# Introduction to Adaptative Experimental Design

Mentor: Zhaoqi Li

Mentee: Zilin Huang

SPA Fall 2022

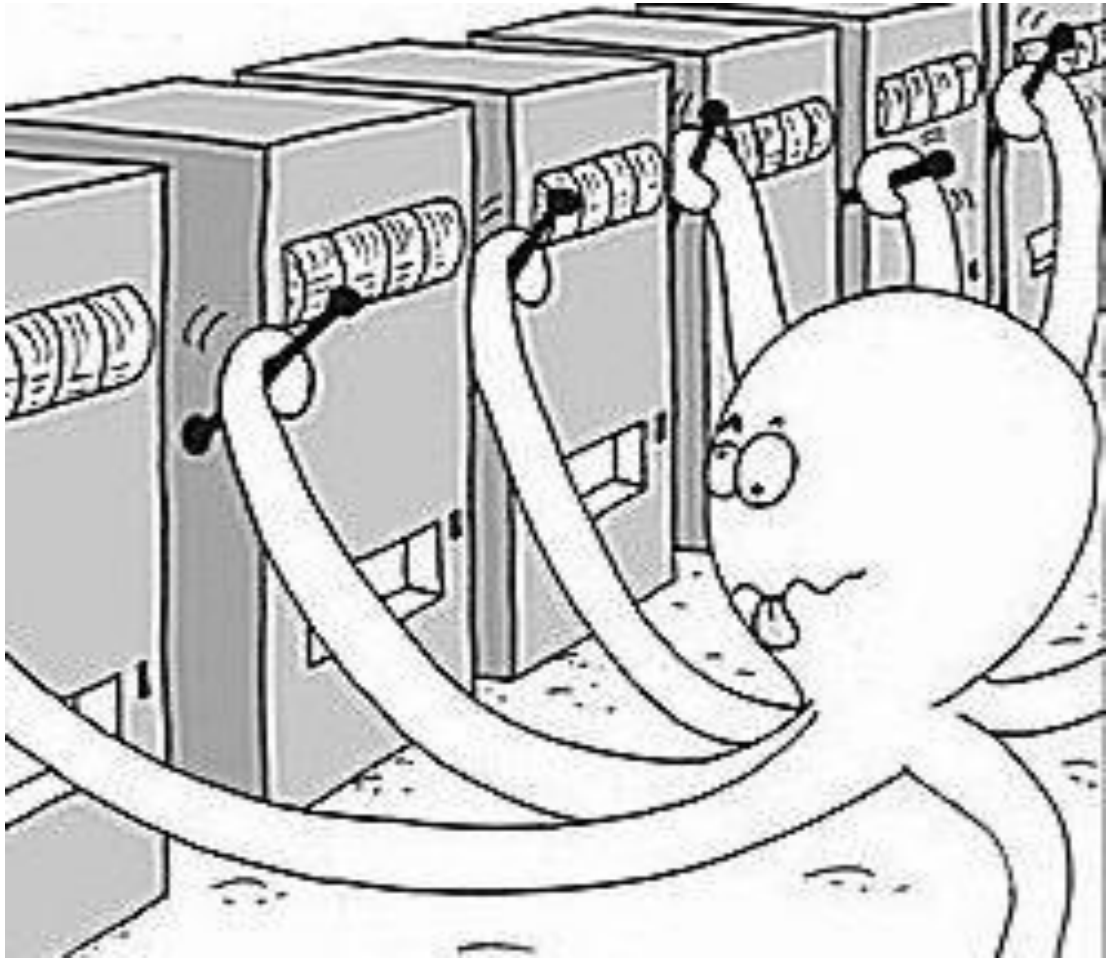
University of Washington

# Motivation

- (Ideal Case) Make as much money during gambling as possible
- No opportunity cost, no loss of money
- Probability of winning different amount of money from different gamble machines vary a lot

# What is Stochastic Bandit?

- Essence: A set of probability distributions ("bandit arms")
- Actions and Rewards
- Ex. Bernoulli Bandit: the simplest case



# Regret of Stochastic Bandit

- Deficiencies between optimal and practical strategy
- Want it to be as small as possible (mean reward as large as possible)
- Suboptimality Gap
- Sum up by rounds
- Sum up by actions?

$$R_n = nu^* - E[\sum_{t=1}^{n} X_t]$$

*($n$ : total number of rounds, $u^*$: largest reward of the "optimal" arm during each round,*
*$X_t$: actual reward during each round)*

# Policy of Stochastic Bandit: Explore-Then-Commit (ETC) Algorithm

- Explore first (play with each of the k rounds for m times)
- Commit next (play with the arm with the largest mean reward only)
- Regret: subject to linear growth
- Ex. Randomly guess makes linear regret occur

1: **Input** $m$.

