Lecture 14: Bandits Recap and Looking Ahead

ID 6002W: Online and Reinforcement Learning

Web-enabled M.Tech. program, IIT Madras

Suryanarayana Sankagiri

24th June, 2026

What is Online Learning?

A data-driven paradigm for making and improving decisions over time

What is Online Learning?

A data-driven paradigm for making and improving decisions over time

What makes this different from other ML?

- data comes in one step at a time, AND
- one must take actions (or make decisions) at each step

What is Online Learning?

A data-driven paradigm for making and improving decisions over time

What makes this different from other ML?

- data comes in one step at a time, AND
- one must take actions (or make decisions) at each step

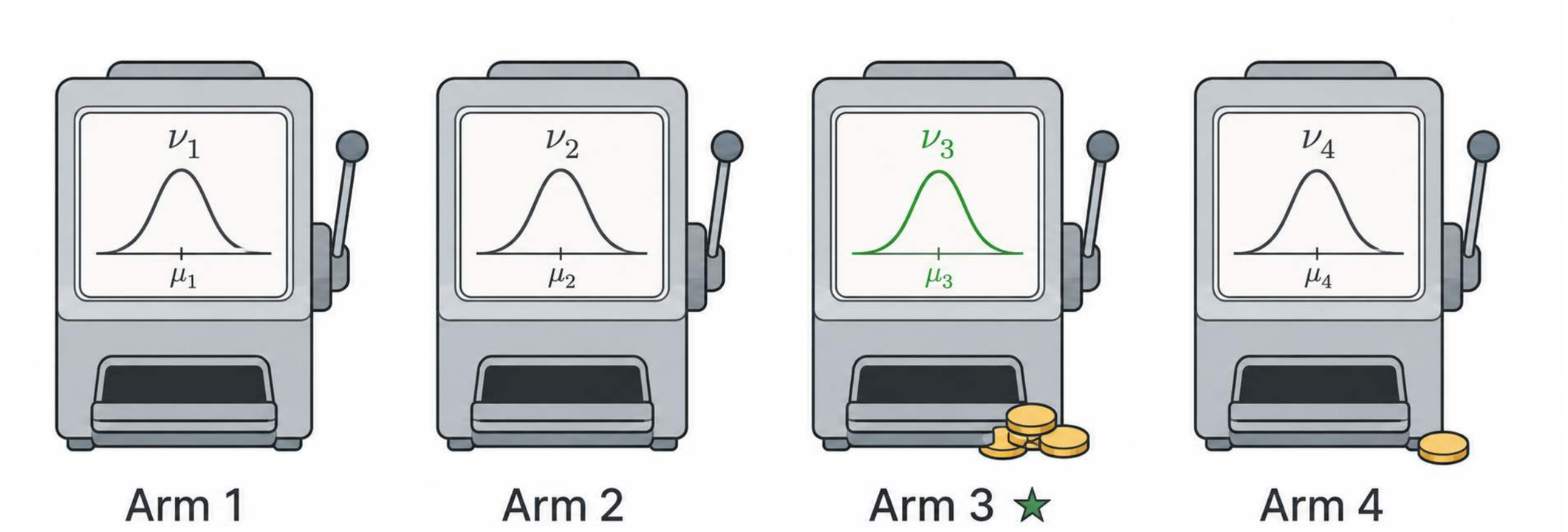
Two natural formalisms

- Find the best action, as quickly and reliably as possible
- Perform well throughout the entire period (regret minimisation)

Stochastic Multi-Armed Bandit

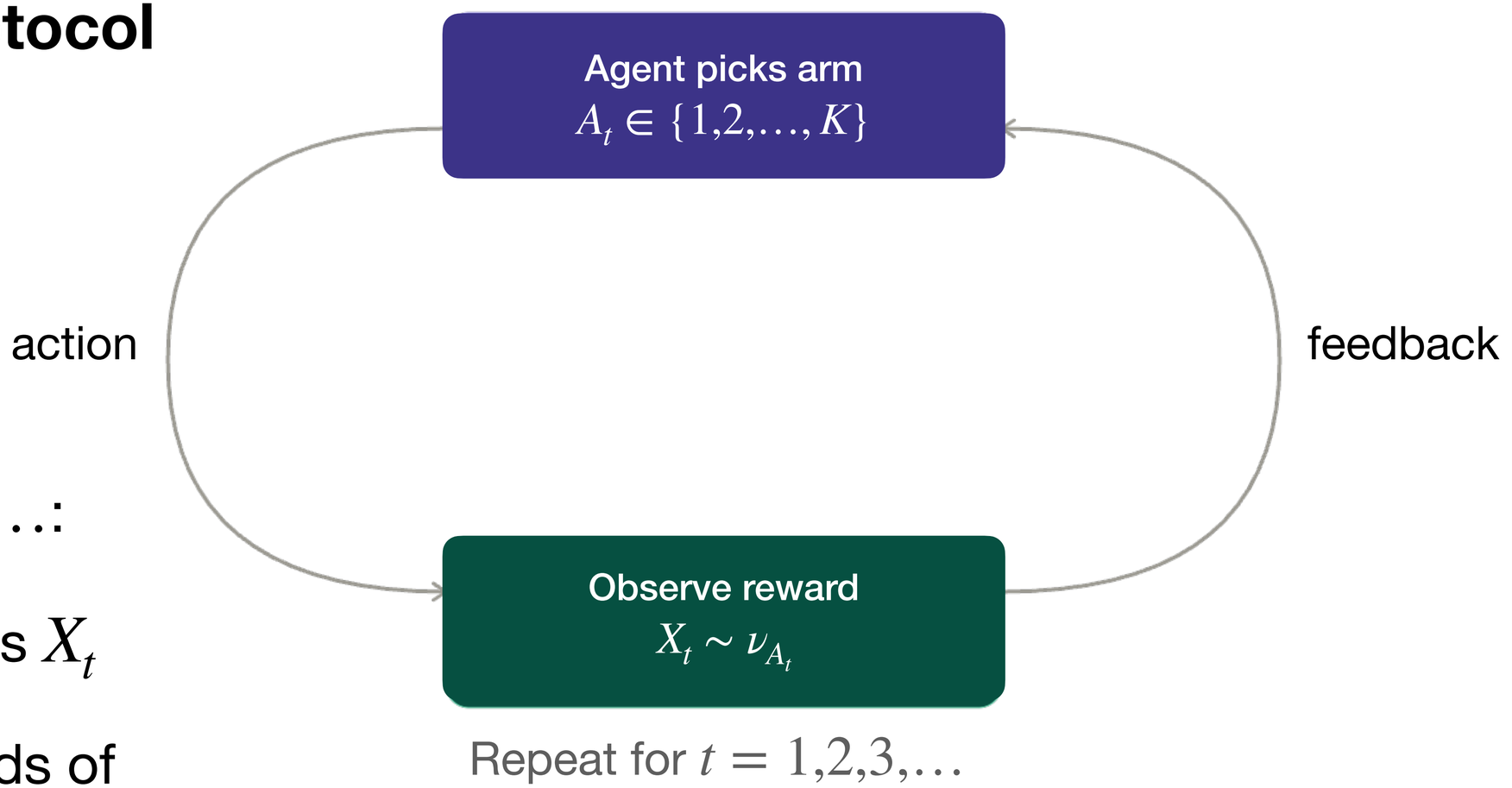
Formal setup

- K arms; arm i has reward distribution ν_i with mean μ_i
- Best arm: $i^* = \arg \max_{i \in \{1,2,\ldots,K\}} \mu_i$



Stochastic Multi-Armed Bandit

The interaction protocol



- At each round $t = 1,2,\ldots$:
- agent picks A_t, observes X_t
- Bandit feedback: rewards of un-pulled arms are not seen

Goal: use observed rewards to make better choices over time

Regret Definition

A useful decomposition

Regret is defined as $R(T) = T\mu^* - \mathbb{E}\left[\sum_{t=1}^{T} X_t\right]$

Regret Definition

A useful decomposition

Regret is defined as $R(T) = T\mu^* - \mathbb{E}\left[\sum_{t=1}^{T} X_t\right]$

- Let A_t denote the action taken in round t
  - Then $\mathbb{E}[X_t \mid A_t] = \mu_{A_t}$
- However, A_t is also random!
  - It usually depends on past rewards, which are random
  - So $\mathbb{E}\left[X_t\right] = \mathbb{E}\left[\mu_{A_t}\right]$

Regret Definition

A useful decomposition

Regret is defined as $R(T) = T\mu^* - \mathbb{E}\left[\sum_{t=1}^{T} X_t\right]$

- Let A_t denote the action taken in round t
  - Then $\mathbb{E}[X_t \mid A_t] = \mu_{A_t}$
- However, A_t is also random!
  - It usually depends on past rewards, which are random
  - So $\mathbb{E}\left[X_t\right] = \mathbb{E}\left[\mu_{A_t}\right] \Rightarrow R(T) = T\mu^* - \mathbb{E}\left[\sum_{t=1}^{T} X_t\right] = \mathbb{E}\left[\sum_{t=1}^{T} (\mu^* - \mu_{A_t})\right]$

Regret Decomposition

Can we simplify the expression $R(T) = \mathbb{E}\left[\sum_{t=1}^{T}(\mu^* - \mu_{A_t})\right]$ further?

- Define pull count of arm i: $T_i(T) \Rightarrow \sum_{i=1}^{K} T_i(T) = T$
- If $T_i(T)$ are known for all i,

$$\sum_{t=1}^{T}(\mu^* - \mu_{A_t}) = \sum_{i=1}^{K}\Delta_i T_i(T)$$

- Taking expectation: $R(T) = \sum_{i=1}^{K}\Delta_i \mathbb{E}[T_i(T)]$

Regret Decomposition

Can we simplify the expression $R(T) = \mathbb{E}\left[\sum_{t=1}^{T}(\mu^* - \mu_{A_t})\right]$ further?

- Define pull count of arm i: $T_i(T) \Rightarrow \sum_{i=1}^{K} T_i(T) = T$
- If $T_i(T)$ are known for all i,

$$\sum_{t=1}^{T}(\mu^* - \mu_{A_t}) = \sum_{i=1}^{K}\Delta_i T_i(T)$$

- Taking expectation: $R(T) = \sum_{i=1}^{K}\Delta_i \mathbb{E}[T_i(T)]$

To minimise regret, we must minimise how often we pull arms with large gaps

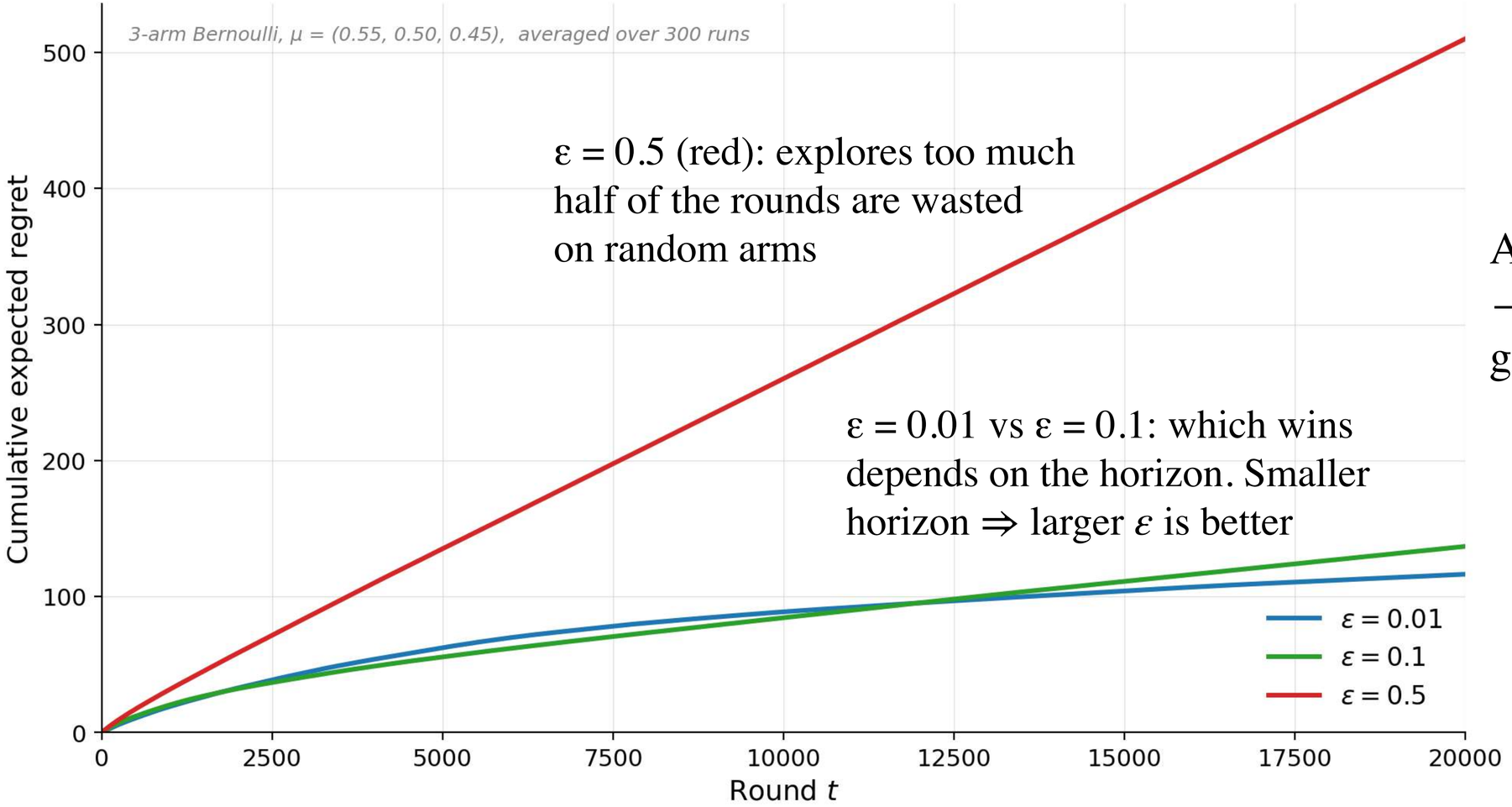
ε-Greedy Algorithm

- With probability ε: pull a uniformly random arm (explore)
- With probability $1 - \varepsilon$: pull the empirically best arm (exploit)
- $\varepsilon \in (0,1)$ is the exploration rate — a knob you tune

