Experimental optimization

Lecture 8: Multi-armed bandits II: Thompson sampling

David Sweet

Review

Versions / arms

- Compare “version A” to “version B”, or
- Compare “arm A” to “arm B”
- Also could compare arms A, B, C, D, …
- Examples
  - A = old ML model, B = new ML model with more features
  - A = place ad on top, B = place ad on side, C = overlay ad
  - A = low threshold, B = medium threshold, C = high threshold

Review

A/B test vs. epsilon-greedy

- Goal: Choose arm/version with highest expected (true) business metric.
- A/B test: Randomize 50/50 between A & B, N times.
- Epsilon-greedy: Randomize 90% to the arm that's "better so far" and ε=10% to the other arm.
  - Decay ε and stop when ε is very small.
  - $\varepsilon_n = \dfrac{kc(BM_0/PS)^2}{n}$

Review

Meta-parameters

- A/B test: Choose FPR, FNR limits (5% and 20%)
- Epsilon-greedy: Choose a value for c, the meta-parameter and a threshold telling us when to stop (when eps is small enough)
- *meta-parameters* determine how the experimental method operates
- Contrast with *parameters* which determine how the engineered system operates
- Prefer not to have to tune meta-parameters since that would require many experiments (a “meta-experiment”?)

Randomization

Epsilon-greedy modifies randomization

- Given n_a individual measurements of A, n_b ind. meas. of B
- A/B: Run A or B with equal probability
- Epsilon-greedy: $\mu_a = \dfrac{\sum a_i}{n_a}, \mu_b = \dfrac{\sum b_i}{n_b}$
  - 90%: If $\mu_a > \mu_b$, run A, else run B
  - 10%: Run A or B with equal probability

Thompson sampling

Also modifies randomization

- Given: n_a individual measurements of A, n_b ind. meas. of B
- Sample from ind. meas. *with replacement*: $\tilde{a}_i$, $\tilde{b}_i$

"Bootstrap"

- $\tilde{\mu}_a = \frac{\sum \tilde{a}_i}{n_a}, \tilde{\mu}_b = \frac{\sum \tilde{b}_i}{n_b}$
- 100%: If $\tilde{\mu}_a > \tilde{\mu}_b$, run A, else run B

Compare

Two multi-armed bandit solutions

Epsilon-greedy	Thompson sampling
—	Resample ind. meas.
Calculate agg. meas. from ind. meas.	Calculate agg. meas. from *resampled* ind. meas
90%: Choose arm with highest agg.	100%: Choose arm with highest agg.
10%: Choose randomly	—

Randomness generates exploration

Randomness generates exploration

Thompson sampling

Bootstrap sampling

- Sample from ind. meas. *with replacement*: $\tilde{a}_i, \tilde{b}_i$
- Kind of like rerunning the experiment up to this point and getting a new set of individual measurements

```
def bootstrap_sample(x):
    return x[np.random.randint(len(x), size=(len(x),))]
```

```
bootstrap_sample(np.array([1,2,3,4]))
```

```
array([3, 4, 4, 2])
```

```
for _ in range(10):
    print (bootstrap_sample(np.array([1,2,3,4])).mean())
```

```
2.0
2.25
2.0
1.5
2.75
2.25
2.0
1.75
2.0
2.75
```

Thompson sampling
Bootstrap sampling

- Recall:
  - Aggregate measurement, μ_a, is an estimate of the expectation of the individual measurements, a_i, i.e., the true BM
  - CLT says agg. meas. approximates a normal distribution (for large N)
  - Each experiment gives a single aggregate measurement
    - (But I did say bootstrap was *like* running another experiment…)