2: In round $t$ choose action

$$A_t = \begin{cases} (t \bmod k) + 1\,, & \text{if } t \le mk\,; \\ \operatorname{argmax}_i \hat{\mu}_i(mk)\,, & t > mk\,. \end{cases}$$

(ties in the argmax are broken arbitrarily)

**Algorithm 1:** Explore-then-commit.

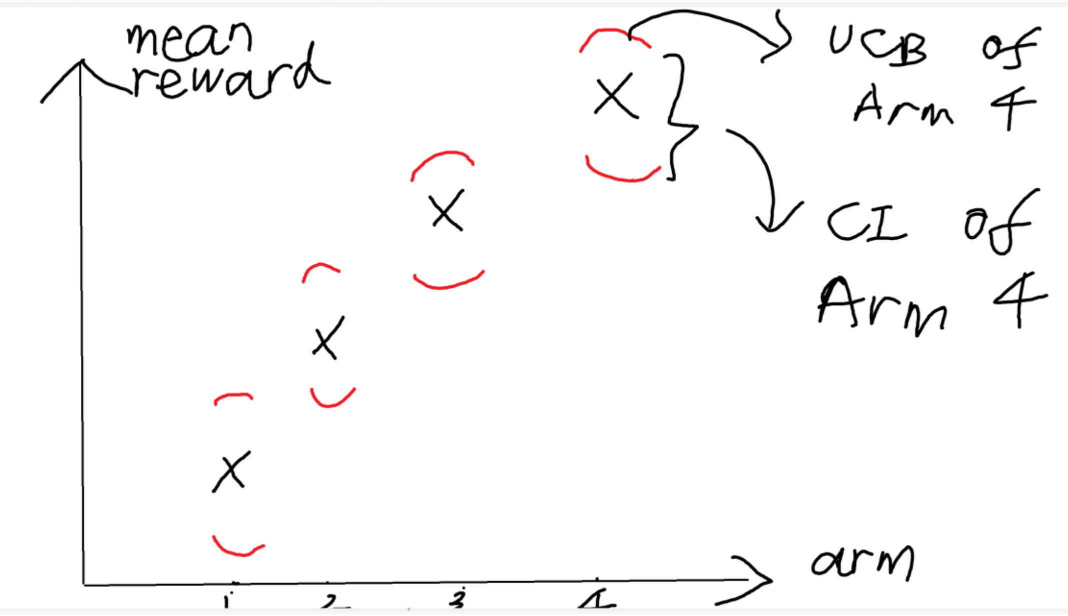
*($m$ : rounds played by each arm during "exploring", $k$: number of arms, $u_i(n)$ : actual mean reward of arm $i$ after $n$ rounds)*

# Policy of Stochastic Bandit: Upper Confidence Bound (UCB) Algorithm

- Define a "UCB" index for each arm
- Play the arm with the largest "UCB"
- Update this arm's "UCB" based on generated rewards
- Regret: subject to sublinear growth
- Bounded by "Good Events" (true value inside Confidence Interval)
- Best for minimizing the overall regret

*(t : current tth round, δ: boundary of Confidence Interval,*
*$T_i(n)$ : total number of rounds (Random Variable)*
*$u_i(n)$ : actual mean reward of arm $i$ after $n$ rounds)*

$$\mathrm{UCB}_i(t-1,\delta) = \begin{cases} \infty & \text{if } T_i(t-1) = 0 \\ \hat{\mu}_i(t-1) + \sqrt{\frac{2\log(1/\delta)}{T_i(t-1)}} & \text{otherwise}. \end{cases}$$



# Policy of Stochastic Bandit: Elimination Algorithm

- Each round represents an updated environment with varied number of arms
- Eliminate the arms whose mean reward has "too large" difference with the optimal arm
- Regret: stick to playing with one arm and calculate the accumulated regret
- Best for identifying the best arm