The simplest algorithm that does both exploration and exploitation by combining pure random and pure greedy strategies

ε-Greedy Algorithm

Behaviour and Limitations



ε = 0.5 (red): explores too much
half of the rounds are wasted
on random arms

ε = 0.01 vs ε = 0.1: which wins
depends on the horizon. Smaller
horizon ⇒ larger ε is better

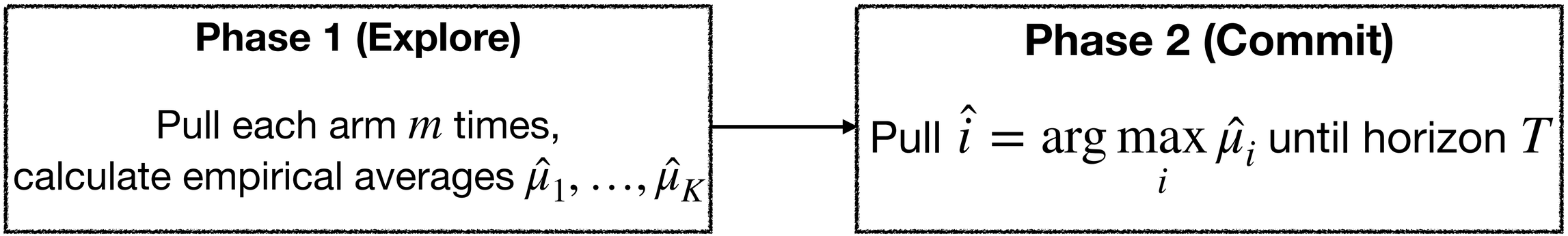
All three curves keep growing
— any fixed ε gives linearly
growing regret

No fixed ε is universally best

For fixed ε, even after we know the best arm, we still explore forever. This leads to linear regret

Explore-Then-Commit

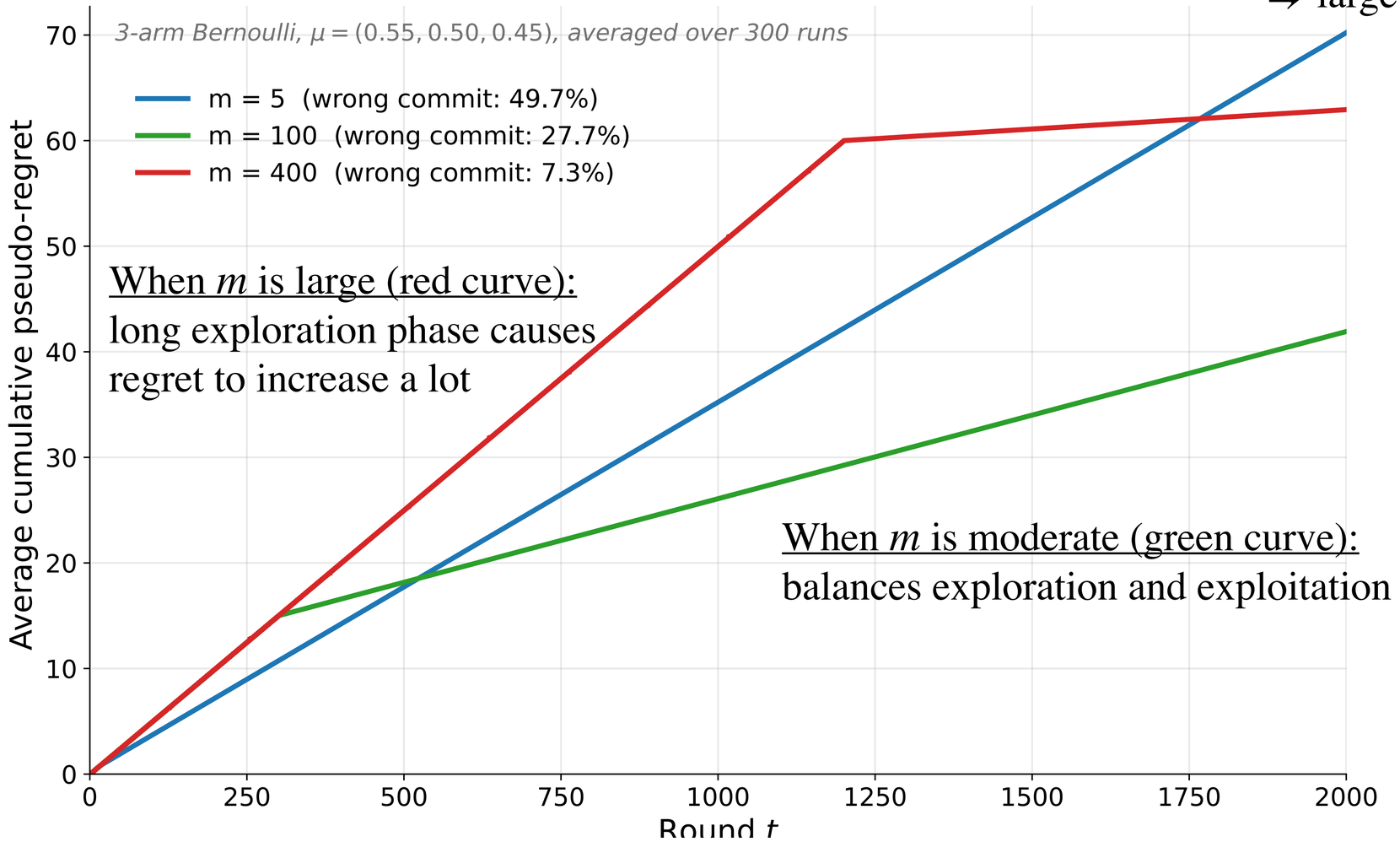
ε-greedy mixes exploration and exploitation at every round.
What if we separate them in time?



Key parameter: m (exploration budget per arm). Total exploration length: Km

Explore-Then-Commit

Behaviour and Limitations



When m is small (blue curve):
many runs commit to wrong arm
⇒ large, linearly growing regret

When m is large (red curve):
long exploration phase causes regret to increase a lot

When m is moderate (green curve):
balances exploration and exploitation

All three curves keep growing — any fixed m gives linearly growing regret

No fixed m is universally best

For any fixed m, there is a chance of permanently committing to the wrong arm. This leads to linear regret

Hoeffding's Inequality

Statement

Let $X_1, \ldots, X_n$ be independent random variables with $X_i \in [a, b]$. Let $\mathbb{E}[X_t] = \mu$.

Let $\hat{\mu}(n) = \frac{1}{n} \sum_{t=1}^{n} X_t$. Then for any $\varepsilon > 0$,

$$\mathbb{P}(\,|\,\hat{\mu}(n) - \mu\,|\,\geq \epsilon) \leq 2\exp\left(\frac{-2n\epsilon^2}{(b-a)^2}\right)$$

Hoeffding's Inequality

Statement

Let $X_1, \ldots, X_n$ be independent random variables with $X_i \in [a, b]$. Let $\mathbb{E}[X_t] = \mu$.

Let $\hat{\mu}(n) = \frac{1}{n} \sum_{t=1}^{n} X_t$. Then for any $\varepsilon > 0$,

$$\mathbb{P}(\,|\,\hat{\mu}(n) - \mu\,|\,\geq \epsilon) \leq 2\exp\left(\frac{-2n\epsilon^2}{(b-a)^2}\right)$$

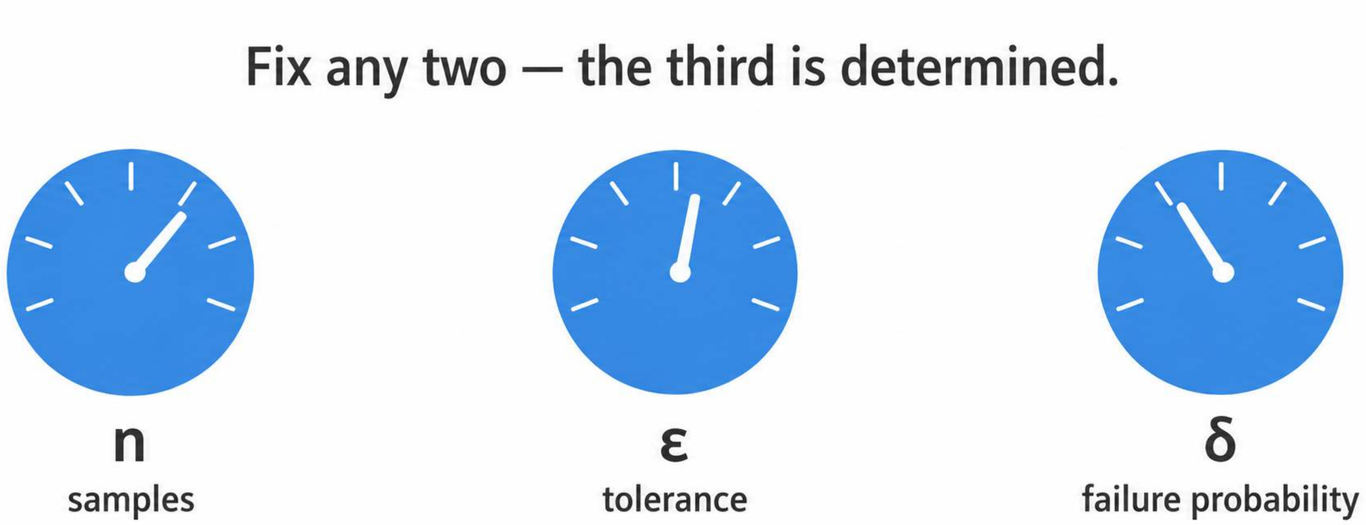
For Bernoulli case, $a = 0,\ b = 1$. So $\mathbb{P}(\,|\,\hat{\mu}(n) - \mu\,|\,\geq \epsilon) \leq 2\exp\left(-2n\epsilon^2\right)$

Hoeffding's Inequality

Interpreting The Bound

For Bernoulli case, $a = 0,\ b = 1$. So $\mathbb{P}(|\hat{\mu}(n) - \mu| \geq \epsilon) \leq 2\exp\left(-2n\epsilon^2\right)$

Multiple ways to interpret this formula.

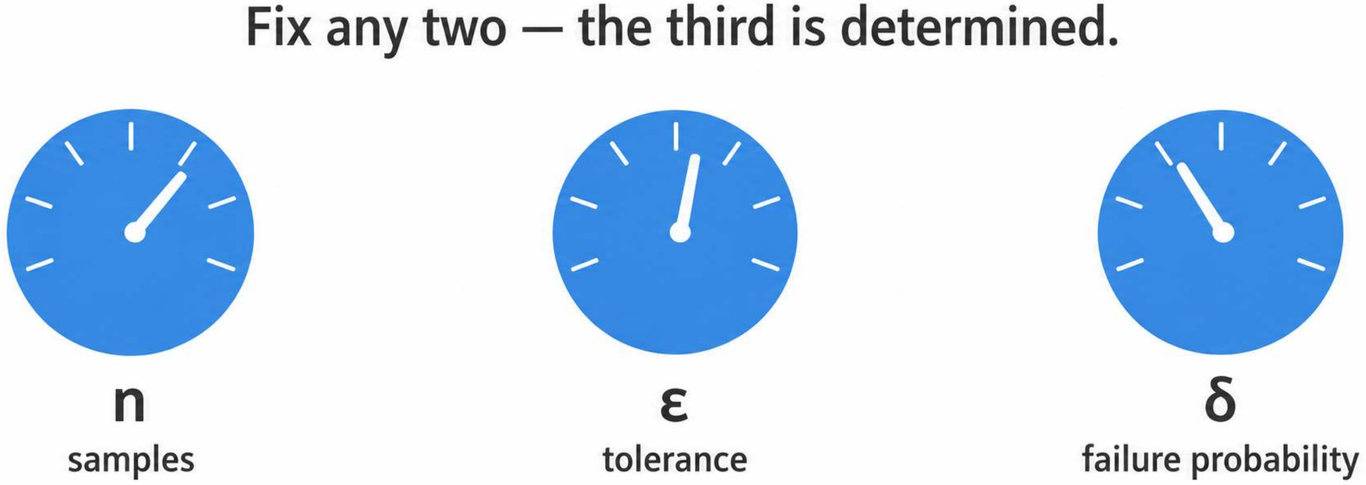


Hoeffding's Inequality

Interpreting The Bound

For Bernoulli case, $a = 0,\ b = 1$. So $\mathbb{P}(|\hat{\mu}(n) - \mu| \geq \epsilon) \leq 2\exp\left(-2n\epsilon^2\right)$

Multiple ways to interpret this formula.



To get confidence intervals, you fix n and δ, and you can get confidence width

$$\epsilon = \sqrt{\frac{2\log(1/\delta)}{2n}}$$

Successive Elimination

The Key Idea

We have K arms. We want to find the best one with the following strategy.

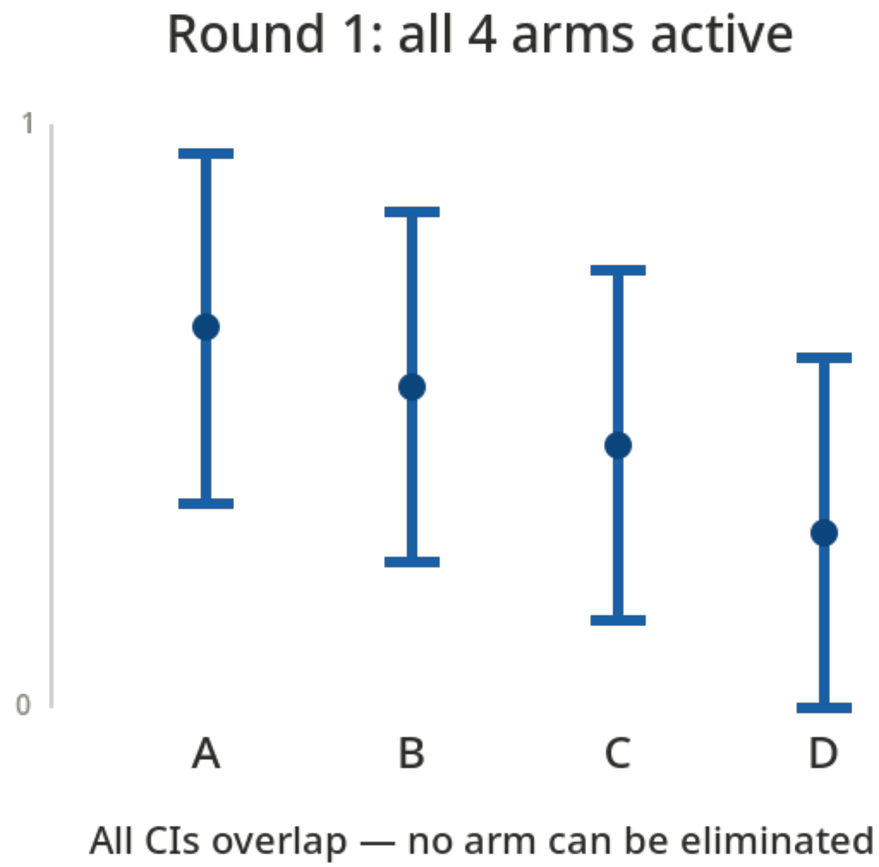
Successive Elimination

The Key Idea

We have K arms. We want to find the best one with the following strategy.