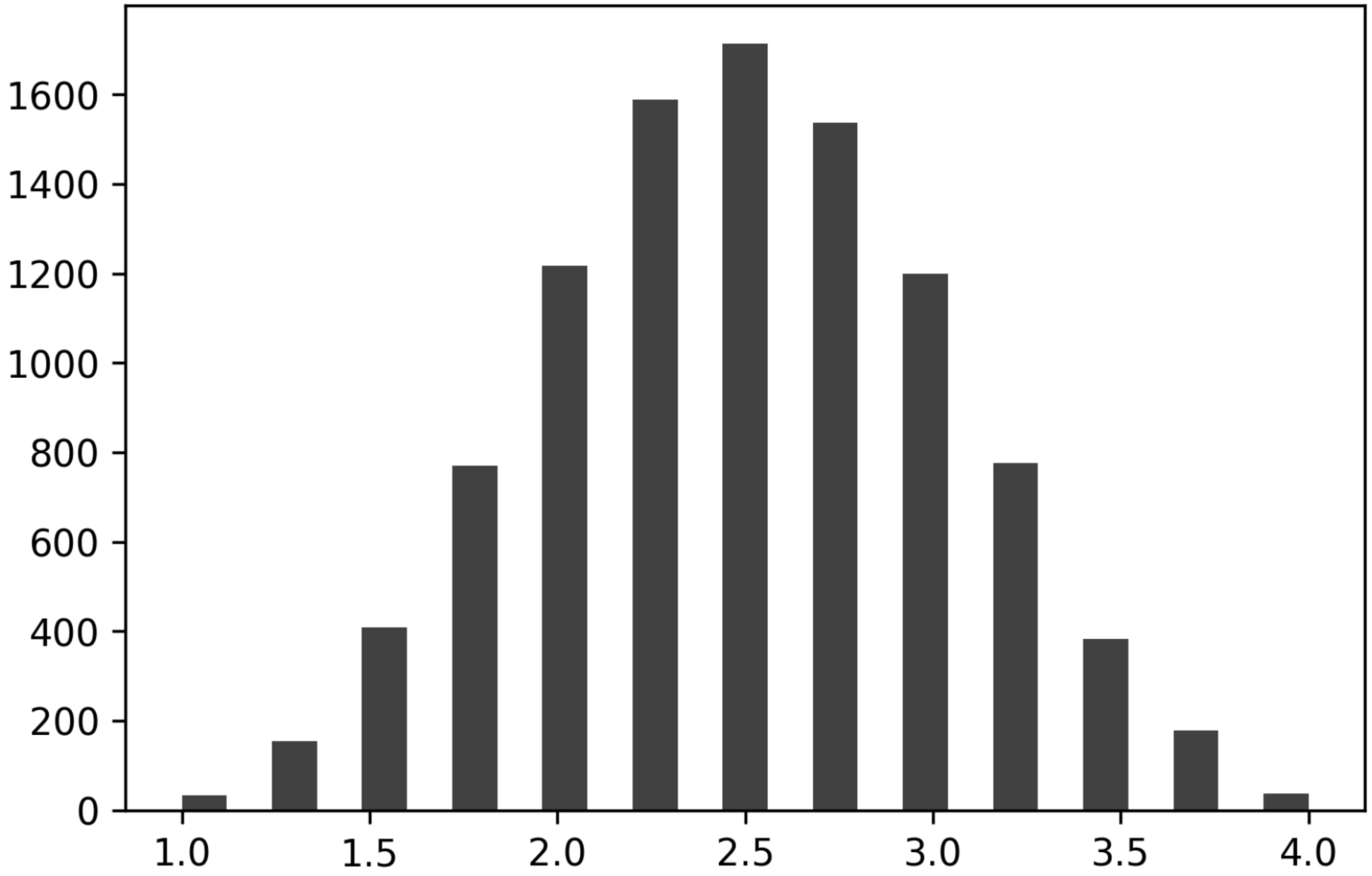
Thompson sampling

Bootstrap sampling

- With B.S. sampling, you can generate many agg. meas., $\tilde{\mu}_a$, from a single set of ind. meas.

- Even for small N

```
agg = np.array([bootstrap_sample(np.array([1,2,3,4])).mean()
                for _ in range(10000)])
plt.hist(agg, 25, color=yu.clr1);
```



