Stochastic Linear Bandits An Empirical Study

Students: Amer Essakine, Théo Molfessis,
Lecturer: Claire Vernade,

Problem 1

The action generation function returns an array of K action vectors uniformly sampled from the unit sphere in $\mathbb{R}^d$. These vectors are first generated with K independant d dimensional standard Gaussian vectors, then normalized. This way of proceeding ensures the isotropy.

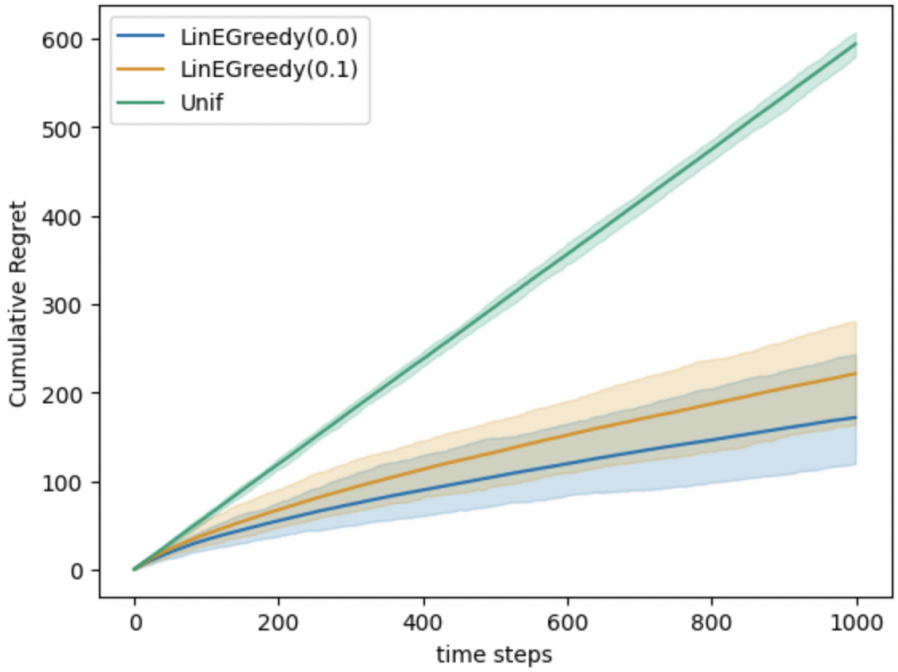

Figure 1: Comparison of regret between a uniform random policy and linear epsilon-greedy algorithms, using $\epsilon = 0$ and $\epsilon = 0.1$.

The epsilon-greedy algorithm was tested on a simple problem where θ is a random vector with values between 0 and 1, and all actions are generated randomly on the unit sphere. This algorithm exhibits a naive behavior, as it continues to explore with a probability $\epsilon > 0$ even when we start to have a good knowledge of θ . As a result, the regret increases when $\epsilon > 0$. To improve performance, we would expect an algorithm that reduces exploration over time.

Matrix inversion

The computational complexity of matrix inversion is $O(d^3)$. To enhance performance, we can apply the Sherman-Morrison formula, which reduces the complexity to $O(d^2)$.

$$(A + aa^\top)^{-1} = A^{-1} - \frac{A^{-1}aa^\top A^{-1}}{1 + a^\top A^{-1}a}$$

where $A = \lambda I_d$ and a represents the chosen arm.

As shown in Figure 2, the computational time is significantly reduced when using the Sherman-Morrison formula compared to the traditional matrix inversion method.