Maintain a set of **active arms** (initially, all K).
At each round t

- pull every active arm once
- Update *empirical means*, *confidence intervals*



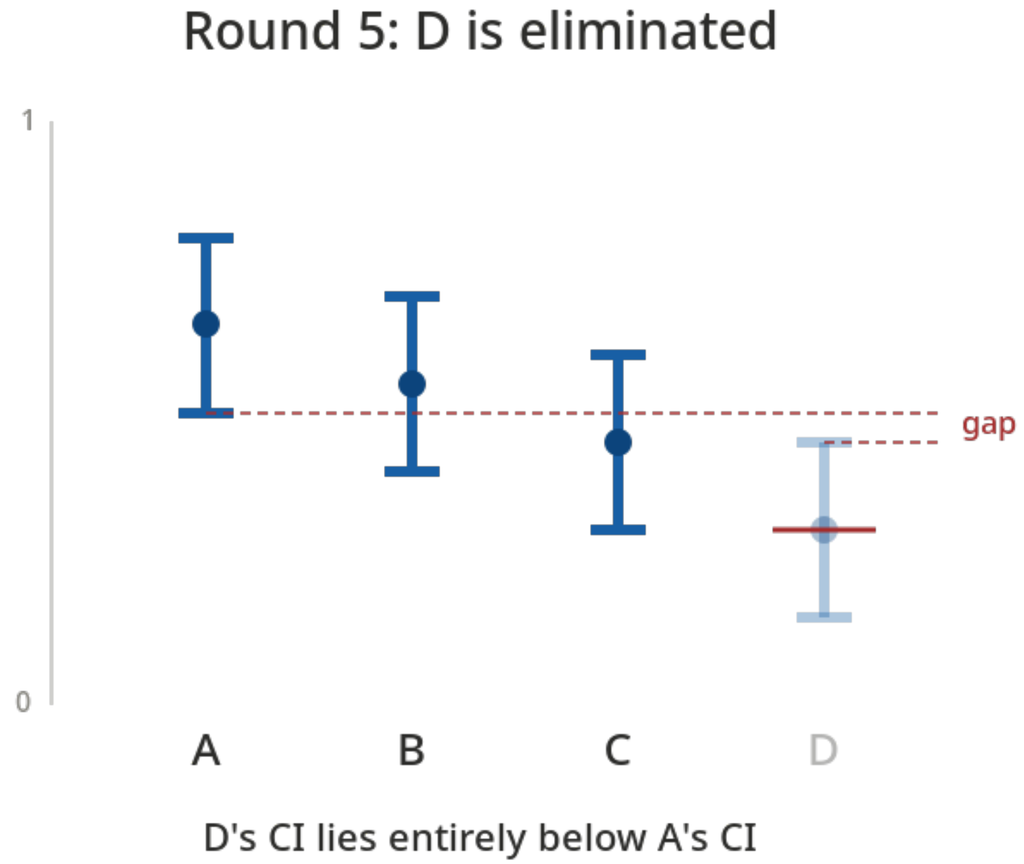
Successive Elimination

The Key Idea

We have K arms. We want to find the best one with the following strategy.

Maintain a set of **active arms** (initially, all K).
At each round t

- pull every active arm once
- Update *empirical means*, *confidence intervals*
- Eliminate any arm whose confidence interval lies *entirely below* another arm's interval



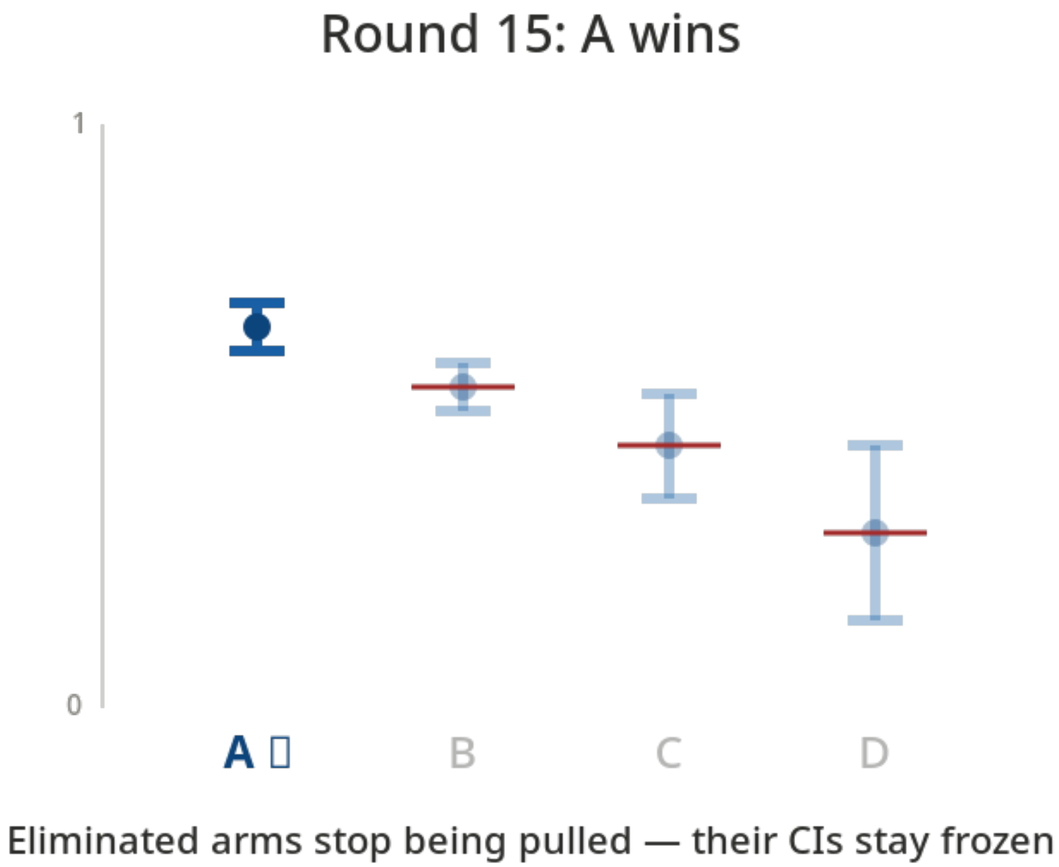
Successive Elimination

The Key Idea

We have K arms. We want to find the best one with the following strategy.

Maintain a set of **active arms** (initially, all K).
At each round t

- pull every active arm once
- Update *empirical means*, *confidence intervals*
- Eliminate any arm whose confidence interval lies *entirely below* another arm's interval
- Stop when only one arm remains

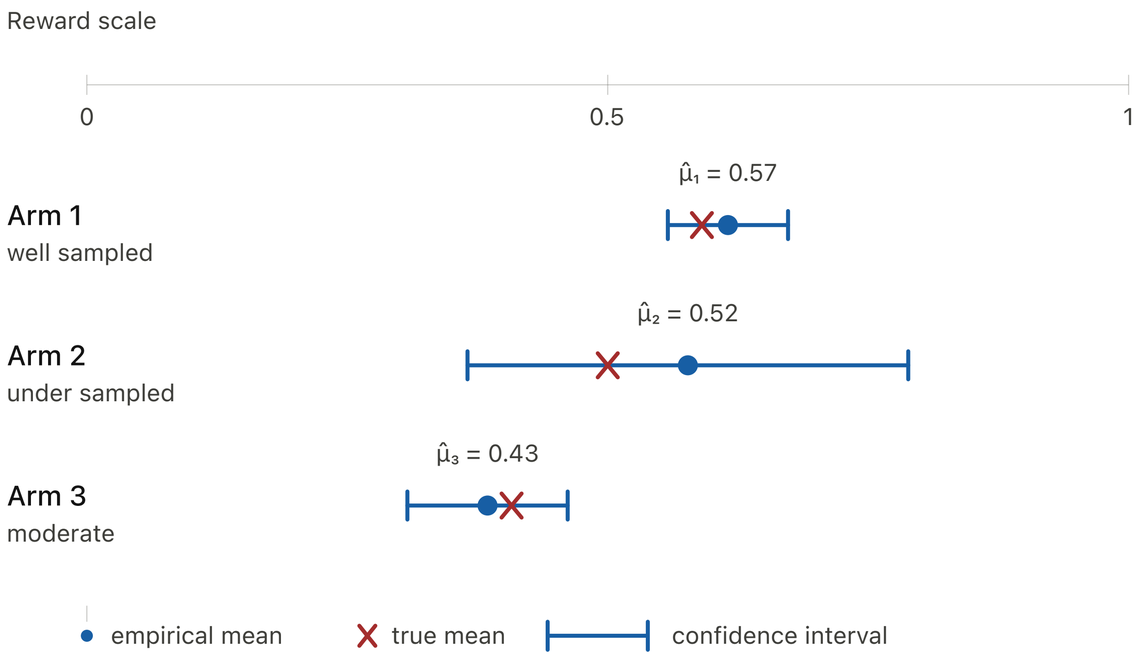


UCB Algorithm

Using confidence intervals

For each arm i, we can easily maintain

- $\hat{\mu}_i(t)$: estimate of the mean reward μ_i
- $\sqrt{\frac{c \log T}{T_i(t)}}$: width of confidence interval



Note:

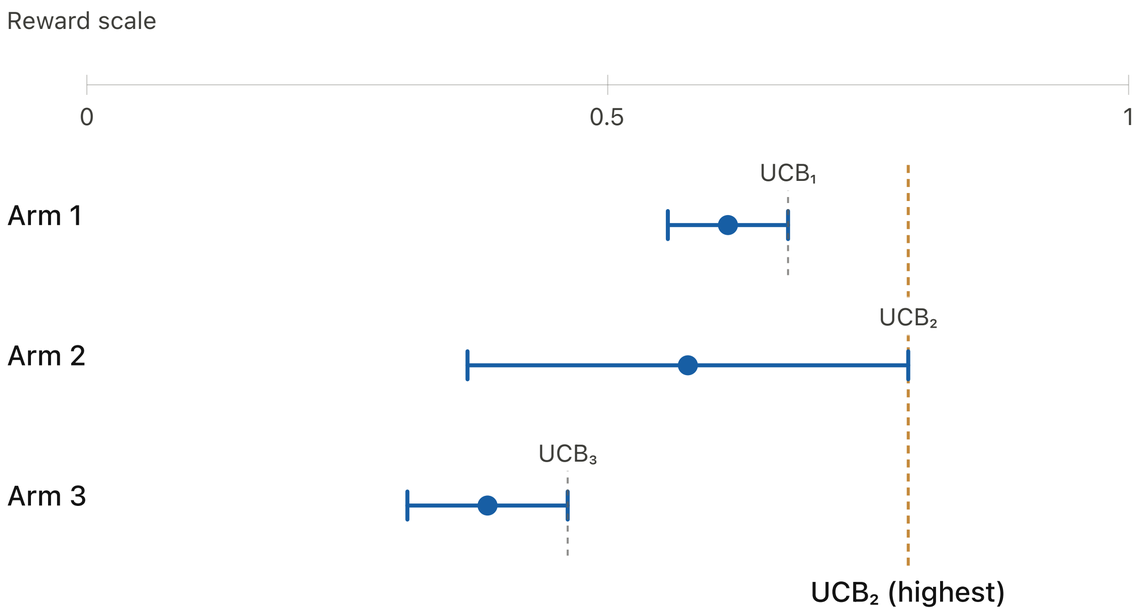
- Hoeffding's inequality gives us confidence intervals of width $\sqrt{\log(2/\delta)/2n}$.
- For a fixed horizon T, we take failure probability $\delta \propto 1/T$; constants absorbed in c

