Lecture 9: LinUCB and LinTS in Action

ID 6002W: Online and Reinforcement Learning

Web-enabled M.Tech. program, IIT Madras

Suryanarayana Sankagiri

9th June, 2026

Where We Are

The Estimation Problem

In linear bandits

- In linear bandits, the mean reward for each arm i is given as $\mu_i = \langle \theta, x_i \rangle$
  - where $\theta \in \mathbb{R}^d$ is an unknown vector, and $x_i \in \mathbb{R}^d$ is known for each arm i

- Instead of directly estimating μ_i, we estimate θ from $\big(x(1), y(1), \ldots x(T), y(T)\big)$
  - $y(t) = \langle \theta, x(t) \rangle + \eta_t$, so we use linear regression (least-squares) to calculate $\hat{\theta}$

- Bandit algorithms also require an uncertainty estimate over $\hat{\theta}$

Quantifying Uncertainty

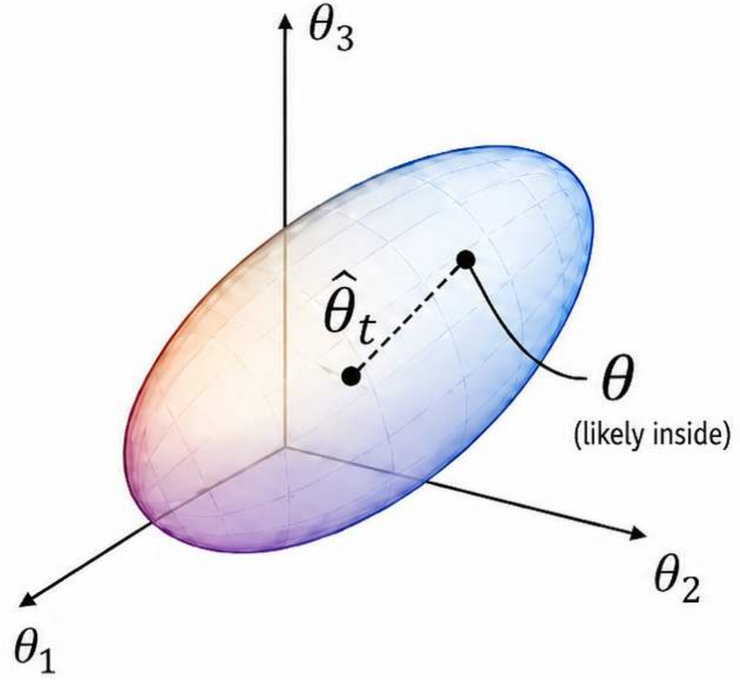
In linear bandits

- 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

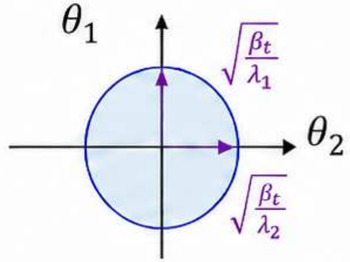
Confidence ellipsoid



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

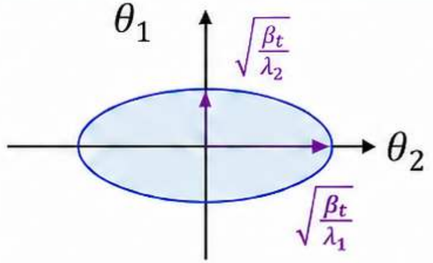
$$V_t = \sum x(s)x(s)^\top$$

How V_t shapes the set



① $\lambda_1 = \lambda_2$

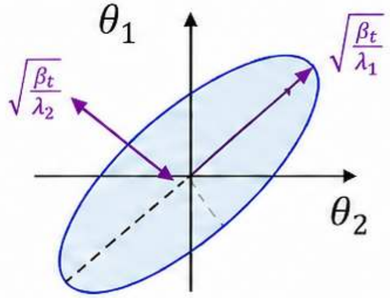
equal uncertainty in all directions



② $\boldsymbol{V}_t = \mathrm{diag}(\lambda_1, \lambda_2)$

larger λ_i → shorter axis

smaller λ_i → longer axis



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

eigenvectors set orientation

Larger eigenvalue → more information → shorter axis.

Optimism Principle for Linear Bandits

The LinUCB algorithm

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

For all arms $i \in [K]$:

Compute $UCB_t(i) = \max_{\theta \in \mathscr{C}_t} \langle \theta, x_i \rangle$

Play $A_t = \arg\max_{i \in [K]} UCB_{t-1}(i)$

Observe $y(t)$, update $\hat{\theta}_t, V_t, \mathscr{C}_t$

- $\mathscr{C}_t = \left\{ \theta \in \mathbb{R}^d : \|\theta - \hat{\theta}_{t-1}\|^2_{V_{t-1}} \leq \beta \right\}$
- $V_t = \lambda I + \sum_{s=1}^{t} x(s)x(s)^\top$
- $\hat{\theta}_t = V_t^{-1} b_t$
- $b_t = \sum_{s=1}^{t} x(s)y(s)$

Making LinUCB Practical

Ease of computation

- The UCB rule requires computing $UCB_t(i) = \max_{\theta \in \mathscr{C}_{t-1}} \langle \theta, x_i \rangle$
- If $\mathscr{C}_t$ is an arbitrary set, this computation can be very hard!
- Here, $\mathscr{C}_t$ is an ellipse centred around $\hat{\theta}_t$. This simplifies the expression:

$$\max_{\theta \in \mathscr{C}_t} \langle \theta, x \rangle = \langle \hat{\theta}_t, x \rangle + \sqrt{\beta} \|x\|_{V_t^{-1}}$$

Making LinUCB Practical

Ease of computation

- The UCB rule requires computing $UCB_t(i) = \max_{\theta \in \mathscr{C}_{t-1}} \langle \theta, x_i \rangle$
- If $\mathscr{C}_t$ is an arbitrary set, this computation can be very hard!
- Here, $\mathscr{C}_t$ is an ellipse centred around $\hat{\theta}_t$. This simplifies the expression:

$$\max_{\theta \in \mathscr{C}_t} \langle \theta, x \rangle = \langle \hat{\theta}_t, x \rangle + \sqrt{\beta} \|x\|_{V_t^{-1}}$$

Cauchy-Schwarz inequality:

$$\langle \theta, x \rangle \leq \|\theta\|_{V_t} \|x\|_{V_t^{-1}}$$

- The maximiser, $\tilde{\theta}_t$, will always lie on the boundary of the ellipse

Practical Pseudocode

Parameter: β

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 $V_t = V_{t-1} + x(t)x(t)^\top,\ b_t = b_{t-1} + x(t)y(t)$