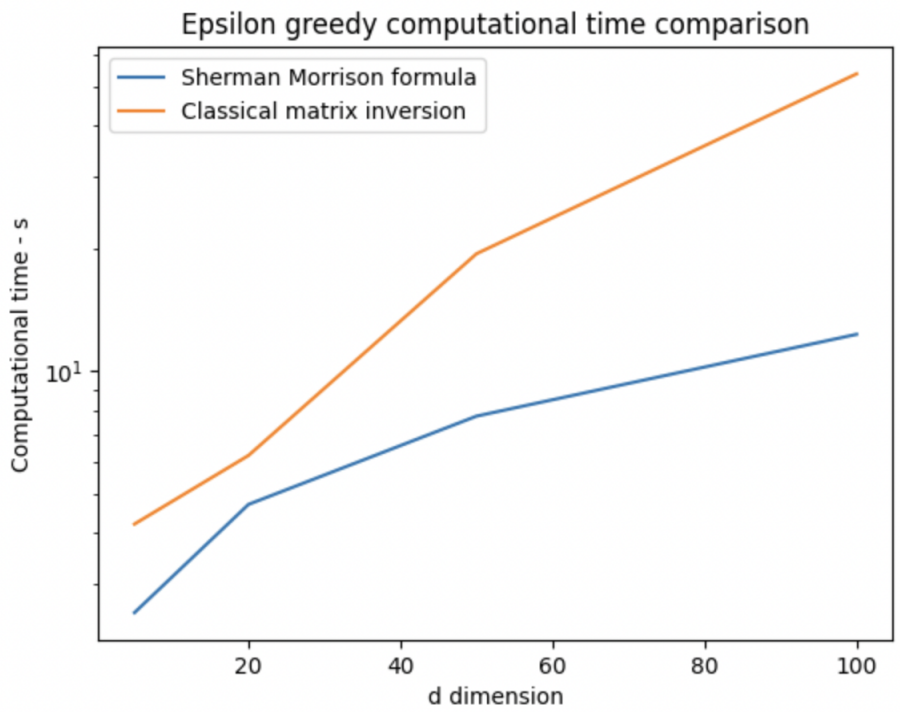

Figure 2: Computational time comparison of the Linear Epsilon-Greedy algorithm using classical matrix inversion versus the Sherman-Morrison inversion.

Problem 2

linUCB : We present an implementation of the LinUCB algorithm, where the UCB strategy is adapted to the context of contextual linear bandits. At each step, the arm is chosen according to the following rule $x_{t+1} = \arg\max_{x \in \mathcal{X}_t} \left[x^T \theta_t^\lambda + \|x\|_{(B_t^\lambda)^{-1}} \beta(t, \delta) \right]$
We use the threshold as provided in the slides. In our implementation, θ_* is approximated by the estimated parameter at time t : θ_t^λ.

LinTS : For the Thompson Sampling (TS) algorithm, we estimate the posterior distribution at every time step t. To derive this distribution, we assume we have a Gaussian prior $\mathcal{N}(\mathbf{0}, \lambda I)$. We write Bayes' theorem to express the posterior distribution:

$$P(\theta | x_1, r_1, ..., x_t, r_t) \propto P(r_1, ..., r_t | \theta, x_1, ..., x_t) \cdot P(\theta)$$

Using the prior distribution and the assumption that the reward follows a linear model: $r_t = \theta_t^T x_t + \epsilon_t$, we find that distribution is Gaussian given by :

$$\theta | x_1, r_1, ..., x_t, r_t \sim \mathcal{N}(\hat{\theta}_t^\lambda, \sigma^2 (B_t^\lambda)^{-1}$$

Where

$$B_t^\lambda = \sum_{s=1}^{t} x_s x_s^T + \frac{\sigma^2}{\lambda} I \text{ and } \hat{\theta}_t^\lambda = (B_t^\lambda)^{-1} (\sum_{s=1}^{t} r_s x_s)$$

Experiments We compare the two strategies alongside the Linear Epsilon-Greedy approach on action sets represented by K vectors. These vectors are sampled from a standard normal distribution in $\mathbb{R}^d$ and subsequently normalized to lie on the unit sphere. We compare strategies across

varying dimensions and numbers of arms to investigate how these factors influence the performance of each model.