UCB Algorithm

The decision rule

$$UCB_i(t) = \hat{\mu}_i(t) + \sqrt{\frac{c \log T}{T_i(t)}}$$

$$\text{Pull } A_t = \arg \max_{i \in \{1,2,\ldots,K\}} UCB_i(t-1)$$

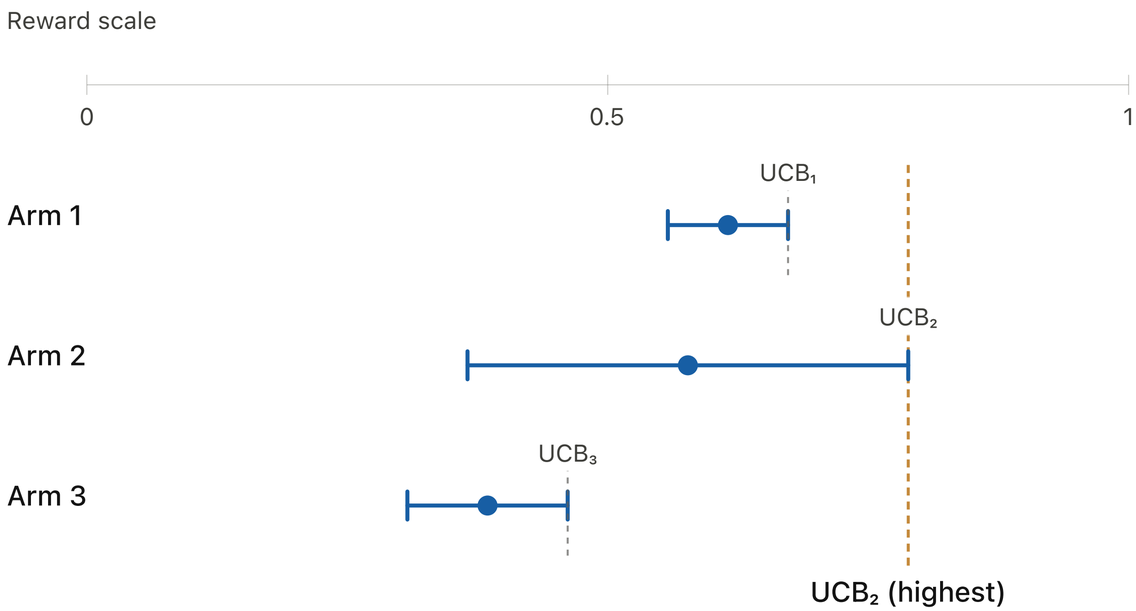


UCB Algorithm

The decision rule

$$UCB_i(t) = \hat{\mu}_i(t) + \sqrt{\frac{c \log T}{T_i(t)}}$$

$$\text{Pull } A_t = \arg \max_{i \in \{1,2,\ldots,K\}} UCB_i(t-1)$$



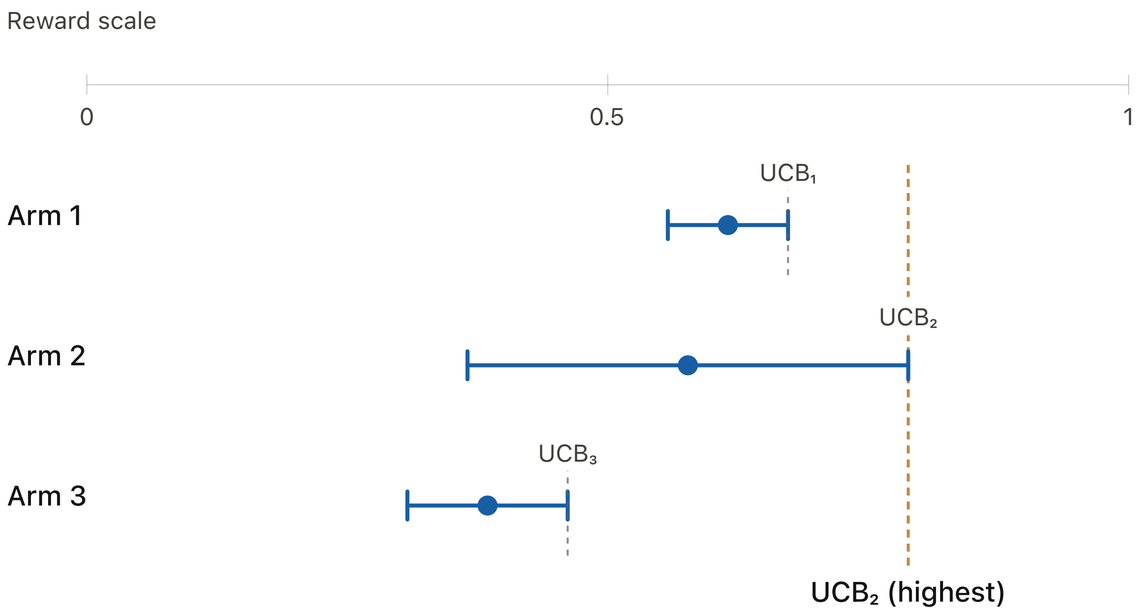
$\sqrt{c \log T / T_i(t)}$ is an exploration bonus term, added to mean estimate

UCB Algorithm

The decision rule

$$UCB_i(t) = \hat{\mu}_i(t) + \sqrt{\frac{c \log T}{T_i(t)}}$$

$$\text{Pull } A_t = \arg \max_{i \in \{1,2,\ldots,K\}} UCB_i(t-1)$$



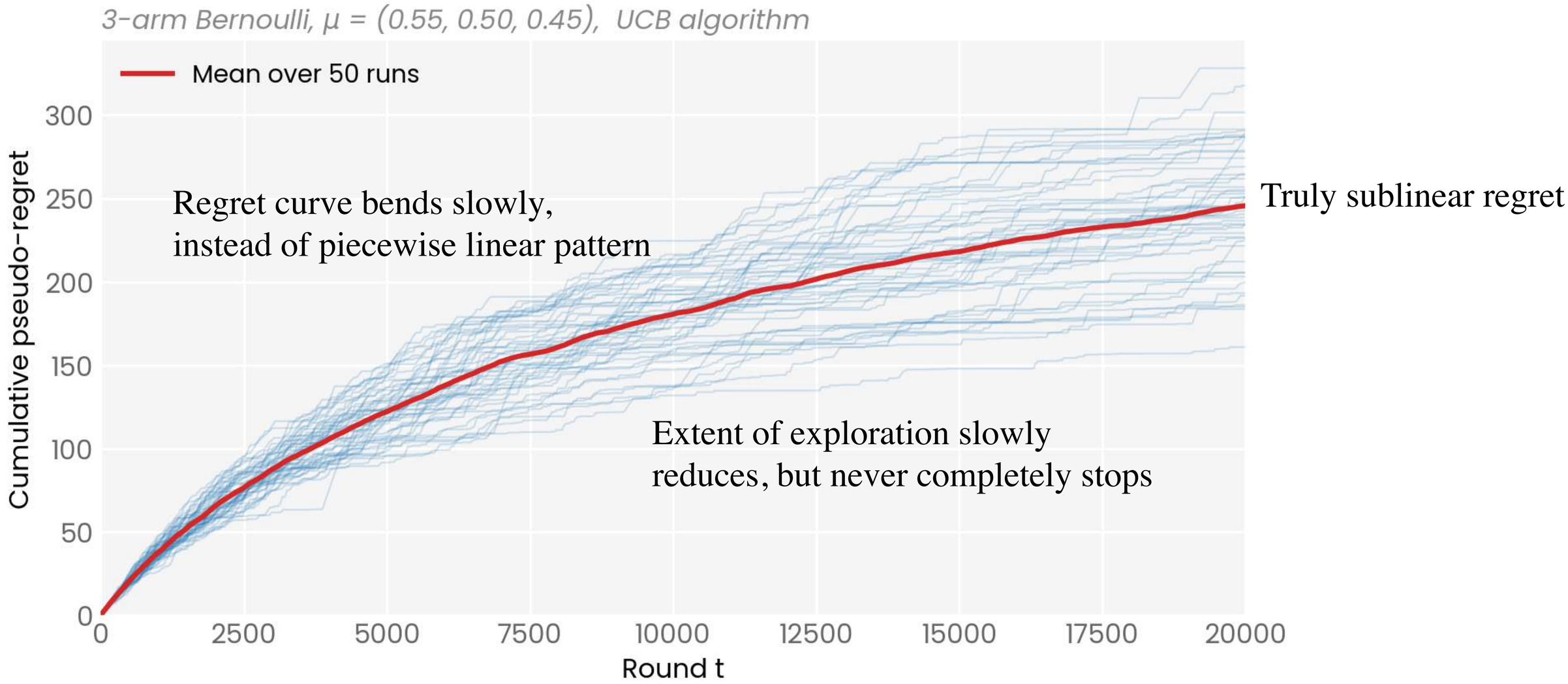
$\sqrt{c \log T / T_i(t)}$ is an exploration bonus term, added to mean estimate

Guiding Principle: Optimism in the Face of Uncertainty

Act as if each arm is as good as it could reasonably be

Empirical Performance

Cumulative regret curves



Thompson Sampling

Pseudocode

Initialise: $\alpha_i = \beta_i = 1$ for all arms i

For $t = 1,2,\ldots,T$

1. For each arm i:

   Sample $\tilde{\mu}_i \sim \text{Beta}(\alpha_i, \beta_i)$

2. Pull arm:

$$A_t = \arg \max_{i \in \{1,2,\ldots,K\}} \tilde{\mu}_i(t-1)$$

3. Observe reward $X_t \in \{0,1\}$ and update:

   if $X_t = 1$: α_{A_t}++, else: β_{A_t}++

Thompson Sampling

Pseudocode

Initialise: $\alpha_i = \beta_i = 1$ for all arms i

For $t = 1,2,\ldots,T$

1. For each arm i:

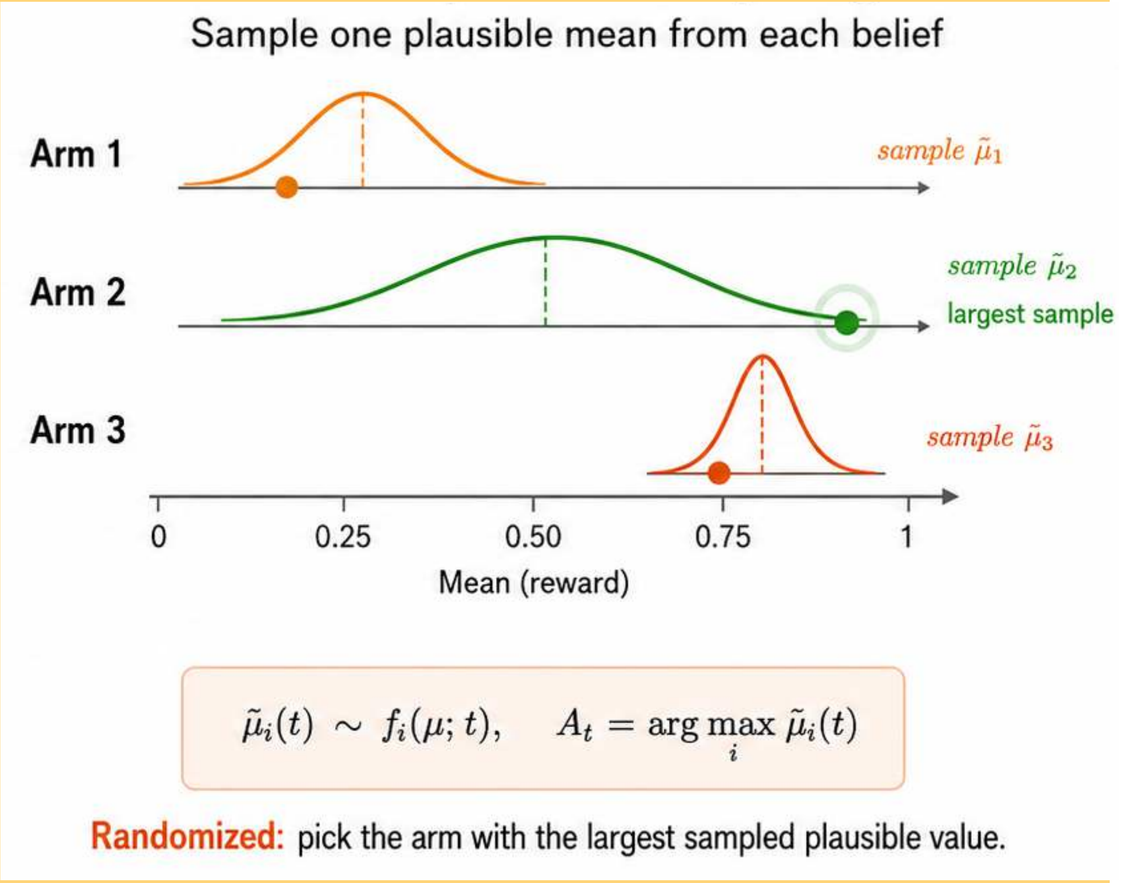
$$\text{Sample } \tilde{\mu}_i \sim \text{Beta}(\alpha_i, \beta_i)$$

2. Pull arm:

$$A_t = \arg \max_{i \in \{1,2,\ldots,K\}} \tilde{\mu}_i(t-1)$$

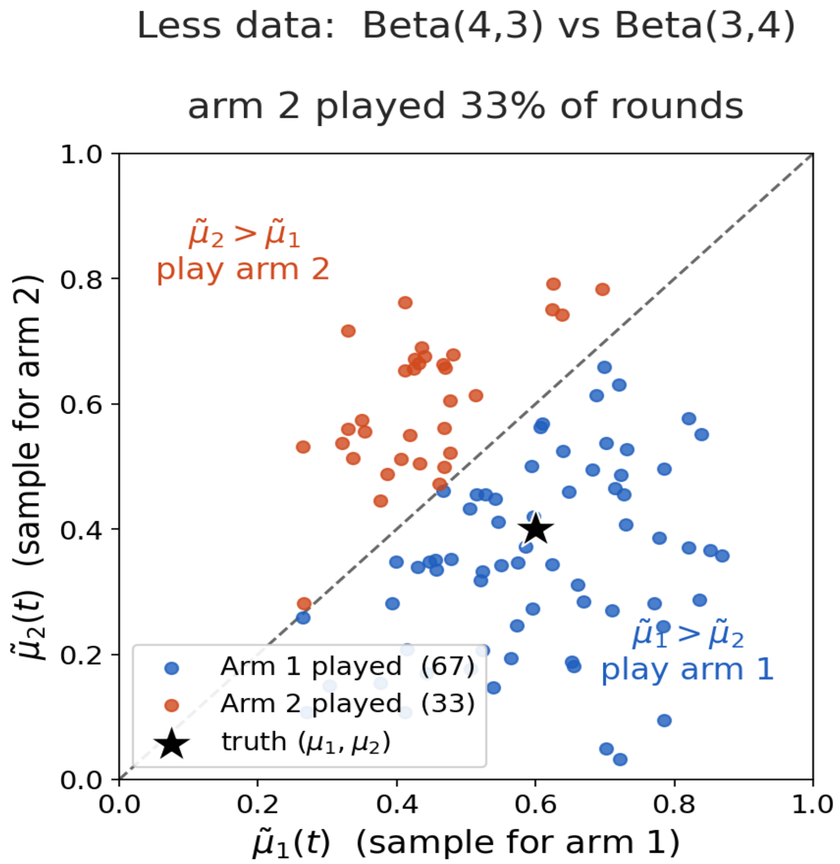
3. Observe reward $X_t \in \{0,1\}$ and update:

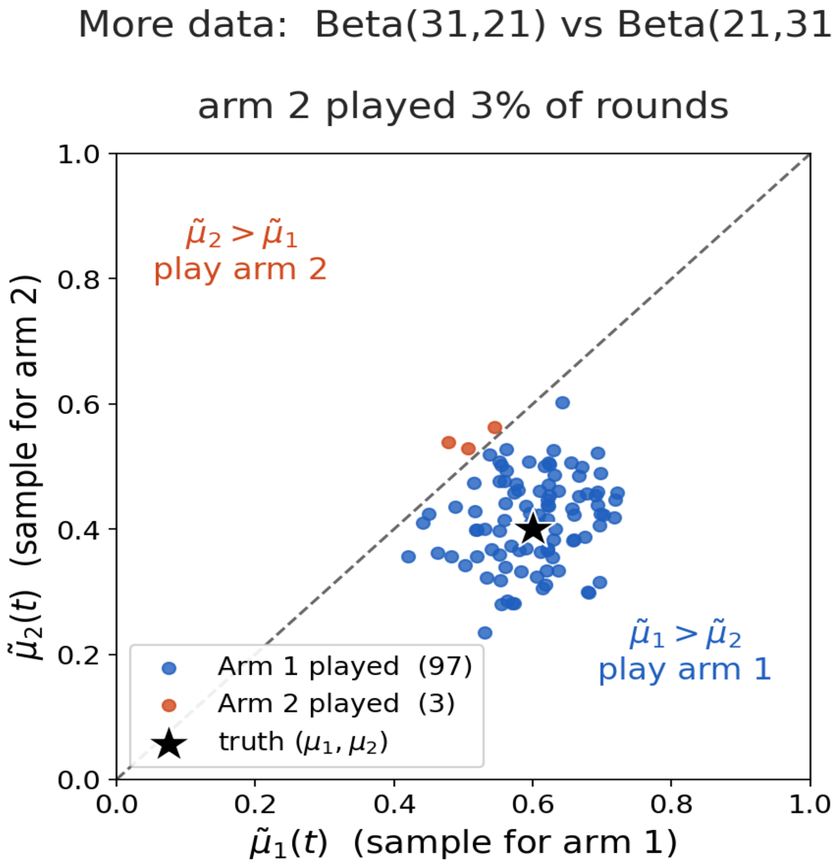
$$\text{if } X_t = 1\text{: } \alpha_{A_t}\text{++, else: } \beta_{A_t}\text{++}$$



Why Thompson Sampling?

