Stochastic Linear Bandits An Empirical Study

Students: Mohammed Raki, Lucas Versini
Lecturer: Claire Vernade

Problem 1: Linear Epsilon Greedy (/5)

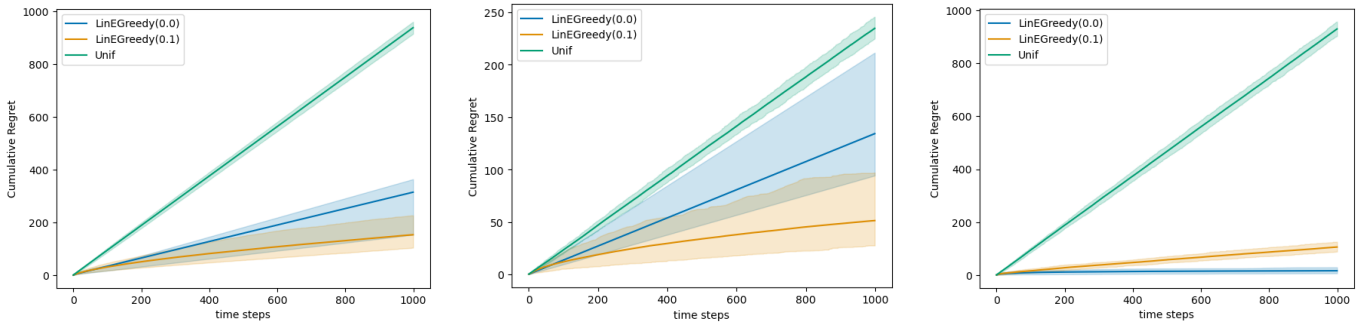
Experiments:

In all our experiments, all actions are vectors with L^2 norm equal to 1. That way, theoretical results, such as the one from the lecture by Abbasi-Yadkori et al., 2011, hold.

When the actions are sampled *at each step* from a Gaussian distribution with mean $\mathbf{0}$, Epsilon Greedy with $\varepsilon = 0$ behaves better than Epsilon Greedy with $\varepsilon > 0$. **In fact, the regret increases when ε increases**, see Figure 2. This is because varying the set of actions automatically leads to some exploration, even for $\varepsilon = 0$, see Figure 1c for instance. Using $\varepsilon > 0$ adds some useless exploration.

However, for *a fixed set of actions*, the regret for $\varepsilon = 0$ often becomes larger than for $\varepsilon > 0$, see Figure 1a. This is because for $\varepsilon = 0$ there is no exploration: once a decent action has been found, it will tend to be kept afterwards, even though it may not the optimal one.

The same thing happens when we use a fixed set of actions, where each action is drawn according to a Gaussian distribution *with mean* θ_* and renormalized, the ε-greedy agent with $\varepsilon = 0$ essentially behaves like the uniform agent (linear regret), while the ε-greedy agent with $\varepsilon = 0.1$ behaves much better, see Figure 1b. In this specific case, most of the actions will be close to θ_*. So when the 0-greedy agent finds an action that is close to θ_*, it will very likely keep it, even though there may be a better action.



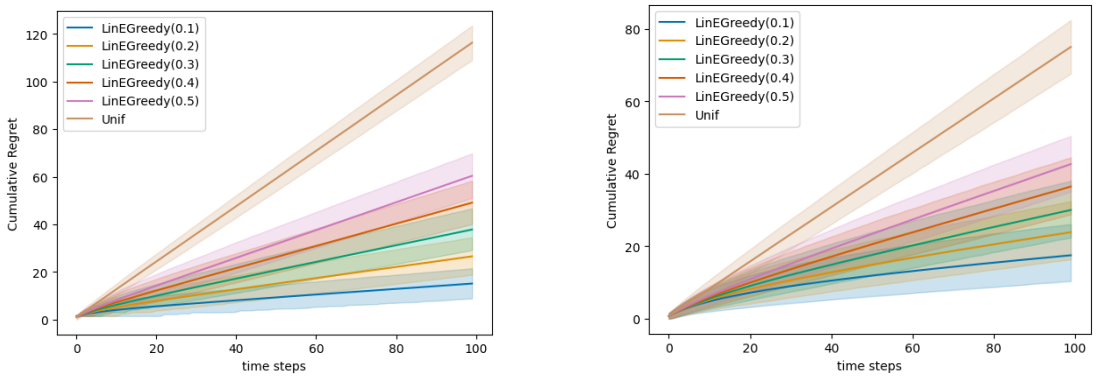
(a) Fixed action set (mean $\mathbf{0}, K = 10$) **(b)** Fixed action set (mean θ_*, $K = 10$) **(c)** Changing action set (mean $\mathbf{0}$, $K = 10$)