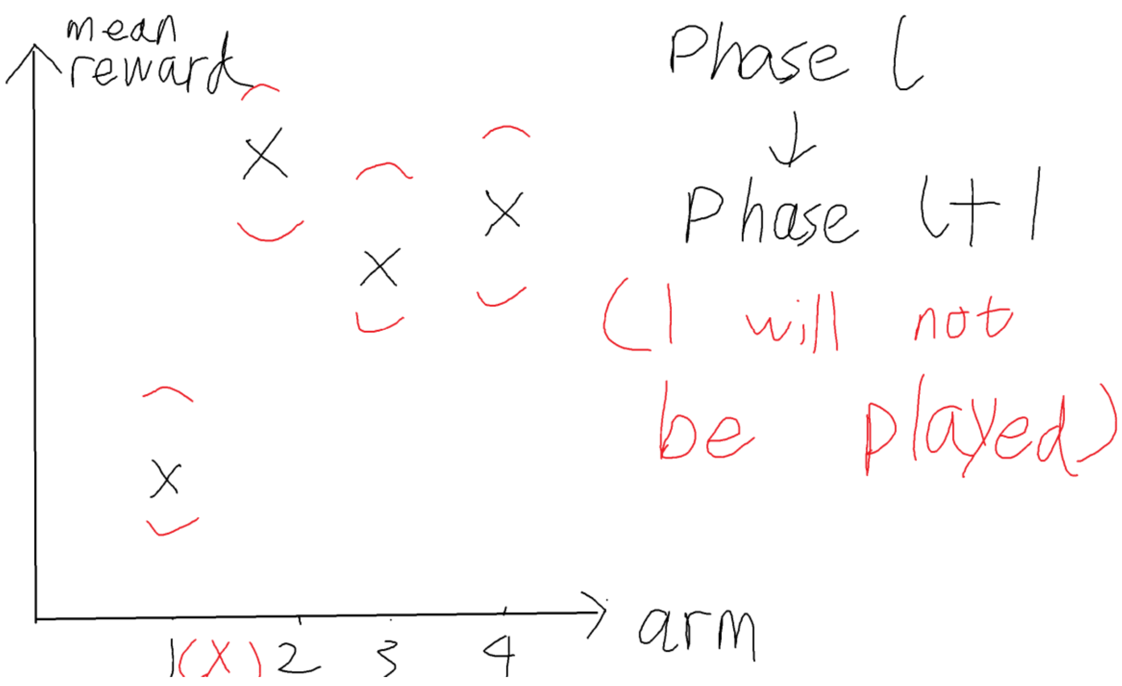
*($l$ : current $l$th phase, $2^{-l}$: defined index of Confidence Interval $u_{i,l}$: actual mean reward of arm $i$ in phase $l$)*

**Input:** $k$ and sequence $(m_\ell)_\ell$
$A_1 = \{1, 2, \ldots, k\}$
**for** $\ell = 1, 2, 3, \ldots$ **do**
  Choose each arm $i \in A_\ell$ exactly $m_\ell$ times
  Let $\hat{\mu}_{i,\ell}$ be the average reward for arm $i$ from this phase only
  Update active set:

$$A_{\ell+1} = \left\{ i : \hat{\mu}_{i,\ell} + 2^{-\ell} \geq \max_{j \in A_\ell} \hat{\mu}_{j,\ell} \right\}$$

