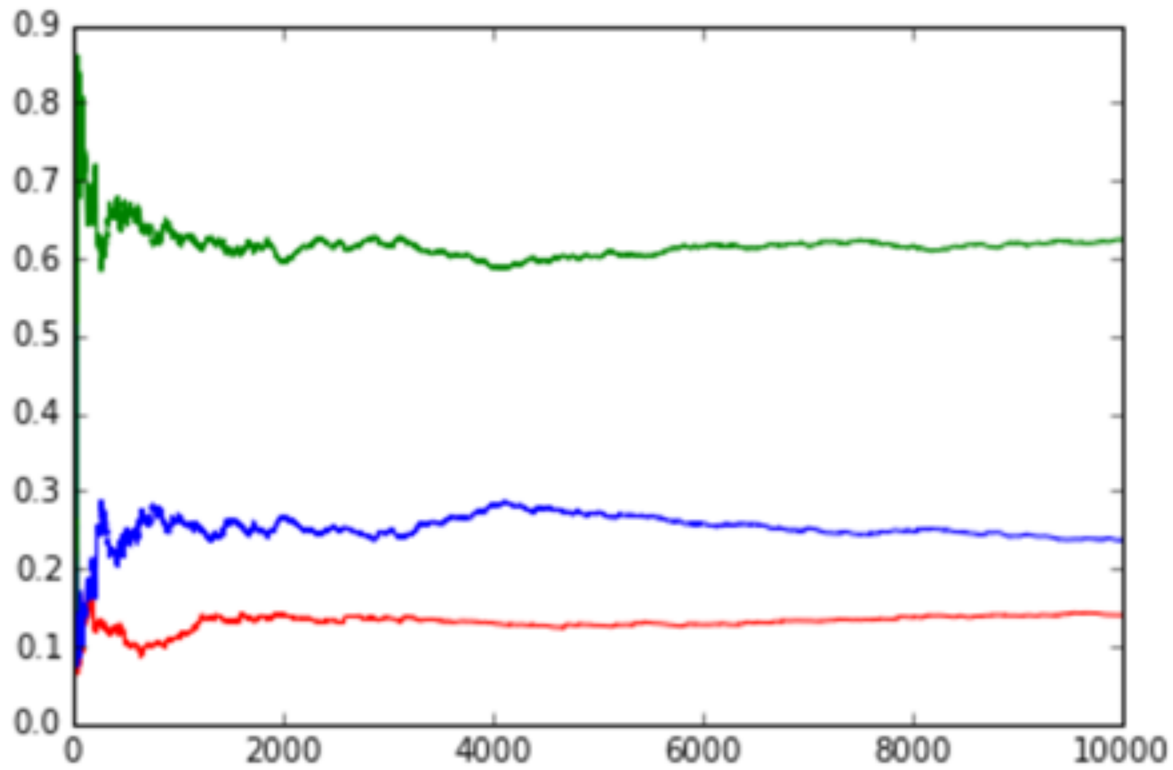
- choose the bandit random in proportion to its estimated value:

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

simulation of softmax bandits payouts are [0.05, 0.1, 0.2]

In [101]:
```
probs = [0.05,0.1,0.2]
played_bandits = play_bandits(bandits,probs)
colors = ['r','b','g']
for i,bandit in enumerate(played_bandits):
    print i, len(bandit['plays']), max(bandit['plays'])
    plt.plot(bandit['softmax'],color=colors[i])
```

```
0 1342 9999
1 2460 9996
2 6198 9998
```