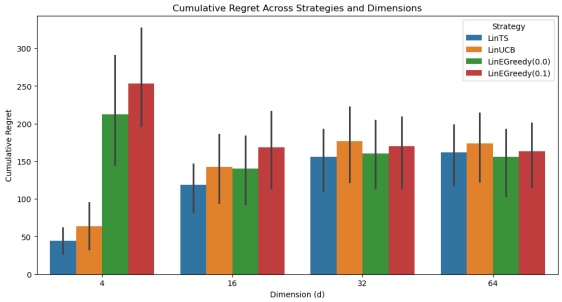

Figure 1: The maximum cumulative regret when varying the dimension

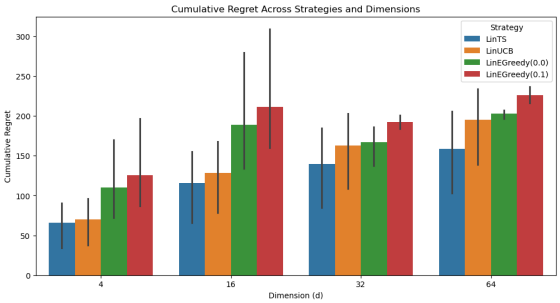

Figure 2: The maximum cumulative regret when variying the number of arms

LinTS is the algorithm that performs the best in general, showing the lowest cumulative regret. It outperforms LinUCB, particularly in lower dimensions (e.g., 4 and 16), where its regret grows more slowly, indicating superior performance. As the number of dimensions and arms increases (for example 64 dimensions and 64 arms), the performances of the algorithms converge, with LinTS remaining slightly ahead. The LinEpsilonGreedy algorithm, especially with epsilon=0.0 (pure exploitation), shows the poorest performance, with significantly higher cumulative regret. Introducing a small amount of exploration (epsilon = 0.1) slightly improves its performance, but it still lags behind both LinTS and LinUCB. However, in higher-dimensional settings (d = 64), the algorithms perform more similarly.

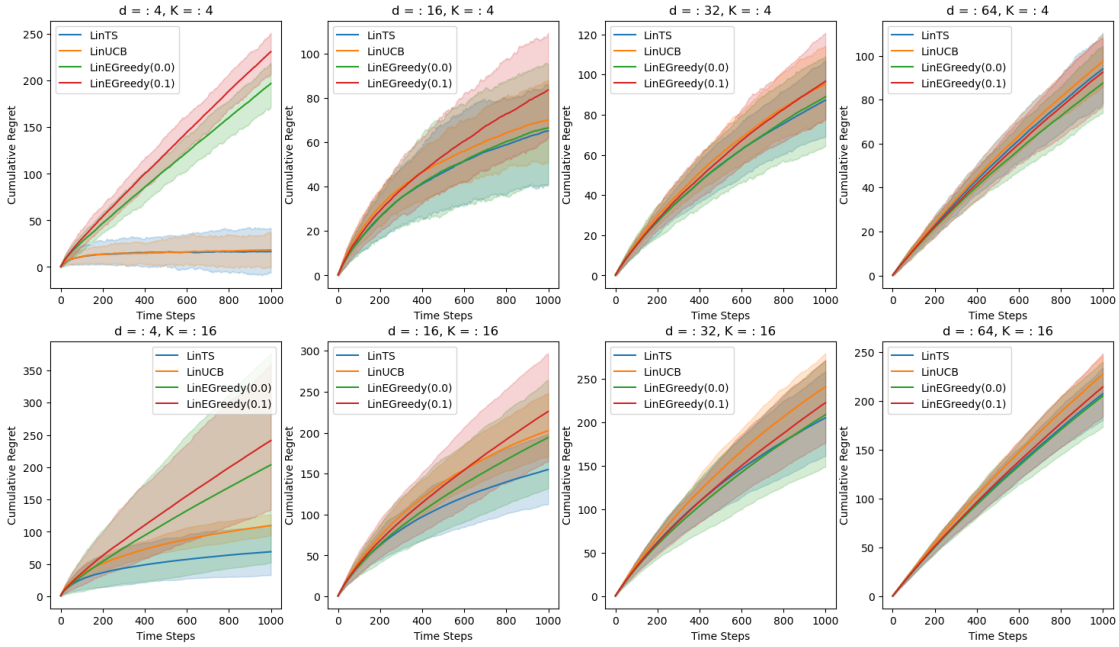

Figure 3: Cumulative regret comparison of the four strategies across varying dimensions and number of arms.

References

Lattimore, T. and Szepesvari, C. (2017). The end of optimism? an asymptotic analysis of finite-armed linear bandits. In *Artificial Intelligence and Statistics*, pages 728–737. PMLR.