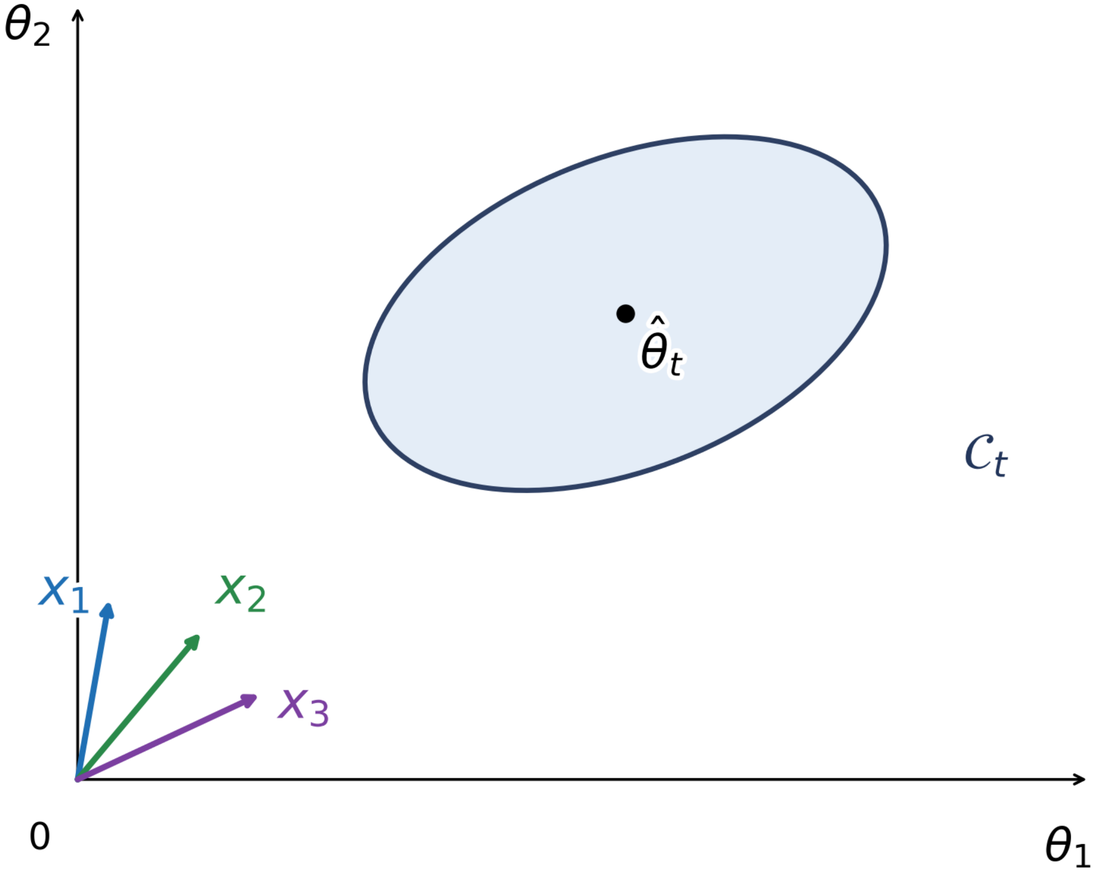
Practical Pseudocode

Parameter: β

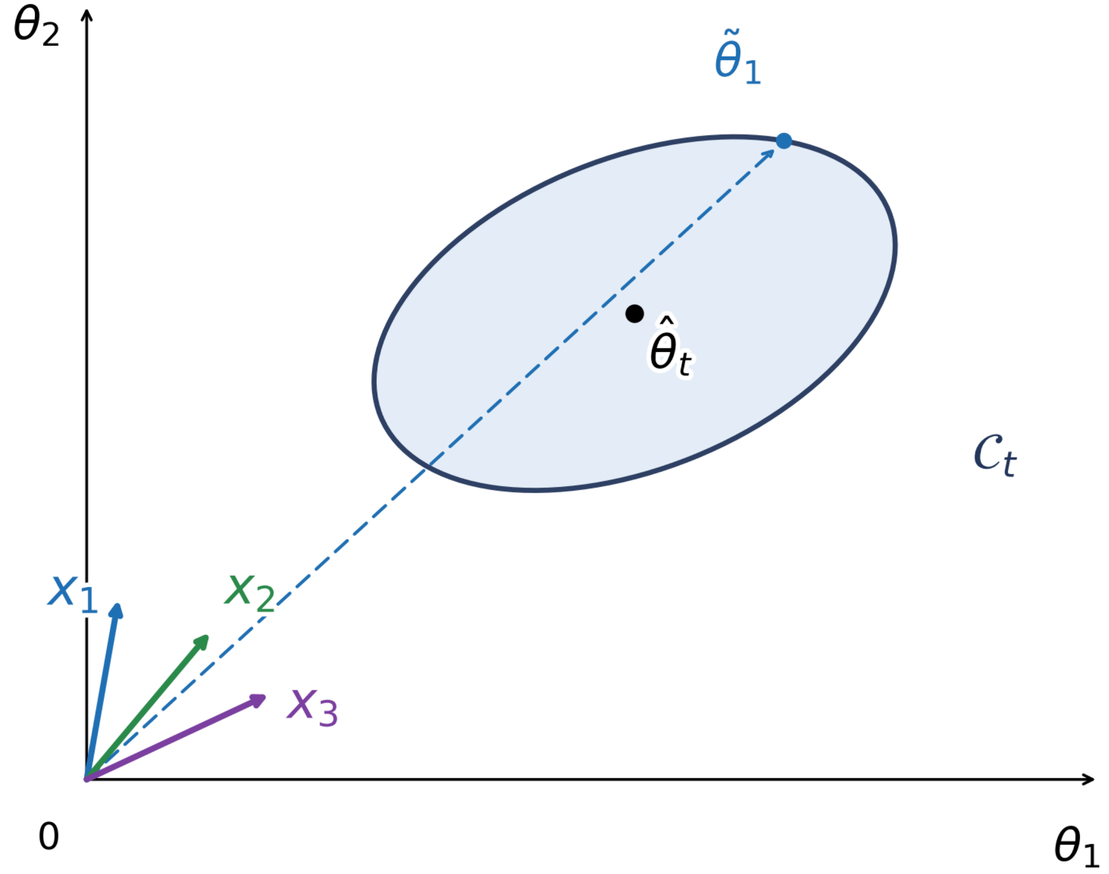
Initialise $V_0 = \lambda I,\ b_0 = 0$

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 $V_t = V_{t-1} + x(t)x(t)^\top,\ b_t = b_{t-1} + x(t)y(t)$



Practical Pseudocode



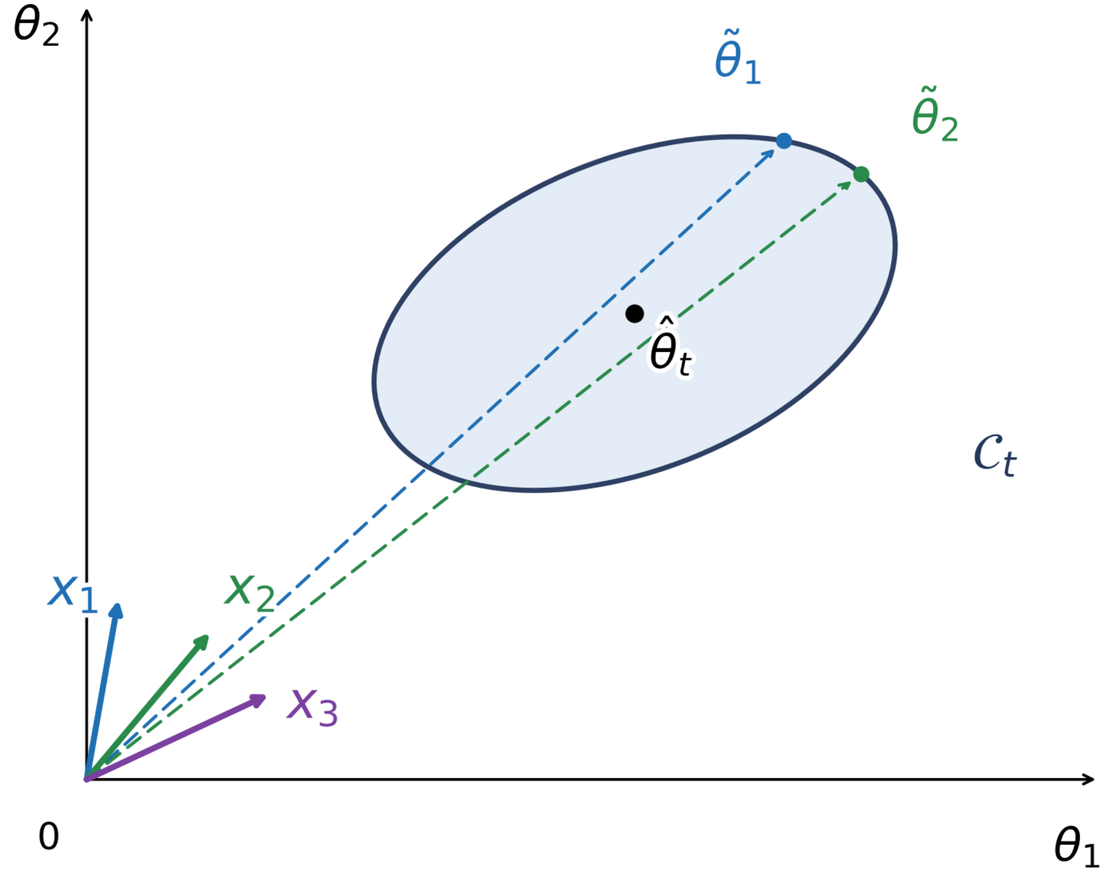
Parameter: β

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 $V_t = V_{t-1} + x(t)x(t)^\top,\ b_t = b_{t-1} + x(t)y(t)$

Practical Pseudocode



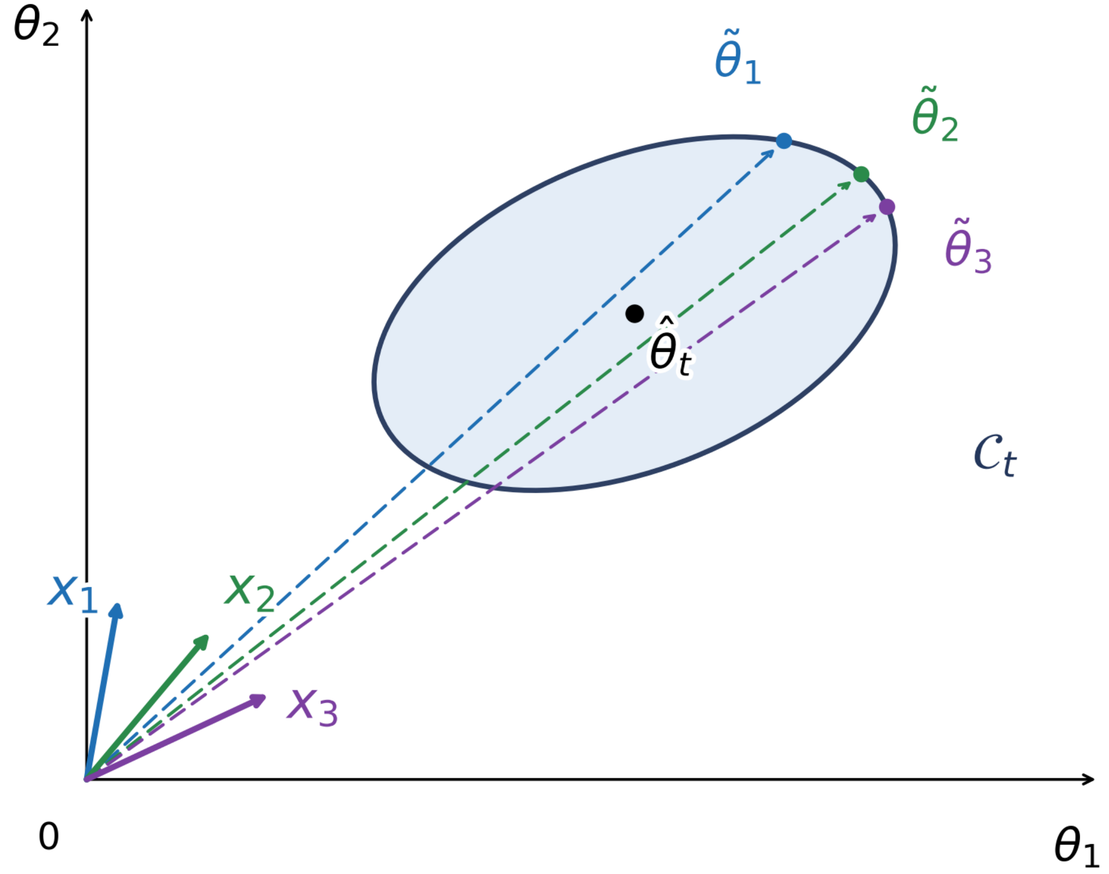
Parameter: β

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 $V_t = V_{t-1} + x(t)x(t)^\top,\ b_t = b_{t-1} + x(t)y(t)$