The set of samples $(\tilde{\mu}_1(t), \tilde{\mu}_2(t))$ are a set of plausible values of $(\mu_1 = 0.6,\ \mu_2 = 0.4)$

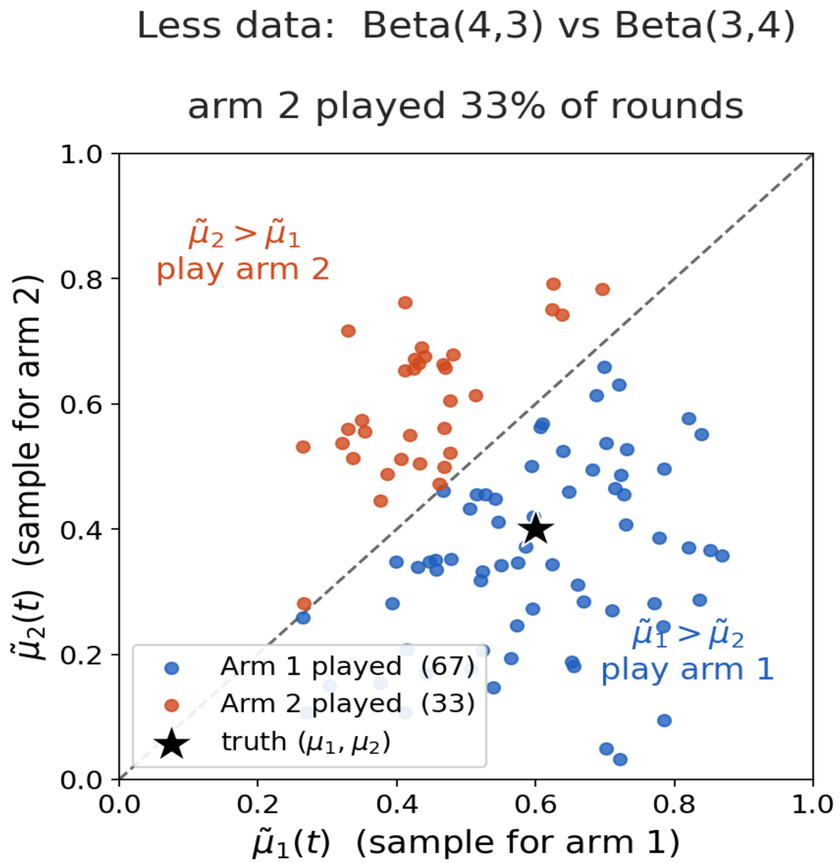


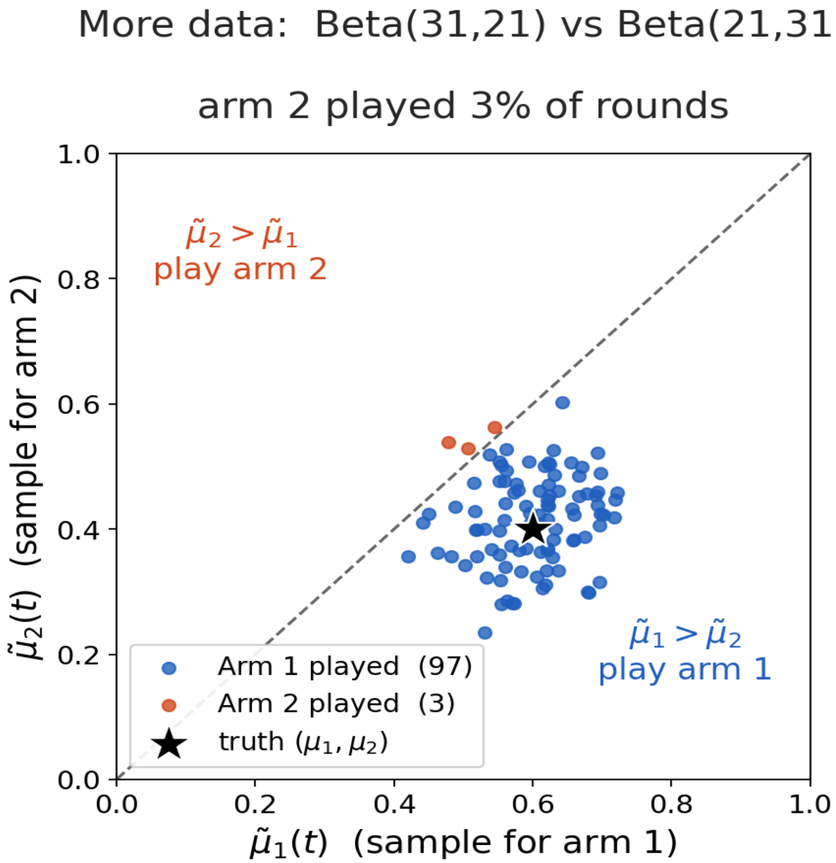


Thompson sampling rule: play as if the sampled world is the true world

Why Thompson Sampling?

The set of samples $(\tilde{\mu}_1(t), \tilde{\mu}_2(t))$ are a set of plausible values of $(\mu_1 = 0.6,\ \mu_2 = 0.4)$





Thompson sampling rule: play as if the sampled world is the true world

Leans towards exploration with less data, leans towards exploitation with more data

The Linear Bandits Model

An extension to multi-armed bandits

The Linear Bandits Model

An extension to multi-armed bandits

- The basic multi-armed bandit setting had K arms
  - each with fixed but unknown rewards μ_i
    - No relation between $\mu_1, \ldots, \mu_K$ is assumed

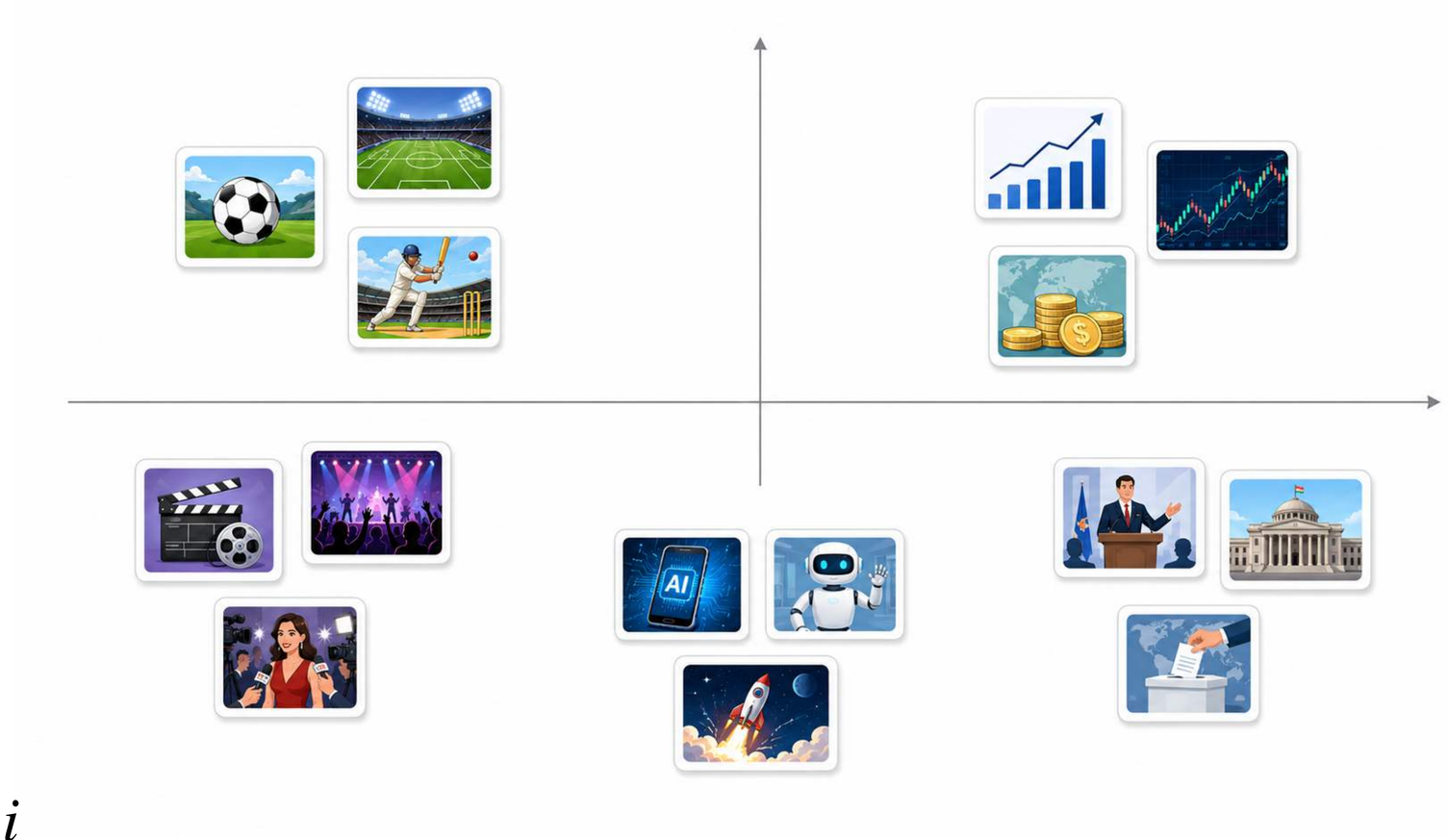
The Linear Bandits Model

An extension to multi-armed bandits

- The basic multi-armed bandit setting had K arms
  - each with fixed but unknown rewards μ_i
    - No relation between $\mu_1, \ldots, \mu_K$ is assumed
- The linear bandit model adds some structure to $\mu_1, \ldots, \mu_K$
  - $\mu_i = \langle \theta, x_i \rangle$: $x_i \in \mathbb{R}^d$ known, $\theta \in \mathbb{R}^d$ unknown
    - Allows us to use data from one arm to learn about other arms

Interpreting Context Vectors

- We know that $\mu_i = \langle \theta, x_i \rangle$
- This means, arms i, j with $x_i \approx x_j$ will have $\mu_i \approx \mu_j$ (for any θ)
- Intuitively, arms that are ‘similar’ should have similar rewards; closer the arms, closer the rewards
- So context vector x_i should capture some semantic meaning about arm i



Representative image of context vectors of different news articles.
Articles of similar topics should have context vectors that are close together

Quantifying Uncertainty

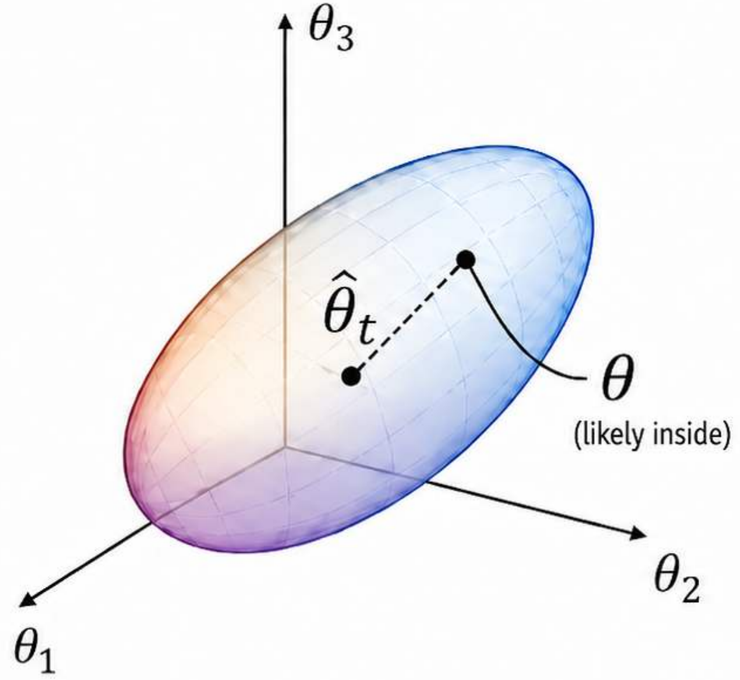
In linear bandits

- In vanilla multi-armed bandits, the unknown quantity μ_i was a scalar
  - Uncertainty was represented via confidence intervals: $\mu_i \in \left[\hat{\mu}_i - \epsilon, \hat{\mu}_i + \epsilon\right]$

- Here, we need to represent uncertainty about a vector quantity θ
  - We calculate confidence sets given by $\mathscr{C}_t = \left\{\theta \in \mathbb{R}^d : \|\theta - \hat{\theta}_t\|^2_{V_t} \leq \beta\right\}$, $V_t = \sum x(s)x(s)^\top$

Illustrating Confidence Ellipsoids

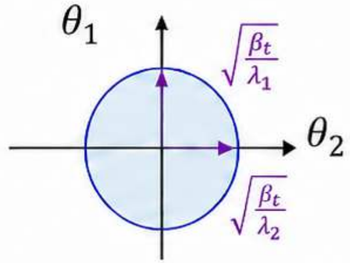
Confidence ellipsoid



$$\mathscr{C}_t = \left\{ \theta \in \mathbb{R}^d : \|\theta - \hat{\theta}_t\|^2_{V_t} \leq \beta \right\}$$

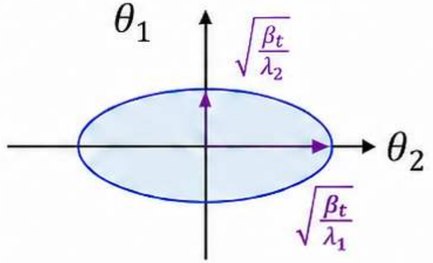
$$V_t = \sum x(s)x(s)^\top$$

How V_t shapes the set



① $\lambda_1 = \lambda_2$

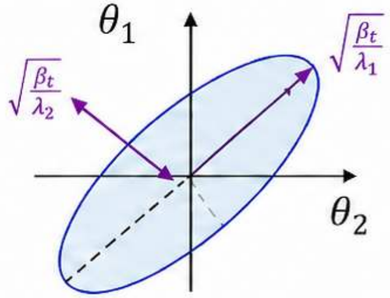
equal uncertainty in all directions



② $V_t = \mathrm{diag}(\lambda_1, \lambda_2)$

larger λ_i → shorter axis

smaller λ_i → longer axis



③ $V_t = U\,\mathrm{diag}(\lambda_1, \lambda_2)\,U^T$

eigenvectors set orientation

Larger eigenvalue → more information → shorter axis.