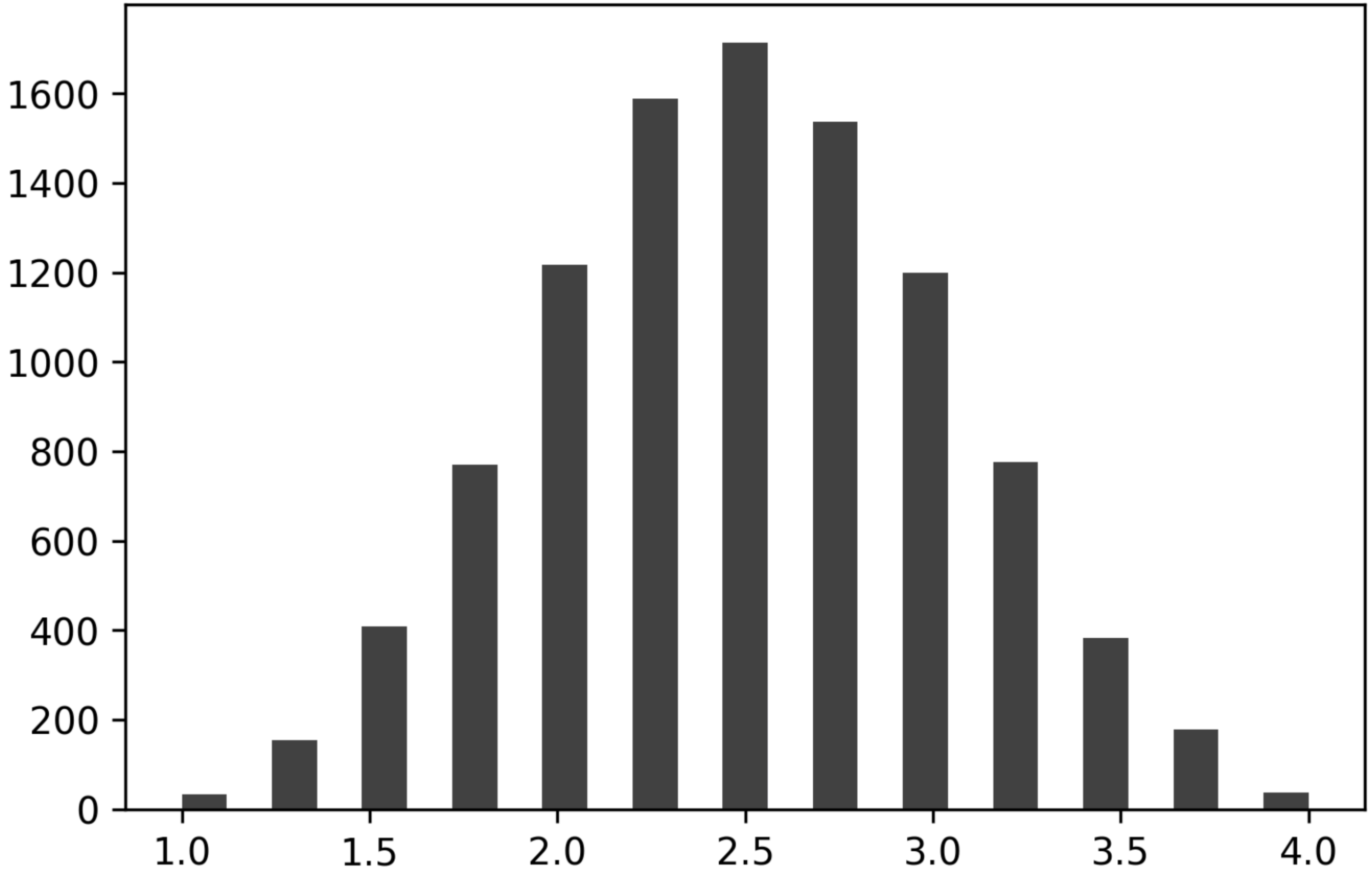
Thompson sampling

Bootstrap sampling

- Each value, $\tilde{\mu}_a$ is an equally-reasonable estimate of BM
- Note: More values near the middle (taller bars), where the true BM is

Do you believe this?

```
agg = np.array([bootstrap_sample(np.array([1,2,3,4])).mean()
                for _ in range(10000)])
plt.hist(agg, 25, color=yu.clr1);
```



Thompson sampling

Bootstrap sampling

- Decision rule: If $\tilde{\mu}_a > \tilde{\mu}_b$, run A.
  - Based on a single bootstrap sample.
  - Different bootstrap sample ==> different decision.
- Think: $P\{\tilde{\mu}_a > \tilde{\mu}_b\}$ == P{I run A}
  - OR, P{A is better} == P{I run A}
  - Calc many B.S. samples and check $\tilde{\mu}_a > \tilde{\mu}_b$ each time.
- The fraction of time that $\tilde{\mu}_a > \tilde{\mu}_b$ can be thought of as the probability that A is better than B **as far as I can tell** from my individual measurements.