Practical Pseudocode



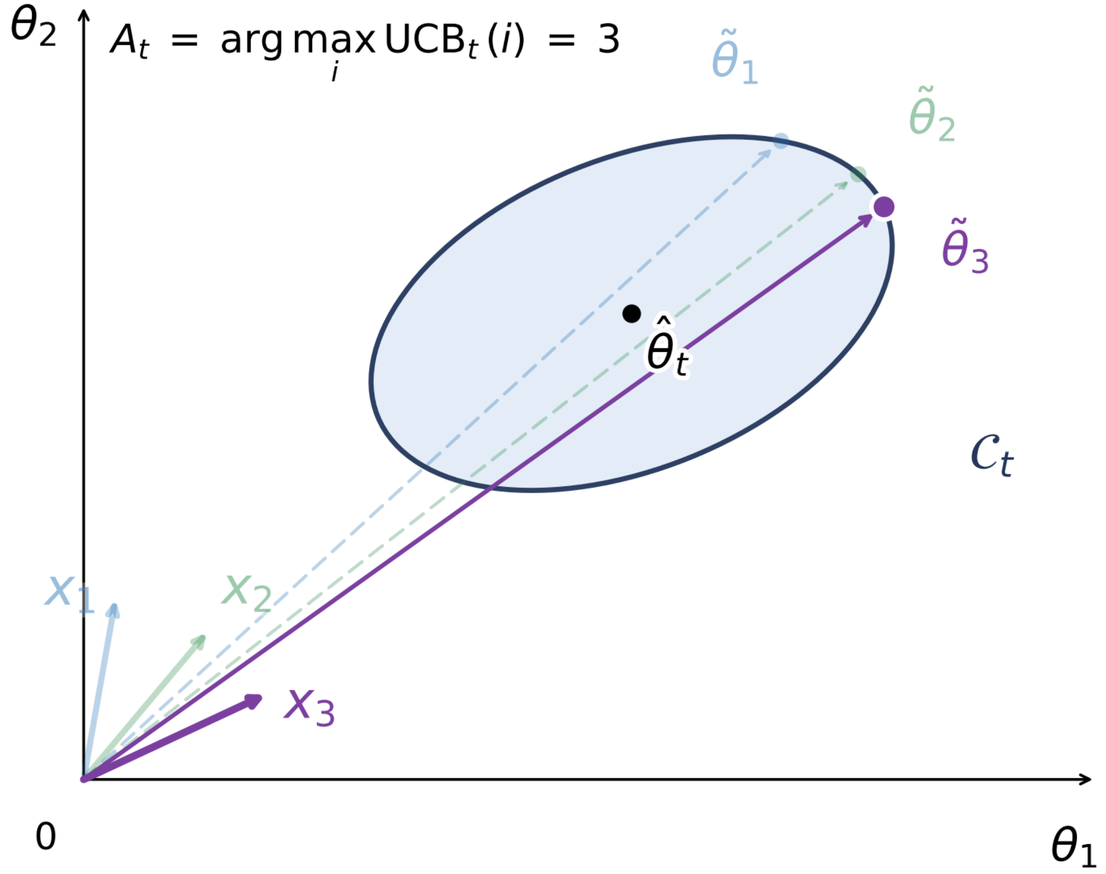
Parameter: β

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 $V_t = V_{t-1} + x(t)x(t)^\top,\ b_t = b_{t-1} + x(t)y(t)$

Practical Pseudocode



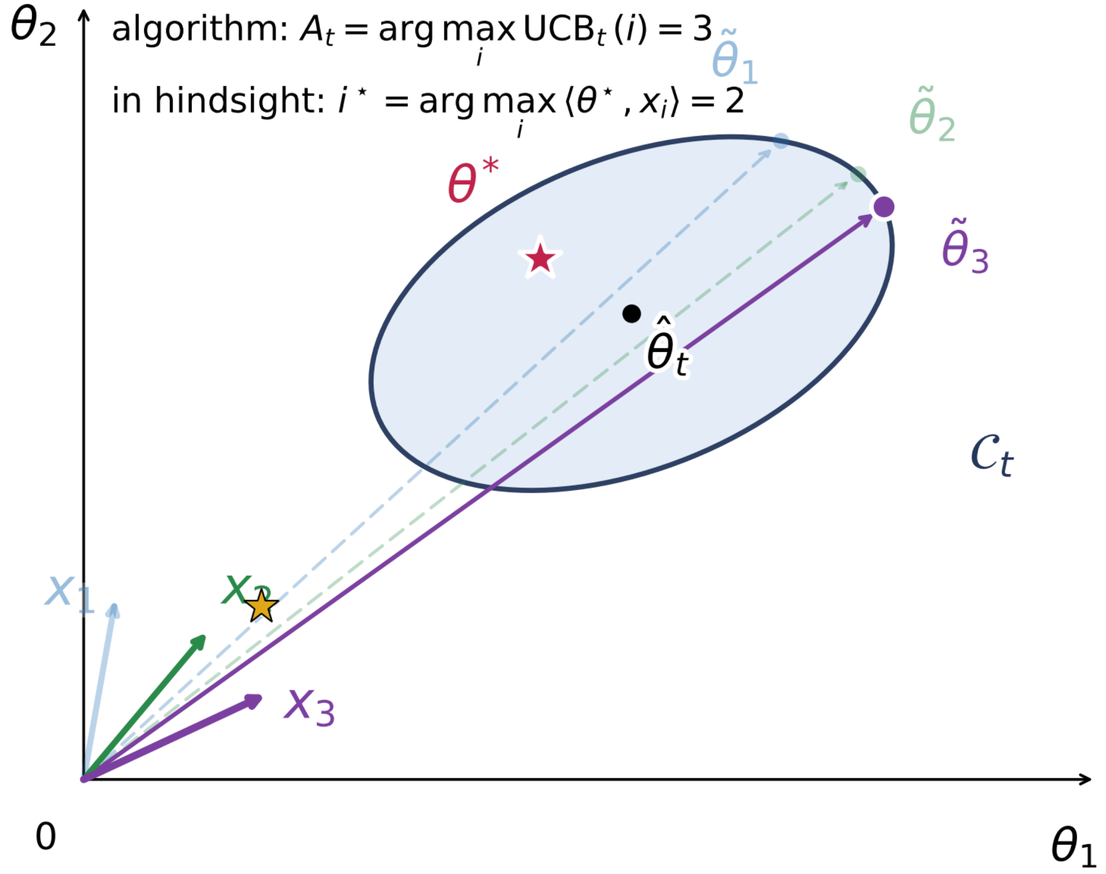
Parameter: β

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 $V_t = V_{t-1} + x(t)x(t)^\top,\ b_t = b_{t-1} + x(t)y(t)$

Practical Pseudocode



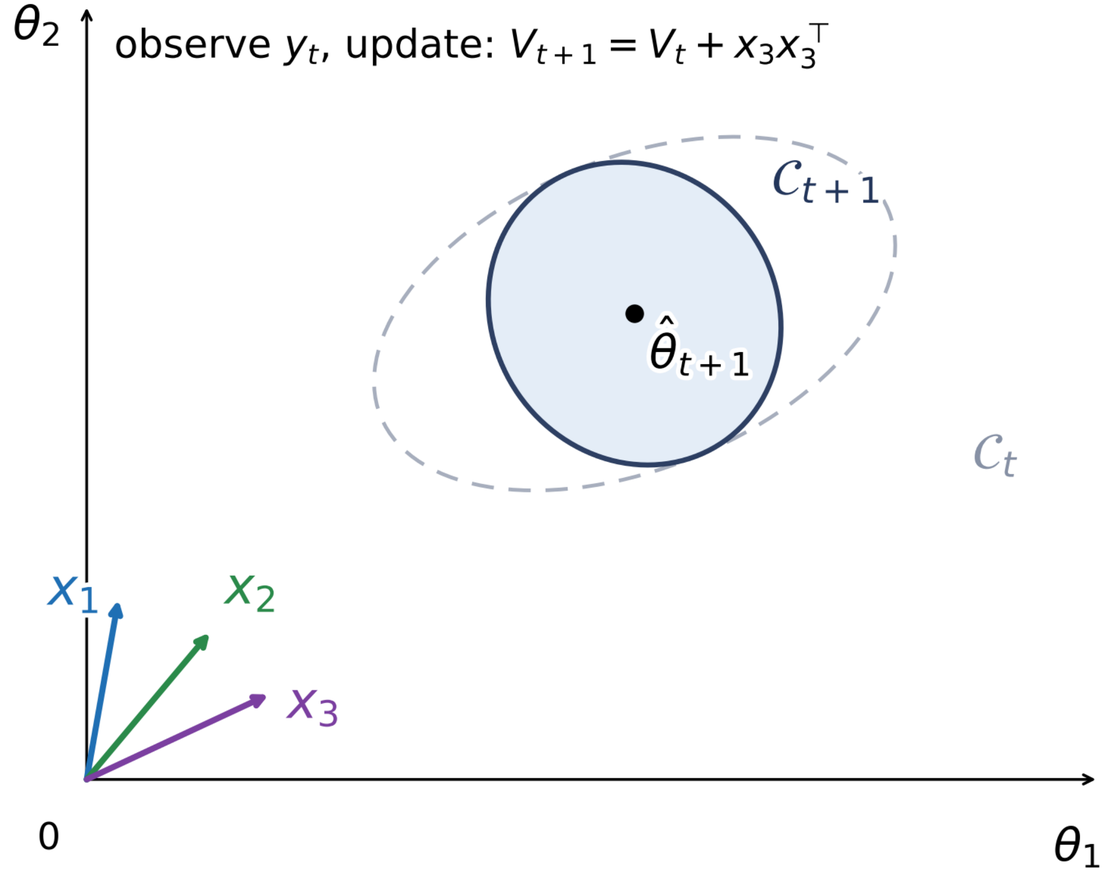
Parameter: β

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 $V_t = V_{t-1} + x(t)x(t)^\top,\ b_t = b_{t-1} + x(t)y(t)$

Practical Pseudocode



Parameter: β

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 $V_t = V_{t-1} + x(t)x(t)^\top,\ b_t = b_{t-1} + x(t)y(t)$

Making LinUCB Practical

Understanding the exploration bonus term

- The core of LinUCB now boils down to a single line:

$$A_t = \arg\max_{i \in [K]} \langle \hat{\theta}_t, x_i \rangle + \sqrt{\beta} \|x_i\|_{V_{t-1}^{-1}}$$