Figure 1: Linear ε-greedy

So overall, the Linear Epsilon Greedy agent is better than the uniform agent, but the value of ε has to be tuned carefully, depending on the exact context (changing set of actions, number of arms, etc.).

When the action set is random at each time step, the best strategy remains to *always* pick the action that seems to be the best, rather than exploring with probability $\epsilon > 0$. This is why ϵ-greedy with $\epsilon = 0$ performs better *when the action set is random*. So overall, ε-greedy with $\varepsilon > 0$ is not such a good baseline, sich it is easily outperformed by the greedy method (that is, the 0-graddy

algorithm).



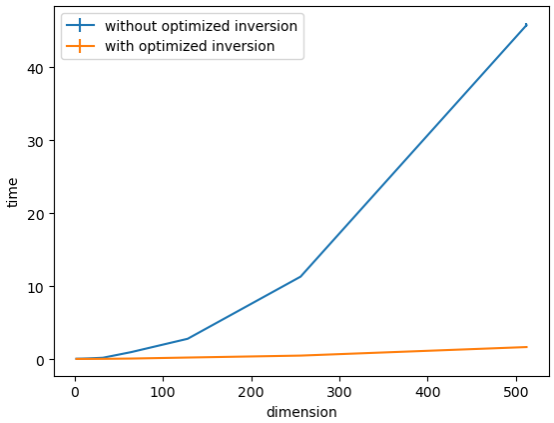
(a) Fixed action set **(b)** Changing action set

Figure 2: Linear ε-greedy, influence of ϵ

Discussion on the matrix inversion step:

At each step, the vector θ^* is estimated by $\hat{\theta}_t^\lambda = (B_t^\lambda)^{-1}\left(\sum_{s=1}^{t} r_s x_s\right)$, where we defined $B_{t+1}^\lambda = \lambda I_d + \sum_{s=1}^{t+1} x_s x_s^T = B_t^\lambda + x_{t+1}x_{t+1}^T$. We can find at least two ways of inverting B_{t+1}^λ:

- Directly inverting it, which can be done in $O(d^3)$ (because B_{t+1}^λ is a $d \times d$ matrix).
- Using Sherman–Morrison formula, $(B_{t+1}^\lambda)^{-1} = (B_t^\lambda)^{-1} - \dfrac{((B_t^\lambda)^{-1}x_{t+1})(x_{t+1}^T(B_t^\lambda)^{-1})}{1 + x_{t+1}^T(B_t^\lambda)^{-1}x_{t+1}}$, which can be done in $O(d^2)$ (because we simply multiply $d\times d$ matrices by d-dimensional vectors). Be careful, writing $(B_t^\lambda)^{-1}(x_{t+1}x_{t+1}^T)(B_t^\lambda)^{-1}$ instead of $((B_t^\lambda)^{-1}x_{t+1})(x_{t+1}^T(B_t^\lambda)^{-1})$ would increase the complexity to $O(d^3)$.



The figure on the left shows the execution time for $d \in \{2, 4, 8, 16, 32, 64, 128, 256, 512\}$, averaged over ten runs.

We clearly see that using Sherman–Morrison formula is much faster.