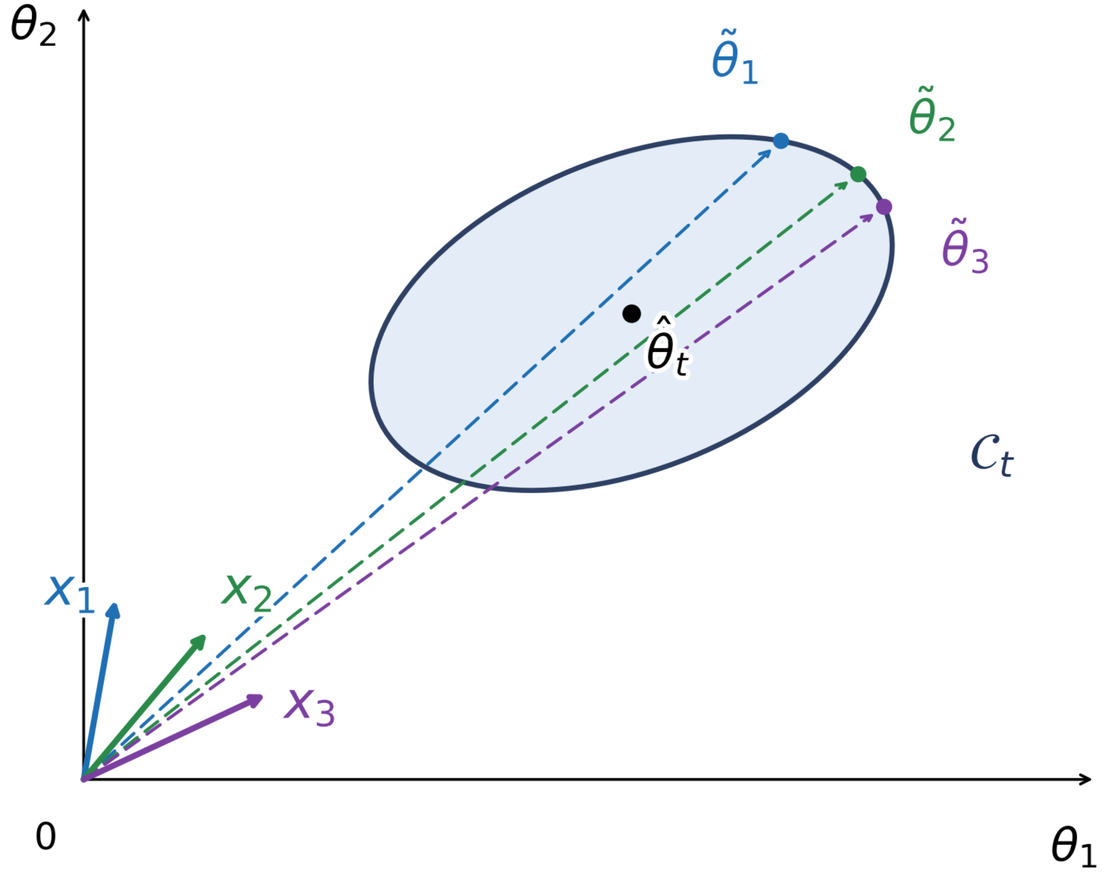
LinUCB Pseudocode

Parameter: β, λ

Initialise $V_0 = \lambda I,\ b_0 = 0$

For $t = 1,2,\ldots,T$:

- Calculate $\hat{\theta}_{t-1} = V_{t-1}^{-1} b_{t-1}$
- Play $A_t = \arg\max_{i \in [K]} \langle \hat{\theta}_{t-1}, x_i \rangle + \sqrt{\beta} \|x\|_{V_{t-1}^{-1}} \qquad (x(t) = x_{A_t})$
- Observe $y(t)$, update $V_t = V_{t-1} + x(t)x(t)^\top,\ b_t = b_{t-1} + x(t)y(t)$



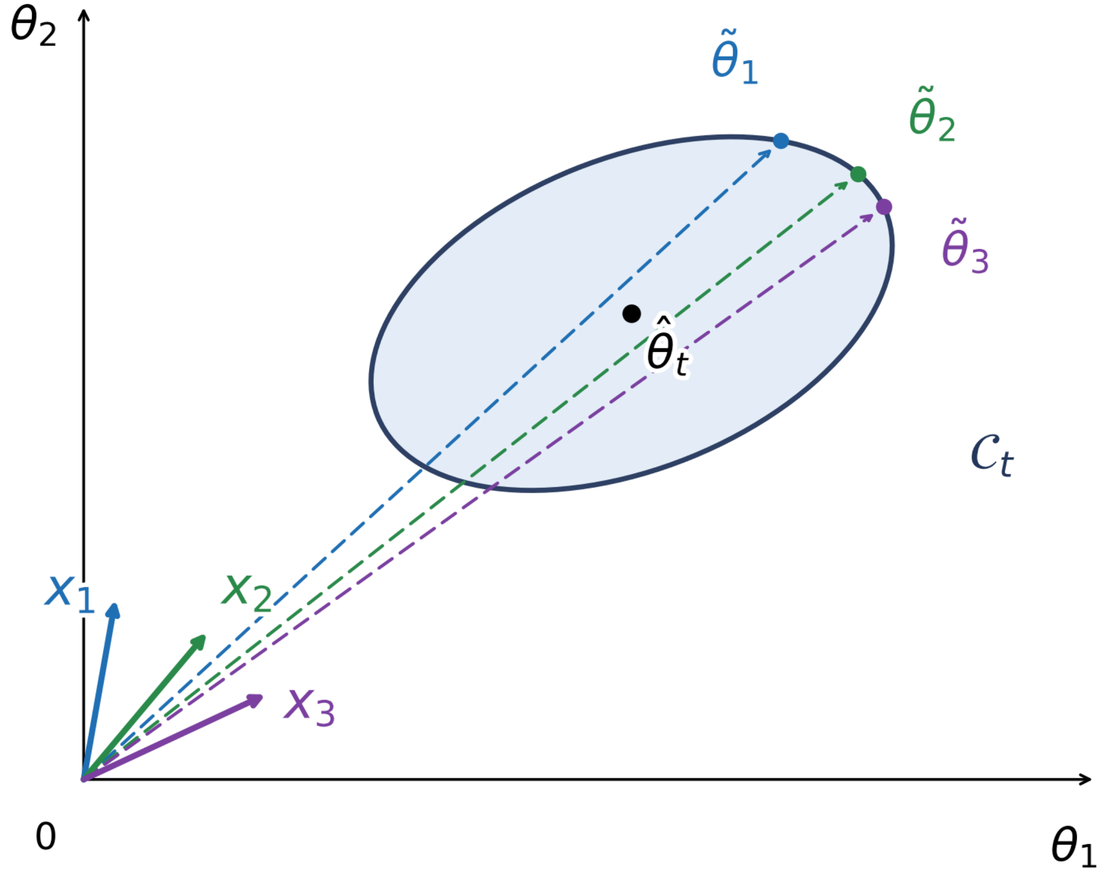
LinUCB Pseudocode

Parameter: β, λ

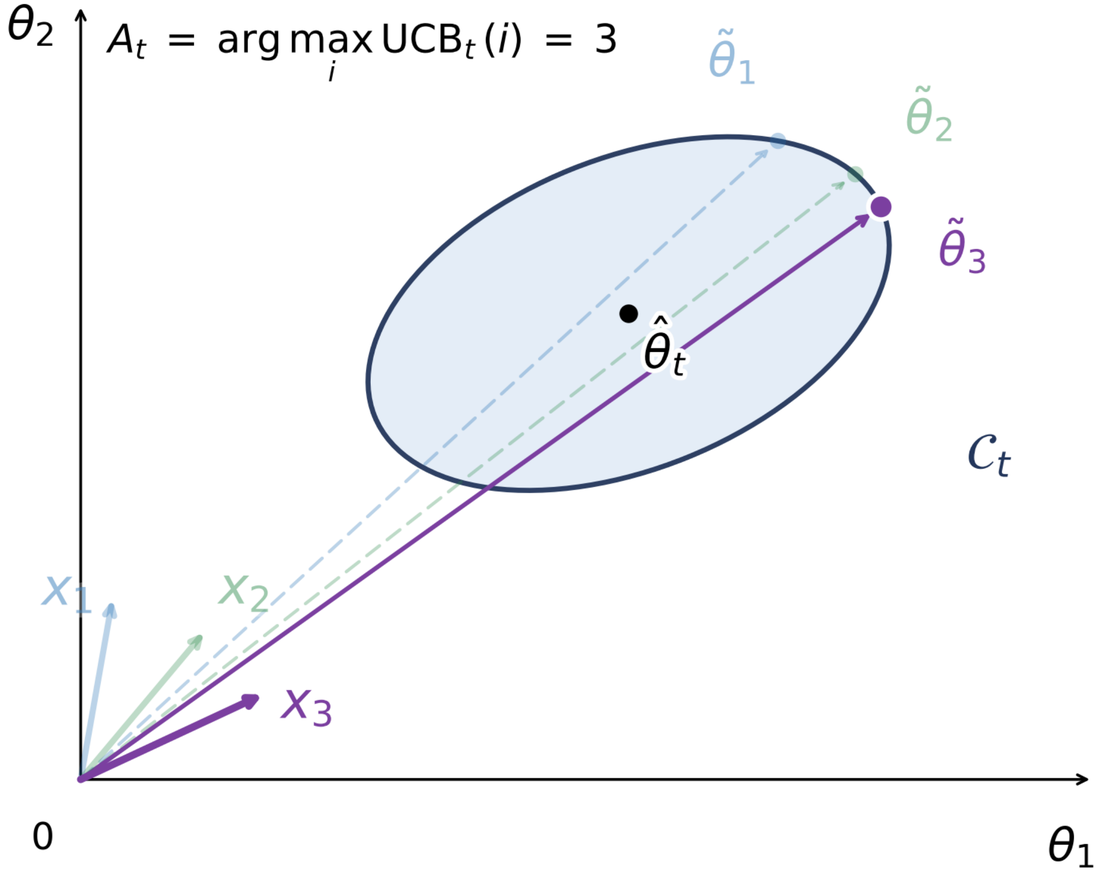
Initialise $V_0 = \lambda I,\ b_0 = 0$

For $t = 1,2,\ldots,T$:

- Calculate $\hat{\theta}_{t-1} = V_{t-1}^{-1} b_{t-1}$
- Play $A_t = \arg\max_{i \in [K]} \langle \hat{\theta}_{t-1}, x_i \rangle + \sqrt{\beta} \|x\|_{V_{t-1}^{-1}} \qquad (x(t) = x_{A_t})$
- Observe $y(t)$, update $V_t = V_{t-1} + x(t)x(t)^\top,\ b_t = b_{t-1} + x(t)y(t)$



LinUCB Pseudocode



Parameter: β, λ

Initialise $V_0 = \lambda I,\ b_0 = 0$

For $t = 1,2,\ldots,T$:

- Calculate $\hat{\theta}_{t-1} = V_{t-1}^{-1} b_{t-1}$
- Play $A_t = \arg\max_{i \in [K]} \langle \hat{\theta}_{t-1}, x_i \rangle + \sqrt{\beta} \|x\|_{V_{t-1}^{-1}} \qquad (x(t) = x_{A_t})$
- Observe $y(t)$, update $V_t = V_{t-1} + x(t)x(t)^\top,\ b_t = b_{t-1} + x(t)y(t)$