Making LinUCB Practical

Understanding the exploration bonus term

- The core of LinUCB now boils down to a single line:

$$A_t = \arg\max_{i \in [K]} \langle \hat{\theta}_t, x_i \rangle + \sqrt{\beta} \|x_i\|_{V_{t-1}^{-1}}$$

- Compare and contrast with the UCB rule of simple bandits:

$$A_t = \arg\max_{i \in [K]} \hat{\mu}_i(t) + \sqrt{\frac{c \log T}{T_i(t)}}$$

Making LinUCB Practical

Understanding the exploration bonus term

- The core of LinUCB now boils down to a single line:

$$A_t = \arg\max_{i \in [K]} \langle \hat{\theta}_t, x_i \rangle + \sqrt{\beta} \|x_i\|_{V_{t-1}^{-1}}$$

- Compare and contrast with the UCB rule of simple bandits:

$$A_t = \arg\max_{i \in [K]} \hat{\mu}_i(t) + \sqrt{\frac{c \log T}{T_i(t)}}$$

- In linear bandits, the uncertainty in the reward for arm x_i *does not depend on* $T_i(t)$
  - It depends on the eigenvectors of V_t, which in turn depends on $x(1), \ldots, x(t)$

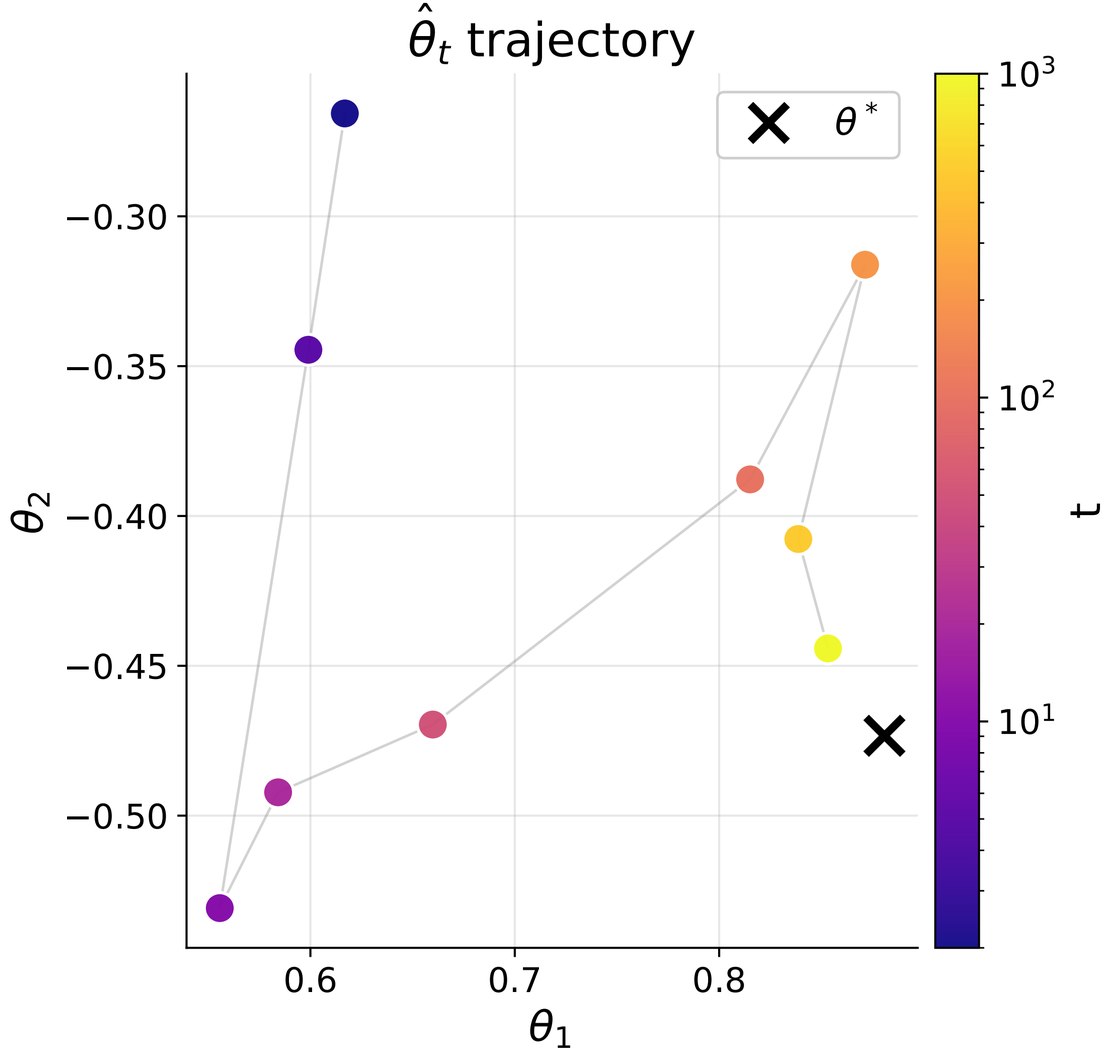
Making LinUCB Practical

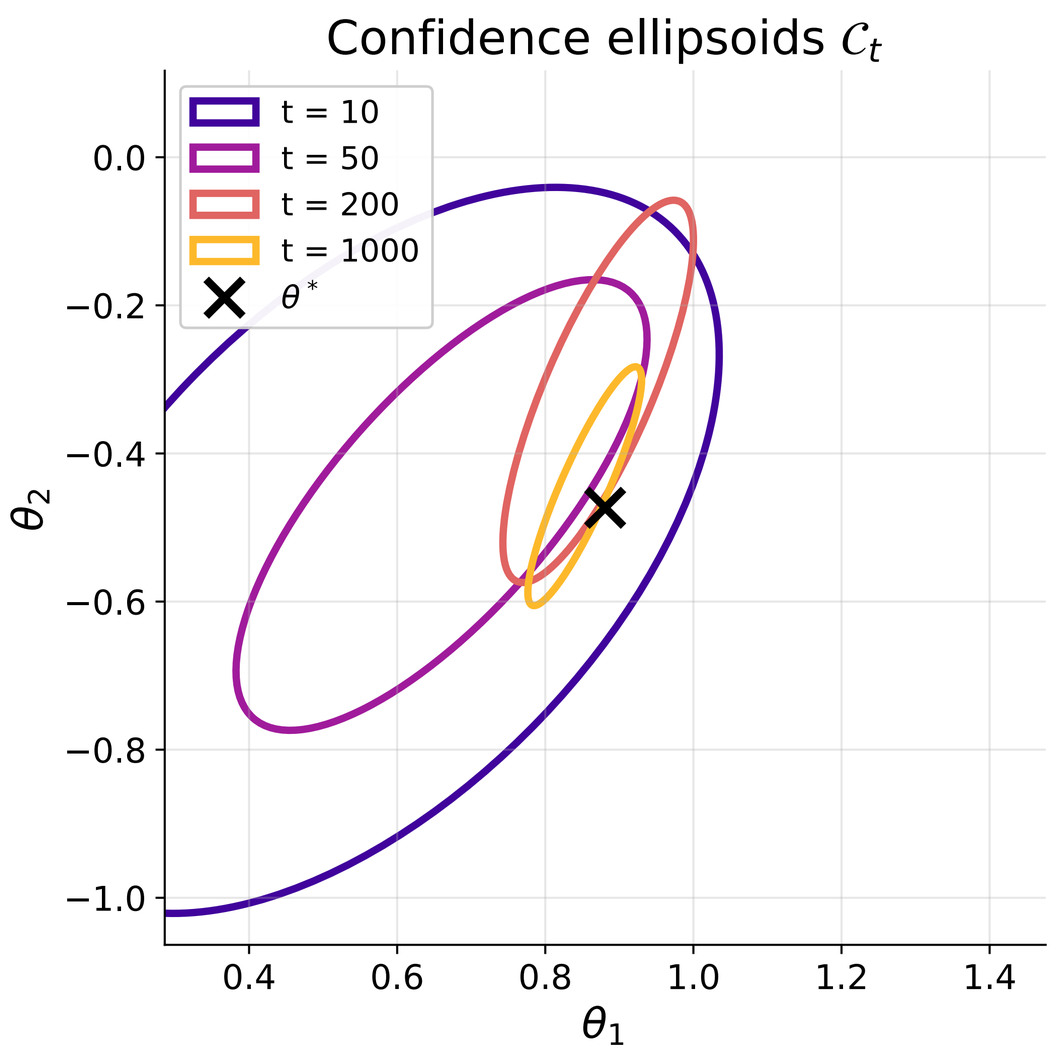
Understanding the exploration bonus term

How does exploration reduce over time?