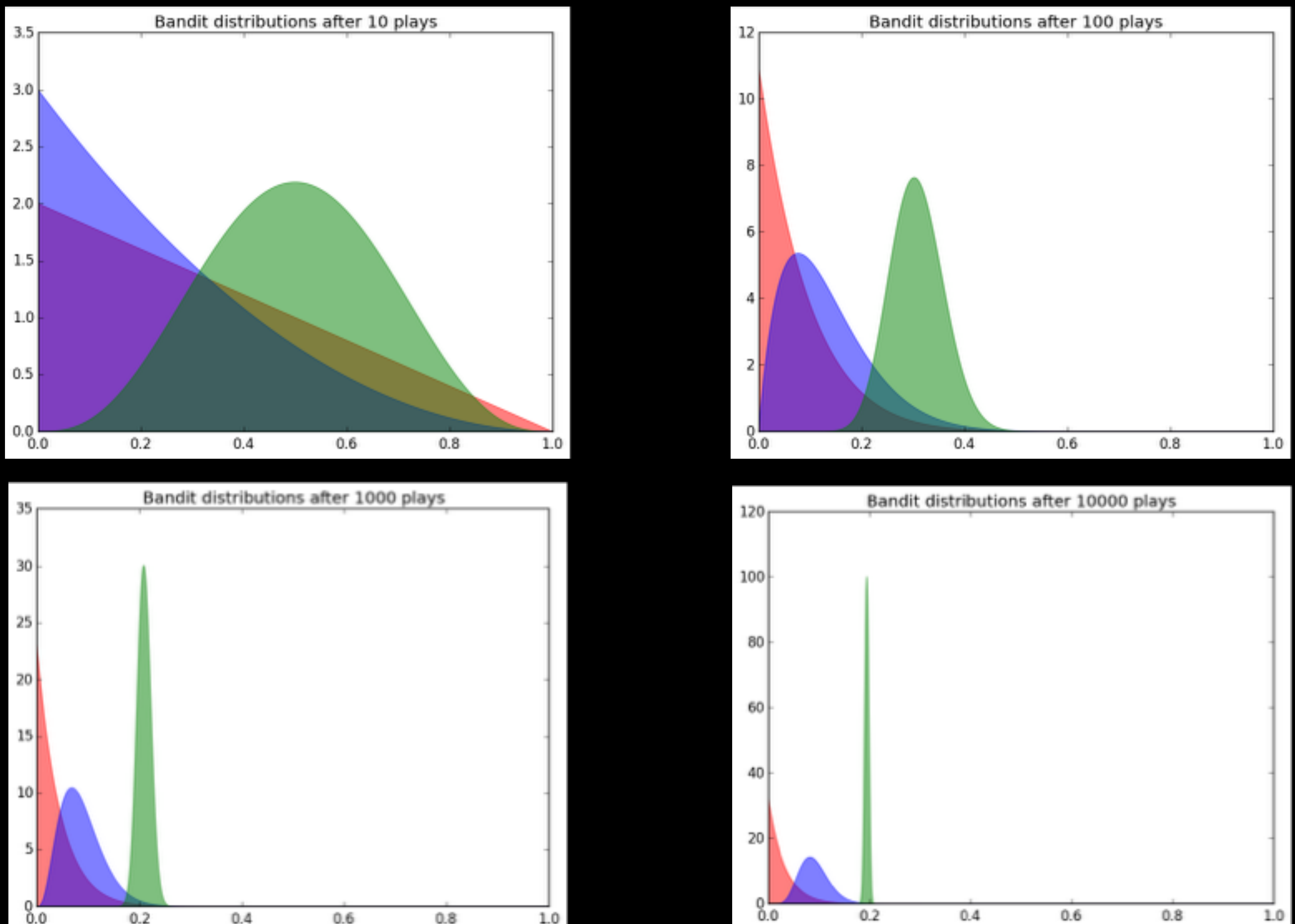


Bayesian bandit

- use bayesian updates to model the bandits
- each bandit has an associated beta distribution with parameters:
  α = 1 + number of times bandit has won
  β = 1 + number of times bandit has lost

- sample from each bandit's distribution and play the bandit with the highest value
- will naturally converge on the bandit with the best payout

simulations of bayesian bandits payouts are [0.05, 0.1, 0.2]



to sum up

- mulit-armed bandit problem maps onto A/B testing naturally
  - can easily expand to testing many things simultaneously
  - can tune exploitation versus exploration
  - can combine with bayesian methods in a natural way

- there are many solutions to the multi-armed bandit problem
  - depend on how you want to make the trade offs
  - if payoffs change over time interpretation of bandits regret changes
  - will play around with different simulations during the sprint