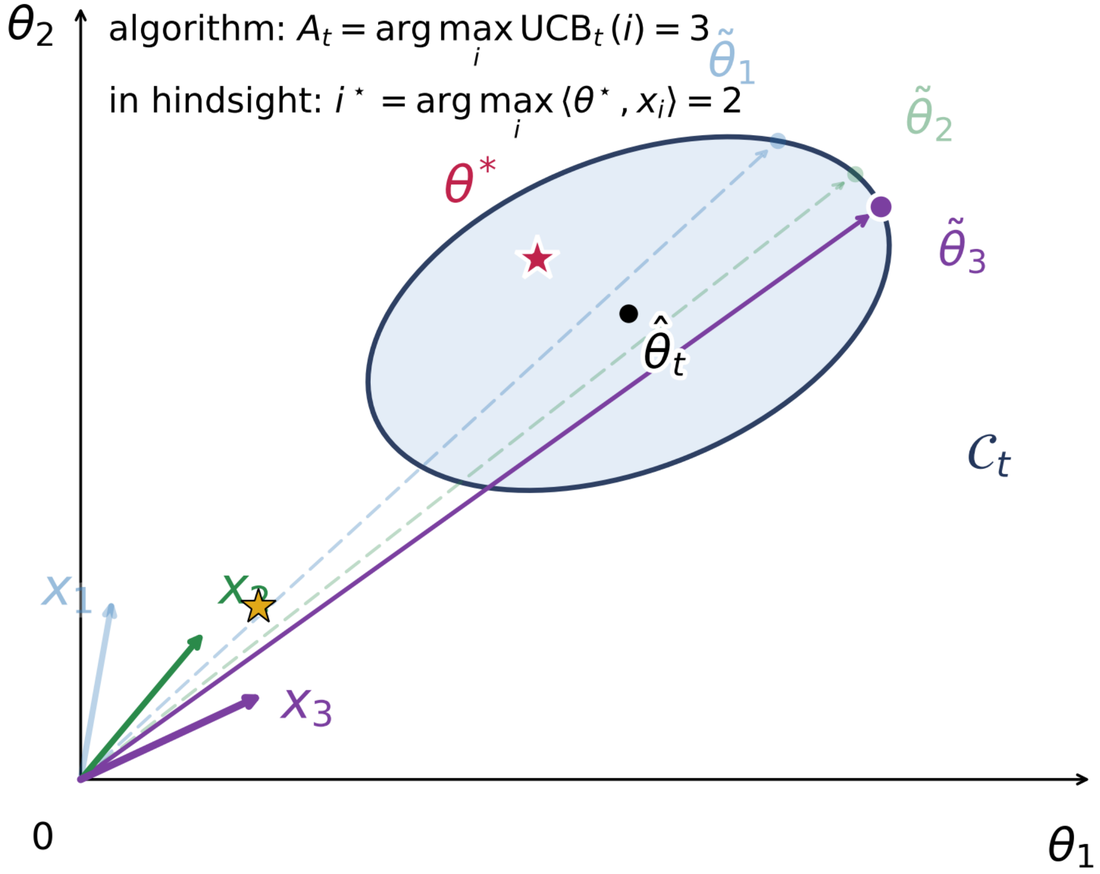
LinUCB Pseudocode

Parameter: β, λ

Initialise $V_0 = \lambda I,\ b_0 = 0$

For $t = 1,2,\ldots,T$:

- Calculate $\hat{\theta}_{t-1} = V_{t-1}^{-1} b_{t-1}$
- Play $A_t = \arg\max_{i \in [K]} \langle \hat{\theta}_{t-1}, x_i \rangle + \sqrt{\beta} \|x\|_{V_{t-1}^{-1}} \qquad (x(t) = x_{A_t})$
- Observe $y(t)$, update $V_t = V_{t-1} + x(t)x(t)^\top,\ b_t = b_{t-1} + x(t)y(t)$



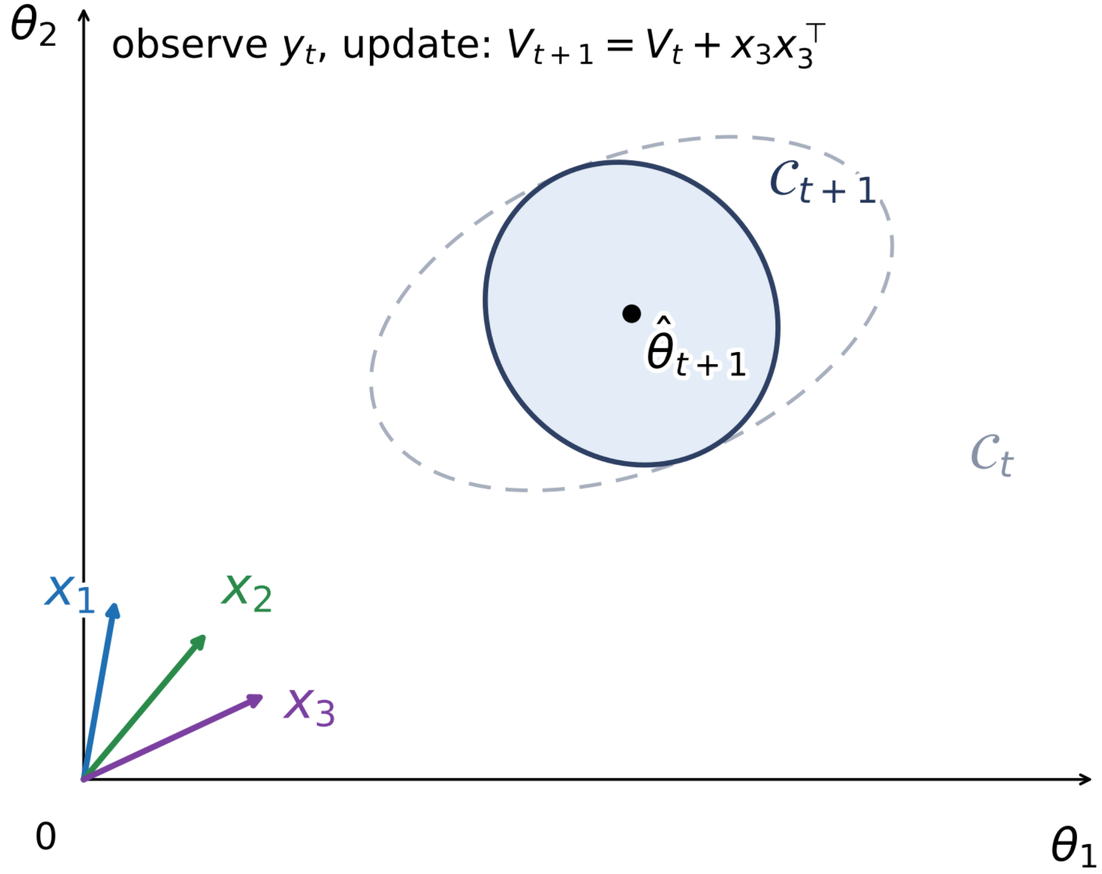
LinUCB Pseudocode

Parameter: β, λ

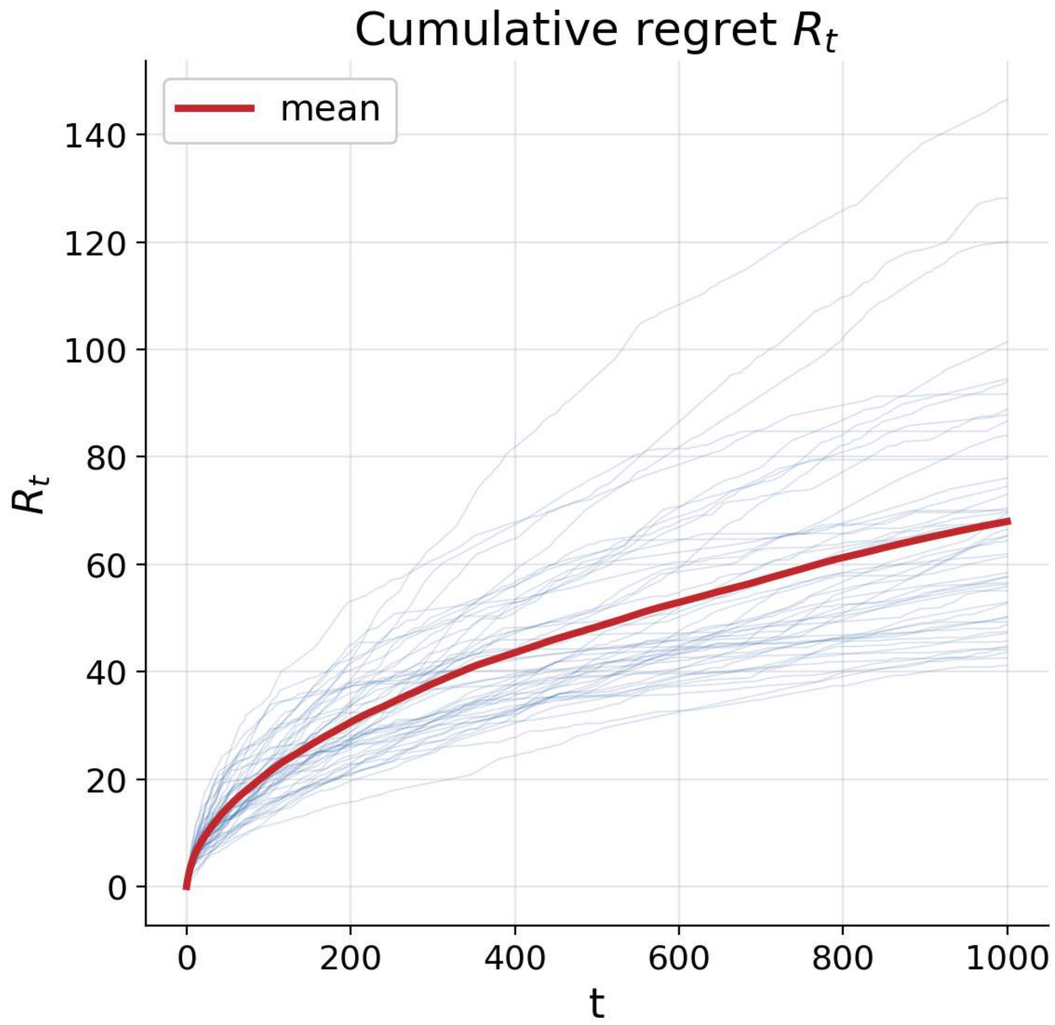
Initialise $V_0 = \lambda I,\ b_0 = 0$

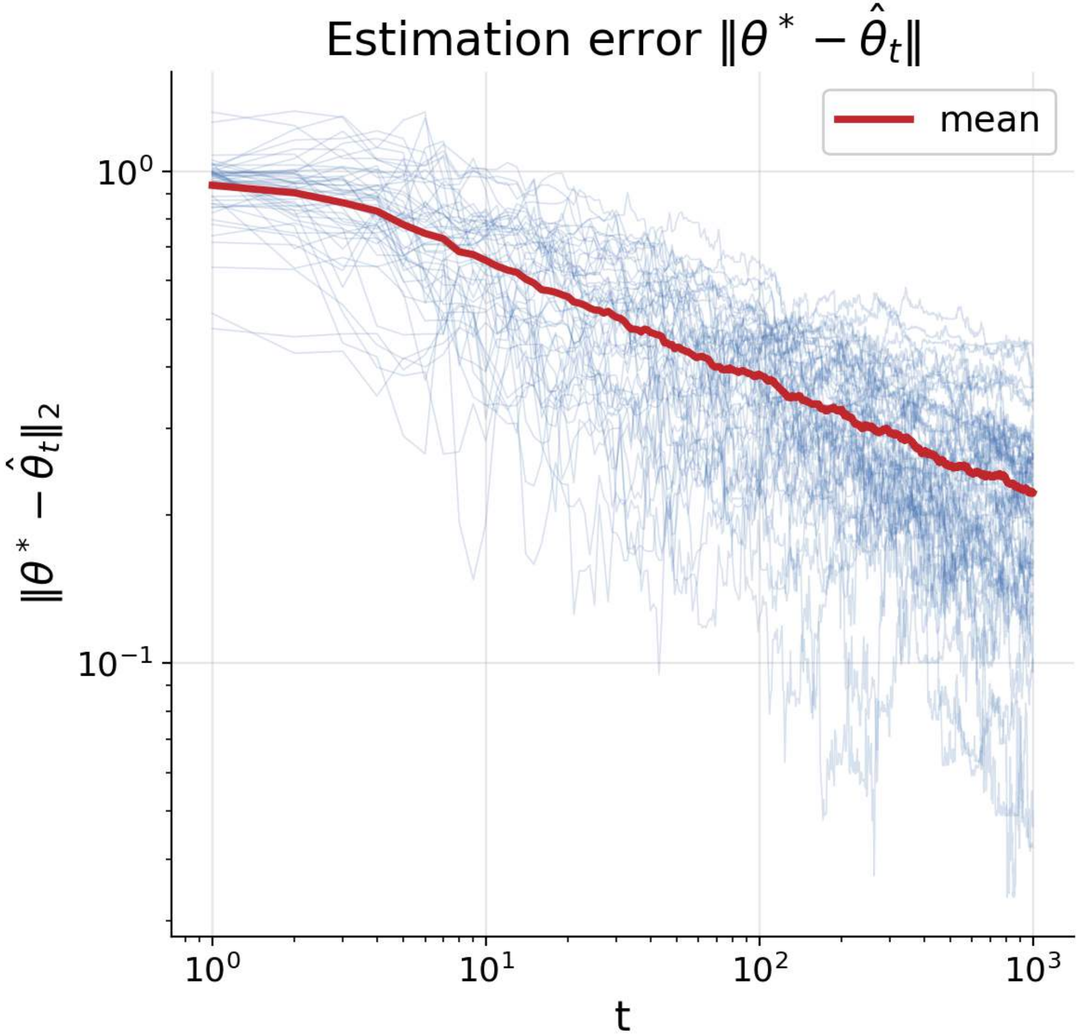
For $t = 1,2,\ldots,T$:

- Calculate $\hat{\theta}_{t-1} = V_{t-1}^{-1} b_{t-1}$
- Play $A_t = \arg\max_{i \in [K]} \langle \hat{\theta}_{t-1}, x_i \rangle + \sqrt{\beta} \|x\|_{V_{t-1}^{-1}} \qquad (x(t) = x_{A_t})$
- Observe $y(t)$, update $V_t = V_{t-1} + x(t)x(t)^\top,\ b_t = b_{t-1} + x(t)y(t)$