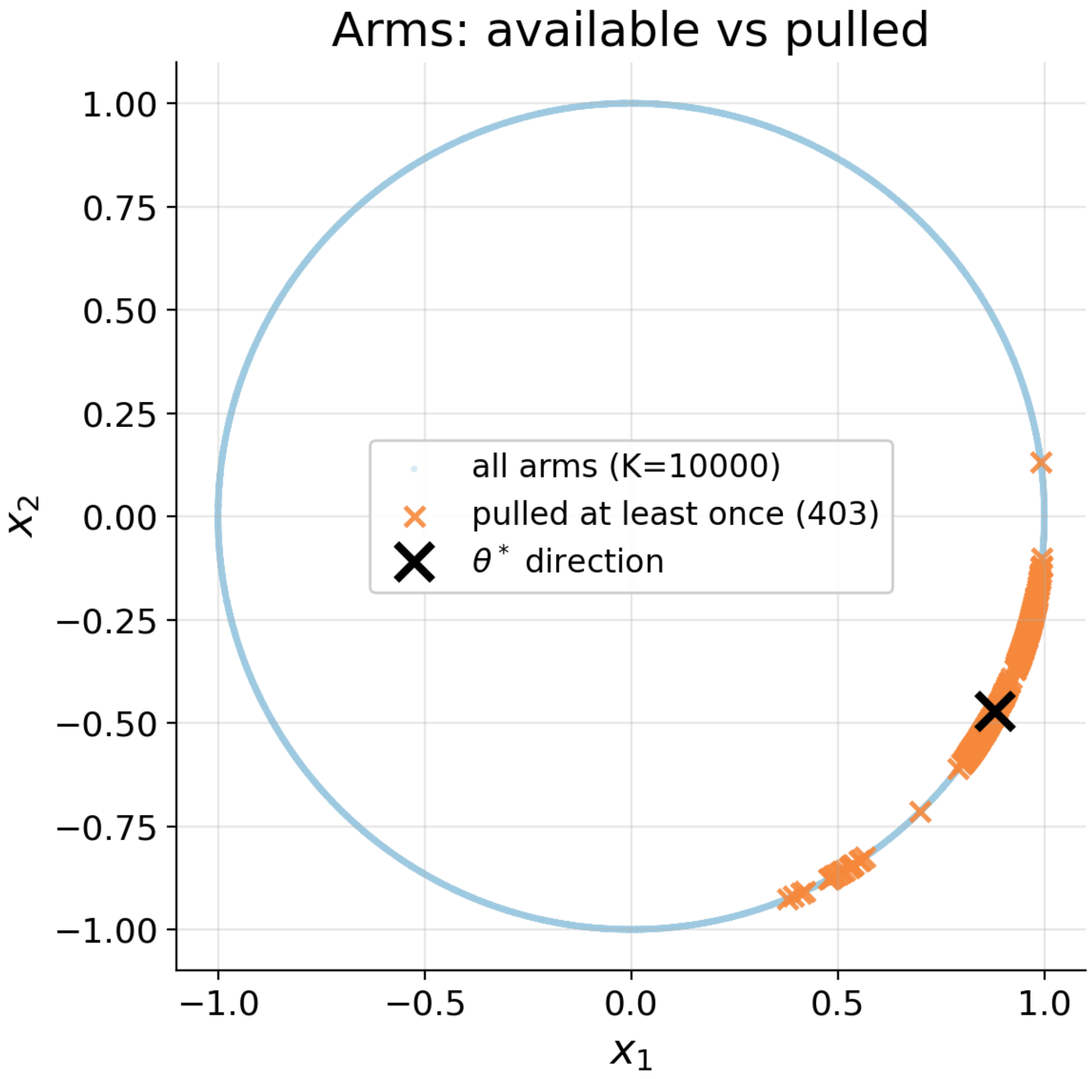
- Playing arm x adds $xx^\top$ to V
- This widens V along x (increases those eigenvalues whose eigenvectors are close to x)
- This shrinks the ellipsoid along x
- Equivalently, reduces the exploration bonus $\|x\|_V$
- Reduces the chance that the arm is pulled again, unless it is genuinely good