Problem 2: LinUCB and LinTS (/5)

We implement the LinUCB and LinTS agents where The Linear UCB upper bound is computed using the β-function presented in the lecture and since the action set is on the unit sphere, we

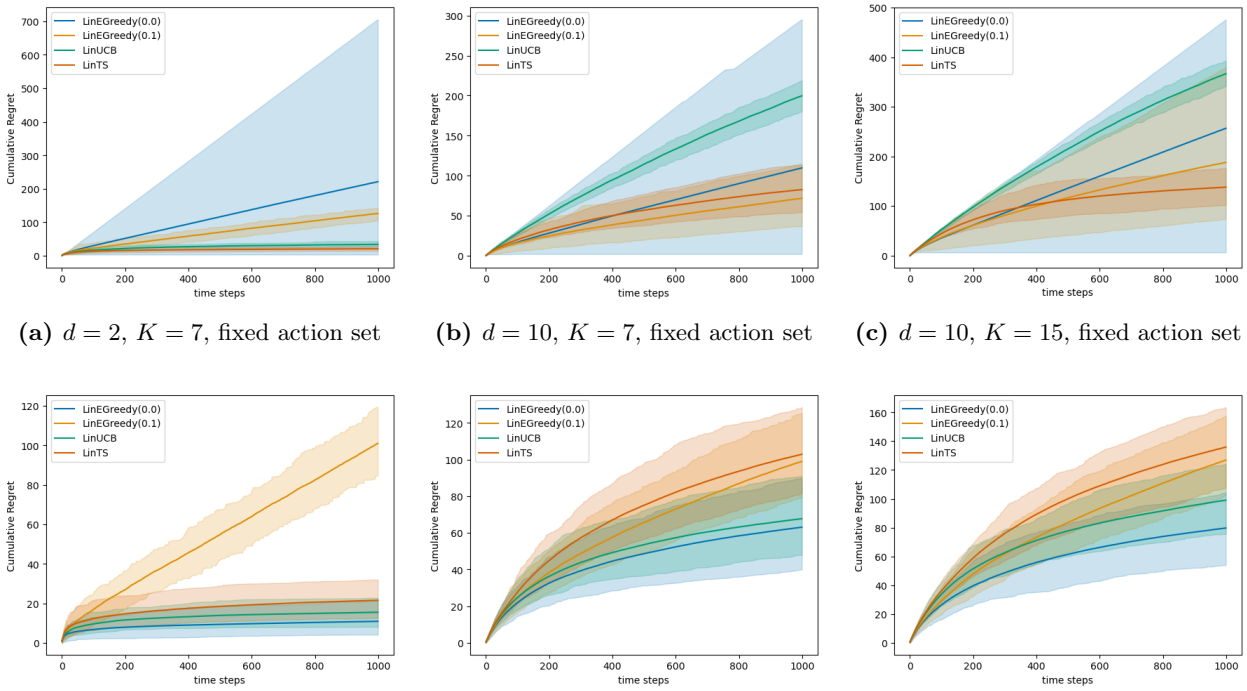
have that $\|\theta^*\| = 1$. On the other hand, The Linear Thompson Sampling is based on a Bayesian approach, as $\tilde{\theta}$ is generated at each time step given the prior information.

Posterior for Thompson sampling: $\boxed{\theta \mid x_1, r_1, \ldots, x_t, r_t \sim \mathcal{N}\left(\widehat{\theta}_t^{\mu}, \sigma^2 \left(B_t^{\mu}\right)^{-1}\right)}$

where: $\mu = \dfrac{\sigma^2}{\lambda}, \quad B_t^{\mu} = \mu I + \sum_{i=1}^{t} x_i x_i^T, \quad \hat{\theta}_t^{\mu} = \left(B_t^{\mu}\right)^{-1} \left(\sum_{i=1}^{t} r_i x_i\right)$

Experiments:

We test different values for the dimension d $(2, 10)$ and various possible numbers of arms K $(7, 15)$. Figure 3 presents the regret plots for several combinations of (d, K) and for each algorithm: Linear Epsilon Greedy ($\epsilon = 0.0$), Linear Epsilon Greedy ($\epsilon = 0.1$), Linear UCB, and Linear Thompson Sampling. In half the experiments, the action set is fixed, and in the other half it changes at each step. The horizon is fixed at $T = 1000$, and all other parameters remain constant across the experiments. For each of the 6 cases studied, the parameter θ is the same random vector of dimension d across all algorithms.