"belief"

Thompson sampling

Randomized probability matching

- The rule: “Run A if $\tilde{\mu}_a > \tilde{\mu}_b$”
  - Randomize to run A in proportion to the probability that A is better than B.
- For multiple arms, “Run arm k if $\tilde{\mu}_k = \max\{\tilde{\mu}_{k'}\}$”
  - Randomize to run arm k in proportion to the probability that arm k is the best arm.
- “probability matching”: P{running an arm} = P{arm is best}

Thompson sampling

Exploration vs. exploitation

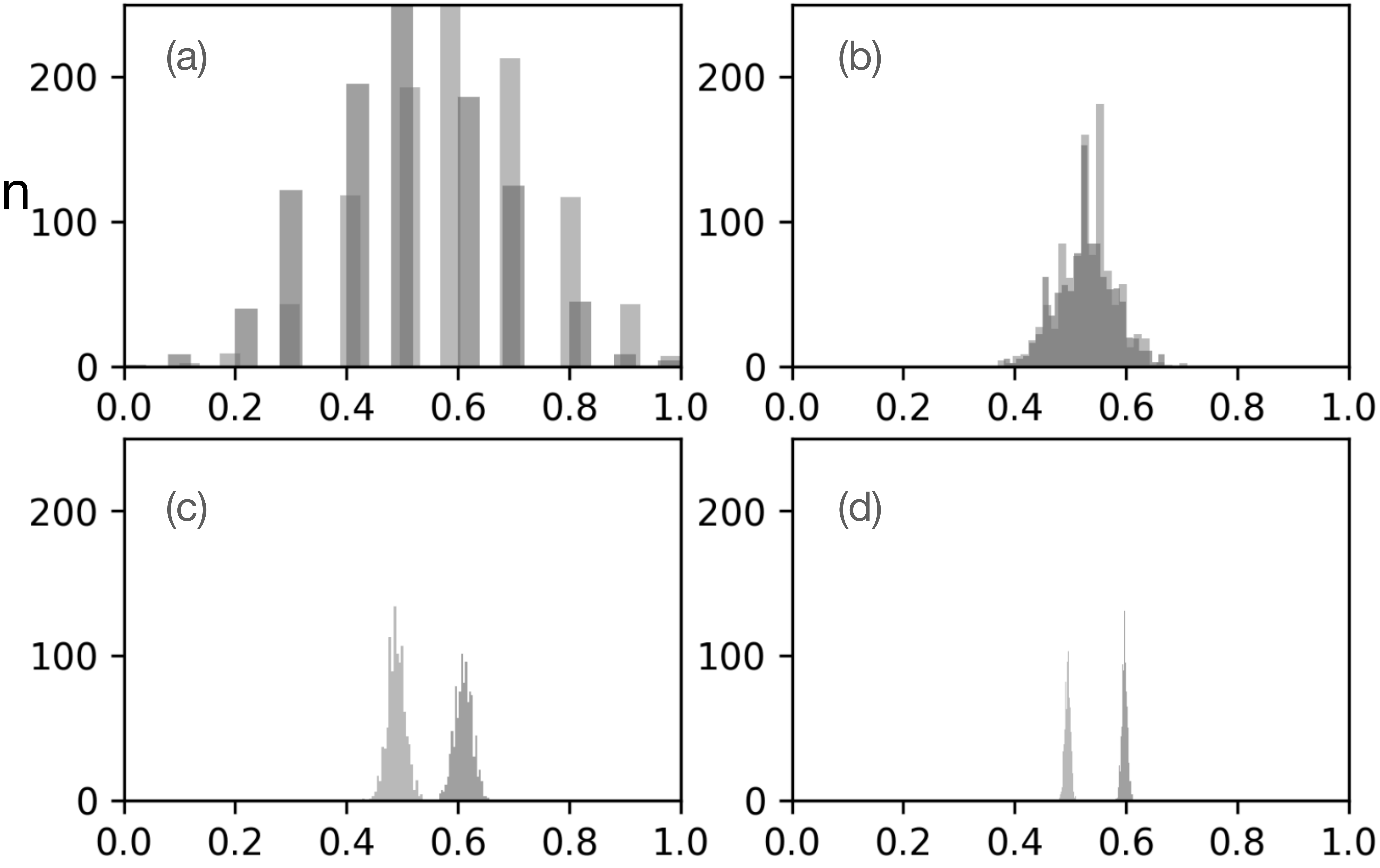
- Exploration: Always allocating some individual measurements to worse arms
- Exploitation: Allocating more individual measurements to better arms
- P{arm k is best} equal for all k at the start, then differentiates as more individual measurements are collected
- Stop when highest P{arm k is best} $>$ 1.0 - threshold
- No meta-parameters besides threshold