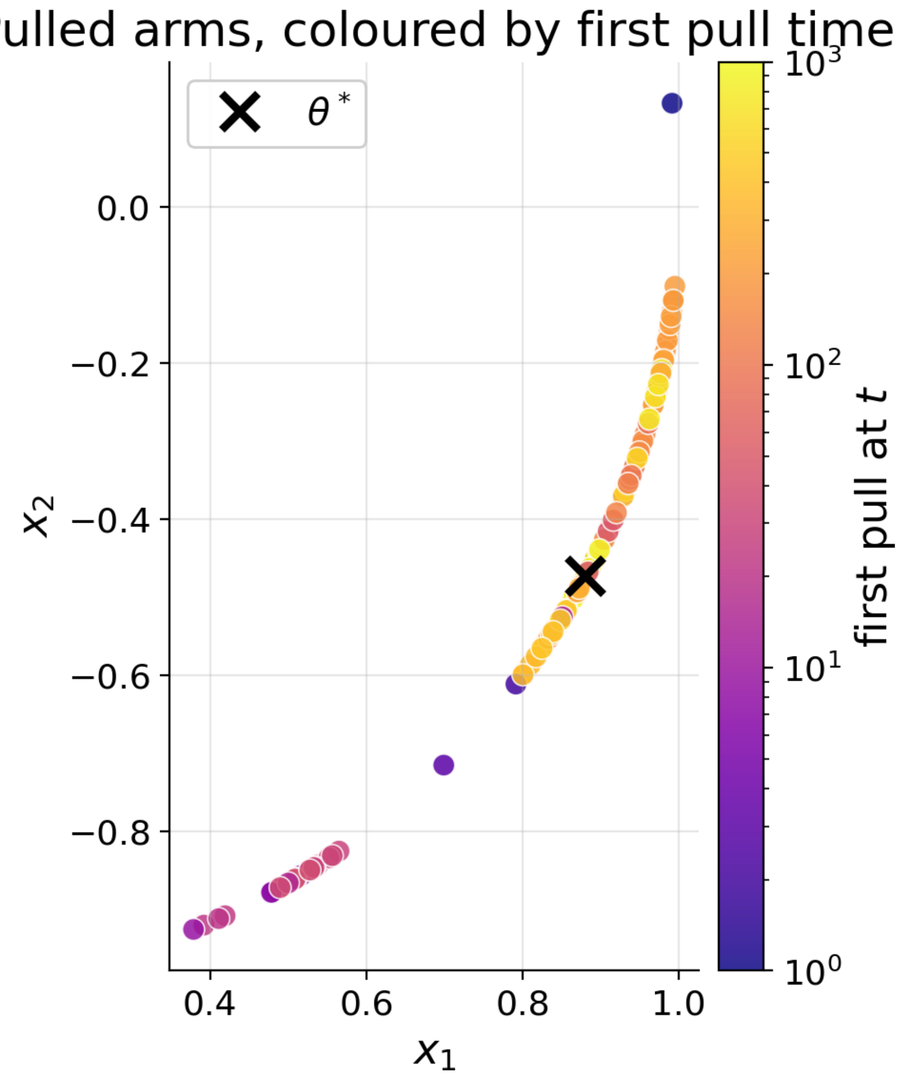
Improving Estimates Over Time



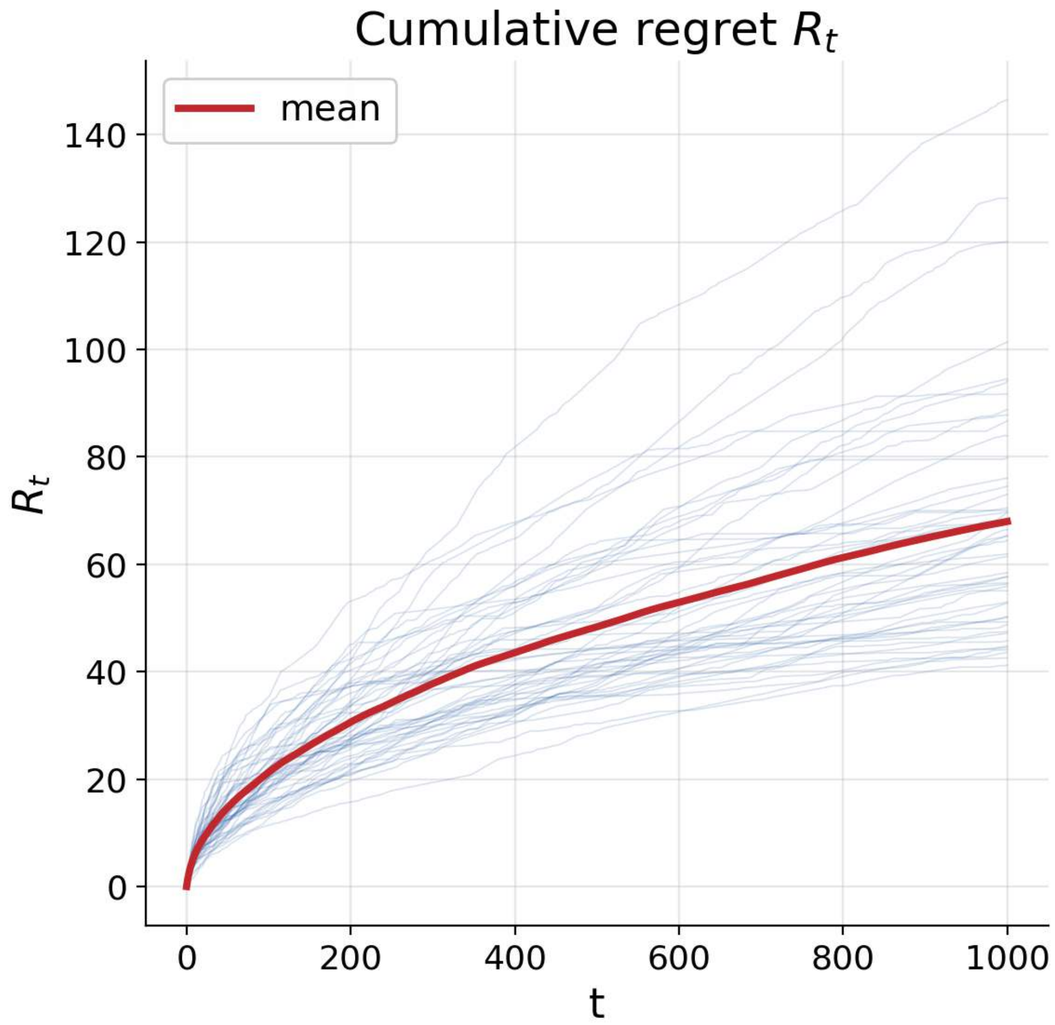


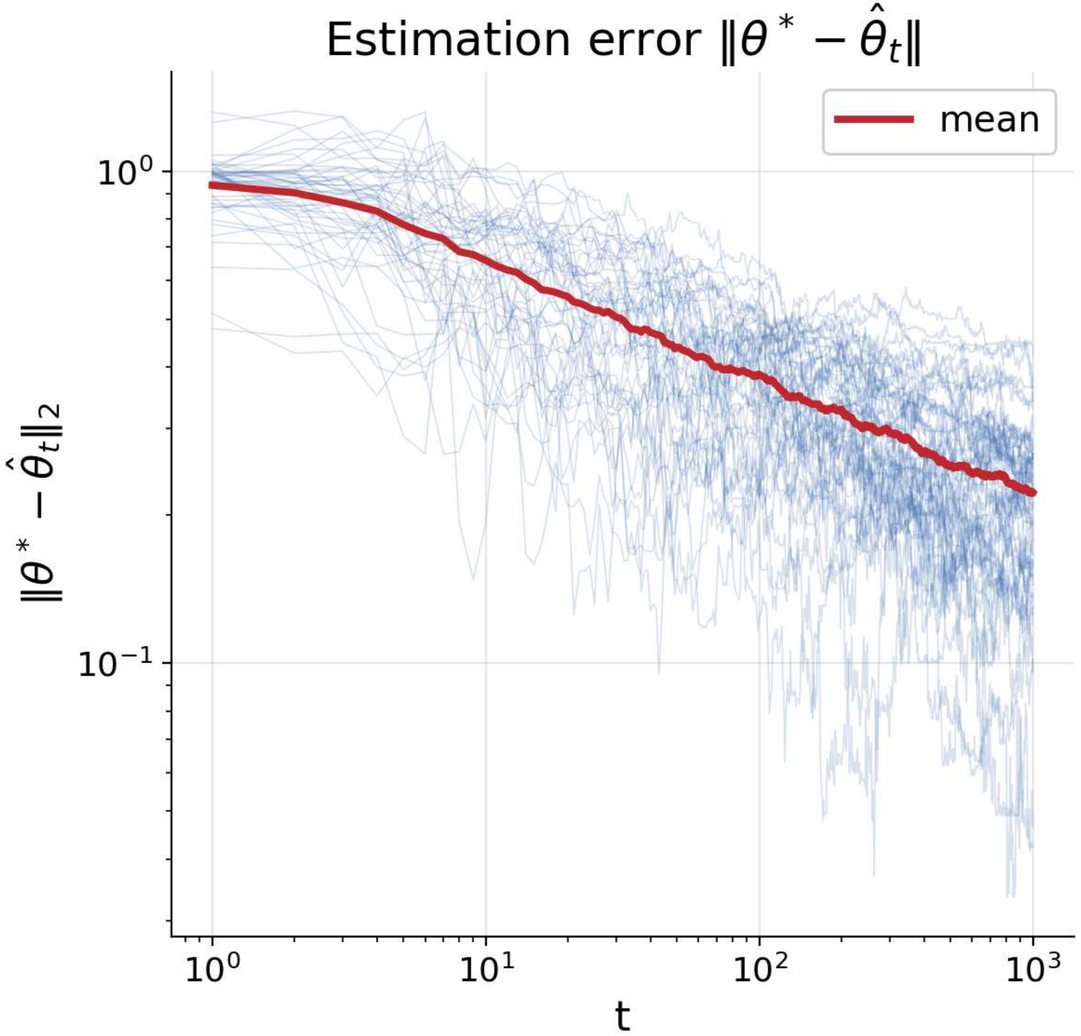
From Exploration to Exploitation





Cumulative Regret Over Time





Regret Guarantee

For linear bandits

- For linear bandits, $R(T) = \sum_{t=1}^{T} \left(\mu^* - \mathbb{E}[\mu_{A_t}] \right) = \sum_{t=1}^{T} \left\langle \theta, x^* - \mathbb{E}[x_{A_t}] \right\rangle$

Regret Guarantee

For linear bandits

- For linear bandits, $R(T) = \sum_{t=1}^{T} \left(\mu^* - \mathbb{E}[\mu_{A_t}] \right) = \sum_{t=1}^{T} \left\langle \theta, x^* - \mathbb{E}[x_{A_t}] \right\rangle$
- Assume $\|x\|,\ \|\theta\| \leq 1 \Rightarrow$ worst-case *instantaneous regret* is at most 1
- Let β be chosen such that with high probability,

$$\forall\ t \in [T], \theta \in \mathscr{C}_t$$

Regret Guarantee

For linear bandits

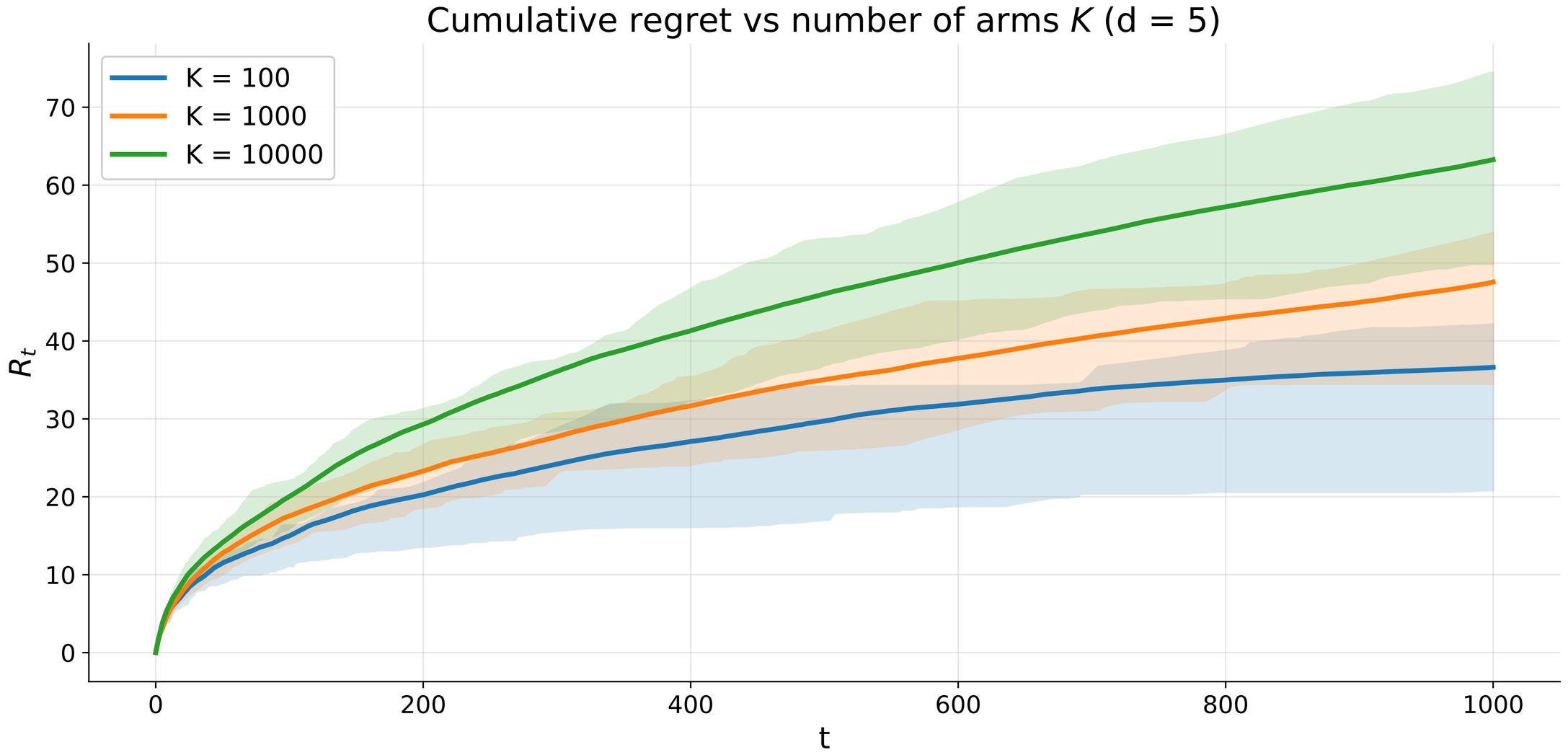
- For linear bandits, $R(T) = \sum_{t=1}^{T} \left(\mu^* - \mathbb{E}[\mu_{A_t}] \right) = \sum_{t=1}^{T} \left\langle \theta, x^* - \mathbb{E}[x_{A_t}] \right\rangle$
- Assume $\|x\|,\ \|\theta\| \leq 1 \Rightarrow$ worst-case *instantaneous regret* is at most 1
- Let β be chosen such that with high probability,