(a) $d = 2$, $K = 7$, fixed action set **(b)** $d = 10$, $K = 7$, fixed action set **(c)** $d = 10$, $K = 15$, fixed action set

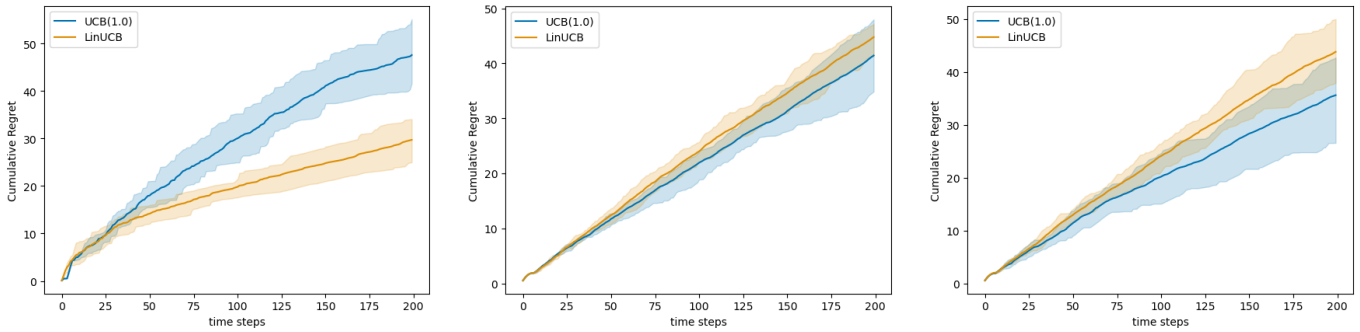
(d) $d = 2$, $K = 7$, changing actions **(e)** $d = 10$, $K = 7$, changing actions **(f)** $d = 10$, $K = 15$, changing actions

Figure 3: Comparison of different agents across different values of d and K.

We observe that for a fixed action set, LinTS and ε-greedy with $\varepsilon > 0$ achieve the best performance in terms of regret. However, for a changing action set, 0-greedy and LinUCB perform better, with an improvement for 0-greedy when K increases.

So overall, there is no agent that is always better than the others. The results are highly dependent on the number of arms K, the dimension d, and whether the action set is fixed or changes at each step.

Appendix - Bonus section: The role of the action set



(a) $K = 7, d = 3$

(b) $K = 7, d = 7$, canonical basis

(c) $K = 7, d = 7$, canonical basis, tighter bound (2 instead of 6)

Figure 4: UCB vs LinUCB

Here, the action set is fixed.

We see in Figure 4 that when $d < K$, LinUCB performs better than UCB, whereas for $d = K$ and actions given by the canonical basis, the two algorithms yield similar results.

The former case is explained by the fact that LinUCB allows some sort of communication between the actions, through the covariance matrix.

The latter case is explained by the fact that the actions are orthogonal, so the covariance between two distinct arms is equal to 0, and therefore using LinUCB or UCB yields similar results.

In fact, when using a factor 6 instead of 2 in the formula for UCB, LinUCB performs worse than UCB.

If we consider the case $d = 2, K = 3$, and actions $a_1 = (1, 0), a_2 = (0, 1), a_3 = (1 - \epsilon, 8\alpha\epsilon)$ (where $0 < \epsilon \ll \alpha$), and $\theta_* = a_1$, then the optimal arm is of course a_1.

What we observe in practice (though it may not really be seen in Figure 5) with UCB is that the arm a_2 is quickly not used by the agent, because it induces a large regret, whereas arms a_1 and a_3 are essentially used the same number of times.

Therefore the regret is often linear (due to the choice of a_3 instead of a_1).

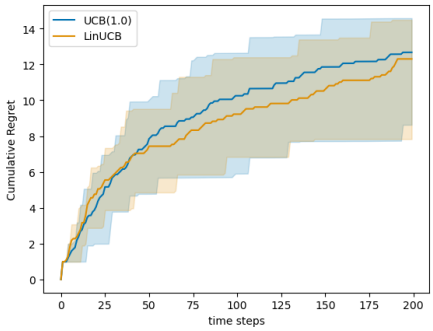


Figure 5: $d = 2, K = 3$, using the actions described previously

This shows that depending on the context, LinUCB may or may not be a good choice.