A Bonus section: The role of the action set

In this bonus section, we compare LinUCB and UCB across fixed action sets, highlighting how LinUCB can underperform compared to UCB on certain action sets.
First, we generate a random action set by sampling from a standard normal distribution and then normalize it to lie on the unit sphere. This action set is kept fixed throughout the experiment. In the first experiment, we set the dimension $d = 3$ and the number of arms to 7. In the second experiment, we set $K = d = 7$ and fix the action set to the canonical basis. We report the finding of the two experiments :

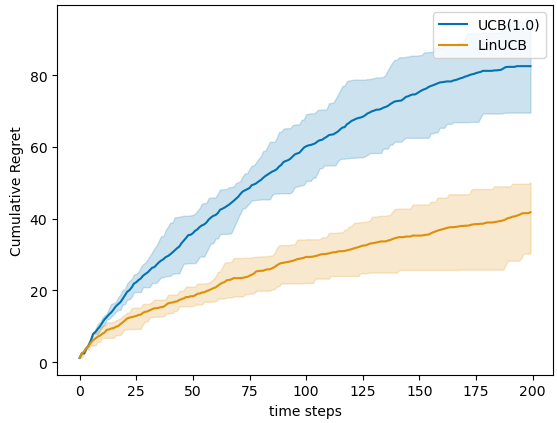

Figure 4: The cumulative regret for a randomly generated fixed set with $d = 3$ and $k = 7$.

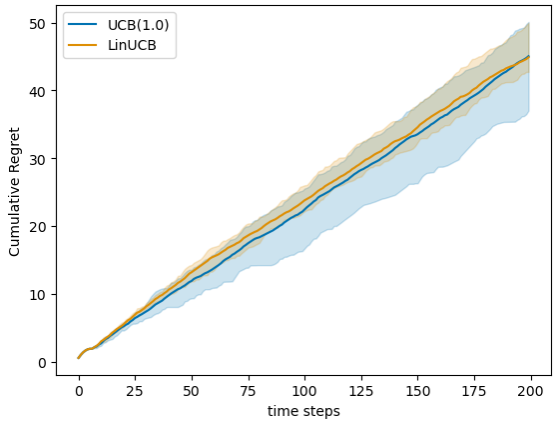

Figure 5: The cumulative regret for a fixed action set using the canonical basis with $d = 7$ and $k = 7$.

In the first experiment, we observe that LinUCB outperforms UCB, as expected, due to its ability to incorporate contextual information effectively. However, for the canonical basis, both algorithms perform similarly. This outcome can be attributed to the nature of the action set, as LinUCB depend heavily on the geometry of the action set. In fact, it is possible to construct an action set for which LinUCB would underperform UCB. This was proposed in Lattimore and Szepesvari (2017), Lattimore et al. We consider $d = 2$ and take $\mathcal{X} = \{x_1, x_2, x_3\}$ to be our fixed action set, where

$$x_1 = e_1, \quad x_2 = e_2, \quad x_3 = (1 - \epsilon, 8\alpha\epsilon),$$

The true parameter is $\theta = e_1$, which means that $x^* = e_1$ and $\delta_x = \epsilon$ while $_2 = 1$ with ϵ being a sufficiently small positive number and $\alpha > 1$. Specifically, we set $\epsilon = 10^{-5}$ and $\alpha = \frac{2}{3}$, which gives the following results.

As shown in the figure, UCB significantly outperforms LinUCB, with even their variances clearly separated. This performance difference arises from the way we defined $\mathcal{X}$. Specifically, the points x_1 and x_3 are nearly aligned and positioned very close to each other. This allows the algorithms

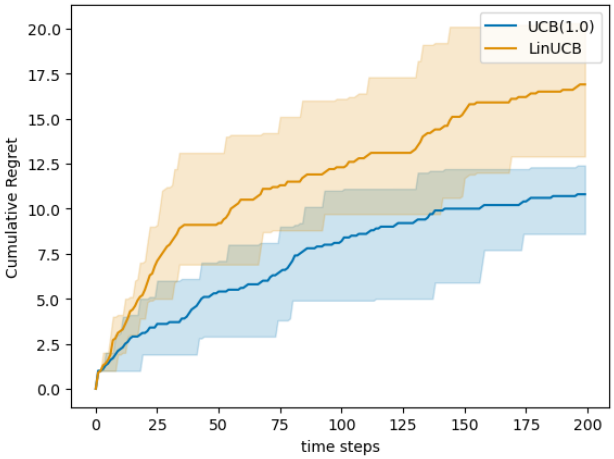

Figure 6: Comparaison between UCB and LinUCB for fixed action set $\mathcal{X}$

to quickly identify that e_2 is suboptimal. However, discarding e_2 results in a loss of valuable information on how to explore effectively.