$$\forall\ t \in [T], \theta \in \mathscr{C}_t$$

Then, the regret of LinUCB satisfies $R(T) \leq C \cdot d\sqrt{T} \log T$

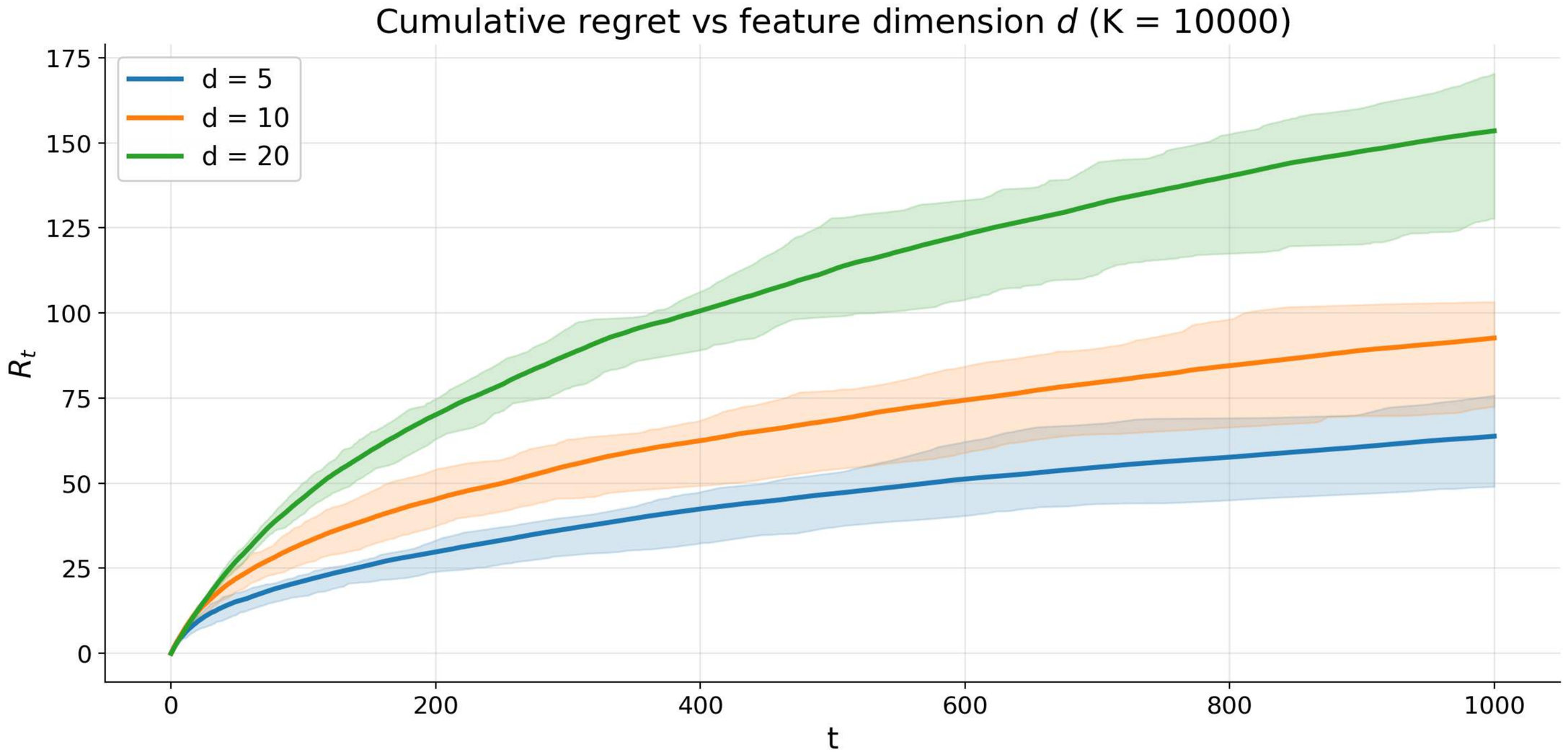
Cumulative Regret

Growth with number of arms

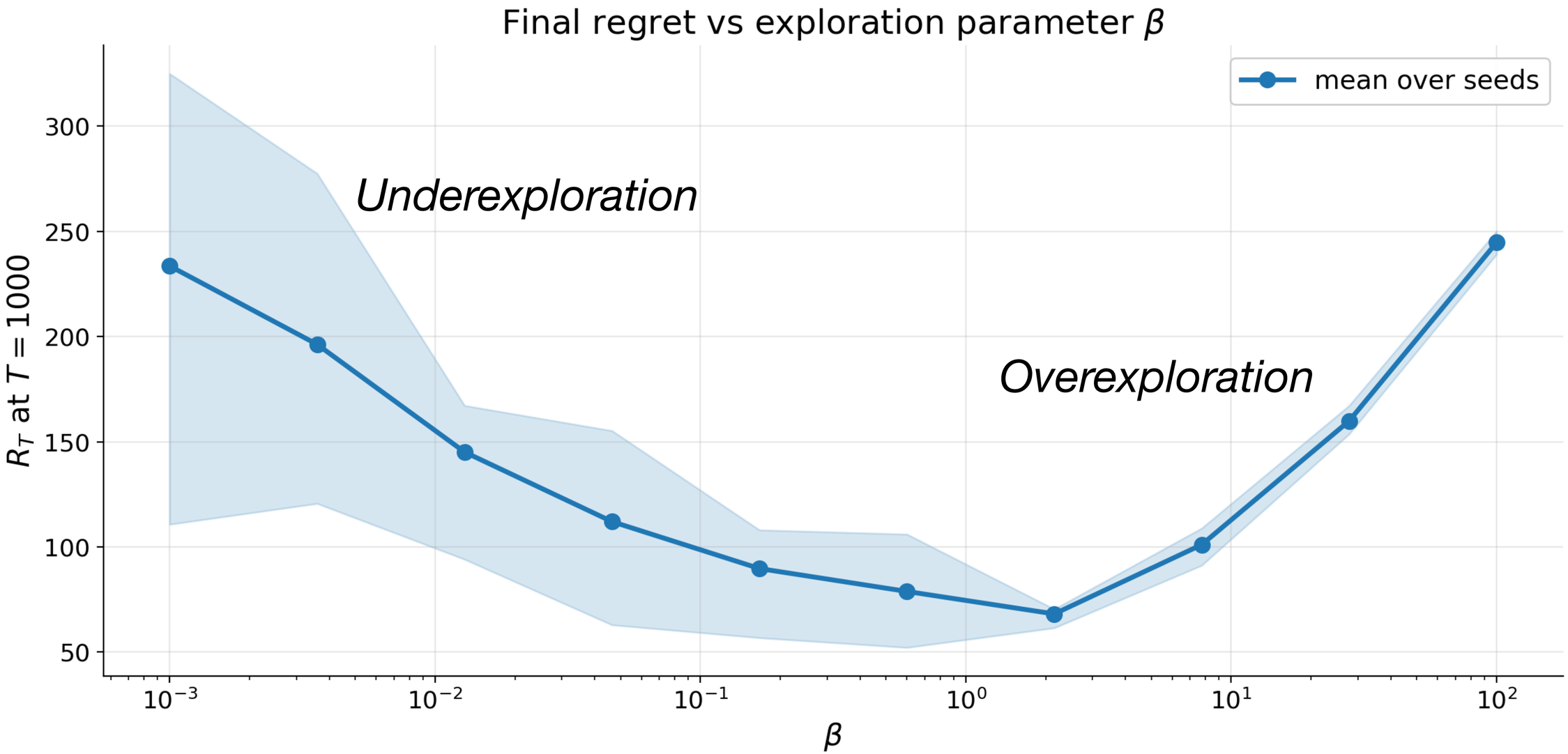


Cumulative Regret

Growth with dimension



Tuning the Exploration Parameter β



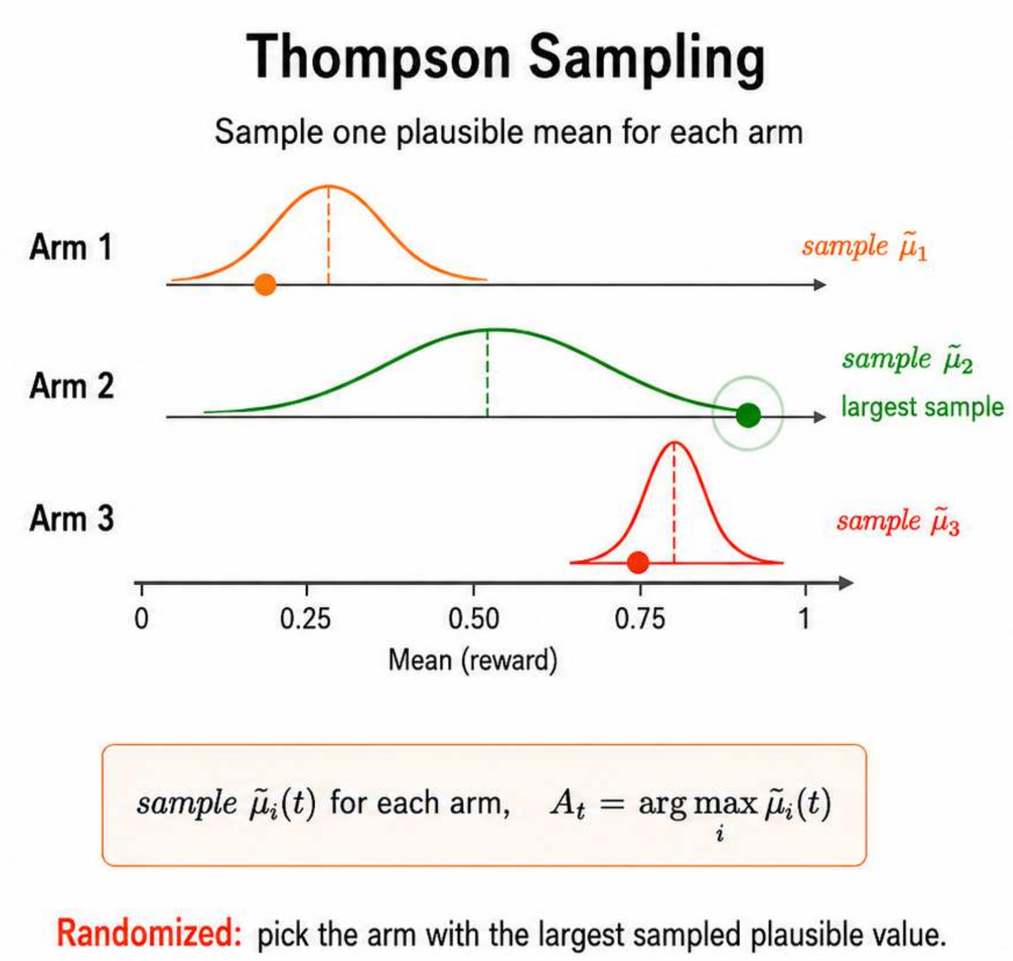
Recap of Linear Bandits

- Linear bandits model adds structure to mean rewards: $\mu_i = \langle \theta, x_i \rangle$
- Pulling one arm gives information about similar arms $\Rightarrow$ much lower regret, if $d \ll K$
- We estimate $\hat{\theta}$ using least-squares (linear regression). But greedy is not enough!
- LinUCB maintains a confidence ellipsoid around $\hat{\theta}$, then act optimistically
- Is optimism the only way to act here?

Can we extend the idea of Thompson Sampling to Linear Bandits?

Thompson Sampling for Linear Bandits

- Recall Thompson Sampling for multi-armed bandits
- We maintain a belief distribution over the unknown arm means μ_i
- We sample from each of these distributions, and play the arm with the largest sampled mean
- In linear bandits, the unknown quantity is θ
- $\Rightarrow$ we need to maintain a belief distribution over θ
- What distribution should we use?