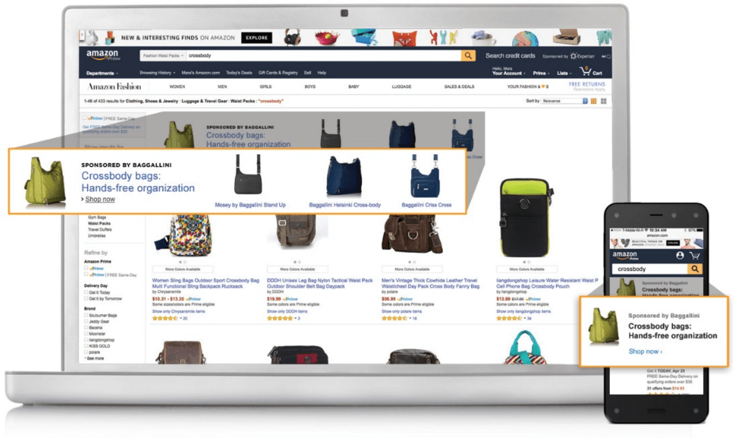
**end for**

**Algorithm 2:** Phased elimination for finite-armed bandits



# Policy of Stochastic Bandit: Thompson Sampling Algorithm

- Each arm's mean is a probability distribution function (pdf), instead of a fixed number
- Extract samples from each arm, which infers another pdf
- P("best arm") RV = P("sample arm")
- These pdf are used to estimate the true pdf of each arm's mean
- Regret: follow Bayesian setting
- Application: Amazon front-page algorithm recommendation



# Implementation in Python (of Bernoulli Bandits)

```python
class BernoulliBandit:
    def __int__(self, means, K, round):
        self.means = means
        self.K = K
        self.round = round

    def pull(self, a):
        # Pull Once Each Time:
        realisation = bernoulli.rvs(sum(self.means[0:a+1])/(a+1), size=1)
        return realisation[0]

    def regret(self, realisation, rounds):
        # Optimal Rewards:
        u_opt = sum(self.means) / self.K
        # Simulate the Learner's Gained Rewards:
        regret = self.round * u_opt - self.expected_value(realisation, rounds)
        return regret

    def expected_value(self, values, weights):
        values = np.asarray(values)
        weights = np.asarray(weights)
        return (logsumexp(values) * logsumexp(weights)).sum() / logsumexp(weights).sum()
```

BernoulliBandit › regret()

```
round 1 's result: arm  2 generates 0
round 2 's result: arm  2 generates 0
round 3 's result: arm  1 generates 1
round 4 's result: arm  1 generates 0
round 5 's result: arm  1 generates 1
round 6 's result: arm  2 generates 0
round 7 's result: arm  1 generates 0
round 8 's result: arm  2 generates 0
round 9 's result: arm  2 generates 1
round 10 's result: arm  1 generates 0
```

# Implementation in Python (of UCB Algorithm)

```
C:\Users\huang\PycharmProjects\venv\Scripts\python.e
in the 1 th round, the 9 th arm is being played
in the 2 th round, the 1 th arm is being played
in the 3 th round, the 5 th arm is being played
in the 4 th round, the 17 th arm is being played
in the 5 th round, the 11 th arm is being played
in the 6 th round, the 13 th arm is being played
in the 7 th round, the 15 th arm is being played
in the 8 th round, the 16 th arm is being played
in the 9 th round, the 18 th arm is being played
in the 10 th round, the 20 th arm is being played
in the 11 th round, the 4 th arm is being played
in the 12 th round, the 12 th arm is being played
in the 13 th round, the 2 th arm is being played
in the 14 th round, the 6 th arm is being played
[0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1]
```

```python
def Berkely(K, delta, mean):
    previous_round = 10
    round = 150
    UCB = []
    time = []
    for_regret = []
    playtime = [0] * K
    for i in range(0, K):
        time.append(math.sqrt(2 * math.log(1 / delta) / previous_round))
    for j in range(0, K):
        UCB.append(mean[j] + time[j])
```

# References

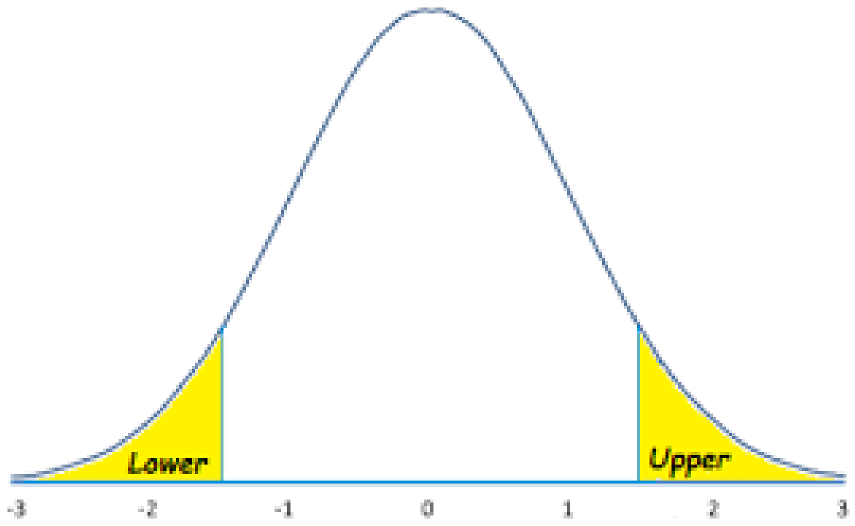
Lattimore, T., & Szepesvári Csaba. (2020). Bandit Algorithms. Cambridge University Press.