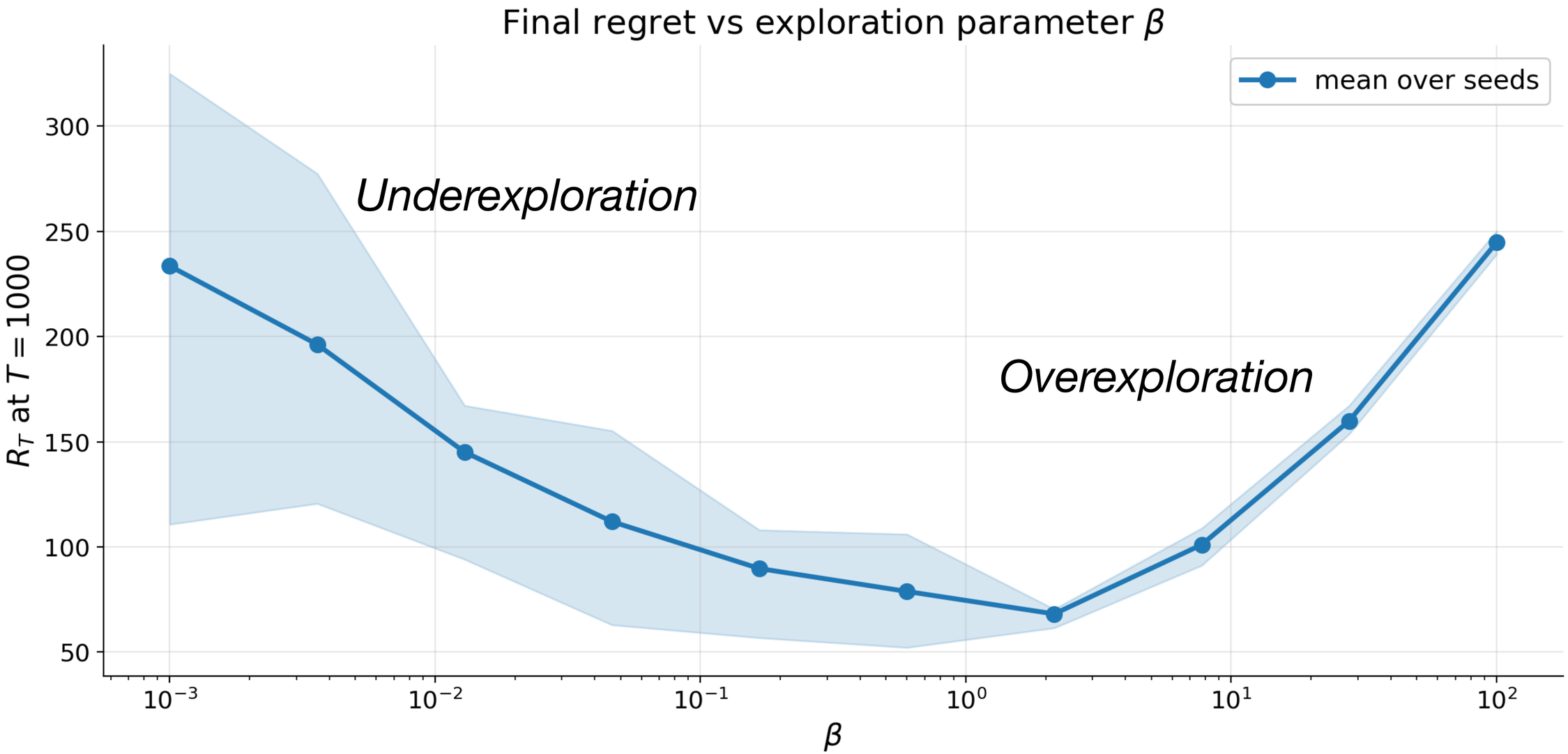


Cumulative Regret Over Time



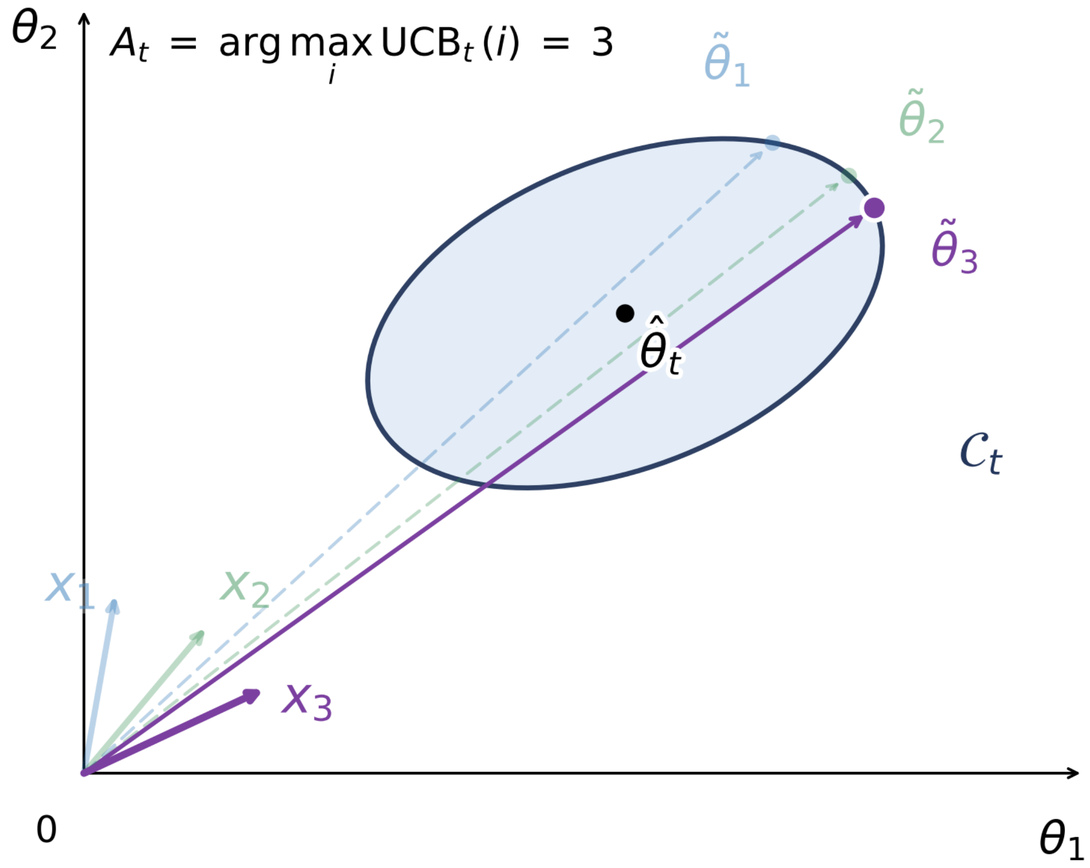


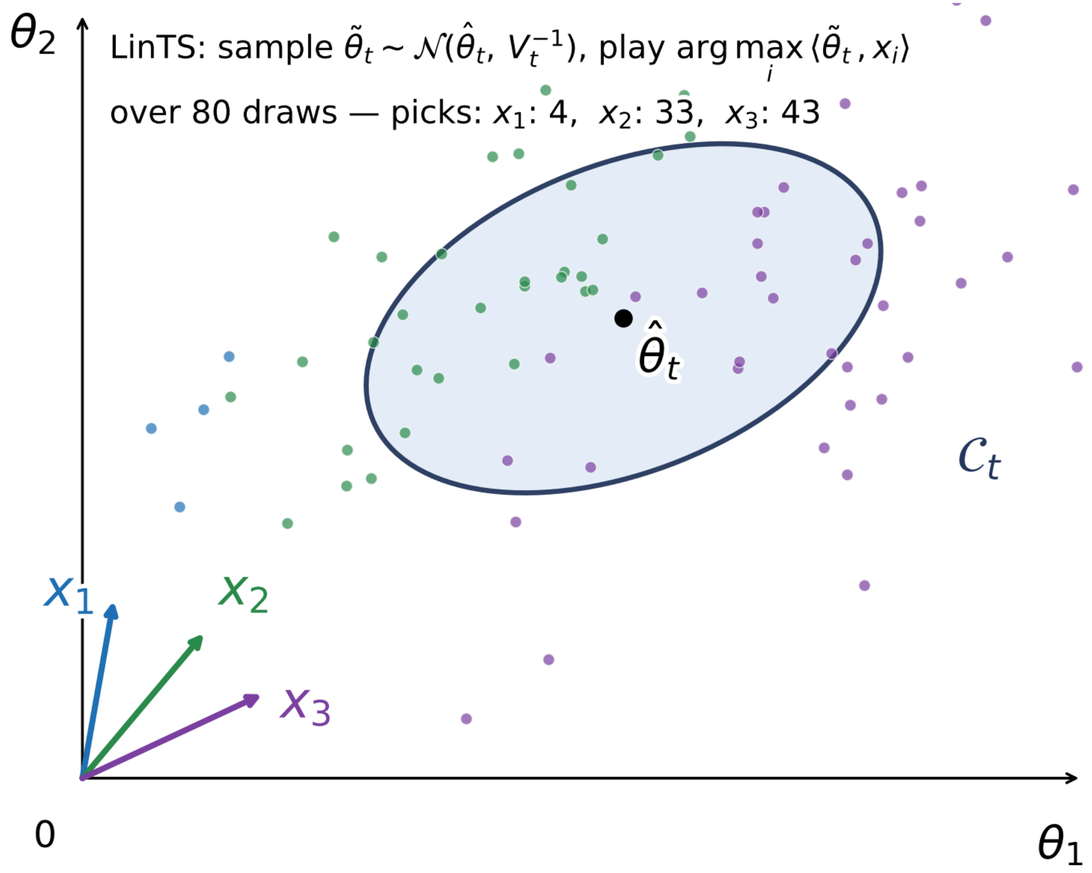
Tuning the Exploration Parameter β



LinUCB vs LinTS

Arm Picking Strategy





Important Aspects of Real-World Systems

- Feature Engineering
  - Using neural networks, incorporating user features, etc.

Important Aspects of Real-World Systems

- Feature Engineering
  - Using neural networks, incorporating user features, etc.
- Reward Modeling
  - What you observe and what you care about can be two different things!

Important Aspects of Real-World Systems

- Feature Engineering
  - Using neural networks, incorporating user features, etc.
- Reward Modeling
  - What you observe and what you care about can be two different things!
- Off-Policy Evaluation
  - Inverse Propensity Scoring (IPS) can judge new policies on prior data

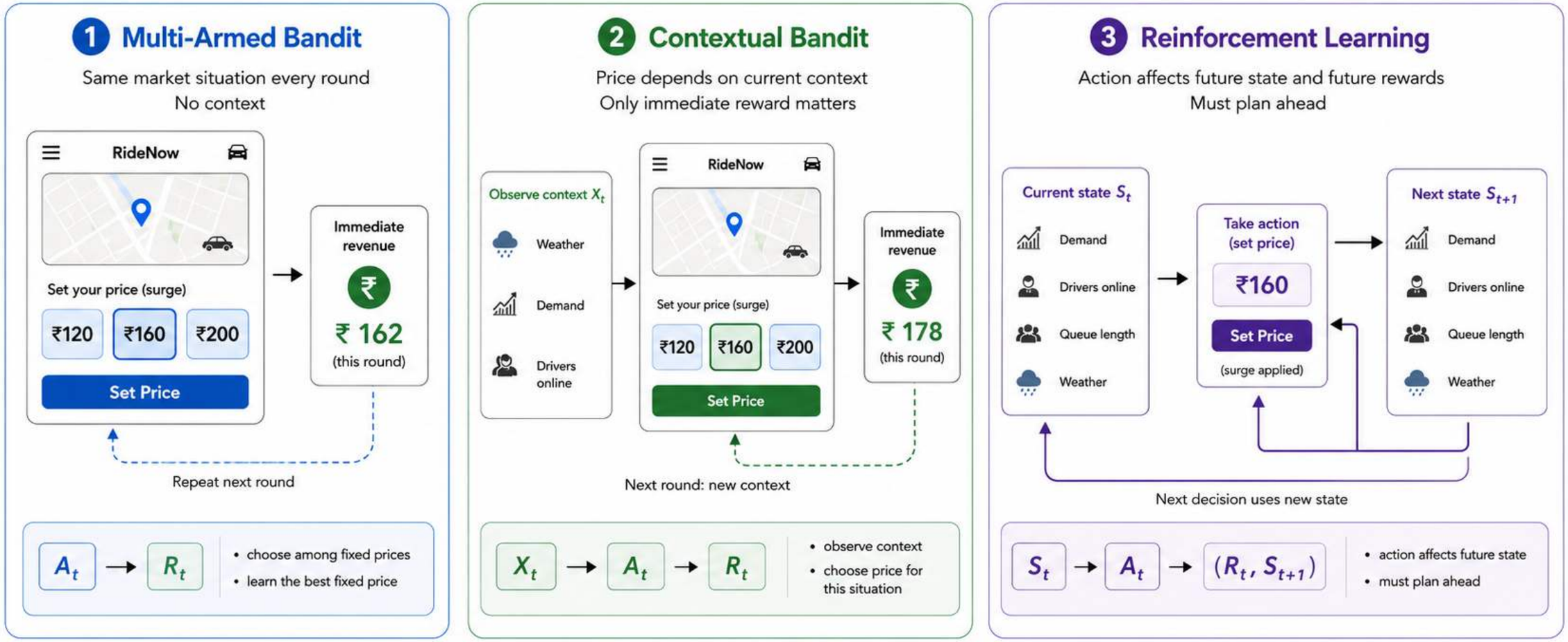
Important Aspects of Real-World Systems

- Feature Engineering
  - Using neural networks, incorporating user features, etc.
- Reward Modeling
  - What you observe and what you care about can be two different things!
- Off-Policy Evaluation
  - Inverse Propensity Scoring (IPS) can judge new policies on prior data
- Beyond single-action bandits: dueling bandits, MNL bandits, Cascade bandits

Coming up next: reinforcement learning

From Bandits to Reinforcement Learning

Understanding via the dynamic pricing example



Same goal: Maximize long-term revenue for the platform

Increasing realism: None → Current context only → Full system dynamics

Increasing complexity: Low → Medium → High

From Bandits to Reinforcement Learning

Key Differences

- A bandit model is appropriate when:
  - past actions (and rewards) give you more data to take better decisions, BUT
  - they do not significantly influence future opportunities
  - each round is essentially a fresh decision

From Bandits to Reinforcement Learning

Key Differences

- A bandit model is appropriate when:
  - past actions (and rewards) give you more data to take better decisions, BUT
  - they do not significantly influence future opportunities
  - each round is essentially a fresh decision
- We need RL when:
  - past actions affect future states, which then affect future rewards
  - there is a need for planning across time
  - being short-term greedy can be highly suboptimal

HEARTFELT THANKS TO ALL OF YOU!