A Gaussian Belief Over θ

Bayesian linear regression viewpoint

- As before, maintain $V_t = \sum x(s)x(s)^\top, \quad b_t = \sum x(s)y(s), \quad \hat{\theta}_t = V_t^{-1} b_t$

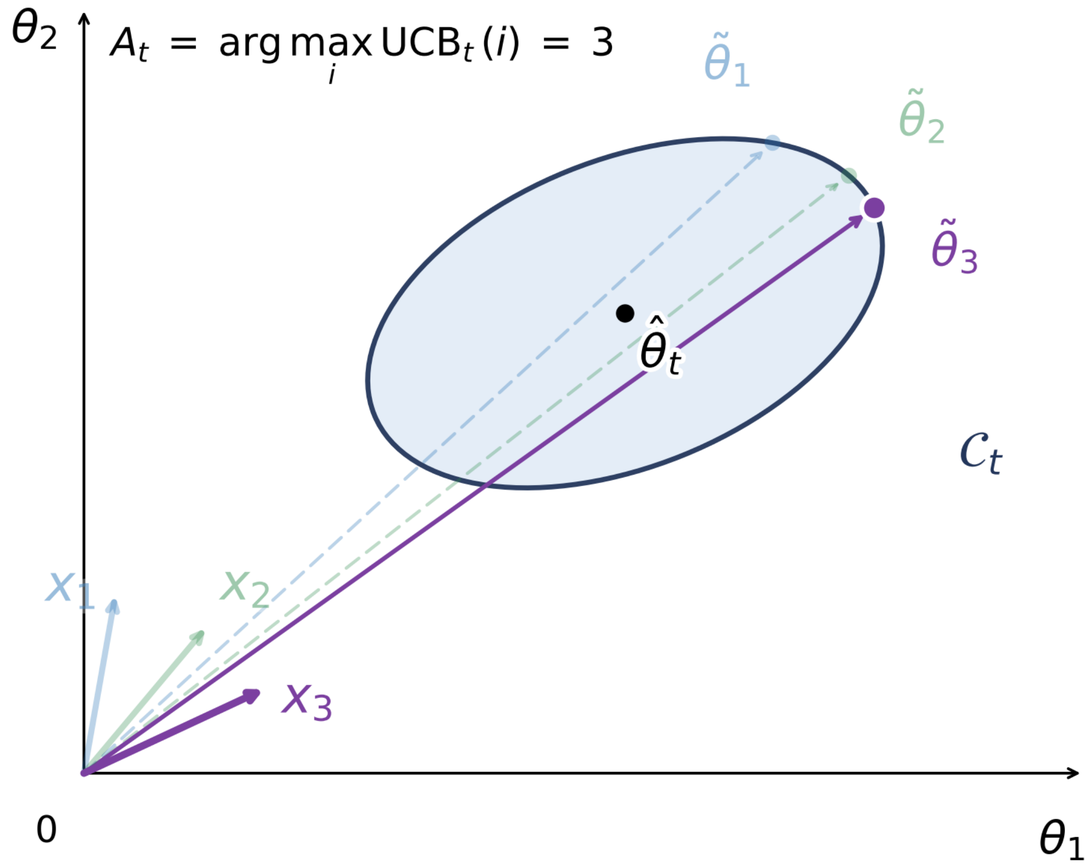
- Intuitively, we believe that θ lies somewhere close to $\hat{\theta}_t$, and uncertainty is given by V_t^{-1}. The following distribution captures this belief:

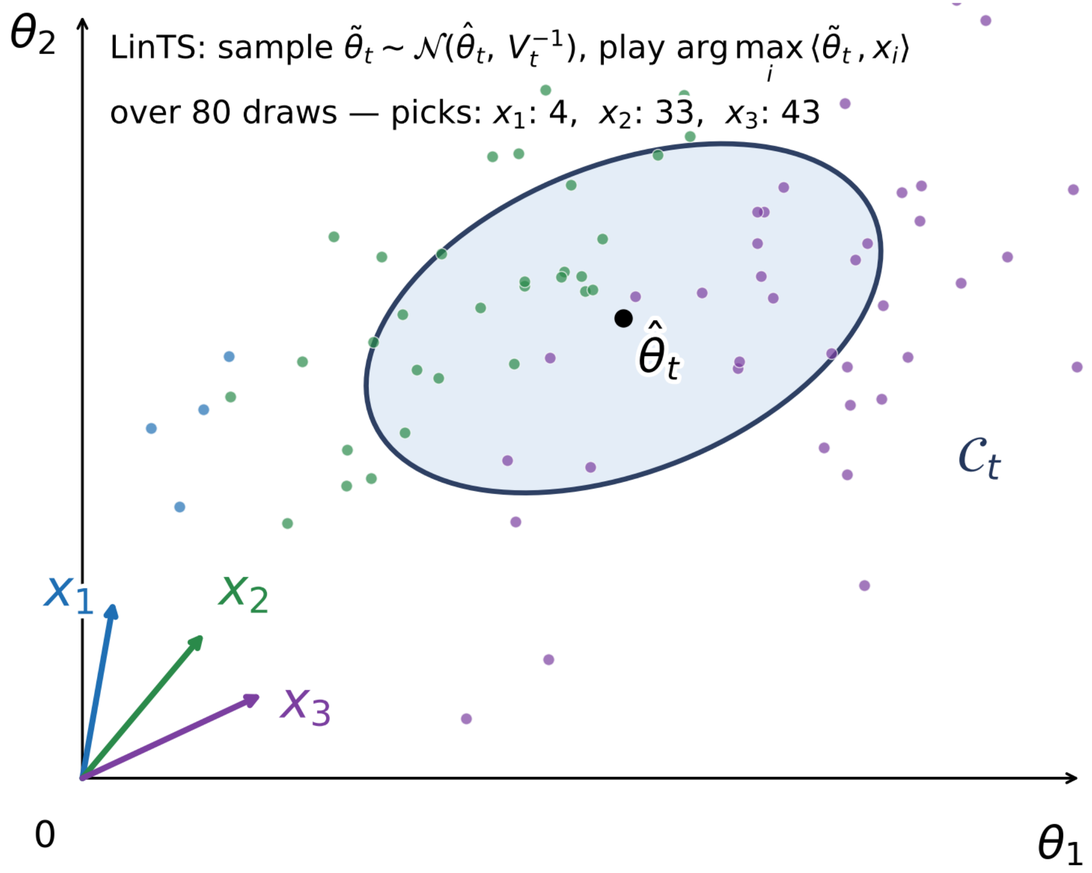
$$f(\theta \mid \text{data}) \approx \mathcal{N}\left(\hat{\theta}_t, \alpha^2 V_t^{-1}\right)$$

- With Gaussian noise, then using Bayes' rule, we get same updates for b_t, V_t as LinUCB

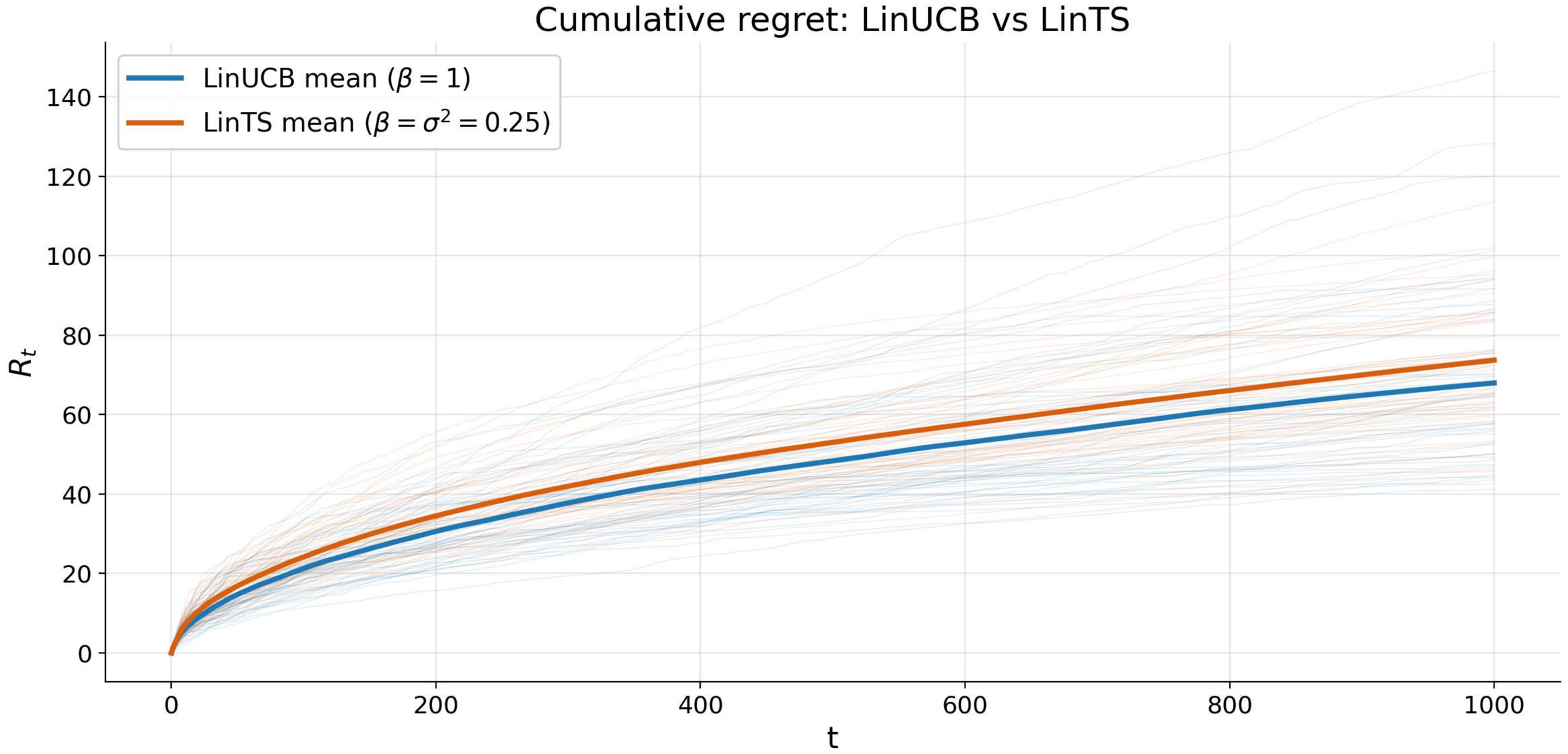
LinUCB vs LinTS

Arm Picking Strategy





LinTS vs LinUCB



Conclusion

- Thompson Sampling works for linear bandits as well
- It is a popular choice in production grade systems
- It’s performance is comparable to that of LinUCB

- Next class: we revisit where context vectors come from, in real-world systems