Kevin Jamieson. (2021). Some Notes on Multi-armed Bandits. University of Washington.

# Thank you for watching

# Any Questions?

# Theory of Stochastic Bandit: Tail Probabilities




- Difference between sample mean and empirical mean

$$\mathbb{P}(\hat{\mu} \geq \mu + \varepsilon) \quad \text{and} \quad \mathbb{P}(\hat{\mu} \leq \mu - \varepsilon) \,.$$

- Bounded upon Subgaussian environment

$$\mathbb{P}(\hat{\mu} \geq \mu + \varepsilon) \leq \exp\left(-\frac{n\varepsilon^2}{2\sigma^2}\right) \quad \textit{and} \quad \mathbb{P}(\hat{\mu} \leq \mu - \varepsilon) \leq \exp\left(-\frac{n\varepsilon^2}{2\sigma^2}\right) ,$$

- sample mean and empirical mean differs by a small amount

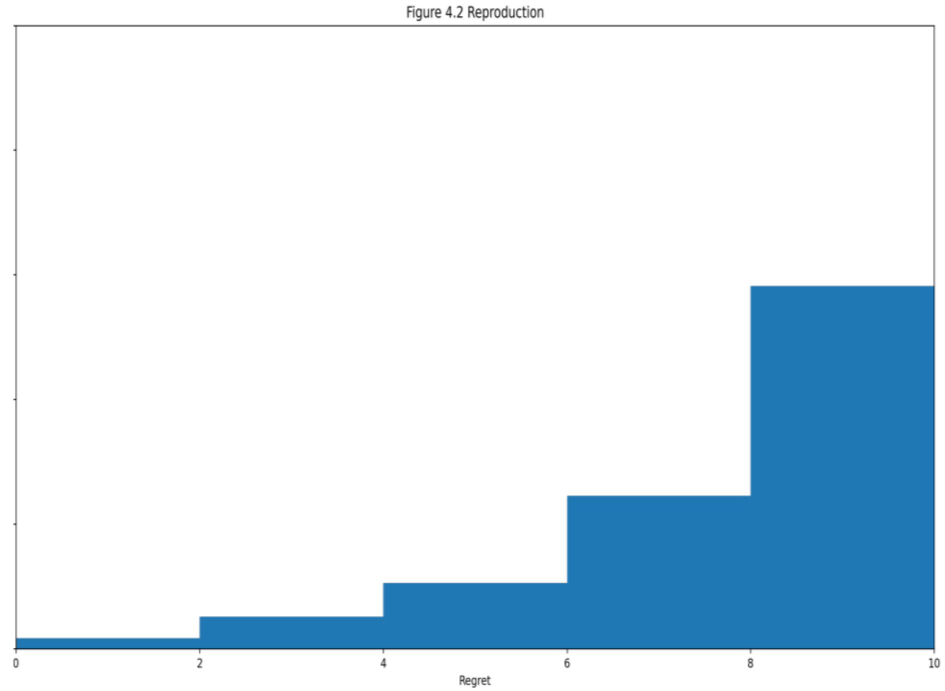
$$\mu \leq \hat{\mu} + \sqrt{\frac{2\sigma^2 \log(1/\delta)}{n}} \,. \tag{5.6}$$