Thompson sampling

Exploration vs. exploitation

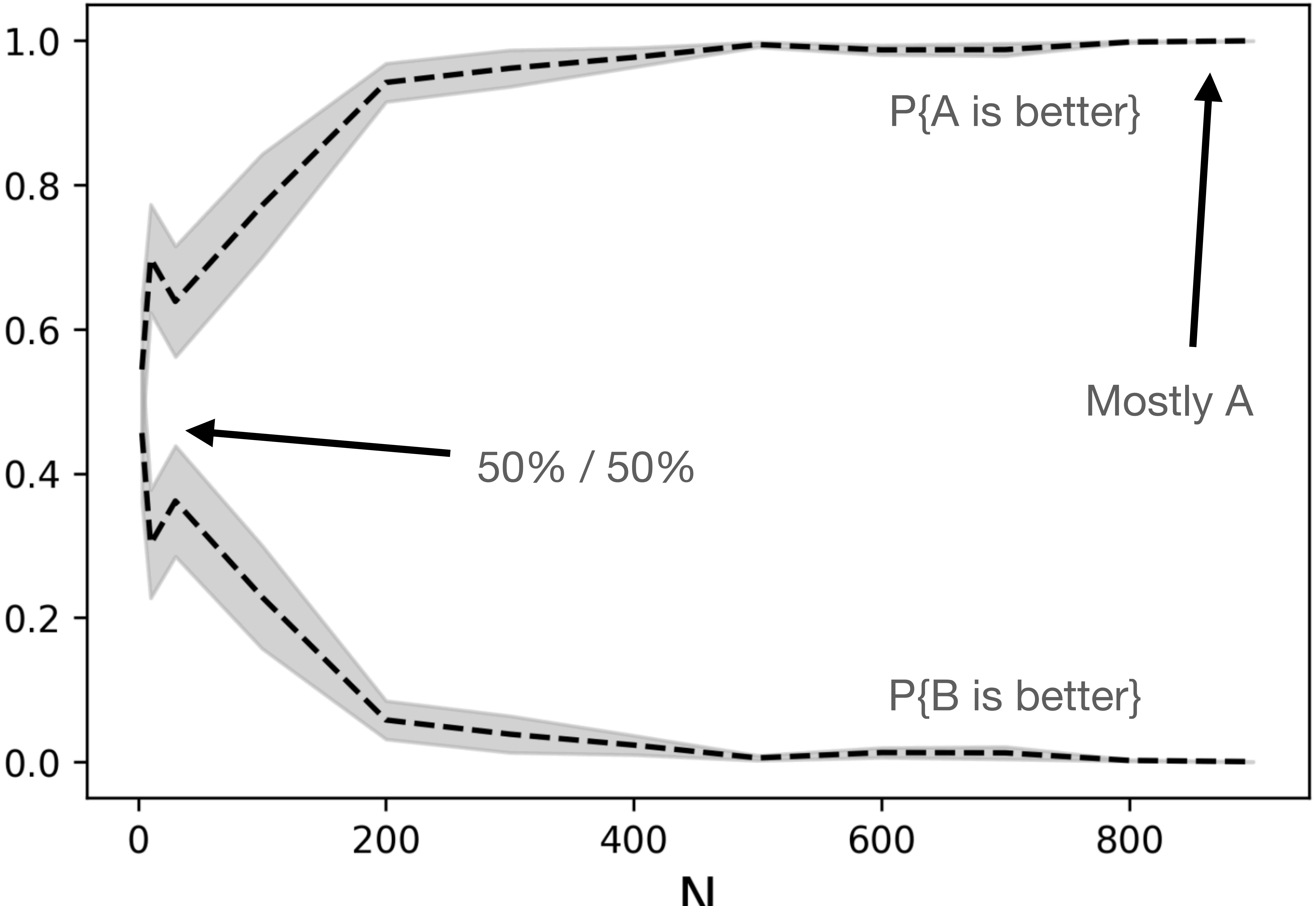
- Gets easier to tell the distributions apart as more ind. meas. are taken

- (a) —> (d) increasing N



Thompson sampling
Stopping

- Estimate P{A is better} by comparing many bootstrap sample means
- P{A is better} = [count of “A is better”] / [total number compared]



Thompson sampling

Compared to epsilon-greedy

- [pro] Thompson sampling does not require you to choose the meta-parameter c
- [con] Thompson sampling is more complex than epsilon-greedy because it needs all of the individual measurements available to make a randomization decision.
    - Chapter 3 discusses a practical solution to this problem

Thompson sampling
Summary

- Randomize like this:
  - Create bootstrap mean for each arm, $\tilde{\mu}_k$
  - Run arm k if $\tilde{\mu}_k = \max\{\tilde{\mu}_{k'}\}$
- Equivalent to randomized probability matching:
  - P{run arm k} = P{arm k has the highest BM}
- Stop when the the highest P{arm k has the highest BM} > 1.0 - threshold