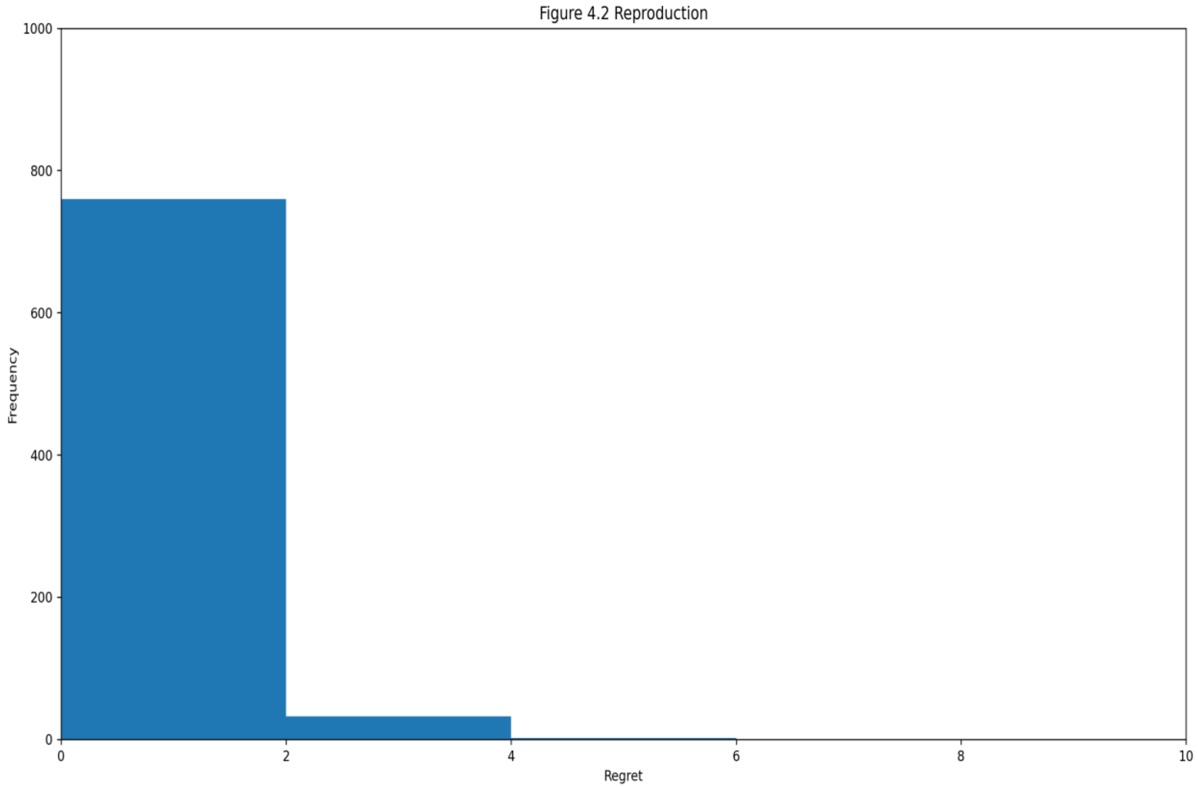
Symmetrically, it also follows that with probability at least $1 - \delta$,

$$\mu \geq \hat{\mu} - \sqrt{\frac{2\sigma^2 \log(1/\delta)}{n}} \,. \tag{5.7}$$

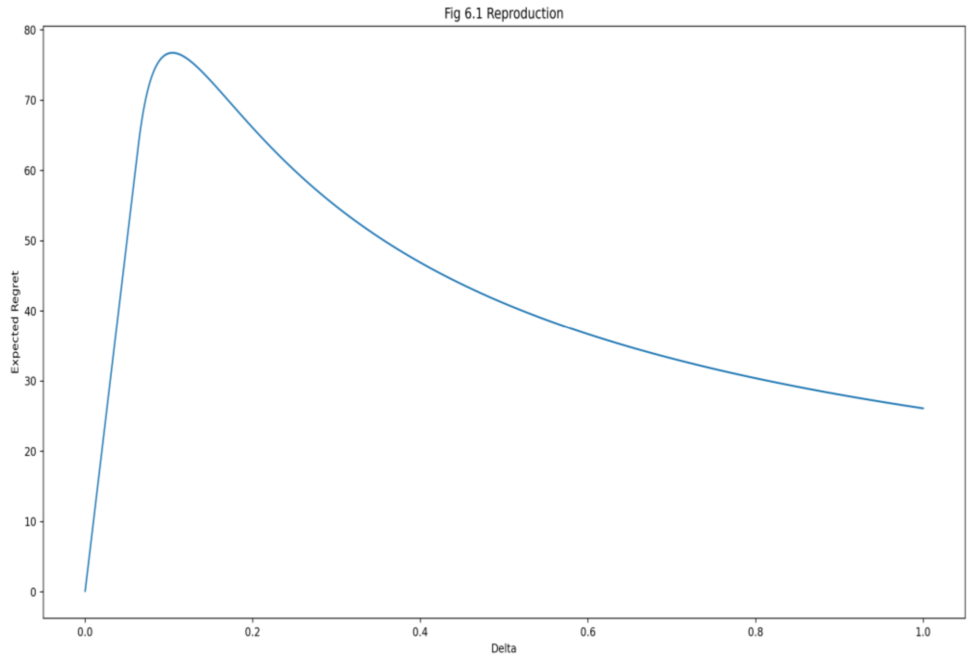
# Special case: Follow-the-leader

$$n\mu^* - \sum_{t=1}^{n} \mu_{A_t},$$



# Some Graphs



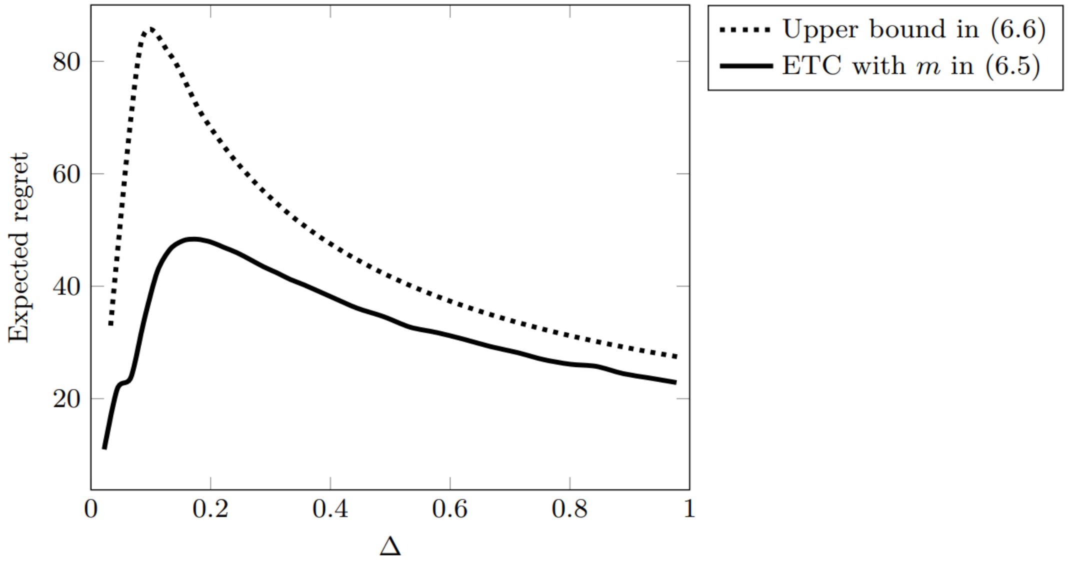




**Figure 6.1** The expected regret of ETC and the upper bound in Eq. (6.6).

# Some Graphs

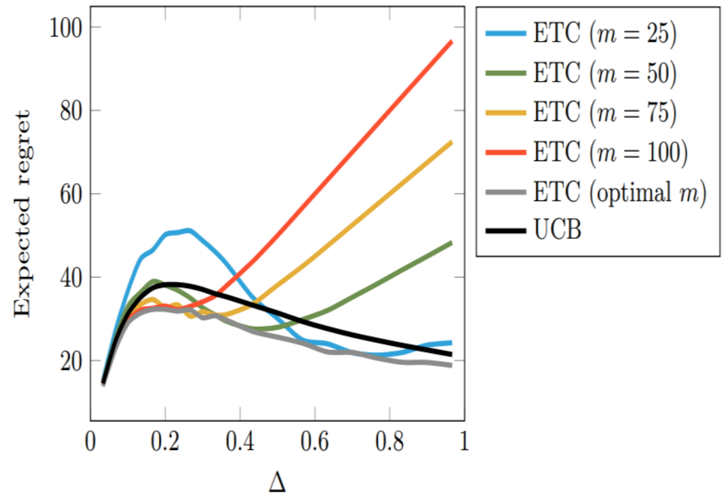



**Figure 7.1** Experiment showing universality of UCB relative to fixed instances of ETC